

TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THÀNH PHỐ HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN CHUYÊN NGÀNH

Đề tài: “Lập trình ứng dụng xem hợp âm bài hát trên nền tảng android”

Giảng viên bộ môn: TS. Nguyễn Chí Kiên,

ThS. Trương Vĩnh Linh,

ThS. Lê Phúc Lữ

Sinh viên thực hiện: Trịnh Thị Bảo Bảo – 19514491 ,

Nguyễn Thị Thanh Hòa – 19429041 ,

Phan Chí Trung - 19499041,

Đoàn Minh Trường - 19519011

Lớp: DHKHDL15A

Tp. Hồ Chí Minh, ngày 21 tháng 12 năm 2021

LỜI CẢM ƠN

Lời đầu tiên, nhóm chúng em xin gửi lời cảm ơn chân thành đến quý thầy – giảng viên bộ môn “Phát triển ứng dụng”. Quý thầy là người đã trực tiếp giảng dạy, chỉ bảo và truyền đạt cho chúng em những kiến thức bổ ích đồng thời đưa ra những lời nhận xét, góp ý để chúng em có thể hoàn thành đề tài “**Lập trình ứng dụng xem hợp âm bài hát trên nền tảng android**”.

Mặc dù bản thân đã rất cố gắng nhưng do thời gian, kiến thức và kinh nghiệm có hạn, và cũng là lần đầu tiên được tiếp xúc và nghiên cứu xây dựng ứng dụng trên nền tảng android nên bài làm của chúng em còn nhiều thiếu sót trong việc trình bày, đánh giá cũng như thành phẩm có thể còn tồn tại lỗi không mong muốn. Chúng em rất mong nhận được sự thông cảm và đóng góp ý kiến của quý thầy cô và các bạn để chúng em hoàn thiện hơn và rút kinh nghiệm cho những đồ án, dự án lớn hơn trong tương lai.

Chúng em xin chân thành cảm ơn!

Mục lục

I.	TỔNG QUAN VỀ ĐỀ TÀI	5
1.	Mục đích của đề tài.....	5
2.	Giới thiệu đề tài	5
3.	Kiến trúc client-server	6
4.	Vòng đời các hoạt động	7
5.	Mô hình ERD miêu tả database.....	7
6.	Cấu trúc của hệ thống	8
II.	CÁC TÍNH NĂNG ĐÃ THIẾT LẬP	8
1.	Giới thiệu ứng dụng.....	9
2.	Đăng nhập và tạo tài khoản và đăng xuất.....	15
3.	Màn hình chờ ứng dụng.....	17
4.	Kết nối dữ liệu từ firebase	19
5.	Chức năng hiện danh sách bài hát	22
6.	Chức năng tìm kiếm bài hát theo tên, lyrics của bài hát.....	27
7.	Tạo danh sách yêu thích của người dùng	30
8.	Hiển thị thông tin của bài hát đang xem và gợi ý những bài tương tự sử dụng Machine Learning.....	32
9.	Chức năng hiển thị top bài hát sắp xếp theo rating và view	41
10.	Chức năng xem các hợp âm có sẵn	43
III.	DEMO	46
IV.	ƯU VÀ NHƯỢC ĐIỂM CỦA ỨNG DỤNG.....	58
1.	Ưu điểm:	58
2.	Nhược điểm:	59
V.	HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI:.....	59

PHẦN I. GIỚI THIỆU

I. TỔNG QUAN VỀ ĐỀ TÀI

1. Mục đích của đề tài

Hợp âm có vai trò rất quan trọng trong những ca khúc được sáng tác – nó như linh hồn quyết định đến sự thành công của một bài hát. Hợp âm chuẩn là một tổ hợp âm thanh có thể giúp người chơi nhạc hiểu và nắm bắt các âm sắc chuẩn nhất, dựa vào đó mà họ sẽ tạo ra những ca khúc có âm thanh nghe hay và đạt dào cảm xúc.

