

TRƯỜNG ĐẠI HỌC SÀI GÒN  
KHOA CÔNG NGHỆ THÔNG TIN



## PHÁT TRIỂN PHẦN MỀM MÃ NGUỒN MỞ

---

Xây dựng phần mềm

HarmonyHub - Ứng dụng nghe nhạc trên PYTHON

---

GVHD: Từ Lăng Phiêu  
SV: Nguyễn Anh Danh - 3121410103  
Phan Duy - 3121410003  
Văn Phú Hiếu - 3121410201  
Đỗ Nguyễn Hoàng Tuấn - 3121410554

TP. HỒ CHÍ MINH, THÁNG 5/2024





## Lời cảm ơn

Chúng em xin gửi lời cảm ơn chân thành nhất đối với các thầy cô ở khoa Công Nghệ Thông Tin, trường Đại học Sài Gòn đã tạo điều kiện cho chúng em tiếp cận và tìm hiểu để hoàn thành đồ án môn học lần này. Và chúng em cũng xin chân thành cảm ơn thầy Từ Lăng Phiêu giáo viên giảng dạy đã nhiệt tình hướng dẫn chúng em hoàn thành đồ án lần này. Trong quá trình thực hiện nghiên cứu và thực hiện làm báo cáo đồ án, do kinh nghiệm thực tế chưa được nhiều, nên bài báo cáo của chúng em có thể vẫn còn những thiếu sót và chưa được hoàn chỉnh nên mong rằng chúng em sẽ nhận được những đóng góp ý kiến đóng góp bổ ích từ thầy để chúng em có thể khắc phục cho những bài báo cáo sau.

Chúng em xin trân trọng cảm ơn thầy!



## Mục lục

<b>1</b>	<b>Phần 1: Mở đầu</b>	<b>5</b>
1.1	Lý do chọn đề tài	5
1.2	Mục đích - mục tiêu của đề tài	5
1.3	Phạm vi đề tài	5
1.4	Nội dung đề tài	5
<b>2</b>	<b>Xây dựng ứng dụng nghe nhạc trên Python bằng các thư viện của Python</b>	<b>6</b>
2.1	Đôi nét về ứng dụng nghe nhạc	6
2.2	Tổng quan và phân tích	6
2.2.1	Khảo sát	6
2.2.2	Phân tích	7
2.3	Xây dựng ứng dụng nghe nhạc	8
2.3.1	Cài đặt các thư viện cần thiết	8
2.4	Các bước khởi tạo ứng dụng nghe nhạc	8
2.4.1	Khai báo thư viện	8
2.5	Sơ đồ cơ sở dữ liệu	9
2.6	Kiến trúc ứng dụng	11
2.7	Xây dựng chức năng	11
2.7.1	Thiết kế socket server và client và cài đặt vài chức năng cho ứng dụng nghe nhạc	28
2.7.2	Xây dựng giao diện và chức năng cho ứng dụng	35
2.7.2.a	Đăng nhập	35
2.7.2.b	Đăng kí	36
2.7.2.c	Đặt lại mật khẩu	36
2.7.3	Trang chủ	37
2.7.4	Tìm kiếm	37
2.7.5	Nghe nhạc	38
2.7.6	Playlist	38
2.7.7	Giao diện và chức năng cho Server	40
2.7.7.a	Quản lý nhạc	40
2.7.7.b	Thêm bài hát mới	41
2.7.7.c	Chỉnh sửa bài hát	42
2.7.7.d	Xóa bài hát	44
2.7.8	Quản lý danh sách Album	44
2.7.8.a	Thêm Album mới	45
2.7.8.b	Chỉnh sửa Album	46
2.7.9	Quản lý thông tin nghệ sĩ	47
2.7.9.a	Hiển thị danh sách nghệ sĩ	47
2.7.9.b	Thêm nghệ sĩ mới	47
2.7.9.c	Chỉnh sửa thông tin nghệ sĩ	48
2.7.10	Quản lý người dùng	50
2.7.10.a	Hiển thị danh sách người dùng	50
2.7.10.b	Thêm người dùng mới	50
2.7.10.c	Chỉnh sửa thông tin người dùng	51
2.7.11	Khởi động server	54
2.7.12	Tắt server	54
2.8	Kết luận	56



2.8.1	Ưu điểm . . . . .	56
2.8.2	Nhược điểm . . . . .	56
2.8.3	Hướng phát triển . . . . .	56

# 1 Phần 1: Mở đầu

## 1.1 Lý do chọn đề tài

Công nghệ thông tin ngày càng trở thành một phần không thể thiếu trong cuộc sống hiện đại, và ngôn ngữ lập trình Python đã và đang đóng một vai trò quan trọng trong việc phát triển các ứng dụng công nghệ thông tin. Python với cấu trúc rõ ràng, dễ đọc và dễ học, đã trở thành một trong những lựa chọn hàng đầu cho nhiều lập trình viên trên toàn thế giới.

Âm nhạc là một phần quan trọng của cuộc sống, mang lại niềm vui, sự thư giãn và là nguồn cảm hứng cho con người. Với sự phát triển của công nghệ, việc nghe nhạc đã trở nên dễ dàng hơn bao giờ hết. Tuy nhiên, việc tìm kiếm một ứng dụng nghe nhạc phù hợp với nhu cầu cá nhân không phải lúc nào cũng dễ dàng.

Chính vì vậy, chúng em đã chọn đề tài "Phát triển ứng dụng nghe nhạc sử dụng ngôn ngữ Python". Mục tiêu của chúng em là tạo ra một ứng dụng nghe nhạc đơn giản nhưng đầy đủ tính năng, dễ sử dụng và có thể tùy chỉnh theo nhu cầu của người dùng. Chúng em tin rằng, với sự linh hoạt và mạnh mẽ của Python, chúng em có thể đạt được mục tiêu này.

## 1.2 Mục đích - mục tiêu của đề tài

### -Mục đích

- Nắm chắc được kỹ năng và kiến thức về ngôn ngữ lập trình Python.
- Tìm hiểu về cách thức hoạt động của một ứng dụng nghe nhạc.
- Tìm hiểu về thư viện Pygame, MySQL Connector, ...
- Cũng cố, áp dụng, nâng cao kiến thức đã học.

### -Mục tiêu

- Vận dụng được tính chất của lập trình hướng đối tượng.
- Xây dựng một ứng dụng nghe nhạc đơn giản, dễ sử dụng.

## 1.3 Phạm vi đề tài

- Ứng dụng có thể tìm kiếm, phát nhạc từ cơ sở dữ liệu.
- Ứng dụng có thể tạo danh sách phát, tạo playlist.

## 1.4 Nội dung đề tài

Bao gồm 2 phần:

- Phần 1: Mở đầu
  - Mở đầu
- Phần 2: Thực hiện ứng dụng nghe nhạc trên Python
  - Xây dựng ứng dụng nghe nhạc bằng cách sử dụng các thư viện của Python:
    - \* Phân tích yêu cầu

- \* Thiết kế kiến trúc
- \* Thiết kế cơ sở dữ liệu
- \* Xây dựng chức năng
- \* Kiểm thử và sửa lỗi

## 2 Xây dựng ứng dụng nghe nhạc trên Python bằng các thư viện của Python

### 2.1 Đôi nét về ứng dụng nghe nhạc

-Ứng dụng nghe nhạc không chỉ là một công cụ giúp người dùng tìm kiếm và phát nhạc từ cơ sở dữ liệu. Nó còn là một nền tảng giúp người dùng trải nghiệm âm nhạc theo cách riêng của họ.

-Tìm kiếm và phát nhạc: Ứng dụng nghe nhạc cho phép người dùng tìm kiếm bài hát, album, nghệ sĩ yêu thích của họ từ một cơ sở dữ liệu lớn. Người dùng có thể phát nhạc trực tiếp từ ứng dụng, điều chỉnh âm lượng, chọn chế độ phát (như phát lại, lặp lại, ngẫu nhiên), và xem thông tin chi tiết về bài hát đang phát.

-Tạo danh sách phát: Người dùng có thể tạo danh sách phát cá nhân, thêm bài hát vào danh sách phát, sắp xếp thứ tự các bài hát trong danh sách phát, và chia sẻ danh sách phát với bạn bè. Điều này giúp người dùng tổ chức bộ sưu tập âm nhạc của họ theo cách mà họ muốn.

-Tạo playlist: Playlist là một tính năng mạnh mẽ giúp người dùng tổ chức và phát nhạc theo chủ đề, tâm trạng, hoặc sự kiện. Người dùng có thể tạo playlist, thêm bài hát vào playlist, và chia sẻ playlist với cộng đồng.

-Khám phá âm nhạc mới: Ứng dụng nghe nhạc thường có tính năng khám phá, giúp người dùng tìm kiếm và khám phá âm nhạc mới dựa trên sở thích âm nhạc của họ. Điều này giúp người dùng mở rộng bộ sưu tập âm nhạc của họ và khám phá những nghệ sĩ, thể loại mới.

-Như vậy, ứng dụng nghe nhạc không chỉ giúp người dùng nghe nhạc, mà còn giúp họ trải nghiệm âm nhạc theo cách riêng của họ, khám phá âm nhạc mới, và chia sẻ niềm đam mê âm nhạc với cộng đồng. Đây chính là lý do mà việc phát triển ứng dụng nghe nhạc sử dụng Python trở nên hấp dẫn và thú vị. Python với khả năng mạnh mẽ và linh hoạt của mình, cho phép chúng ta tạo ra những ứng dụng nghe nhạc phong phú và đa dạng, phục vụ cho nhu cầu ngày càng đa dạng của người dùng.

### 2.2 Tổng quan và phân tích

#### 2.2.1 Khảo sát

-Ứng dụng nghe nhạc là một nền tảng trực tuyến phổ biến, đặc biệt trong cộng đồng người yêu âm nhạc và các nhóm cộng đồng trực tuyến khác. Một trong những lợi ích của ứng dụng nghe nhạc là tính linh hoạt và đa dạng của nó. Người dùng có thể tùy chỉnh các danh sách phát và quyền truy cập cho từng bài hát, tạo ra các thể loại khác nhau để quản lý bộ sưu tập âm nhạc và tùy chỉnh các cài đặt âm thanh cho phù hợp với nhu cầu của mình.

-Việc tạo ứng dụng nghe nhạc cũng là một lợi ích lớn, giúp việc quản lý bộ sưu tập âm nhạc trở nên dễ dàng hơn và giảm thiểu thời gian và công sức cho các hoạt động quản lý. Ứng dụng có thể tự động thực hiện các nhiệm vụ như kiểm tra và cập nhật thông tin bài hát, quản lý danh sách phát và nhiều tính năng khác ...

-Tuy nhiên, ứng dụng nghe nhạc cũng có một số hạn chế như việc không thể tùy chỉnh giao diện của ứng dụng hoặc các danh sách phát quá nhiều.



### 2.2.2 Phân tích

Thư viện	Mô tả
mysql.connector	Thư viện mysql.connector trong Python cung cấp các hàm để tương tác với cơ sở dữ liệu MySQL. Cho phép tạo ra các ứng dụng có khả năng tương tác mạnh mẽ với cơ sở dữ liệu, thực hiện các tác vụ như truy vấn, cập nhật, và quản lý dữ liệu. Thư viện mysql.connector giúp lập trình viên tạo ra các ứng dụng có khả năng tương tác mạnh mẽ với cơ sở dữ liệu MySQL.
pygame	Thư viện pygame trong Python cung cấp các hàm để tạo ra các ứng dụng đồ họa, bao gồm các trò chơi và các ứng dụng đa phương tiện khác. Tạo ra các ứng dụng có đồ họa mạnh mẽ, tương tác với người dùng qua các sự kiện đầu vào, và tạo ra các hiệu ứng âm thanh và hình ảnh. Thư viện pygame giúp lập trình viên tạo ra các ứng dụng đồ họa mạnh mẽ và tương tác.
tkinter	Thư viện tkinter trong Python cung cấp các hàm để tạo ra các ứng dụng đồ họa, các ứng dụng đa phương tiện khác. Tạo ra các ứng dụng có đồ họa mạnh mẽ, tương tác với người dùng qua các sự kiện đầu vào, và tạo ra các hiệu ứng âm thanh và hình ảnh.
PIL	Thư viện PIL trong Python cung cấp các hàm để tạo ra các ứng dụng đồ họa, các ứng dụng đa phương tiện khác. Tạo ra các ứng dụng có đồ họa mạnh mẽ, tương tác với người dùng qua các sự kiện đầu vào, và tạo ra các hiệu ứng âm thanh và hình ảnh.

Bảng 1: Các thư viện Python được sử dụng cho việc phát triển ứng dụng



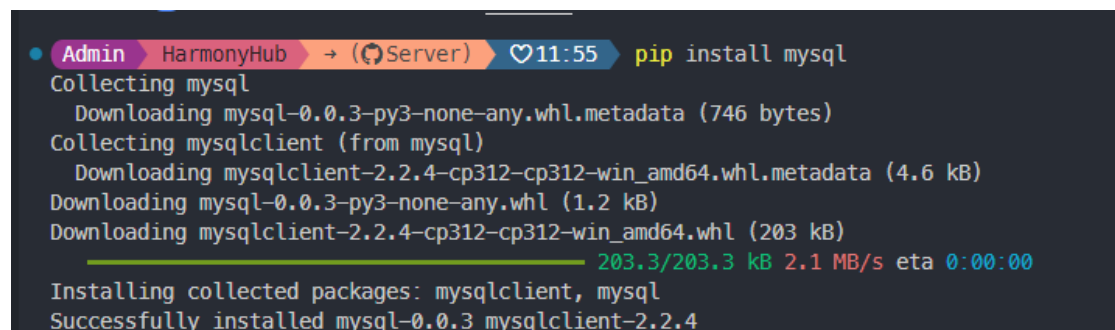
## 2.3 Xây dựng ứng dụng nghe nhạc

### 2.3.1 Cài đặt các thư viện cần thiết

-Visual Studio Code (64-bit).

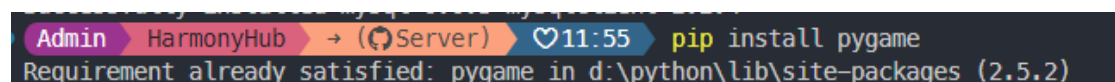
-Điều đầu tiên cần làm để lập trình ứng dụng nghe nhạc trên Python là cài đặt các thư viện cần thiết. Các thư viện này giúp chúng ta tương tác với hệ thống, tạo ra các ứng dụng đa luồng, tương tác với cơ sở dữ liệu, và tạo ra các ứng dụng đồ họa mạnh mẽ.