Ứng dụng chạy trên nền android với hai mục đích chính:

- Nghiên cứu các công nghệ lập trình android
- Phát triển ứng dụng xem hợp âm, hỗ trợ cho người chơi nhạc dễ dàng sử dụng mà không cần tra cứu website, bên cạnh đó giao diện app cũng như các chức năng được đề xuất hỗ trợ người dùng thao tác dễ dàng và tiện lợi

2. Giới thiệu đề tài

Music R(Recommand Music) là một ứng dụng xem hợp âm và gợi ý những bài tương tự cho người chơi nhạc. Ý tưởng dựa trên website hopamchuan.com.

Sử dụng các kiến thức về lập trình android, firebase và machine learning để hiện thực ứng dụng này bằng ngôn ngữ kotlin, chạy trên hệ điều hành android.

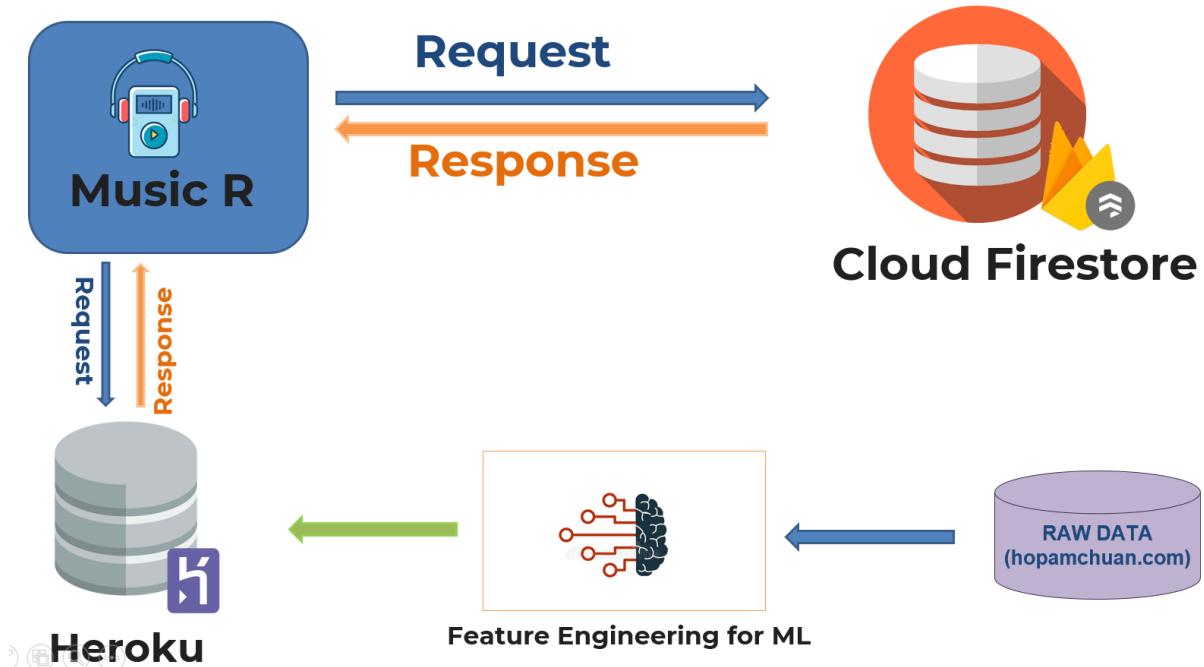
Nhóm em chủ yếu dùng firebase để có thể rút ngắn thời gian triển khai và mở rộng quy mô của ứng dụng cũng như việc lưu trữ dữ liệu, đồng bộ và hỗ trợ xác thực thông tin. Trong đồ án này, chúng em dùng cloud storage để lưu trữ dữ liệu và authentication để quản lý người dùng thông qua phương pháp xác thực email và mật khẩu.

Ứng dụng việc thu thập dữ liệu từ website hopamchuan để làm dữ liệu chính cho ứng dụng. Bên cạnh đó việc ứng dụng ML vào việc phân loại nhóm bài hát dựa trên topic được hỗ trợ bởi thuật toán phân cụm phân cấp trong thư viện scipy của python. Cụ thể hơn là từ dữ liệu ban đầu gồm 100 bài để demo sau khi được đánh topic cho các bài hát, chúng em sẽ biểu diễn nó về dạng cấu trúc sử dụng TF-IDF và sau khi sử dụng model phân cấp, tập dữ liệu được chia làm 8 cụm dựa trên mức độ tương quan của topic bài hát.

Ứng dụng với giao diện thân thiện, tích hợp các tính năng hữu ích chính như:

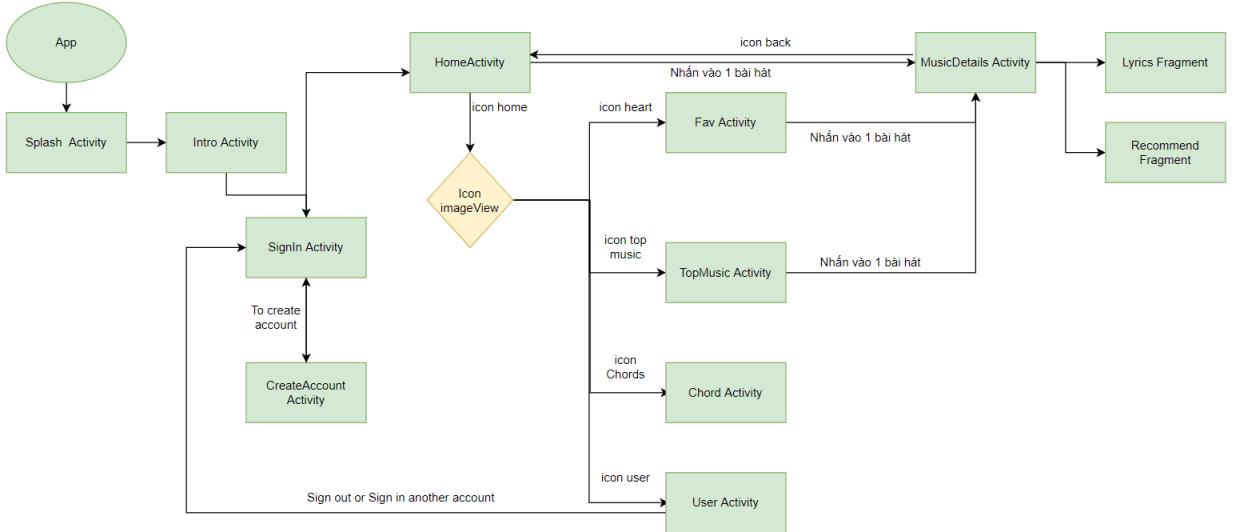
- Lưu trữ, hiển thị và truy xuất, tìm kiếm các bài hát theo tên và lyrics: Sử dụng các chức năng của firebase
- Gợi ý các bài hát tương tự cho người dùng: Sử dụng mô hình phân cụm phân cấp để phân loại bài hát sau đó sử dụng recommend API kết hợp firebase để truy xuất các bài hát tương tự.