Sau đó hệ thống sẽ tự động cài đặt các thư viện cần thiết:



```
Admin HarmonyHub → (Server) 11:55 pip install mysql
Collecting mysql
  Downloading mysql-0.0.3-py3-none-any.whl.metadata (746 bytes)
Collecting mysqlclient (from mysql)
  Downloading mysqlclient-2.2.4-cp312-win_amd64.whl.metadata (4.6 kB)
  Downloading mysql-0.0.3-py3-none-any.whl (1.2 kB)
  Downloading mysqlclient-2.2.4-cp312-cp312-win_amd64.whl (203 kB)
    203.3/203.3 kB 2.1 MB/s eta 0:00:00
Installing collected packages: mysqlclient, mysql
Successfully installed mysql-0.0.3 mysqlclient-2.2.4
```

Hình 1: Cài đặt thư viện MySQL



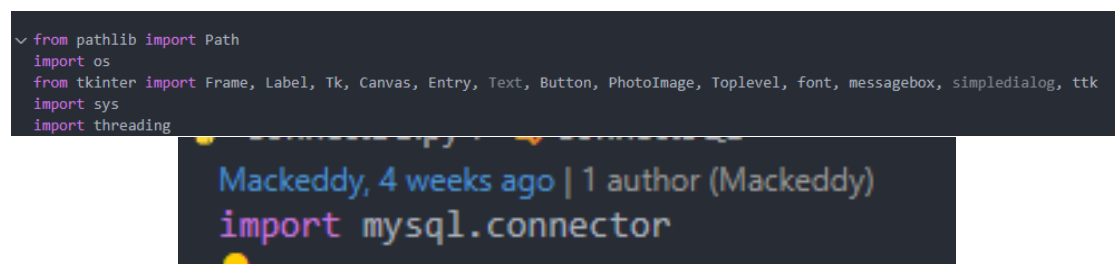
```
Admin HarmonyHub → (Server) 11:55 pip install pygame
Requirement already satisfied: pygame in d:\python\lib\site-packages (2.5.2)
```

Hình 2: Cài đặt thư viện Pygame (Do đã cài đặt nên ta sẽ không cần cài nữa)

## 2.4 Các bước khởi tạo ứng dụng nghe nhạc

### 2.4.1 Khai báo thư viện

Sau khi cài đặt các thư viện cần thiết, ta tiến hành khai báo các thư viện cần dùng:



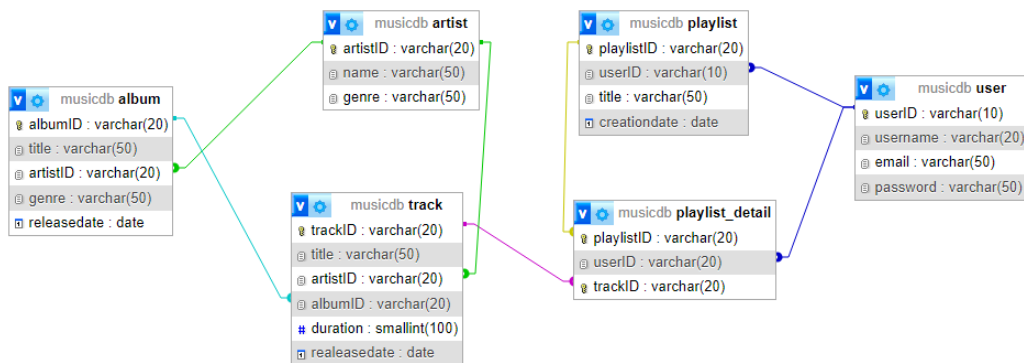
```
from pathlib import Path
import os
from tkinter import Frame, Label, Tk, Canvas, Entry, Text, Button, PhotoImage, Toplevel, font, messagebox, simpledialog, ttk
import sys
import threading

Mackeddy, 4 weeks ago | 1 author (Mackeddy)
import mysql.connector
```

Hình 3: Khai báo thư viện

## 2.5 Sơ đồ cơ sở dữ liệu

- Để lưu trữ thông tin về bài hát, album, nghệ sĩ, và các thông tin khác, chúng ta cần tạo một cơ sở dữ liệu.
- Chúng ta đồng thời sẽ sử dụng XAMPP để tạo cơ sở dữ liệu MySQL và thiết kế cơ sở dữ liệu cho ứng dụng trên đó.
- Chúng ta có sơ đồ cơ sở dữ liệu như sau:



Hình 4: Sơ đồ cơ sở dữ liệu cho ứng dụng nghe nhạc

- Sau khi chúng ta đã có sơ đồ cơ sở dữ liệu, chúng ta sẽ tiến hành kết nối cơ sở dữ liệu với ứng dụng thông qua thư viện MySQL Connector và thực hiện các thao tác truy vấn, cập nhật dữ liệu từ cơ sở dữ liệu.

```
connectdb.py 7/11
Mackeddy, 4 weeks ago | 1 author (Mackeddy)
import mysql.connector

Mackeddy, 4 weeks ago | 1 author (Mackeddy)
class ConnectSQL():
    @staticmethod
    def connect_mysql():
        try:
            connection = mysql.connector.connect(
                host='localhost',
                database='musicdb', "musicdb": Unknown word.
                user='root',
                password=''
            )
            if connection.is_connected():
                print("Connected to MySQL database")
                return connection
            else:
                print("Connection failed")
                return None
        except mysql.connector.Error as error:
            print("Error while connecting to MySQL", error)
            return None
```

Hình 5: Khai báo thư viện và kết nối cơ sở dữ liệu thông qua class ConnectDB.py

-Giải thích:

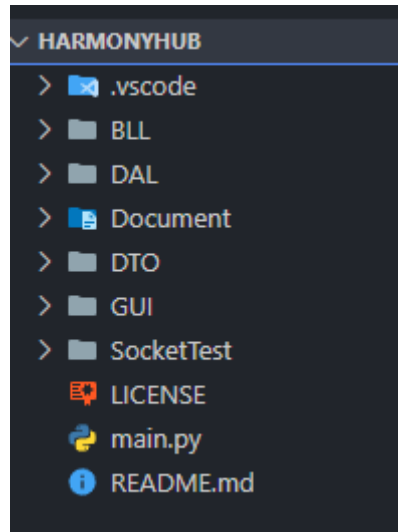
+Class ConnectDB chứa hàm khởi tạo và hàm kết nối cơ sở dữ liệu gồm các thông số sau:

- host: địa chỉ IP của máy chủ cơ sở dữ liệu.
- user: tên người dùng để truy cập cơ sở dữ liệu.
- password: mật khẩu để truy cập cơ sở dữ liệu.
- database: tên cơ sở dữ liệu.

+Nếu kết nối thành công, hàm kết nối sẽ trả về một đối tượng connection, ngược lại sẽ trả về None.

## 2.6 Kiến trúc ứng dụng

Về kiến trúc thiết kế mô hình phát triển ứng dụng nghe nhạc, chúng ta sẽ sử dụng mô hình 3 lớp (3-tier architecture) bao gồm 3 lớp chính: DAL, BLL, và GUI.



Hình 6: Kiến trúc ứng dụng nghe nhạc

## 2.7 Xây dựng chức năng

Để xây dựng chức năng cho ứng dụng nghe nhạc, chúng ta sẽ tạo ra các class đối tượng (Object) tương ứng với các bảng trong cơ sở dữ liệu ở thư mục DTO (Data Transfer Object):

```
MrDemoa, 2 weeks ago | 2 authors (Mackeddy and others)
▼ class AlbumDTO:
  ▼ def __init__(self, albumID, title, artistID, genre, releasedate):
      self.albumID = albumID
      self.title = title
      self.artistID = artistID
      self.genre = genre
      self.releasedate = releasedate MrDemoa, 2 weeks ago • updat
```

```
MrDemoa, 2 weeks ago | 2 authors (Mackeddy and others)
class ArtistDTO:
    Mackeddy, 4 weeks ago • Them chuc nang DAL, BLI

def __init__(self, artistID, name, genre):
    self.artistID = artistID
    self.name = name
    self.genre = genre

Mackeddy, last week | 1 author (Mackeddy)
class PLDetailDTO:
    Mackeddy, last week • Them chuc nang DAL, BLI

def __init__(self, PlaylistID, UserID, trackID):
    self.PlaylistID = PlaylistID
    self.UserID = UserID
    self.trackID = trackID

Mackeddy, last week | 1 author (Mackeddy)
class PlayListDTO:
    Mackeddy, last week • Them chuc nang DAL, BLI

def __init__(self, playlistID, userID, trackID, title, createiondate):
    self.playlistID = playlistID    "playlist": Unknown word.
    self.userID = userID
    self.trackID = trackID
    self.title = title
    self.createiondate = createiondate    Mackeddy, 4 weeks ago • Them chuc nang DAL, BLI

MrDemoa, 2 weeks ago | 2 authors (Mackeddy and others)
class TrackDTO:
    Mackeddy, 4 weeks ago • Them DTO, hoan thien chuc nang DAL, BLI

def __init__(self, trackID, title, artistID, albumID, duration, releasedate):
    self.trackID = trackID
    self.title = title
    self.artistID = artistID
    self.albumID = albumID
    self.duration = duration
    self.releasedate = releasedate    "releasedate": Unknown word.
```

```
Mackeddy, 4 weeks ago | 1 author (Mackeddy)
class UserDTO:
    def __init__(self, userID, username, email, password):
        self.userID = userID
        self.username = username
        self.email = email
        self.password = password
```

Hình 7: Các class DTO

-Sau khi chúng ta đã tạo các class DTO, chúng ta sẽ tiến hành tạo các class ở tầng DAL (Data Access Layer) để thực hiện các thao tác truy vấn, cập nhật dữ liệu từ cơ sở dữ liệu:



```
PlaylistDAL.py
DAL > PlaylistDAL.py > PlaylistDAL
You, 11 hours ago | 4 authors (Mackeddy and others)
1  import os
2  import sys
3  sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6  from DAL.ConnectDB import ConnectSQL
7  from DTO.PlayListDTO import PlayListDTO
8
9  You, 11 hours ago | 4 authors (Mackeddy and others)
10 class PlaylistDAL:
11     con = ConnectSQL.connect_mysql()
12
13     def getAllData(self):
14         global con
15         cursor = PlaylistDAL.con.cursor()
16         cursor.execute("select * from playlist")
17         records = cursor.fetchall()
18         cursor.close()
19         return records
20
21     def getPlaylistIDByUserID(self, userID):
22         cursor = self.con.cursor()
23         cursor.execute("select playlistID from playlist where userID = %s", (userID,))
24         records = cursor.fetchall()
25         cursor.close()
26         return records
27
28     def getDataPlaylistFromUserID(self, userID):
29         cursor = self.con.cursor()
30         cursor.execute("select playlistID, title, creationdate from playlist where userID = %s", (userID,))
31         records = cursor.fetchall()
32         cursor.close()
33         return records
34
35     def generatePlaylistID(self):
36         cursor = self.con.cursor()
37         cursor.execute("select PlaylistID from playlist order by playlistID desc limit 1")
38         playlist_id = cursor.fetchone()
39         if playlist_id:
40             id = playlist_id[0]
41             id = int(id[2:]) + 1
42         else:
43             id = 1
44         return "PL" + str(id).zfill(4)
```

```
44     def insert(self, playlist_dto):
45         cursor = self.con.cursor()
46         cursor.execute("insert into playlist values(%s, %s, %s, %s)", (playlist_dto.playlistID, playlist_dto.userID,
47                                                                                               playlist_dto.title, playlist_dto.creationdate))
48         self.con.commit()
49         cursor.close()
50
51     def delete(self, id):
52         global con
53         cursor = self.con.cursor()
54         cursor.execute("delete from playlist where playlistID = %s", (id,))
55         count = int(cursor.rowcount)
56         self.con.commit()
57         cursor.close()
58
59         if count > 0:
60             print("Xoa thanh cong")
61             return True
62         else:
63             print("Ma khong ton tai")
64             return False
65
66     def update(playlist_dto):
67         global con
68         cursor = con.cursor()
69         cursor.execute("update playlist set userID = %s, title = %s, creationdate = %s where playlistID = %s", ("cre
70                                                                                               (playlist_dto.userID, playlist_dto.title, playlist_dto.creationdate, playlist_dto.playlistID))
71         con.commit
72         cursor.close()
```

Hình 8: Code cho class PlaylistDAL gồm các phương thức CRUD (Create, Read, Update, Delete)

-Giải thích:

- Phương thức **getAllData(self)** trả về danh sách tất cả các playlist trong cơ sở dữ liệu.
  - +Chi tiết hơn:
    - Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu thông qua class ConnectDB.
    - Sau đó, ta sẽ tạo 1 cursor để thực hiện các thao tác truy vấn dữ liệu.
    - Cuối cùng, ta sẽ thực hiện truy vấn dữ liệu và trả về kết quả lưu về records
- Phương thức **getPlaylistIDByUserID(self, userID)** trả về danh sách playlist dựa vào id của user.
  - +Chi tiết hơn:
    - Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu.
    - Sau đó, ta sẽ tạo 1 cursor để thực hiện các thao tác truy vấn dữ liệu và lấy ra toàn bộ playlist dựa vào id của user.
    - Cuối cùng, ta sẽ thực hiện truy vấn dữ liệu và trả về kết quả lưu về records (trong trường hợp này, ta sẽ truy vấn dữ liệu dựa vào id của user).
- Phương thức **getDataPlayListFromUserId(self, userID)** trả về thông tin chi tiết của một playlist dựa vào id.
  - +Chi tiết hơn:
    - Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu thông qua class ConnectDB.
    - Sau đó, ta sẽ tạo 1 cursor để thực hiện các thao tác truy vấn dữ liệu.



-Cuối cùng, ta sẽ thực hiện truy vấn dữ liệu và trả về kết quả lưu về records (trong trường hợp này, ta sẽ truy vấn dữ liệu dựa vào id của user).

- Phương thức **generatePlaylistID(self)** tạo một ID duy nhất cho playlist mới.

+Chi tiết hơn:

-Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu thông qua class ConnectDB.

-Sau đó, ta sẽ tạo 1 cursor để thực hiện các thao tác truy vấn dữ liệu.

-Lấy ra kết quả đầu tiên của query, kiểm tra xem nếu nó tồn tại thì lấy ID đó, loại bỏ ký tự đầu tiên (giả sử là 'PL'), chuyển phần còn lại thành số và cộng thêm một. Điều này đảm bảo rằng mỗi playlist có một ID duy nhất.

Nếu không nhận được ID trả về, điều đó có nghĩa là chưa có playlist nào trong cơ sở dữ liệu. Vì vậy, nó đơn giản là đặt ID cho playlist mới là 1.

Cuối cùng, nó thêm 'PL' vào đầu ID (để chỉ ra rằng đó là Playlist ID) và trả về nó. Đây sẽ là ID duy nhất cho playlist mới.

- Phương thức **insert(self, playlist \_dto)** thêm một playlist mới vào cơ sở dữ liệu.

+Chi tiết hơn:

-Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu thông qua class ConnectDB.

-Sau đó, ta sẽ tạo 1 cursor để thực hiện các thao tác truy vấn dữ liệu.

-Sau khi chuẩn bị xong câu lệnh, nó được thực thi bằng cách sử dụng con trỏ (Cursor). Điều này thêm playlist mới vào cơ sở dữ liệu.

-Tuy nhiên, chỉ thực thi câu lệnh không đủ. Các thay đổi cần được lưu lại. Đó là lý do tại sao self.con.commit() được sử dụng - nó lưu lại bất kỳ thay đổi nào được thực hiện kể từ lần cuối cùng thay đổi được lưu lại.

-Cuối cùng, con trỏ được đóng. Điều này được thực hiện khi chúng ta đã hoàn thành với nó, để giải phóng tài nguyên.

- Phương thức **deletePlaylist(playlistID)** xóa một playlist dựa vào id.

+Chi tiết hơn:

-Đầu tiên, ta thiết lập kết nối và tạo con trỏ (Cursor).

-Sau đó, ta thực thi câu lệnh DELETE để xóa playlist dựa vào id và đếm số lượng dòng bị ảnh hưởng.

-Nếu số lượng dòng bị ảnh hưởng lớn hơn 0, điều đó có nghĩa là playlist đã được xóa thành công.

-Cuối cùng, ta lưu lại các thay đổi và đóng con trỏ.

- Phương thức **updatePlaylist(playlist \_dto)** cập nhật thông tin của một playlist.

+Cách thức hoạt động tương tự như phương thức insert, nhưng thay vì thêm mới, nó sẽ cập nhật thông tin của playlist đã tồn tại.

```
AlbumDAL.py
DAL > AlbumDAL.py > AlbumDAL
...
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.ConnectDB import ConnectSQL
7 from DTO.AlbumDTO import AlbumDTO
8
9 ...
10 class AlbumDAL:
11     con = ConnectSQL.connect_mysql()
12
13     def getAllData(self):
14         global con
15         cursor = AlbumDAL.con.cursor()
16         cursor.execute("select * from album")
17         records = cursor.fetchall()
18         cursor.close()
19         return records
20
21     def generateAlbumID(self):
22         cursor = self.con.cursor()
23         cursor.execute("select albumID from album order by albumID desc limit 1")
24         album_id = cursor.fetchone()
25         if album_id:
26             id = album_id[0]
27             id = int(id[2:]) + 1
28         else:
29             id = 1
30         return "AT" + str(id)
31
32     def insert(self, album_dto):
33         cursor = self.con.cursor()
34         cursor.execute("insert into album values(%s, %s, %s, %s, %s)", (album_dto.albumID,
35 album_dto.title, album_dto.artistID, album_dto.genre, album_dto.releasedate)) "releasedate": Unknown word.
36         self.con.commit()
37         cursor.close()
38
39     def update(self, album_dto):
40         cursor = self.con.cursor()
41         cursor.execute("update album set title = %s, artistID = %s, genre = %s, releasedate = %s where albumID = %s",
42 (album_dto.title, album_dto.artistID, album_dto.genre, album_dto.releasedate, album_dto.albumID))
43         self.con.commit()
44         cursor.close()
45
46     def getTracksFromAlbumID(self, albumID):
47         cursor = self.con.cursor()
48         cursor.execute("select * from track where albumID = %s", (albumID))
49         records = cursor.fetchall()
50         cursor.close()
51         return records
52
```

Hình 9: Code cho class AlbumDAL gồm các phương thức đọc, thêm, sửa

-Giải thích:

- Phương thức **getAllData(self)** trả về danh sách tất cả các album trong cơ sở dữ liệu.  
+ Tương tự như PlaylistDAL chỉ khác là trả về danh sách album.
- Phương thức **generateAlbumID(self, albumID)** tạo một ID duy nhất cho album mới.  
+Chi tiết hơn:  
-Đầu tiên, chúng ta thiết lập kết nối với cơ sở dữ liệu.