3. Kiến trúc client-server



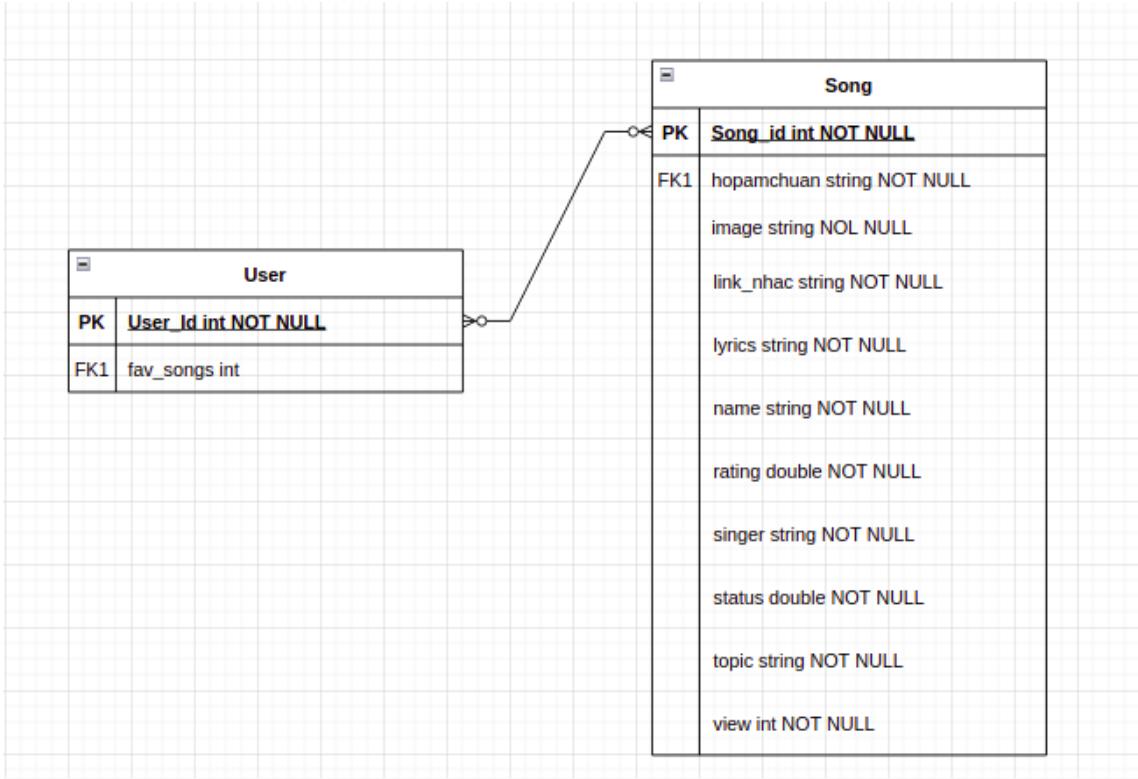
Hình 1. Kiến trúc client-server

4. Vòng đời các hoạt động



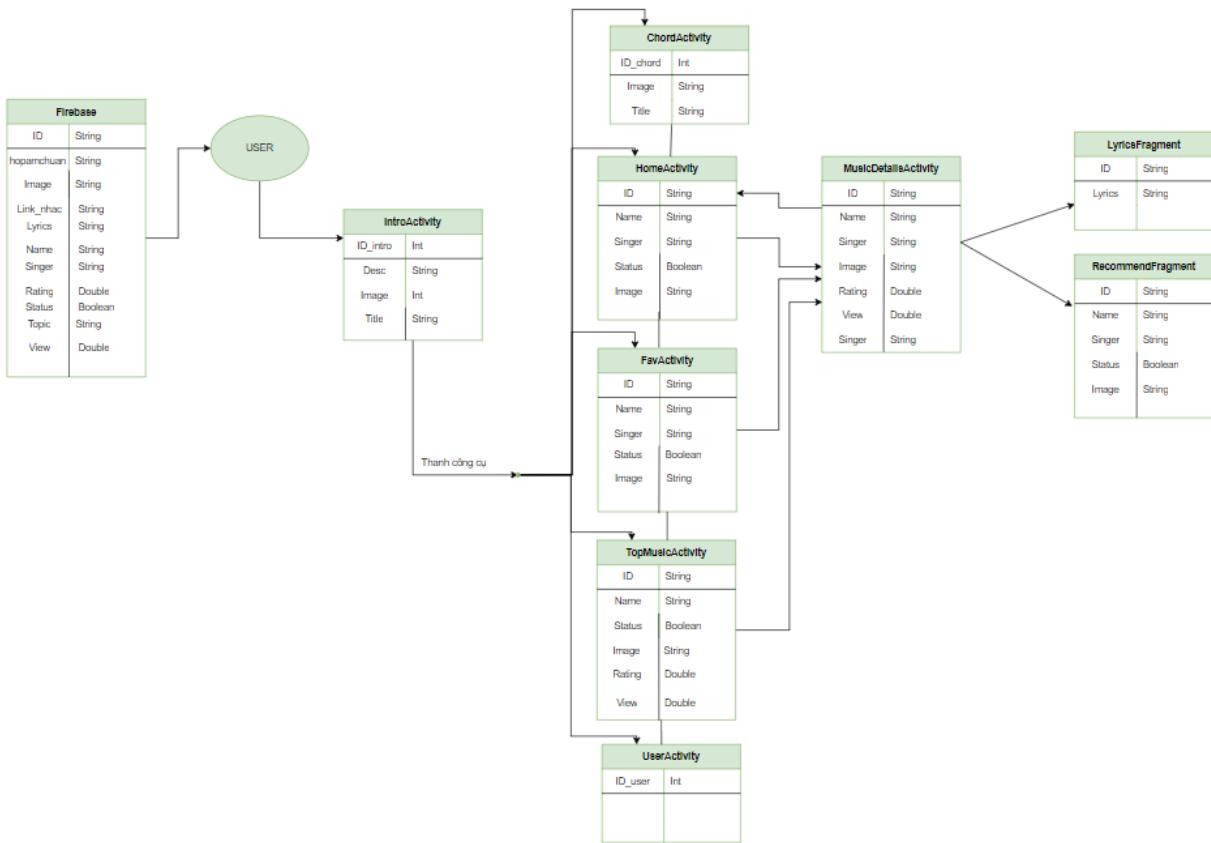
Hình 1. Vòng đời các hoạt động

5. Mô hình ERD miêu tả database



Hình 1. Mô hình ERD miêu tả database

6. Cấu trúc của hệ thống



Hình 1. Cấu trúc của hệ thống