-Sau đó, chúng ta yêu cầu cơ sở dữ liệu trả về ID của album được thêm gần đây nhất. Chúng ta làm điều này bằng cách chạy một lệnh yêu cầu "cho tôi albumID từ bảng album, nhưng hãy đảm bảo rằng bạn đưa cho tôi cái cuối cùng bạn có".

-Bây giờ, có hai khả năng: hoặc chúng ta nhận được một ID trả về, hoặc không. Nếu chúng ta nhận được một ID, điều đó có nghĩa là đã có một số album trong cơ sở dữ liệu. Vì vậy, chúng ta lấy ID đó, loại bỏ hai ký tự đầu tiên (giả sử là 'AT'), chuyển phần còn lại thành một số và cộng thêm một vào nó. Điều này đảm bảo rằng mỗi album có một ID duy nhất.

-Nếu chúng ta không nhận được ID trả về, điều đó có nghĩa là chưa có album nào trong cơ sở dữ liệu. Vì vậy, chúng ta đơn giản là đặt ID cho album mới là 1.

-Cuối cùng, chúng ta thêm 'AT' vào đầu ID (để chỉ ra rằng đó là Album ID), và trả về nó. Đây sẽ là ID duy nhất cho album mới.

- Phương thức **insert(self, album\_dto)** tạo một album mới trong cơ sở dữ liệu.

+Tương tự như hàm insert của PlaylistDAL chỉ khác là thêm mới album.

- Phương thức **update(album\_dto)** sửa thông tin của một album.

+Tương tự như hàm update của PlaylistDAL chỉ khác là sửa album.

- Phương thức **getTracksFromAlbumID(self, albumID)** lấy ra danh sách các bài hát theo albumID.

+Tương tự như hàm getDataPlaylistFromUserId của PlaylistDAL chỉ khác là lấy ra danh sách bài hát theo albumID.

```
ArtistDAL.py
DAL > ArtistDAL.py > ArtistDAL > update
...
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5 from DAL.ConnectDB import ConnectSQL
6 from DTO.ArtistDTO import ArtistDTO
7
8 ...
9
10 class ArtistDAL:
11
12     con = ConnectSQL.connect_mysql()
13
14     def getAllData(self):
15         global con
16         cursor = ArtistDAL.con.cursor()
17         cursor.execute("select * from artist")
18         records = cursor.fetchall()
19         cursor.close()
20         return records
21
22     def generateArtistID(self):
23         cursor = self.con.cursor()
24         cursor.execute("select artistID from artist order by artistID desc limit 1")
25         artist_id = cursor.fetchone()
26         if artist_id :
27             id = artist_id[0]
28             id = int(id[2:]) + 1
29         else:
30             id = 1
31         return "AT" + str(id)
32
33     def insert(self,artist_dto):
34         cursor = self.con.cursor()
35         cursor.execute("insert into artist values(%s, %s, %s)", (artist_dto.artistID, artist_dto.name, artist_dto.genre))
36         self.con.commit()
37         cursor.close()
38
39     def update(self, artist_dto):
40         cursor = self.con.cursor()
41         cursor.execute("update artist set name = %s, genre = %s where artistID = %s", (artist_dto.name,
42         artist_dto.genre, artist_dto.artistID))
43         self.con.commit()
44         cursor.close()
45
46     def getTracksFromArtistID(self, artistID):
47         cursor = self.con.cursor()
48         cursor.execute("select * from track where artistID = %s", (artistID))
49         records = cursor.fetchall()
50         cursor.close()
51         return records
```

Hình 10: Code cho class ArtistDAL gồm các phương thức đọc, thêm, sửa

-Giải thích:

- Phương thức **getAllData(self)** trả về danh sách tất cả các nghệ sĩ trong cơ sở dữ liệu.
- Phương thức **generateArtistID(self)** tạo một ID duy nhất cho nghệ sĩ mới.
- Phương thức **insert(self, artist\_dto)** thêm một nghệ sĩ mới vào cơ sở dữ liệu.
- Phương thức **update(artist\_dto)** sửa thông tin của một nghệ sĩ.
- Phương thức **getTracksFromArtistID(self, artistID)** lấy ra danh sách các bài hát theo artistID.

```
L > PLDetailDAL.py > PLDetailDAL > deleteTrackInPlaylist
...
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.ConnectDB import ConnectSQL
7 from DTO.PLDetailDTO import PLDetailDTO
8
9 ...
10 class PLDetailDAL:
11     con = ConnectSQL.connect_mysql()
12
13     def insertTracktoPlaylist(self, pldetail_dto): "Trackto": Unknown word.
14         cursor = self.con.cursor()
15         cursor.execute("insert into detail_playlist values(%s, %s, %s)", (pldetail_dto.playlistID, pldetail_dto.userID,
16                                                                           pldetail_dto.trackID)) "pldetail": Unknown w
17         self.con.commit()
18         cursor.close()
19
20     def deleteTrackInPlaylist(self, trackID):
21         cursor = self.con.cursor()
22         cursor.execute("delete from detail_playlist where tracktrackID = %s", (trackID,)) "tracktrack": Unknown word.
23         count = int(cursor.rowcount)
24         self.con.commit()
25         cursor.close()
26
27         if count > 0:
28             return True
29         return False
```

Hình 11: Code cho class PLDetailDAL gồm 2 phương thức thêm và xóa

-Giải thích:

- Phương thức **insertTracktoPlaylist(self, pldetail\_dto)** thêm một bài hát vào playlist.
- Phương thức **deleteTrackInPlaylist(playlistID, trackID)** xóa một bài hát khỏi playlist theo trackID.

```
TrackDAL.py
DAL > TrackDAL.py > TrackDAL > update
You, yesterday | 3 authors (Mackeddy and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.ConnectDB import ConnectSQL
7 from DTO.TrackDTO import TrackDTO
8
9 You, yesterday | 3 authors (Mackeddy and others)
10 class TrackDAL():
11     con = ConnectSQL.connect_mysql()
12
13     def getAllData(self):
14         global con
15         cursor = TrackDAL.con.cursor()
16         cursor.execute("select * from track")
17         records = cursor.fetchall()
18         cursor.close()
19         return records
20
21     def generateTrackID(self):
22         cursor = self.con.cursor()
23         cursor.execute("select trackID from track order by trackID desc limit 1")
24         track_id = cursor.fetchone()
25         if track_id:
26             id = track_id[0]
27             id = int(id[1:]) + 1
28         else:
29             id = 1
30         return "T" + str(id).zfill(4)
31
32     def insert(self, track_dto):
33         try:
34             cursor = self.con.cursor()
35             if track_dto.albumID == "None":
36                 track_dto.albumID = None
37             cursor.execute("insert into track values(%s, %s, %s, %s, %s, %s)",
38                             (track_dto.trackID, track_dto.title, track_dto.artistID, track_dto.albumID,
39                              track_dto.duration, track_dto.releasedate)) "releasedate": Unknown word.
40             self.con.commit()
41             cursor.close()
42         except Exception as e:
43             print("DAL TRACK:", str(e))
44
45     def delete(self, trackID):
46         cursor = self.con.cursor()
47         cursor.execute("delete from track where trackID = %s", (trackID,))
48         count = int(cursor.rowcount)
49         self.con.commit()
50         cursor.close()
51
52         if count > 0:
53             print("Xoa thanh cong")
54         else:
55             print("ma khong ton tai")
56
57     def update(self, track_dto):
58         cursor = self.con.cursor()
59         if track_dto.albumID == "None":
60             track_dto.albumID = None
61         cursor.execute("update track set title = %s, artistID = %s, albumID = %s, duration = %s, releasedate = %s where trackID = %s",
62                         (track_dto.title, track_dto.artistID, track_dto.albumID, track_dto.duration, track_dto.releasedate,
63                          track_dto.trackID))
64         self.con.commit()
65         cursor.close()
```

Hình 12: Code cho class TrackDAL gồm các phương thức CRUD (Create, Read, Update, Delete)



-Giải thích:

- Phương thức **getAllData(self)** trả về danh sách tất cả các bài hát trong cơ sở dữ liệu.
- Phương thức **generateTrackID(self)** tạo một ID duy nhất cho bài hát mới.

+Chi tiết hơn:

-Đầu tiên, ta sẽ thiết lập kết nối đến cơ sở dữ liệu.

-Sau đó, ta sẽ tạo 1 cursor từ kết nối này để thực hiện các thao tác truy vấn dữ liệu.

-Ta thực hiện câu lệnh SQL để lấy ID cao nhất (mới nhất) từ bảng track.

-Chúng ta lấy kết quả ở dòng đầu tiên của câu lệnh truy vấn. Nếu kết quả tồn tại (tức là bảng không trống), ta lấy ID này, loại bỏ ký tự đầu tiên (giả sử là 'T'), chuyển phần còn lại thành số và cộng thêm một để tạo ID mới.

-Nếu kết quả không tồn tại (tức là bảng trống), ta sẽ bắt đầu với ID là 1.

-Cuối cùng, ta trả về ID mới này dưới dạng chuỗi, thêm tiền tố 'T' và đảm bảo rằng nó có độ dài 4 ký tự bằng cách thêm các số 0 vào đầu.

- Phương thức **insert(self, track\_dto)** thêm một bài hát mới vào cơ sở dữ liệu.
- Phương thức **update(track\_dto)** sửa thông tin của một bài hát.
- Phương thức **deleteTrack(trackID)** xóa một bài hát dựa vào id.

```
UserDAL.py M
DAL > UserDAL.py > ...
...
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4 from DAL.ConnectDB import ConnectSQL
5 from DTO.UserDTO import UserDTO
...
6 class UserDAL():
7
8     con = ConnectSQL.connect_mysql()
9
10    def getAllData(self):
11        global con
12        cursor = UserDAL.con.cursor()
13        cursor.execute("select * from user")
14        records = cursor.fetchall()
15        cursor.close()
16        return records
17
18    def generateUserID(self):
19        cursor = self.con.cursor()
20        cursor.execute("select userID from user order by userID desc limit 1")
21        user_id = cursor.fetchone()
22        if user_id :
23            id = user_id[0] + 1
24        else:
25            id = 1
26        return id
27
28    def insert(self, user_dto):
29        cursor = self.con.cursor()
30        cursor.execute("insert into user values(%s, %s, %s, %s)", (user_dto.userID, user_dto.username,
31                                                                    user_dto.email, user_dto.password))
32        self.con.commit()
33        cursor.close()
34
35    def checkUsernameAndPass(self, username, password):
36        cursor = self.con.cursor()
37        cursor.execute("select * from user where username = %s and password = %s", (username, password))
38        records = cursor.fetchall()
39        self.con.commit()
40        cursor.close()
41        if records:
42            return True
43        else:
44            return False
45
46    def update(self, user_dto):
47        cursor = self.con.cursor()
48        cursor.execute("update user set username = %s, email = %s, password = %s where userID = %s", (user_dto.username,
49                                                                    user_dto.email, user_dto.password, user_dto.userID))
50        self.con.commit()
51        cursor.close()
52
53    def checkUsername(self, username):
54        cursor = self.con.cursor()
55        cursor.execute("select * from user where username = %s", (username,))
56        records = cursor.fetchall()
57        self.con.commit()
58        cursor.close()
59        if records:
60            return True
61        else:
62            return False
63
64    def resetPassWord(self, username, password):
65        cursor = self.con.cursor()
66        cursor.execute("update user set password = %s where username = %s", (password, username))
67        self.con.commit()
68        cursor.close()
```

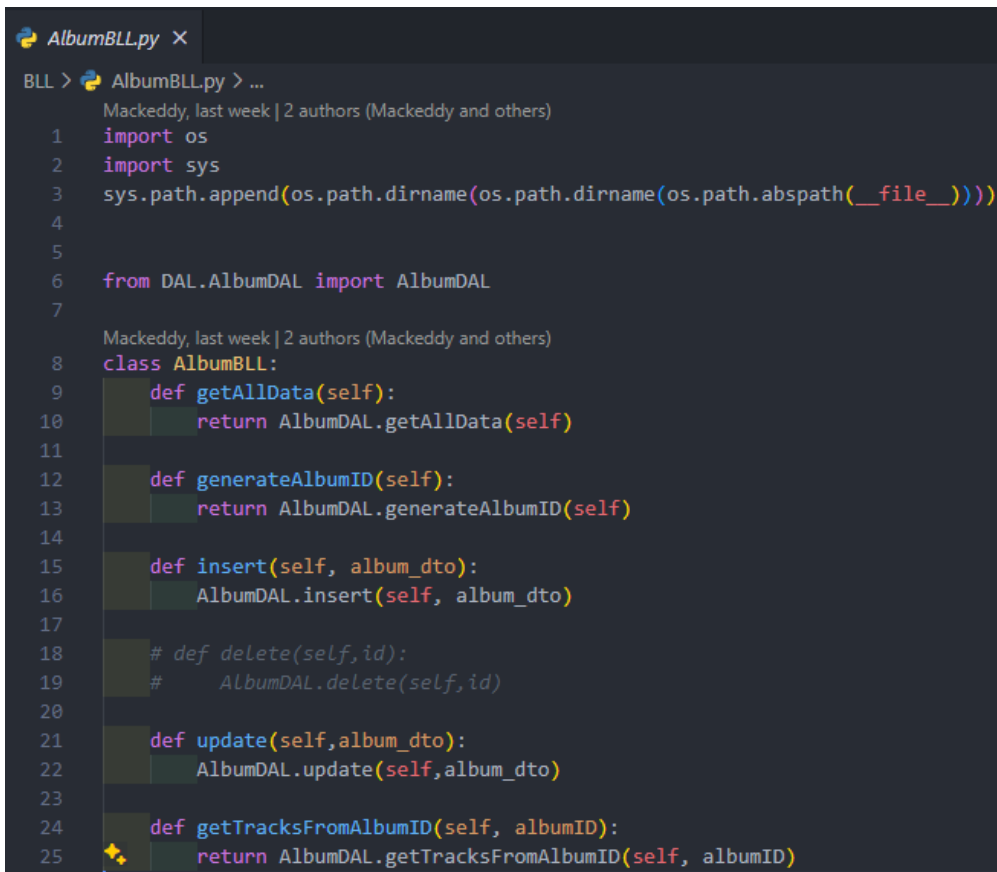
Hình 13: Code cho class UserDAL gồm các phương thức CRUD (Create, Read, Update, Delete)



-Giải thích:

- Phương thức **getAllData(self)** trả về danh sách tất cả các user trong cơ sở dữ liệu.
- Phương thức **generateUserID(self)** tạo một ID duy nhất cho user mới.
- Phương thức **checkUsernameAndPass(self, user\_dto)** kiểm tra username và password của user.
- Phương thức **insert(self, user\_dto)** thêm một user mới vào cơ sở dữ liệu.
- Phương thức **update(self, user\_dto)** sửa thông tin của một user.
- Phương thức **checkUsername(self, userID)** kiểm tra username của user.
- Phương thức **resetPassword(self, username, password)** reset password của user.