II. CÁC TÍNH NĂNG ĐÃ THIẾT LẬP

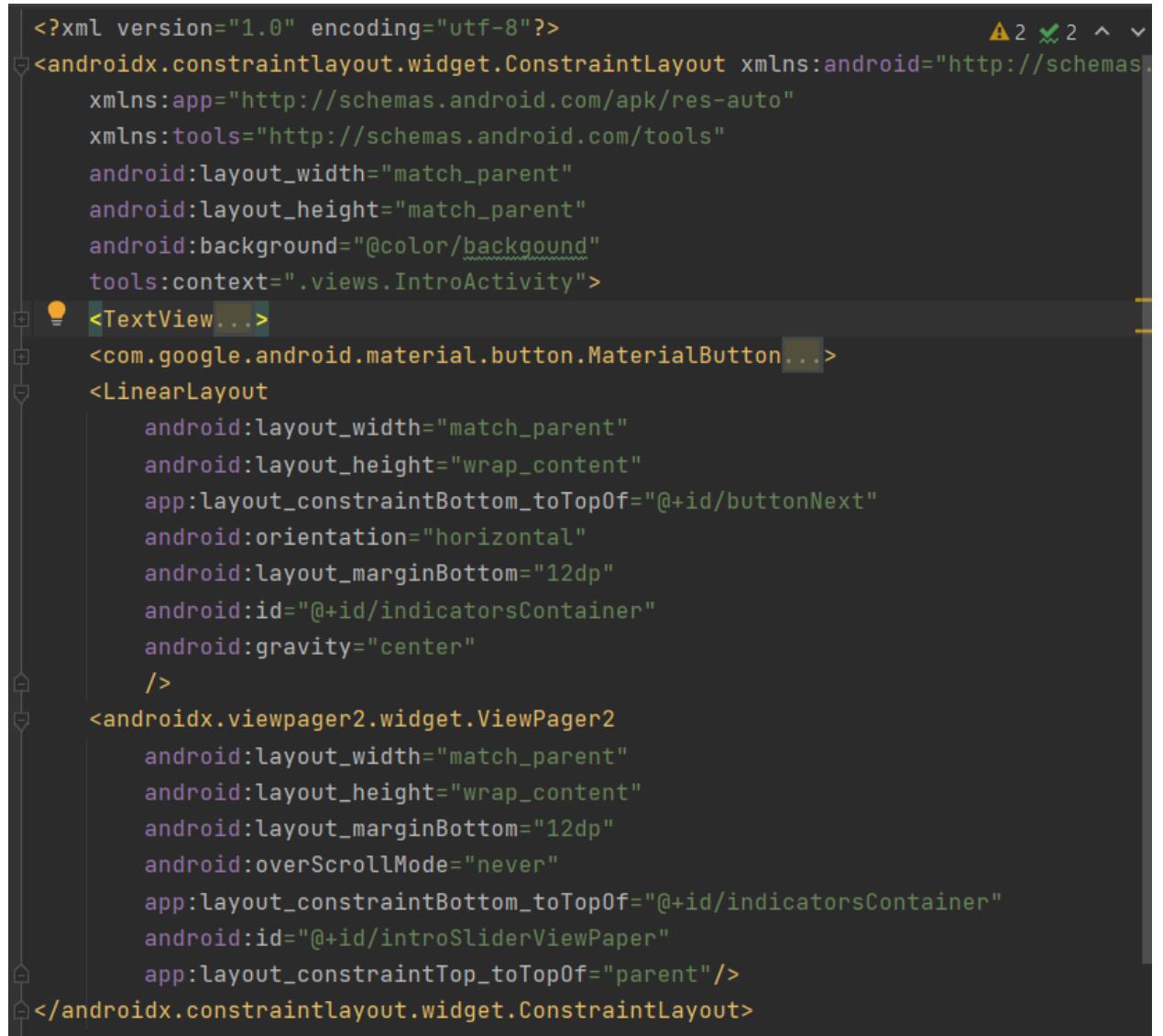
Để có thể sử dụng các tính năng ứng với từng màn hình, nhóm chúng em sử dụng thanh công cụ sử dụng CoordinatorLayout bao gồm các icon home - đây là màn hình thể hiện danh sách tất cả bài hát, icon favorite - màn hình thể hiện danh sách bài hát mà người dùng thích, icon top music - màn hình thể hiện danh sách top bài hát theo rating và theo lượt xem, icon chords - màn hình xem các hợp âm cơ bản và icon users để đăng xuất tài khoản, thoát ứng dụng hoặc đăng nhập bằng một tài khoản khác. Và setOnClickListener các button này cho các activity để có thể chuyển qua lại giữa các màn hình.