-Sau khi hoàn thành các class ở tầng DAL, chúng ta sẽ tiến hành tạo các class ở tầng BLL (Business Logic Layer) để thực hiện các thao tác xử lý logic cho ứng dụng.



```
AlbumBLL.py x
BLL > AlbumBLL.py > ...
Mackedy, last week | 2 authors (Mackedy and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.AlbumDAL import AlbumDAL
7
8 Mackedy, last week | 2 authors (Mackedy and others)
9 class AlbumBLL:
10     def getAllData(self):
11         return AlbumDAL.getAllData(self)
12
13     def generateAlbumID(self):
14         return AlbumDAL.generateAlbumID(self)
15
16     def insert(self, album_dto):
17         AlbumDAL.insert(self, album_dto)
18
19     # def delete(self, id):
20     #     AlbumDAL.delete(self, id)
21
22     def update(self, album_dto):
23         AlbumDAL.update(self, album_dto)
24
25     def getTracksFromAlbumID(self, albumID):
26         return AlbumDAL.getTracksFromAlbumID(self, albumID)
```

Hình 14: Code cho class AlbumBLL gồm các phương thức xử lý logic

```
ArtistBLL.py X
BLL > ArtistBLL.py > ...
Mackdeddy, last week | 2 authors (Mackdeddy and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.ArtistDAL import ArtistDAL
7
8 Mackdeddy, last week | 2 authors (Mackdeddy and others)
9 class ArtistBLL:
10     def getAllData(self):
11         return ArtistDAL.getAllData(self)
12
13     def generateArtistID(self):
14         return ArtistDAL.generateArtistID(self)
15
16     def insert(self,artist_dto):
17         ArtistDAL.insert(self,artist_dto)
18
19     def update(self,artist_dto):
20         ArtistDAL.update(self,artist_dto)
21
22     def getTracksFromArtistID(self, artistID):
23         return ArtistDAL.getTracksFromArtistID(self, artistID)
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
263
```

```
PLDetailBLL.py X
BLL > PLDetailBLL.py > ...
PhanDuy, 8 hours ago | 3 authors (Mackeddy and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.PLDetailDAL import PLDetailDAL
7
8 PhanDuy, 8 hours ago | 3 authors (Mackeddy and others)
9 class PLDetailBLL:
10
11     def insertTracktoPlaylist(self, playlist_dto): "Trackto": Unknown word.
12         PLDetailDAL.insertTracktoPlayList(self, playlist_dto) "Trackto": Unknown word.
13
14     def getTrackinPlayListofUserID(self, playlistID, userID): "Trackin": Unknown word.
15         return PLDetailDAL.getTrackinPlayListofUserID(self, playlistID, userID)
16
17     def udeleteTrackInPlayList(self, trackID): "udelete": Unknown word.
18         PLDetailDAL.update(self, trackID)

TrackBLL.py X
BLL > TrackBLL.py > ...
Mackeddy, 2 weeks ago | 2 authors (Mackeddy and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.TrackDAL import TrackDAL
7
8 Mackeddy, 2 weeks ago | 2 authors (Mackeddy and others)
9 class TrackBLL:
10     def getAllData(self):
11         return TrackDAL.getAllData(self)
12
13     def generateTrackID(self):
14         return TrackDAL.generateTrackID(self)
15
16     def insert(self, track_dto):
17         TrackDAL.insert(self, track_dto)
18
19     def delete(self,Trackid): "Trackid": Unknown word.
20         TrackDAL.delete(self,Trackid) "Trackid": Unknown word.
21
22     def update(self,track_dto):
23         TrackDAL.update(self,track_dto)
```

Hình 16: Code cho class PLDetailBLL, TrackBLL gồm các phương thức xử lý logic

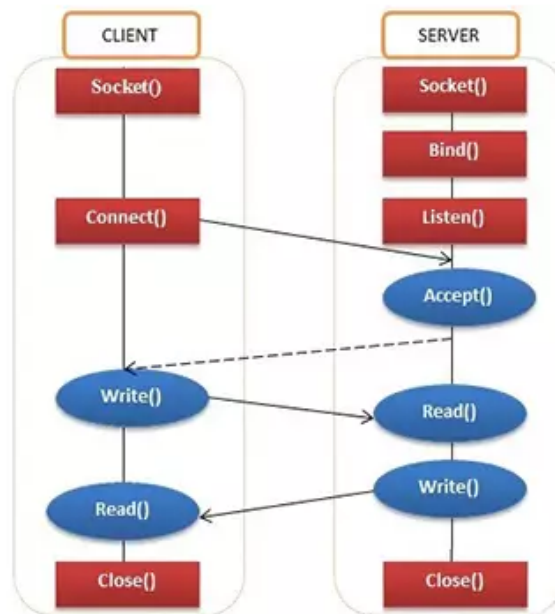
```
UserBLL.py M x
BLL > UserBLL.py > UserBLL > resetPassWord
You, 1 second ago | 4 authors (MrDemoa and others)
1 import os
2 import sys
3 sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
4
5
6 from DAL.UserDAL import UserDAL
7
8 You, 1 second ago | 4 authors (MrDemoa and others)
9 class UserBLL:
10     def getAllData(self):
11         return UserDAL.getAllData(self)
12
13     def hasUsername(self, username):
14         return UserDAL.hasUsername(self, username)
15
16     def insert(self, user_dto):
17         UserDAL.insert(self, user_dto)
18
19     def generateUserID(self):
20         return UserDAL.generateUserID(self)
21
22     def checkUsernameAndPass(self, username, password):
23         return UserDAL.checkUsernameAndPass(self, username, password)
24
25     def checkUsername(self, username):
26         return UserDAL.checkUsername(self, username)
27
28     def getUserIDByUsername(self, username):
29         user_id = UserDAL.getUserIDByUsername(self, username)
30         if user_id is None:
31             print("User not found")
32             return None
33         else:
34             return user_id
35
36     def getUsernameByUserId(self, userID):
37         username = UserDAL.getUsernameByUserID(self, userID)
38         if username is None:
39             print("User not found")
40             return None
41         else:
42             return username
43
44     def update(self, user_dto):
45         UserDAL.update(self, user_dto)
46
47     def resetPassWord(self, username, password):
48         UserDAL.resetPassWord(self, username, password)
49         You, 1 second ago
```

Hình 17: Code cho class UserBLL gồm các phương thức xử lý logic

- Sau khi hoàn thành việc thiết lập các phương thức xử lý tại các tầng BLL và DAL, chúng ta sẽ tiếp tục tạo các lớp (class) tại tầng GUI (Graphical User Interface). Mục đích của việc này là để thực hiện các thao tác giao diện cho ứng dụng.
- Đồng thời, chúng ta sẽ sử dụng thư viện Tkinter để tạo giao diện cho ứng dụng.
- Thư viện PIL (Python Imaging Library) sẽ được sử dụng để hiển thị hình ảnh trong ứng dụng.
- Thư viện Pygame.mixer sẽ được sử dụng để phát nhạc trong ứng dụng.
- Thư viện mutagen sẽ được sử dụng để lấy thông tin về file nhạc như tên bài hát, tên nghệ sĩ, và album.
- Ngoài ra, chúng ta cũng sẽ sử dụng thư viện socket để tạo kết nối giữa client và server. Điều này giúp cho việc truyền dữ liệu giữa client và server trở nên dễ dàng hơn.

### 2.7.1 Thiết kế socket server và client và cài đặt vài chức năng cho ứng dụng nghe nhạc

- Để tạo kết nối giữa client và server, chúng ta sẽ tạo một socket server và một socket client.
- Server sẽ lắng nghe các yêu cầu từ client và thực hiện các thao tác tương ứng.
- Client sẽ gửi các yêu cầu đến server và nhận kết quả trả về từ server.
- Chúng ta sẽ tuân theo mô hình client-server để tạo kết nối giữa client và server như sau:



Hình 18: Thiết kế socket server và client cho ứng dụng nghe nhạc

- Trước tiên, chúng ta sẽ mở một cổng kết nối trên server và lắng nghe các yêu cầu từ client.

```
#####  
ip = "localhost"  
  
port = 6767  
  
def __init__(self):  
    # Khởi tạo socket của server  
    self.host_ip = Server.ip  
    self.port = Server.port  
    self.con = ConnectSQL.connect_mysql()  
    # Bắt đầu lắng nghe các kết nối đến server  
    self.runServer()  
  
def runServer(self):  
    # Tạo socket của server  
    self.server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)  
  
    # Bind socket với host và port  
    self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)  
    self.server_socket.bind((self.host_ip, self.port))  
    print("HOST IN SERVER: " + self.host_ip)  
  
    self.server_socket.listen(5)  
    print("Server listening on port", self.port)  
  
    while True:  
        self.getSignal()
```

Hình 19: Code server gồm phương thức khởi tạo và chạy server

+Giải thích:

-Phương thức **init(self)** khởi tạo server, gán ip và port, thiết lập kết nối đến cơ sở dữ liệu thông qua class ConnectDB.

-Phương thức **run(self)** chạy server, lắng nghe các yêu cầu từ client và thực hiện các thao tác tương ứng.

```
def getSignal(self):
    # Chấp nhận kết nối từ client và khởi tạo xử lý
    print("Đang chờ kết nối: ")
    client, address = self.server_socket.accept()
    signal = client.recv(1024).decode("utf-8")
    if ("LOGIN" in signal):
        signal, username, password = signal.split("|")
        self.sendDataUser(client, username, password)
    elif ("RESET_PASSWORD" in signal):
        signal, username, new_password = signal.split("|")
        self.resetPassword(client, username, new_password)
    elif ("REGISTER" in signal):
        signal, username, email, password = signal.split("|")
        self.Register(client, username, email, password)
    elif ("DATA_PLAYLIST_USERID" in signal):
        signal, userID = signal.split("|")
        self.sendDataPlaylistWithUserID(client, userID)
    elif (signal == "DATA_TRACK"):
        self.sendDataTrack(client)
    elif (signal == "DATA_ALBUM"):
        self.sendDataAlbum(client)
    elif (signal == "DATA_TRACK_ALBUM"):
        self.sendDataTrackInAlbum(client)
    elif (signal == "DATA_ARTIST"):
        self.sendDataArtist(client)
    elif (signal == "DATA_TRACK_ARTIST"):
        self.sendDataTrackOfArtist(client)
    elif (signal == "PLAY_SONG_"):
        self.sendAudio(client, address)
    elif ("ADD_PLAYLIST" in signal):
        signal, userID, title, creationdate = signal.split("|")
        self.addPlayList(userID, title, creationdate)
    elif ("DELETE_PLAYLIST" in signal):
        signal, playlistID = signal.split("|")
        self.deletePlayList(client, playlistID)
    elif ("ADD_TRACK_PLAYLIST" in signal):
        signal, playlistID, userID, trackID = signal.split("|")
        self.addTrackToPlayList(playlistID, userID, trackID)
    elif (signal == "DELETE_TRACK_PLAYLIST"):
        self.deleteTrackInPlayList(client)
    elif (signal == "GET_USERNAME_USERID"):
        self.sendDataUserNameByUserID(client)
    elif ("GET_PLAYLISTID" in signal):
        signal, userID = signal.split("|")
        self.sendDataPlaylistID(client, userID)
    elif ("GET_TRACK_PL_USERID" in signal):
        signal, playlistIDs, userID = signal.split("|")
        self.sendDataInPlaylistofTrack(client, playlistIDs, userID)
```

Hình 20: Code server-side phương thức lấy phản hồi từ client

```
def checkLogin(self, username, password):
    # Assign a default value to Notification_Server
    Notification_Server = ""
    try:
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # gửi yêu cầu connect
        self.client_socket.connect((self.host_ip, self.port))
        signal = "LOGIN"
        message = signal + "|" + username + "|" + password
        self.client_socket.sendall(message.encode())
        #self.client_socket.recv(1024).decode("utf-8")

    except socket.timeout as e:
        print("TIMEOUT ERROR:", str(e))
    except OSError as e:
        print("FAILED TO RECEIVE DATA:", str(e))
        return
    except Exception as e:
        print("ERROR:", str(e))
        return

    Notification_Server = self.client_socket.recv(1024).decode()
    # Check the length of the sequence
    if "|" in Notification_Server:
        print("Notification_Server:", Notification_Server)
        Notification, userID = Notification_Server.split("|")
        return Notification, userID
    else:
        return False, None
```

Hình 21: Code cho client-side phương thức kiểm tra thông tin đăng nhập của user ở client-side

```
# Gửi dữ liệu kiểm tra đăng nhập
def sendDataUser(self, client, username, password):
    checkLogin = str(UserBLL.checkUsernameAndPass(self, username, password))
    userID = str(UserBLL.getUserIDByUsername(self, username))
    print("CHECK LOGIN: ", checkLogin)
    print("USER ID: ", userID)
    message = checkLogin + "|" + userID
    client.sendall(message.encode())
```

Hình 22: Code cho server-side phương thức kiểm tra thông tin đăng nhập của user ở server-side

-Ví dụ:

- +Ở chức năng đăng nhập tài khoản, khi client gửi yêu cầu đến server, server sẽ kiểm tra thông tin đăng nhập của user và trả về kết quả cho client.
- +Server sẽ gọi xuống cơ sở dữ liệu để kiểm tra thông tin đăng nhập của user và trả về kết quả cho server.
- +Server sẽ trả về kết quả cho client và client sẽ hiển thị thông báo tương ứng với kết quả đó.
- +Tương tự như vậy với các hàm khác như Reset Password, Register, Send PlayList, Send Track, Send Album, Send Artist, ...



```
def resetPassword(self, username, new_password):
    try:
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # Gửi yêu cầu connect
        self.client_socket.connect((self.host_ip, self.port))
        signal = "RESET_PASSWORD"
        message = signal + "|" + username + "|" + new_password
        self.client_socket.sendall(message.encode())

    except socket.timeout as e:
        print("TIMEOUT ERROR:", str(e))
    except OSError as e:
        print("FAILED TO RECEIVE DATA:", str(e))
        return
    except Exception as e:
        print("ERROR:", str(e))
        return
    # Thông báo thành công hoặc thất bại
    Notification_Server = self.client_socket.recv(1024)

    Notification = bool(int.from_bytes(Notification_Server, byteorder='big'))

    return Notification
```

Hình 23: Code cho client-side xử lý đặt lại mật khẩu của user và gửi yêu cầu đến server-side

```
# Gửi dữ liệu kiểm tra đăng nhập
def resetPassword(self, client, username, new_password):
    print("DANG GUI DU LIEU USER!!!")
    # username = client.recv(1024).decode()
    # new_password = client.recv(1024).decode()
    user = UserBLL.checkUsername(self, username) #Lấy dữ liệu

    if user:
        UserBLL.resetPassWord(self, username, new_password)
        flag = True
        client.send(bytes([flag]))
    else:
        client.sendall("Username is Wrong!!!".encode())
        flag = False
        client.send(bytes([flag]))
```

Hình 24: Code cho server-side đặt lại mật khẩu của user và trả về kết quả cho client-side

-Giải thích:

Hàm resetPassword cho client:

- +Đầu tiên, hàm tạo một socket mới và kết nối đến server thông qua IP và cổng đã cho.
- +Sau đó, hàm tạo một chuỗi tin nhắn bằng cách nối chuỗi "RESET\_PASSWORD" với tên người dùng và mật khẩu mới, rồi gửi tin nhắn này đến Server.
- +Nếu có lỗi xảy ra trong quá trình này (như timeout hoặc lỗi hệ thống), hàm sẽ in ra thông báo lỗi và trả về.
- +Cuối cùng, hàm nhận thông báo từ server, chuyển đổi nó thành boolean và trả về. Thông báo này cho biết việc đặt lại mật khẩu có thành công hay không.

Hàm resetPassword cho server:

- +Đầu tiên, hàm in ra thông báo "DANG GUI DU LIEU USER!!!" để cho biết rằng nó đang xử lý dữ liệu người dùng.
- +Sau đó, hàm kiểm tra xem tên người dùng có tồn tại trong cơ sở dữ liệu hay không bằng cách gọi hàm UserBLL.checkUsername.
- +Nếu tên người dùng tồn tại, hàm sẽ đặt lại mật khẩu bằng cách gọi hàm UserBLL.resetPassWord, gán cờ thành True và gửi cờ này đến client.
- +Nếu tên người dùng không tồn tại, hàm sẽ gửi thông báo "Username is Wrong!!!" đến client và gán biến False.

```
def Register(self, UserName, Email, Password):
    try:
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # gửi yêu cầu connect
        self.client_socket.connect((self.host_ip, self.port))
        signal = "REGISTER"
        message = signal + "|" + UserName + "|" + Email + "|" + Password
        self.client_socket.sendall(message.encode())
        msg = self.client_socket.recv(1024).decode("utf-8")
    except socket.timeout as e:
        print("TIMEOUT ERROR:", str(e))
    except OSError as e:
        print("FAILED TO RECEIVE DATA:", str(e))
        return
    except Exception as e:
        print("ERROR:", str(e))
        return
    print("msg:", msg)
    return msg
```

Hình 25: Code cho client-side xử lý đăng ký tài khoản và gửi yêu cầu đến server-side

```
def Register(self, client, username, email, password):
    userID = UserBLL.generateUserID(self)
    new_user = UserDTO(userID=userID, username=username, email=email, password=password)

    if UserBLL.hasUsername(self, username):
        msg = "Username already exists"
        client.sendall(msg.encode())
    else:
        UserBLL.insert(self, new_user)
        msg = "Register successfully"
        client.sendall(msg.encode())
        print(msg)
```

Hình 26: Code cho server-side xử lý đăng ký tài khoản và trả về kết quả cho client-side

-Giải thích:

-Hàm Register cho client:

- +Đầu tiên, hàm tạo một socket mới và kết nối đến server thông qua IP và cổng đã cho.
- +Sau đó, hàm tạo một chuỗi tin nhắn bằng cách nối chuỗi "REGISTER" với tên người dùng, email và mật khẩu, rồi gửi tin nhắn này đến server.
- +Nếu có lỗi xảy ra trong quá trình này (như timeout hoặc lỗi hệ thống), hàm sẽ in ra thông báo lỗi và trả về.
- +Cuối cùng, hàm nhận thông báo từ server và trả về. Thông báo này cho biết việc đăng ký có thành công hay không.