1. Giới thiệu ứng dụng

Xây dựng intro slider trong ứng dụng là một cách hiệu quả và tuyệt vời để giới thiệu các tính năng nổi bật của ứng dụng.

Bước 1. Tạo 1 Empty Activity và chúng em sẽ đặt tên là IntroActivity.kt và đây sẽ là Activity khởi chạy thứ hai khi mở ứng dụng sau phần splash screen.

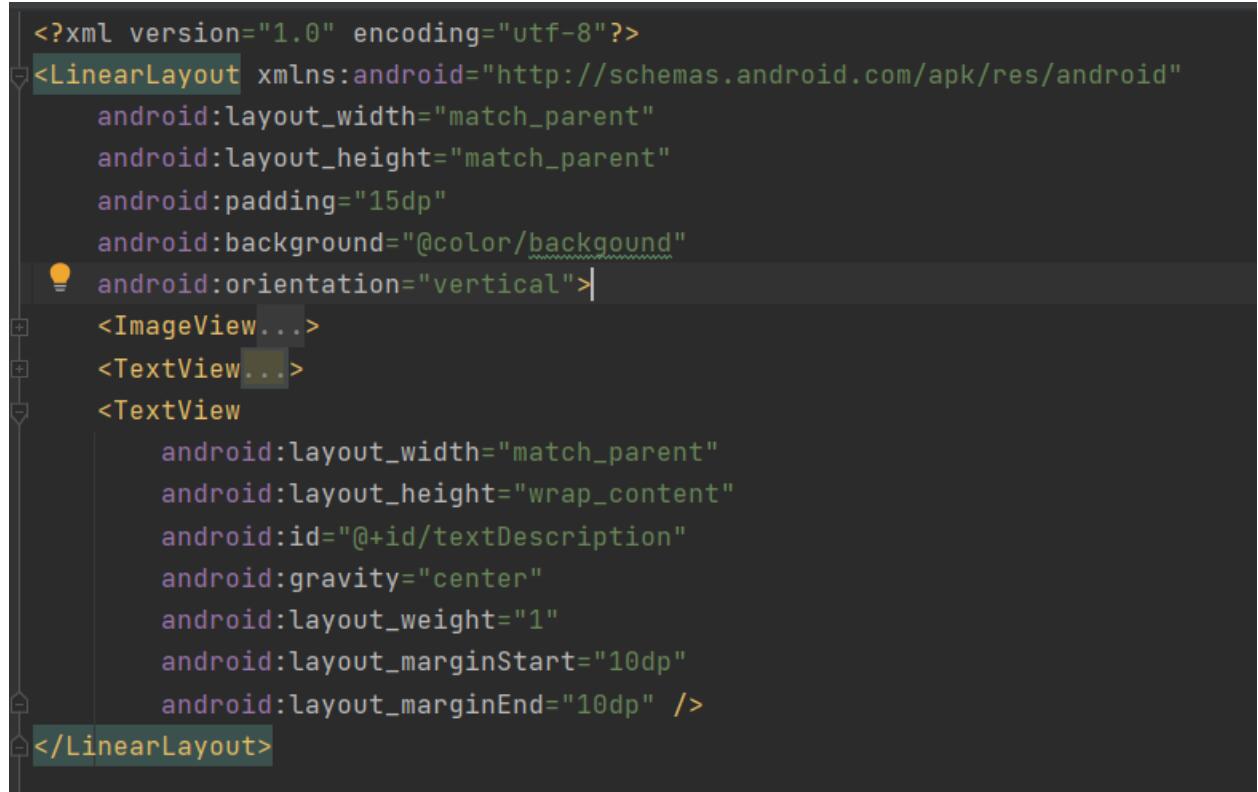
Mở tập tin activity_intro.xml và sửa đổi như sau:



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/backgound"
    tools:context=".views.IntroActivity">
    <TextView...>
    <com.google.android.material.button.MaterialButton...>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toTopOf="@+id/buttonNext"
        android:orientation="horizontal"
        android:layout_marginBottom="12dp"
        android:id="@+id/indicatorsContainer"
        android:gravity="center"
    />
    <androidx.viewpager2.widget.ViewPager2
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="12dp"
        android:overScrollMode="never"
        app:layout_constraintBottom_toTopOf="@+id/introSliderViewPager"
        android:id="@+id/introSliderViewPager"
        app:layout_constraintTop_toTopOf="parent"/>
</androidx.constraintlayout.widget.ConstraintLayout>
```

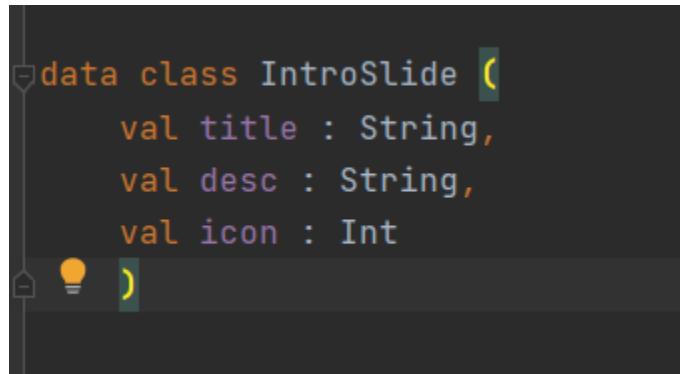
Cùng với đó là sử dụng ViewPagers để chuyển giao giữa các trang bằng cách cho phép người dùng vuốt sang trái hoặc sang phải. Để có thể hiển thị nhiều màn hình giới thiệu(cụ thể trong ứng dụng này là 5 màn hình, chúng em tạo 1 file

slide_item_container.xml để lưu thông tin của từng màn hình giới thiệu gồm có tên của màn hình, miêu tả ngắn về màn hình đó và ảnh minh họa như sau:



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="15dp"
    android:background="@color/background"
    android:orientation="vertical">
    <ImageView...>
    <TextView...>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/textDescription"
        android:gravity="center"
        android:layout_weight="1"
        android:layout_marginStart="10dp"
        android:layout_marginEnd="10dp" />
</LinearLayout>
```