-Hàm Register cho server:

- +Đầu tiên, hàm tạo một ID người dùng mới bằng cách gọi hàm UserBLL.generateUserID.
- +Sau đó, hàm tạo một đối tượng người dùng mới với ID người dùng, tên người dùng, email và mật khẩu.
- +Hàm kiểm tra xem tên người dùng đã tồn tại trong cơ sở dữ liệu hay chưa bằng cách gọi hàm UserBLL.hasUsername.
- +Nếu tên người dùng đã tồn tại, hàm sẽ gửi thông báo "Username already exists" đến client.
- +Nếu người dùng chưa tồn tại, hàm sẽ thêm người dùng mới vào cơ sở dữ liệu qua gọi hàm UserBLL.insert, gửi thông báo "Register successfully" đến client và in thông báo này ra màn hình.

```
def getDataPlaylistFromServer(self, userID):
    try:
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        # gửi yêu cầu connect
        self.client_socket.connect((self.host_ip, self.port))
        signal = "DATA_PLAYLIST_USERID"
        message = signal + "|" + userID
        self.client_socket.sendall(message.encode())
        # timeout, 8 hours ago + finish show playlist detail and searching funct...
    except socket.timeout as e:
        print("A timeout error occurred while trying to connect to the server:", str(e))
    except OSError as e:
        print("An error occurred while trying to receive data:", str(e))
        return
    except Exception as e:
        print("An error occurred while trying to connect to the server:", str(e))
        return

    # Nhận dữ liệu từ server
    print("NHẬN DỮ LIỆU TỪ SERVER!!!")
    received_data = self.client_socket.recv(4096)
    # decode dữ liệu
    json_data_track = received_data.decode()
    print(f"Received data: {json_data_track}")

    # chuyển đổi dữ liệu từ dạng JSON thành danh sách từ điển
    data_track = json.loads(json_data_track)

    # Print the received data
    print("Received data track:")
    for record in data_track:
        print(record)

    return data_track
```

Hình 27: Code cho client-side lấy danh sách playlist đến client-side qua user-id

```
# Gửi dữ liệu album
def sendDataPlaylistWithUserID(self, client, userID):
    Mackdeddy, 2 weeks ago + them ham gọi dữ liệu của album, user, artist,
    data_playlist = PlaylistBLL.getDataPlaylistFromUserID(self, userID)
    print("DATA PLAYLIST: ", data_playlist)
    def tuple_to_dict(tpl):
        return {
            'playlistID': tpl[0],
            'title': tpl[1],
            'creationdate': tpl[2].strftime("%Y-%m-%d")
        }

    #Convert to JSON string using map and dumps
    json_string = json.dumps(list(map(tuple_to_dict, data_playlist)))

    client.send(json_string.encode())
```

Hình 28: Code cho server-side gửi dữ liệu playlist đến client-side qua user-id

-Hàm getDataPlaylistFromServer cho client:

- +Đầu tiên, hàm tạo một socket mới và kết nối đến server thông qua IP và cổng đã cho.
- +Sau đó, hàm tạo một chuỗi tin nhắn bằng cách nối chuỗi "DATA\_PLAYLIST\_USERID" với userID, rồi gửi tin nhắn này đến server.
- +Nếu có lỗi xảy ra trong quá trình này (như timeout hoặc lỗi hệ thống), hàm sẽ in ra thông báo lỗi và trả về.
- +Cuối cùng, hàm nhận dữ liệu từ server, giải mã dữ liệu này từ dạng JSON thành danh sách từ điển, và trả về danh sách này.

-Hàm sendDataPlaylistWithUserID cho server:

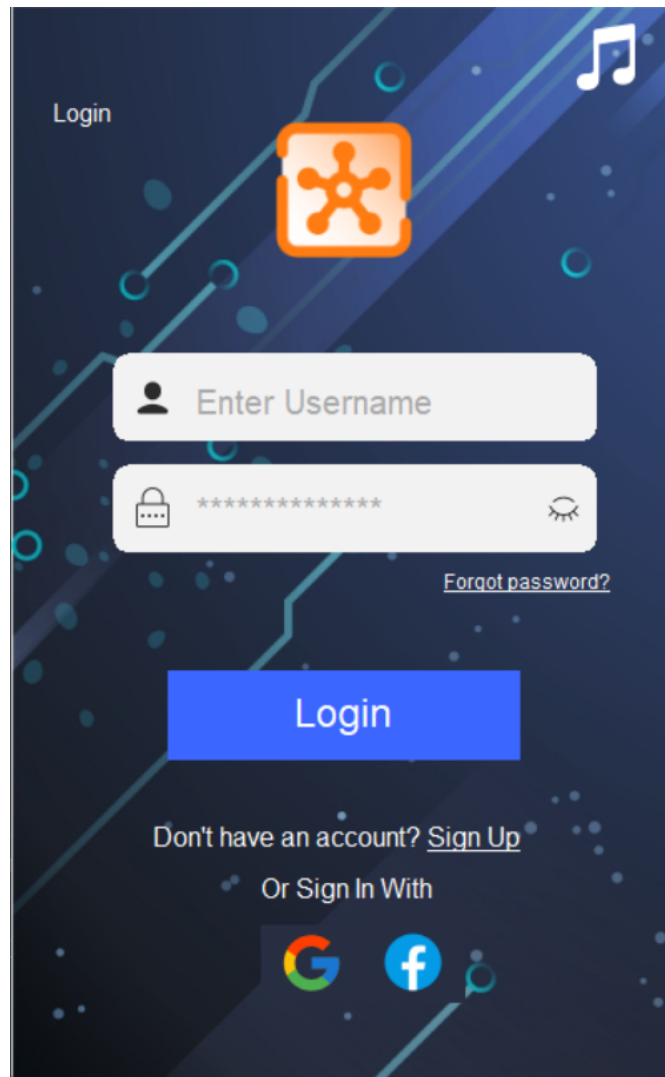
- +Đầu tiên, hàm lấy dữ liệu playlist từ cơ sở dữ liệu bằng cách gọi hàm PlaylistBLL.getDataPlaylistFromUserID.
- +Sau đó, hàm chuyển đổi dữ liệu này từ dạng tuple thành từ điển, rồi chuyển đổi từ điển này thành chuỗi JSON.
- +Cuối cùng, hàm gửi chuỗi JSON này đến client.

## 2.7.2 Xây dựng giao diện và chức năng cho ứng dụng

### 2.7.2.a Đăng nhập

-Sau khi chúng ta đã hoàn thành thiết lập các chức năng cần thiết cho Server và Client để tạo kết nối giữa chúng, chúng ta sẽ tiến hành tạo giao diện cho ứng dụng và kết hợp các chức năng đã thiết lập ở trên vào giao diện.

-Đầu tiên, chúng ta có giao diện đăng nhập cho ứng dụng:

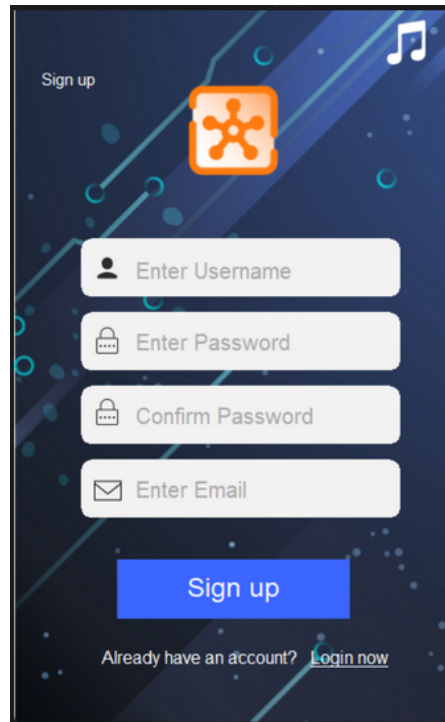


Hình 29: Giao diện đăng nhập cho ứng dụng

-Ở đây chúng ta có 2 ô nhập liệu là username và password, và 3 nút chức năng là Login, Register và Reset Password.

### 2.7.2.b Đăng kí

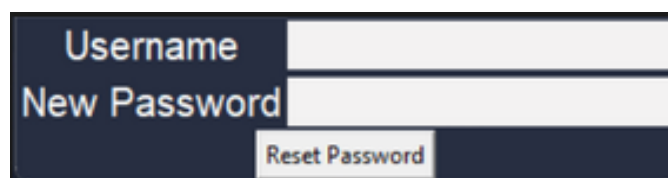
- Khi chúng ta bấm Sign Up, chúng ta sẽ chuyển sang giao diện đăng ký tài khoản như phía dưới:

The image shows a mobile app sign-up screen with a dark blue background and a network diagram. At the top left is the text "Sign up" and at the top right is a musical note icon. In the center is an orange square icon with a white network diagram. Below this are four input fields: "Enter Username" (with a person icon), "Enter Password" (with a lock icon), "Confirm Password" (with a lock icon), and "Enter Email" (with an envelope icon). Below the input fields is a blue "Sign up" button. At the bottom, it says "Already have an account? [Login now](#)".

Hình 30: Giao diện đăng ký tài khoản cho ứng dụng

### 2.7.2.c Đặt lại mật khẩu

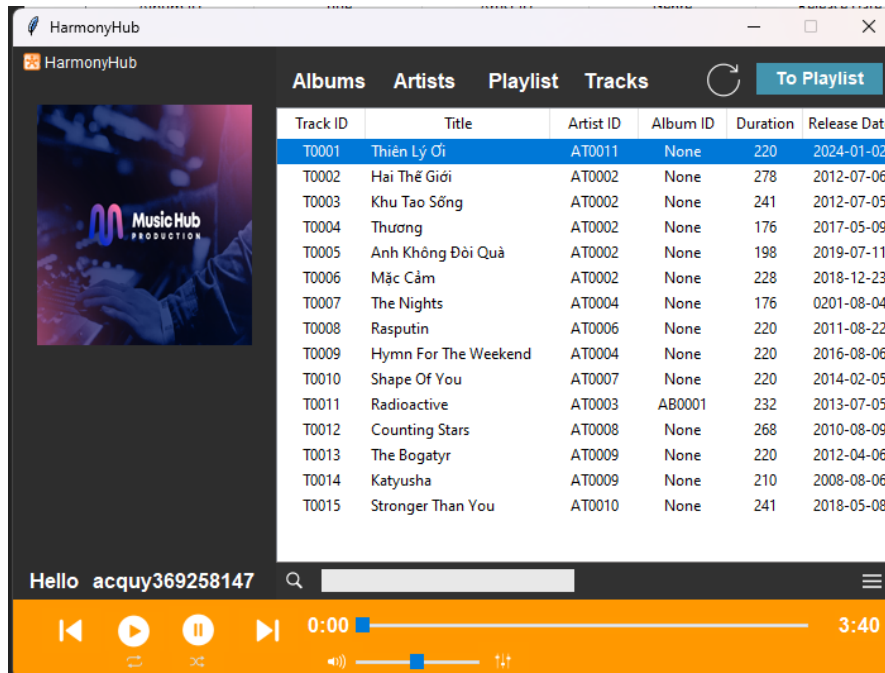
- Khi chúng ta bấm Reset Password, chúng ta sẽ chuyển sang giao diện đặt lại mật khẩu như phía dưới:

The image shows a mobile app reset password screen with a dark blue background. It has two input fields: "Username" and "New Password". Below the "New Password" field is a "Reset Password" button.

Hình 31: Giao diện đặt lại mật khẩu cho ứng dụng

### 2.7.3 Trang chủ

- Sau khi đăng nhập thành công, chúng ta sẽ chuyển sang giao diện chính của ứng dụng như phía dưới:

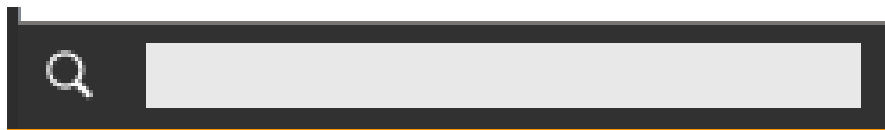


Hình 32: Giao diện chính của ứng dụng sau khi đăng nhập thành công

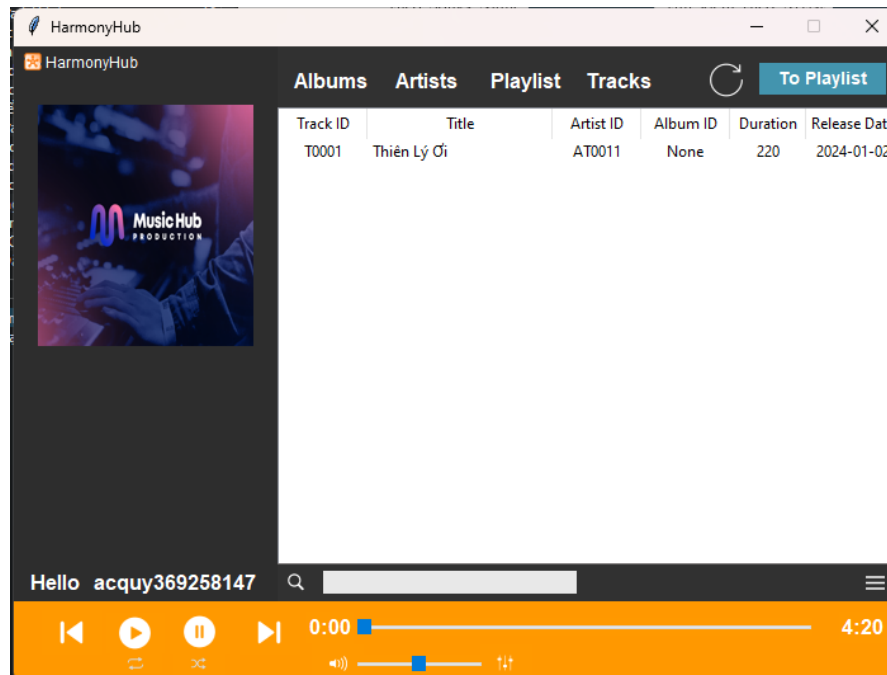
-Ở đây chúng ta có 4 tab chính là Playlist, Albums, Artists và Tracks và nhiều nút chức năng khác như Play, Pause, Next, Previous, Volume, thêm bài hát vào Playlist ...

### 2.7.4 Tìm kiếm

-Tìm kiếm ở đây là tìm kiếm theo tên bài hát, chúng ta chỉ cần nhập tên bài hát vào ô tìm kiếm và bấm nút Search để tìm kiếm.









Hình 33: Giao diện tìm kiếm bài hát theo tên



Hình 34: Kết quả tìm kiếm bài hát theo tên


### 2.7.5 Nghe nhạc

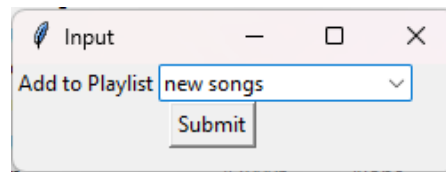
- Để nghe nhạc, chúng ta chỉ cần chọn bài hát từ danh sách bài hát và bấm nút Play.
- Giao diện bài hát đang phát gồm các chức năng như Play , Pause , Next , Previous , Volume , thêm bài hát vào Playlist ...
- Giao diện chính để thao tác với bài hát đang phát:
- Để dừng nghe nhạc, chúng ta chỉ cần bấm nút tạm ngừng .



Hình 35: Giao diện chính mỗi bài nhạc

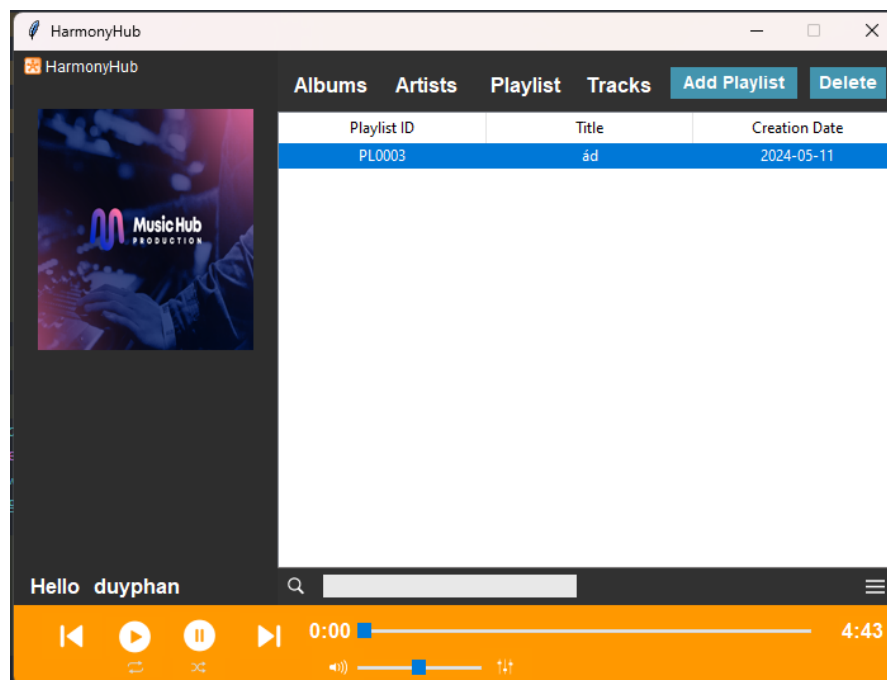
### 2.7.6 Playlist

- Ngoài các thao tác với mỗi bài nhạc, chúng ta có thể thêm bài hát vào Playlist bằng cách chọn bài hát và bấm vào nút playlist .
- Sau khi bấm vào, nó sẽ hiện cửa sổ chọn Playlist để thêm bài hát vào Playlist đó.



Hình 36: Giao diện thêm bài hát vào Playlist

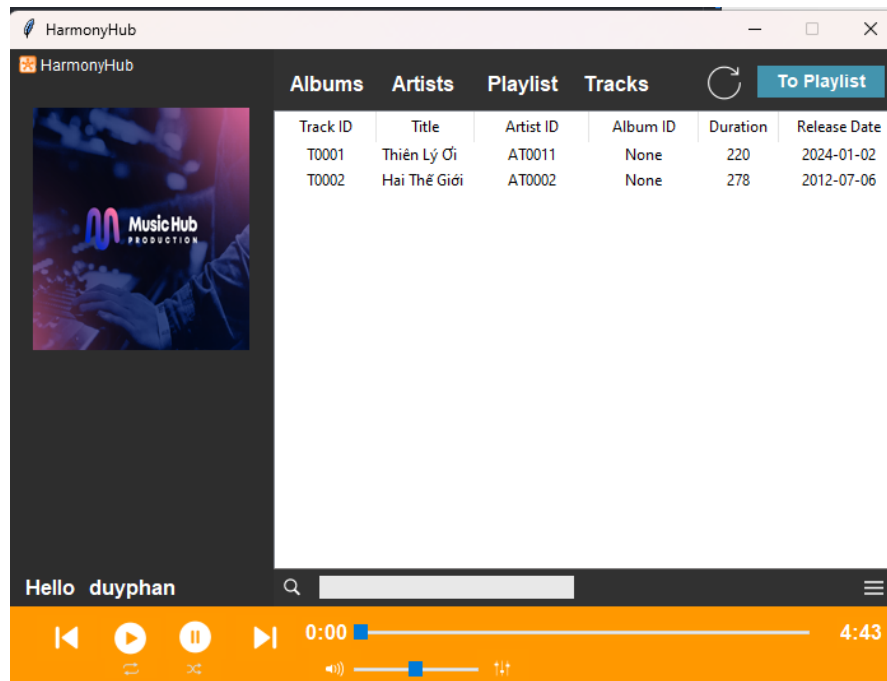
-Sau khi chọn Playlist, bài hát được chọn sẽ được thêm vào Playlist đó.



Hình 37: Giao diện Playlist

-Ta nhấn đúp vào playlist để xem danh sách bài hát trong playlist đó.



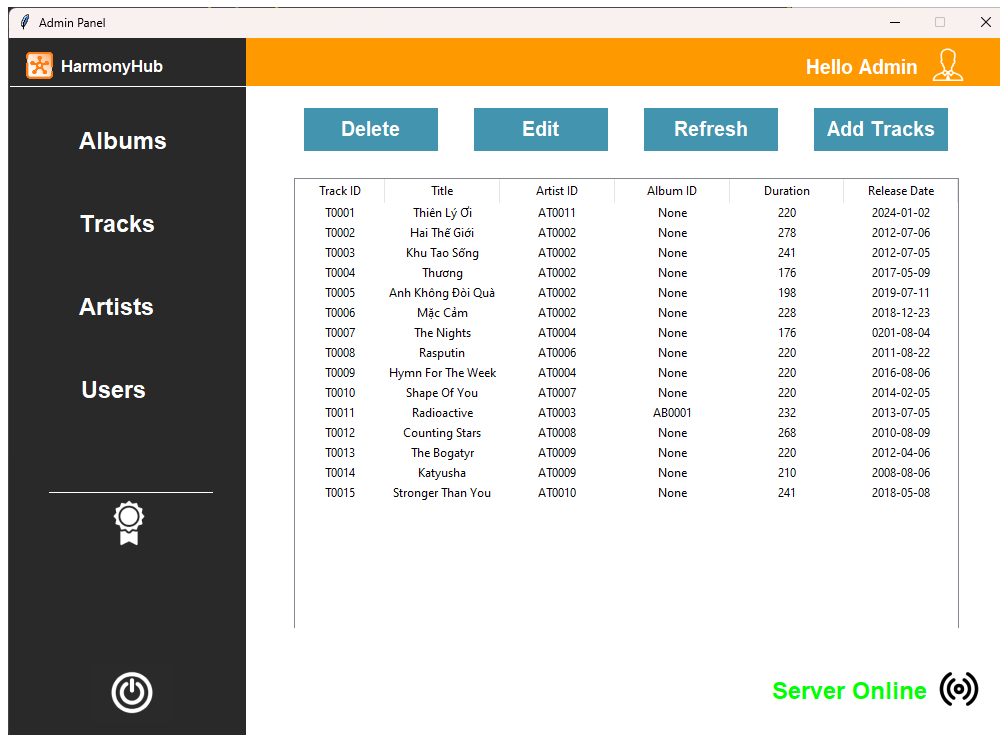


Hình 38: Danh sách bài hát trong Playlist

## 2.7.7 Giao diện và chức năng cho Server

### 2.7.7.a Quản lý nhạc

-Hiển thị danh sách bài hát:



Hình 39: Danh sách bài hát

-Danh sách bài hát trong kho lưu trữ gồm id, tên bài hát, mã nghệ sĩ, mã album, thời lượng, và ngày phát hành.

#### 2.7.7.b Thêm bài hát mới

-Sau khi điền đầy đủ các thông tin, chúng ta bấm nút Add để thêm bài hát mới vào kho lưu trữ.

Input

Title

Artist ID

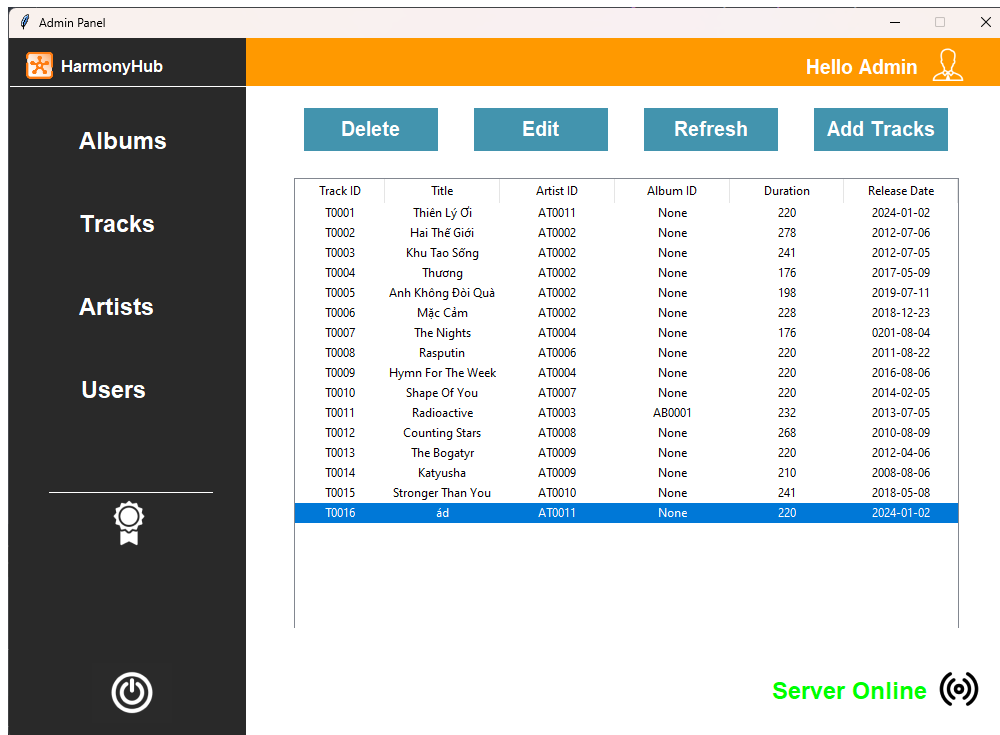
Album ID: None

Duration

Release Date

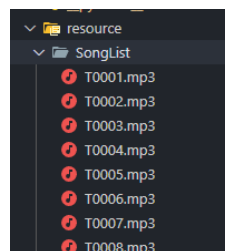
Submit

Hình 40: Thêm bài hát mới



Hình 41: Thêm bài hát mới thành công

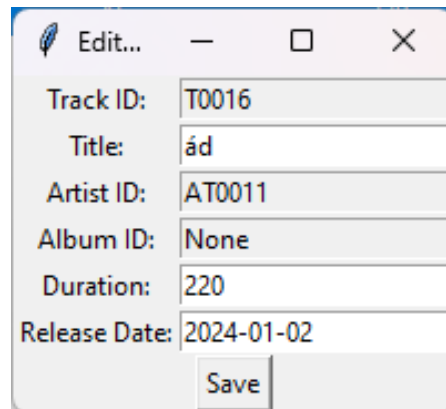
- Sau khi thêm bài hát mới thành công, nó sẽ hiện ra trong danh sách bài hát và chúng ta kéo thả file nhạc vào thư mục SongList của Resource để thêm bài hát mới vào kho lưu trữ.
- Đổi tên thư mục ứng với ID của bài hát mới.



Hình 42: Thêm bài hát mới vào thư mục SongList

### 2.7.7.c Chính sửa bài hát

-Để chỉnh sửa bài hát, chúng ta chọn bài hát cần chỉnh sửa và bấm nút **Edit**. Sau khi chỉnh các thông tin cần thiết, chúng ta bấm nút **Save** để lưu lại.



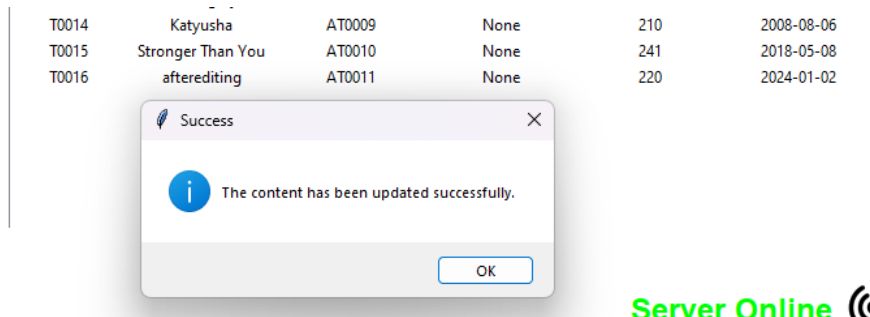
Hình 43: Giao diện chỉnh sửa bài hát

Track ID	Title	Artist ID	Album ID	Duration	Release Date
T0001	Thiên Lý Ở	AT0011	None	220	2024-01-02
T0002	Hai Thế Giới	AT0002	None	278	2012-07-06
T0003	Khu Táo Sợng	AT0002	None	241	2012-07-05
T0004	Thương	AT0002	None	176	2017-05-09
T0005	Anh Không Đòi Quà	AT0002	None	198	2019-07-11
T0006	Mặc Cảm	AT0002	None	228	2018-12-23
T0007	The Nights	AT0004	None		
T0008	Rasputin	AT0006	None		
T0009	Hymn For The Week	AT0004	None		
T0010	Shape Of You	AT0007	None		
T0011	Radioactive	AT0003	AB0001		
T0012	Counting Stars	AT0008	None		
T0013	The Bogatyr	AT0009	None		
T0014	Katyusha	AT0009	None		
T0015	Stronger Than You	AT0010	None		
T0016	asdasd	AT0011	None	220	2024-01-02




Hình 44: Thông tin bài hát trước khi chỉnh sửa

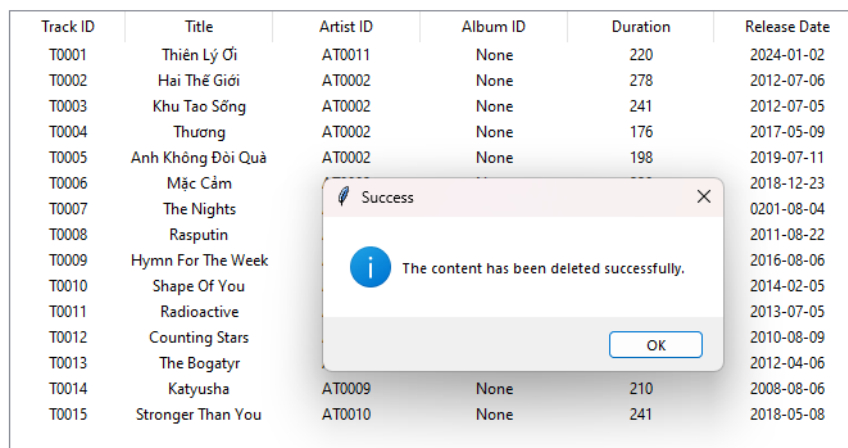
-Sau khi chỉnh sửa bài hát, nó sẽ hiện ra thông báo chỉnh sửa thành công.



Hình 45: Thông tin bài hát sau khi chỉnh sửa

#### 2.7.7.d Xóa bài hát

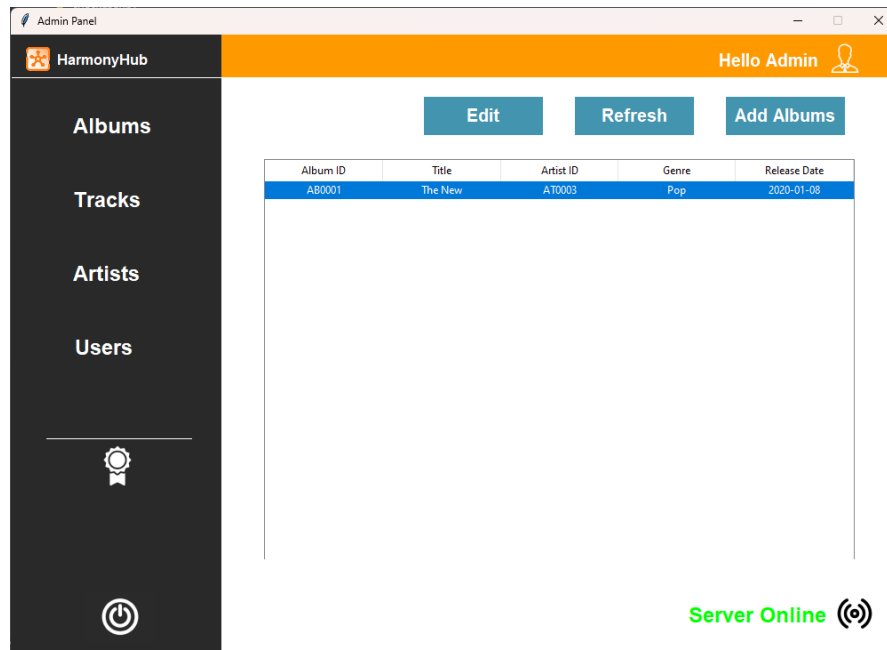
-Để xóa bài hát, chúng ta chọn bài hát cần xóa và bấm nút Delete .



Hình 46: Bài hát được chọn sau khi xóa (mã T0016)

#### 2.7.8 Quản lý danh sách Album

-Hiện thị danh sách các Album trong cơ sở dữ liệu.

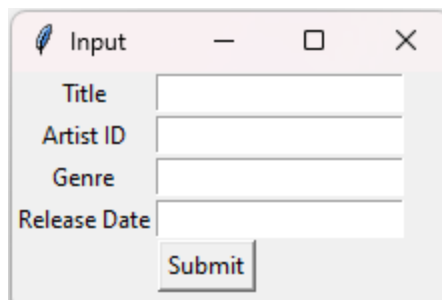


Hình 47: Danh sách Album

-Danh sách được hiển thị gồm mã album, tên album, mã nghệ sĩ, thể loại và ngày phát hành.

#### 2.7.8.a Thêm Album mới

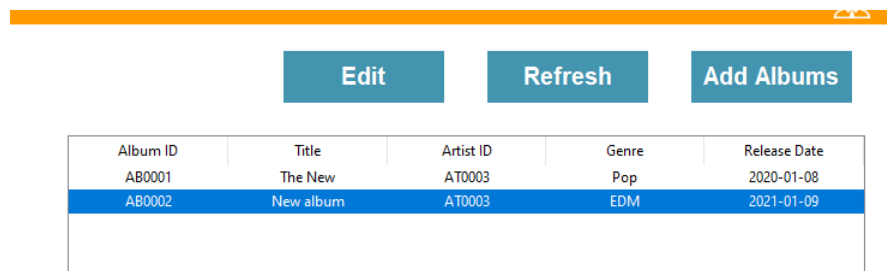
-Chúng ta bấm vào nút Add Albums [Add Albums](#) sau đó hệ thống sẽ hiện ra giao diện thêm Album mới.



The screenshot shows the "Input" form for adding a new album. It has four input fields: Title, Artist ID, Genre, and Release Date. Below the fields is a "Submit" button.

Hình 48: Thêm Album mới

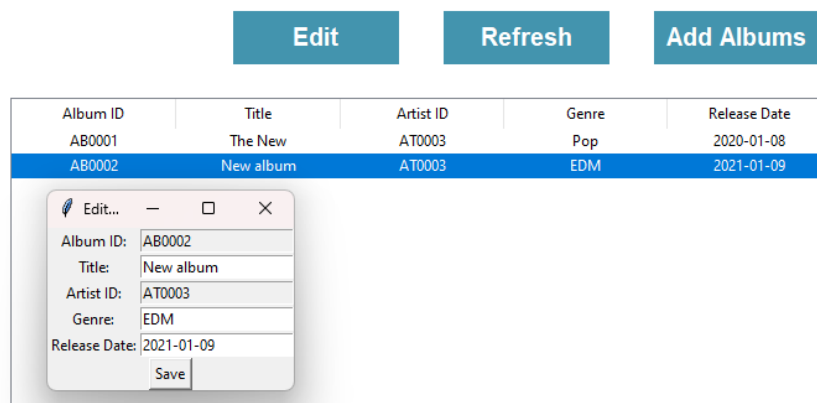
-Chúng ta điền toàn bộ thông tin cần thiết sau đó bấm Submit để thêm Album mới vào cơ sở dữ liệu.



Hình 49: Giao diện sau khi thêm Album mới thành công

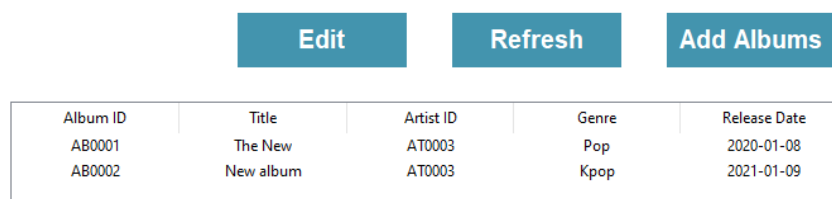
### 2.7.8.b Chỉnh sửa Album

-Để chỉnh sửa Album, chúng ta chọn Album cần chỉnh sửa và bấm nút Edit .



Hình 50: Giao diện chỉnh sửa Album

-Sau khi chỉnh sửa xong các thông tin cần thay đổi, chúng ta bấm nút Save để lưu lại.

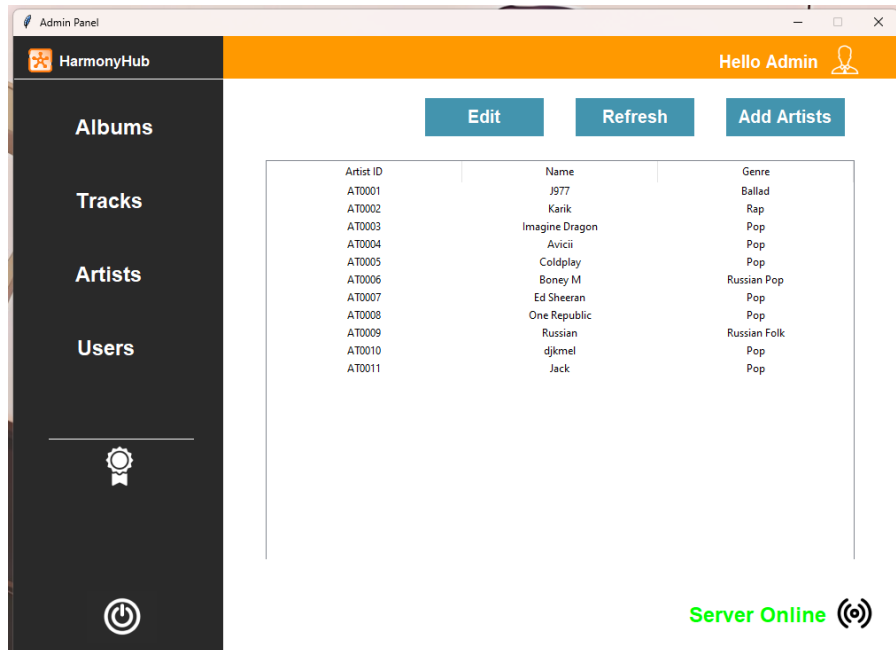


Hình 51: Giao diện sau khi chỉnh sửa Album

## 2.7.9 Quản lý thông tin nghệ sĩ

### 2.7.9.a Hiển thị danh sách nghệ sĩ

-Chúng ta bấm vào mục Artist để hiển thị danh sách nghệ sĩ.



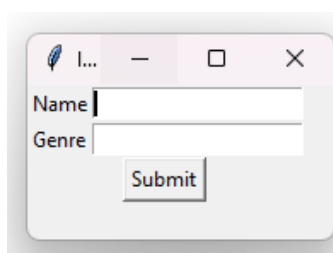
Hình 52: Danh sách các nghệ sĩ

-Danh sách nghệ sĩ gồm mã nghệ sĩ, tên nghệ sĩ, và thể loại nhạc mà nghệ sĩ đó thể hiện.

### 2.7.9.b Thêm nghệ sĩ mới

-Chúng ta bấm vào nút Add Artists để thêm nghệ sĩ mới.

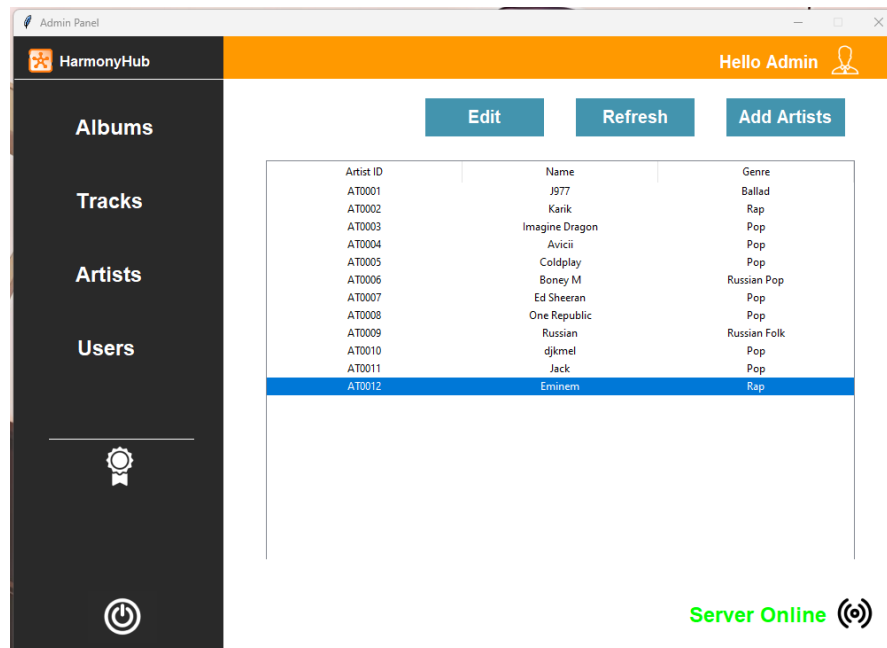
-Sau đó, hệ thống sẽ hiển thị giao diện thêm nghệ sĩ mới.



Hình 53: Thêm nghệ sĩ mới

-Ta điền các thông tin cần thiết sau đó bấm Submit để thêm nghệ sĩ mới vào cơ sở dữ liệu.

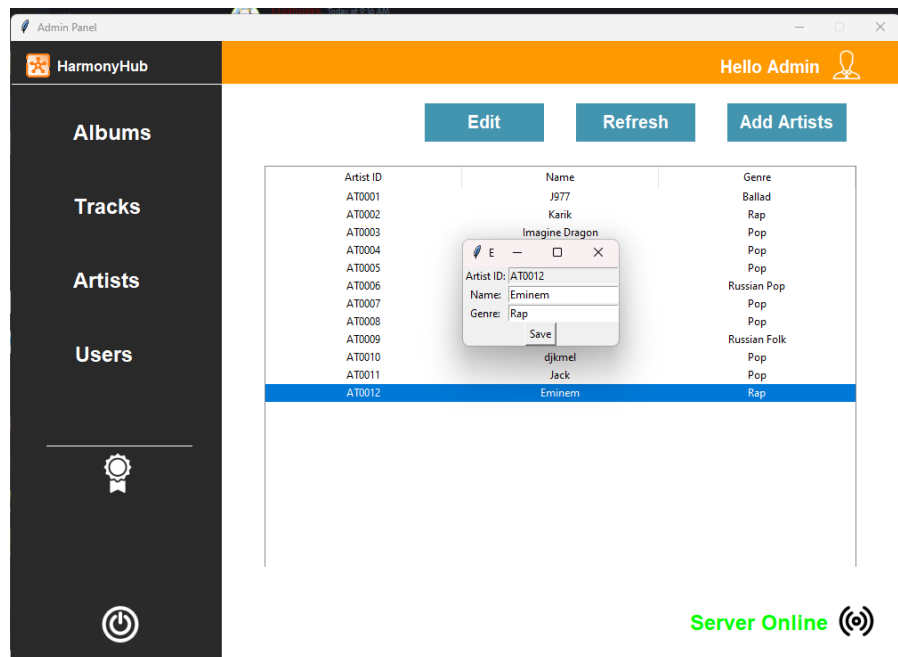




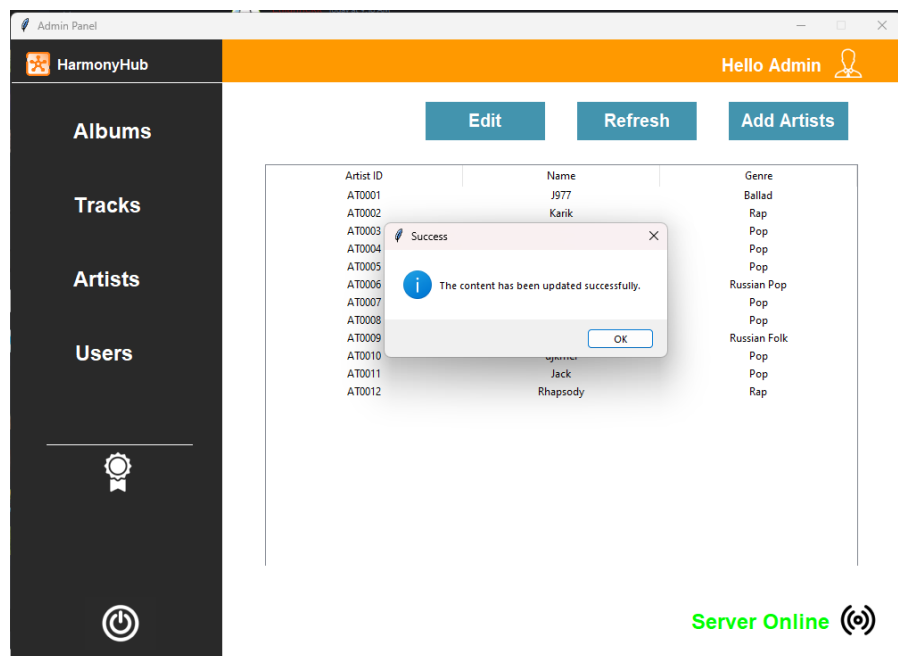
Hình 54: Giao diện sau khi thêm nghệ sĩ mới thành công

### 2.7.9.c Chỉnh sửa thông tin nghệ sĩ

- Để chỉnh sửa thông tin nghệ sĩ, chúng ta chọn nghệ sĩ cần chỉnh sửa và bấm nút Edit.
- Sau khi chỉnh sửa xong, ta thấy tên nghệ sĩ đã được thay đổi.



Hình 55: Giao diện chỉnh sửa thông tin nghệ sĩ trước khi chỉnh sửa

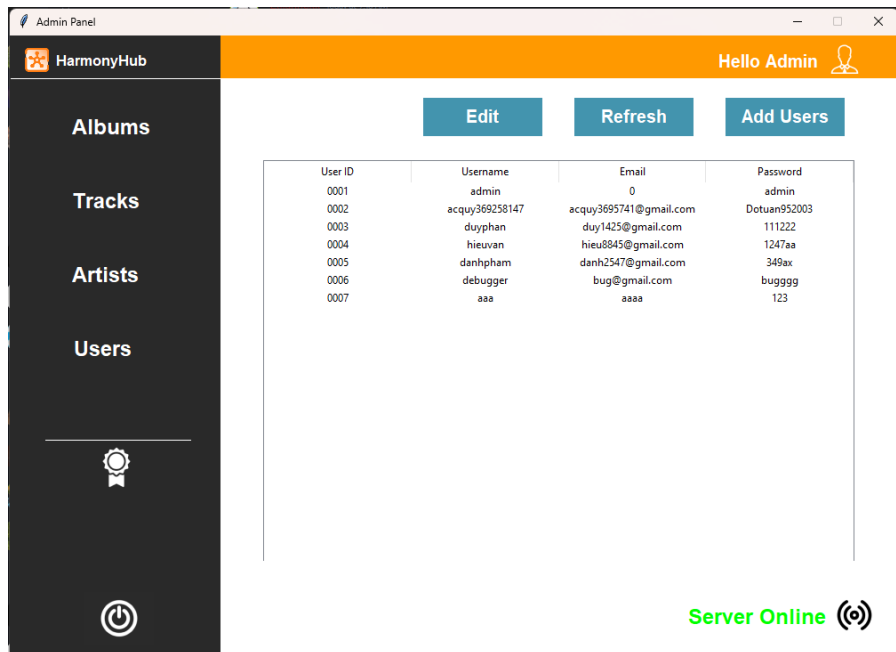


Hình 56: Giao diện chỉnh sửa thông tin nghệ sĩ sau khi chỉnh sửa

## 2.7.10 Quản lý người dùng

### 2.7.10.a Hiển thị danh sách người dùng

-Chúng ta bấm vào mục User để hiển thị danh sách người dùng.



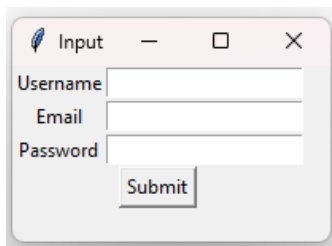
Hình 57: Danh sách người dùng

-Danh sách người dùng gồm mã người dùng, tên người dùng, email, và mật khẩu.

### 2.7.10.b Thêm người dùng mới

-Chúng ta bấm vào nút Add Users để thêm người dùng mới.

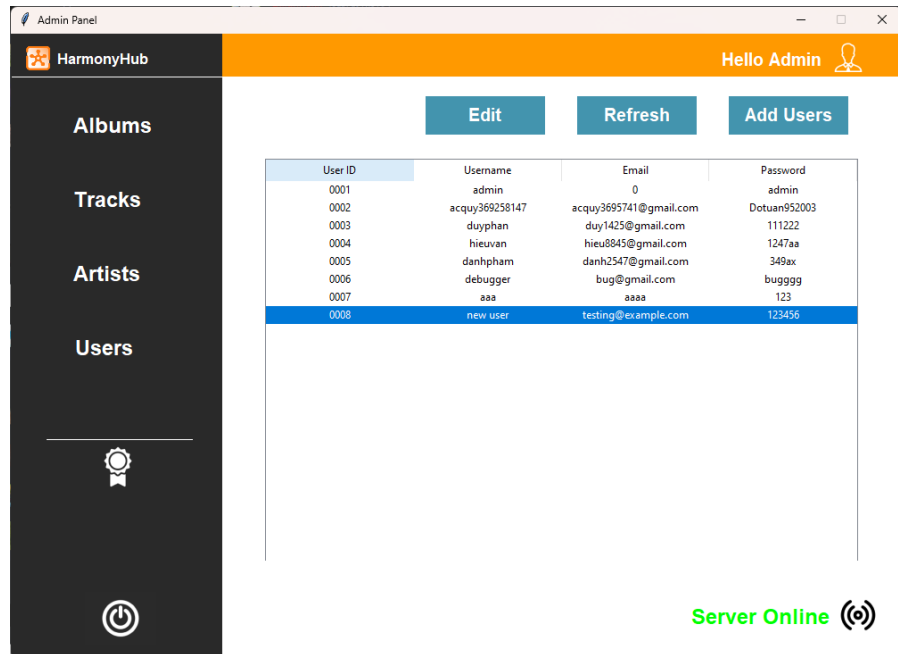
-Sau đó, hệ thống sẽ hiển thị giao diện thêm người dùng mới.



Hình 58: Thêm người dùng mới



-Ta điền các thông tin cần thiết sau đó bấm Submit để thêm người dùng mới vào cơ sở dữ liệu.



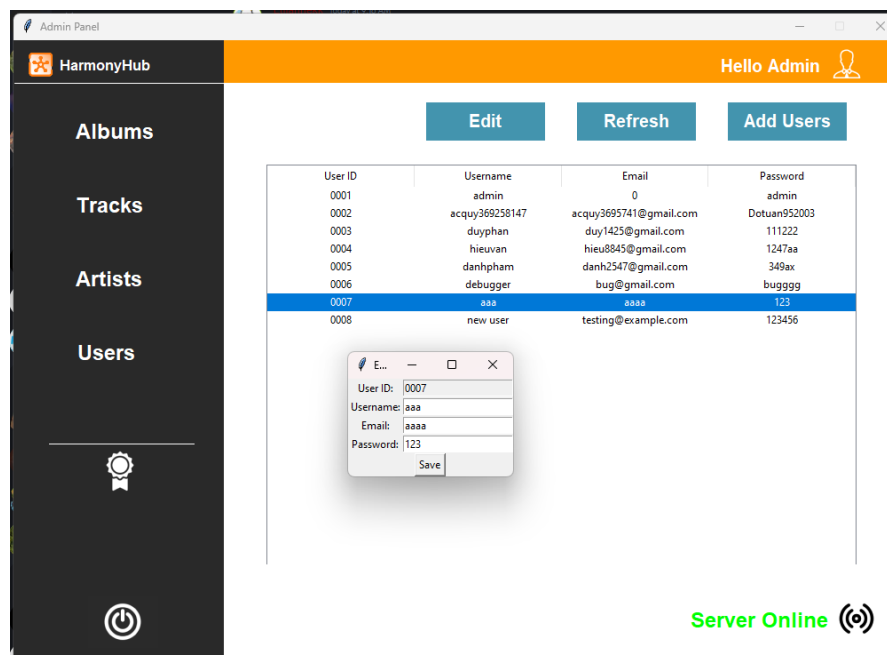
Hình 59: Giao diện sau khi thêm người dùng mới thành công

-Sau khi thêm người dùng mới thành công, hệ thống sẽ hiển thị thông báo thêm người dùng mới thành công và hiển thị người dùng mới trong danh sách người dùng.

#### 2.7.10.c Chỉnh sửa thông tin người dùng

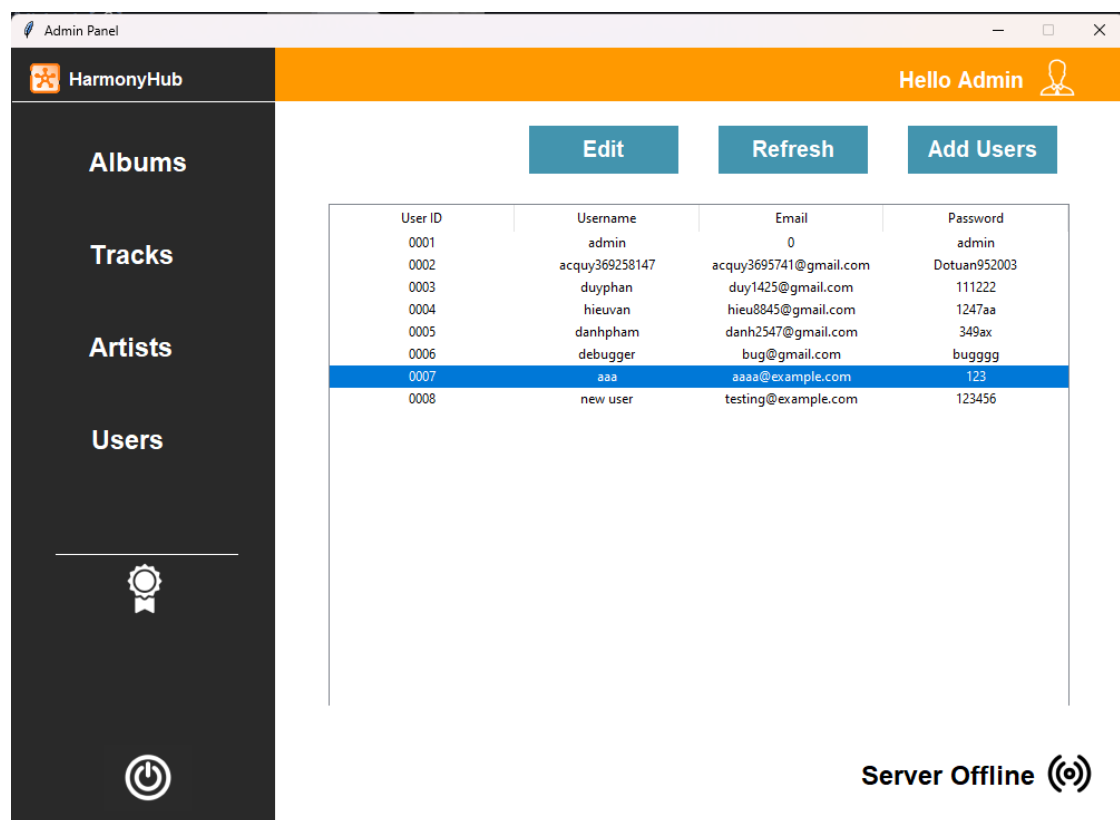
-Để chỉnh sửa thông tin người dùng, chúng ta chọn người dùng cần chỉnh sửa và bấm nút Edit.

-Lúc này hệ thống sẽ hiển thị giao diện chỉnh sửa thông tin người dùng. Giả sử ở đây ta chọn UserID = 0007



Hình 60: Giao diện chỉnh sửa thông tin người dùng trước khi chỉnh sửa


-Sau khi chỉnh sửa xong, ta thấy thông tin người dùng đã được thay đổi ở email




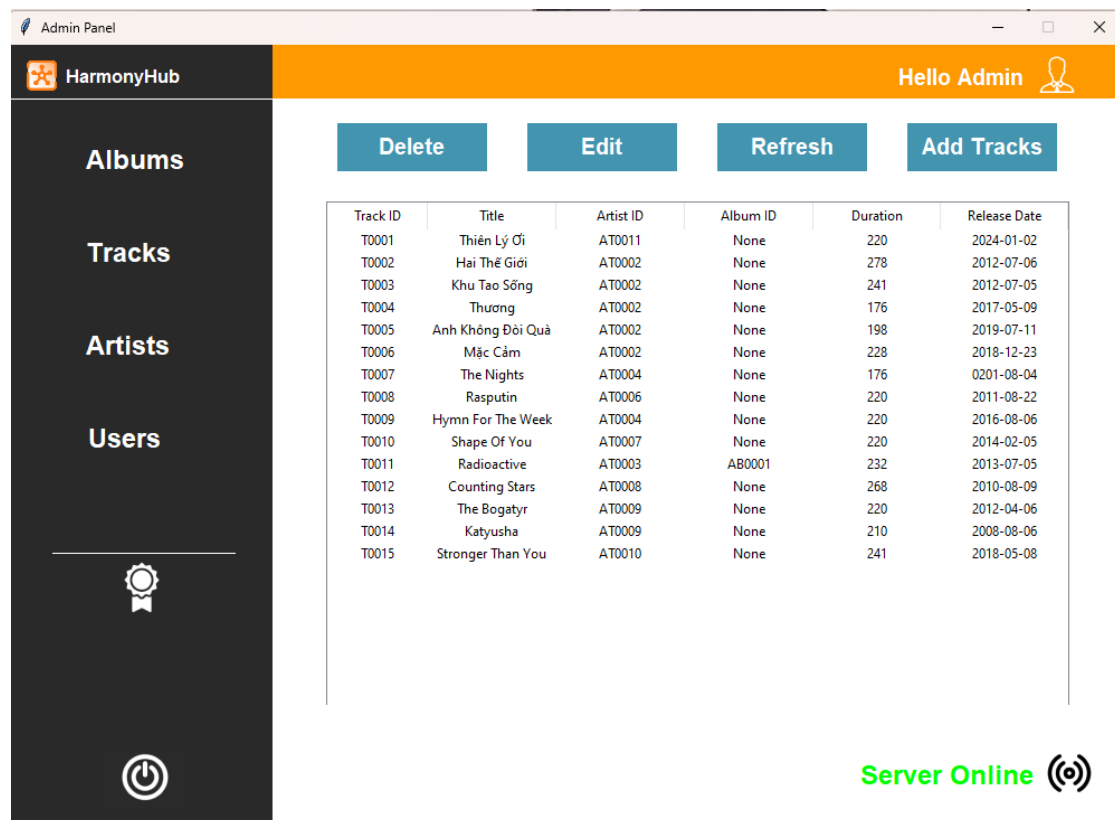
Hình 61: Giao diện chỉnh sửa thông tin người dùng sau khi chỉnh sửa



### 2.7.11 Khởi động server

-Để khởi động server, chúng ta chỉ cần bấm vào nút Server Offline .

-Ngay lập tức, hệ thống sẽ chuyển nút Server Offline thành Server Online  tức server đã được khởi động.





The screenshot shows the HarmonyHub Admin Panel. The left sidebar contains navigation links for Albums, Tracks, Artists, and Users. The main content area displays a table of tracks with columns for Track ID, Title, Artist ID, Album ID, Duration, and Release Date. The table contains 15 rows of data. Above the table are buttons for Delete, Edit, Refresh, and Add Tracks. The top right of the panel shows 'Hello Admin' and a user profile icon. The bottom right of the panel shows the status 'Server Online' with a green power icon.

Track ID	Title	Artist ID	Album ID	Duration	Release Date
T0001	Thiên Lý Gi	AT0011	None	220	2024-01-02
T0002	Hai Thế Giới	AT0002	None	278	2012-07-06
T0003	Khu Táo Sổng	AT0002	None	241	2012-07-05
T0004	Thương	AT0002	None	176	2017-05-09
T0005	Anh Không Đòi Quà	AT0002	None	198	2019-07-11
T0006	Mặc Cảm	AT0002	None	228	2018-12-23
T0007	The Nights	AT0004	None	176	0201-08-04
T0008	Rasputin	AT0006	None	220	2011-08-22
T0009	Hymn For The Week	AT0004	None	220	2016-08-06
T0010	Shape Of You	AT0007	None	220	2014-02-05
T0011	Radioactive	AT0003	AB0001	232	2013-07-05
T0012	Counting Stars	AT0008	None	268	2010-08-09
T0013	The Bogatyr	AT0009	None	220	2012-04-06
T0014	Katyusha	AT0009	None	210	2008-08-06
T0015	Stronger Than You	AT0010	None	241	2018-05-08

Hình 62: Giao diện sau khi khởi động server

### 2.7.12 Tắt server

-Để tắt server, chúng ta chỉ cần bấm vào nút Server Online  và hệ thống sẽ chuyển nút Server Online thành Server Offline  tức server đã được tắt.



Admin Panel

HarmonyHub

Hello Admin

Albums

Tracks

Artists

Users

Delete

Edit

Refresh

Add Tracks

Track ID	Title	Artist ID	Album ID	Duration	Release Date
T0001	Thiên Lý Ở	AT0011	None	220	2024-01-02
T0002	Hai Thế Giới	AT0002	None	278	2012-07-06
T0003	Khu Tao Sống	AT0002	None	241	2012-07-05
T0004	Thương	AT0002	None	176	2017-05-09
T0005	Anh Không Đòi Quà	AT0002	None	198	2019-07-11
T0006	Mặc Cảm	AT0002	None	228	2018-12-23
T0007	The Nights	AT0004	None	176	0201-08-04
T0008	Rasputin	AT0006	None	220	2011-08-22
T0009	Hymn For The Week	AT0004	None	220	2016-08-06
T0010	Shape Of You	AT0007	None	220	2014-02-05
T0011	Radioactive	AT0003	AB0001	232	2013-07-05
T0012	Counting Stars	AT0008	None	268	2010-08-09
T0013	The Bogatyr	AT0009	None	220	2012-04-06
T0014	Katyusha	AT0009	None	210	2008-08-06
T0015	Stronger Than You	AT0010	None	241	2018-05-08

Server Offline

Hình 63: Giao diện sau khi tắt server



## 2.8 Kết luận

### 2.8.1 Ưu điểm

- **Dễ sử dụng:** Ứng dụng nghe nhạc HarmonyHub cung cấp 2 ứng dụng con là App Server và App Client có giao diện thân thiện, dễ tương tác.
- **Đa dạng chức năng:**
  - App Server:**
    - Quản lý nhạc: giúp admin có thể quản lý nhạc trong hệ thống.
    - Quản lý user: Xem được thông tin người dùng bên phía Client đăng ký vào hệ thống.
    - Quản lý server: Khởi động/Tắt server linh hoạt đồng thời theo dõi hoạt động của Client.
    - Quản lý Album: Quản lý các Album trong hệ thống.
    - Quản lý nghệ sĩ: Quản lý thông tin nghệ sĩ trong hệ thống.
  - App Client:**
    - Nghe nhạc: các người dùng có thể nghe được các bài nhạc lưu trữ ở server mà không cần phải tải về.
    - Tìm kiếm nhạc: Giúp người dùng tìm kiếm các bài hát theo yêu cầu.
    - Thêm vào playlist: Người dùng có thể lưu trữ các bài hát yêu thích bằng cách thêm các bài hát vào danh sách playlist.
    - Đăng ký tài khoản: Để có thể lưu các danh sách yêu thích người dùng cần đăng nhập vào ứng dụng với tài khoản đã đăng ký.
- **Phản hồi nhanh:** Việc gửi/nhận dữ liệu của Client và Server nhanh chóng giúp tăng trải nghiệm của người dùng, dữ liệu nhạc được chia nhỏ và gửi liên tục để tránh thời gian chờ khi dữ liệu quá lớn.

### 2.8.2 Nhược điểm

- **Bảo mật:** Ứng dụng chưa có chức năng bảo mật dữ liệu, dữ liệu người dùng chưa được mã hóa khi gửi từ Client đến Server.
- **Chưa hoàn thiện:** Ứng dụng còn nhiều chức năng chưa hoàn thiện, chưa có chức năng xóa bài hát, xóa Album, xóa nghệ sĩ, xóa người dùng.
- **Giao diện:** Giao diện của ứng dụng còn đơn giản, chưa được tối ưu hóa cho người dùng.

### 2.8.3 Hướng phát triển

- **Bảo mật:** Cần phát triển chức năng bảo mật dữ liệu, mã hóa dữ liệu người dùng khi gửi từ Client đến Server. Admin có thể quản lý người dùng, phân quyền người dùng.
- **Hoàn thiện:** Cần hoàn thiện các chức năng còn thiếu như xóa bài hát, xóa Album, xóa nghệ sĩ, xóa người dùng.
- **Giao diện:** Cần tối ưu hóa giao diện cho người dùng, thêm các chức năng giúp người dùng dễ sử dụng hơn.