Để có thể lấy các thông tin đó, chúng em đã tạo 1 data class để khởi tạo các biến là IntroSlide



```
data class IntroSlide (
    val title : String,
    val desc : String,
    val icon : Int
)
```

Và tạo adapter để lưu trữ dữ liệu này để hiển thị trong IntroSlideAdapter như sau:

```
class IntroSlideAdapter(private val inTroSlides: List<IntroSlide>): RecyclerView.Adapter<IntroSlideAdapter.IntroSlideViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): IntroSlideViewHolder {
        return IntroSlideViewHolder(
            LayoutInflater.from(parent.context).inflate(
                R.layout.slide_item_container,
                parent,
                attachToRoot: false
            )
        )
    }
    override fun onBindViewHolder(holder: IntroSlideViewHolder, position: Int) {
        holder.bind(inTroSlides[position])
    }
    override fun getItemCount(): Int {
        return inTroSlides.size
    }
    inner class IntroSlideViewHolder(view: View): RecyclerView.ViewHolder(view){
        private val textTitle = view.findViewById<TextView>(R.id.textTitle)
        private val textDescription = view.findViewById<TextView>(R.id.textDescription)
        private val imageIcon = view.findViewById<ImageView>(R.id.imageSlideIcon)
        fun bind(inTroSlide: IntroSlide){
            textTitle.text = inTroSlide.title
            textDescription.text = inTroSlide.desc
            imageIcon.setImageResource(inTroSlide.icon)
        }
    }
}
```

Vậy là xong phần chuẩn bị nơi lưu trữ và khai báo dữ liệu cũng như layout hiển thị, bây giờ quay lại IntroActivity, để có thể hiển thị như mong muốn, cần chuẩn bị code những chi tiết sau:

- setupIndicator để chuyển đổi trạng thái khi xem màn hình bắt kè

```
77     private fun setupIndicators(){
78         val indicator = arrayOfNulls<ImageView>(introSlideAdapter.itemCount)
79         val layoutParams: LinearLayout.LayoutParams =
80             LinearLayout.LayoutParams(
81                 ViewGroup.LayoutParams.WRAP_CONTENT,
82                 ViewGroup.LayoutParams.WRAP_CONTENT
83             )
84         layoutParams.setMargins( left: 8, top: 0, right: 8, bottom: 0)
85         for(i in indicator.indices){
86             indicator[i] = ImageView(applicationContext)
87             indicator[i].apply { this: ImageView?
88                 this?.setImageDrawable(
89                     ContextCompat.getDrawable(
90                         applicationContext,
91                         R.drawable.indicator_2
92                     )
93                 )
94                 this?.layoutParams = layoutParams
95             }
96             indicatorsContainer.addView(indicator[i])
97         }
98     }
```

```
9     private fun setCurrentIndicator(index:Int){
10         val childCount = indicatorsContainer.childCount
11         for ( i in 0 until childCount){
12             val imageView = indicatorsContainer[i] as ImageView
13             if (i == index){
14                 imageView.setImageDrawable(
15                     ContextCompat.getDrawable(
16                         applicationContext,
17                         R.drawable.indicator
18                     )
19                 )
20             } else{
21                 imageView.setImageDrawable(
22                     ContextCompat.getDrawable(
23                         applicationContext,
24                         R.drawable.indicator_2
25                     )
26                 )
27             }
28         }
29     }
30 }
```

- Tạo dữ liệu cho các màn hình

```
private val introSlideAdapter = IntroSlideAdapter(  
    listOf(  
        IntroSlide(  
            title: "Home App",  
            desc: "Welcome to our app!! Here you can view and search songs",  
            R.drawable.home  
)  
,  
        IntroSlide(  
            title: "Favorite Songs",  
            desc: "This page displays your favorite songs",  
            R.drawable.favorite  
)  
,  
        IntroSlide(  
            title: "Lyrics",  
            desc: "This page displays lyrics and song information",  
            R.drawable.lyrics  
)  
,  
        IntroSlide(  
            title: "Recommended Song",  
            desc: "This page recommend songs similar to the one you're watching",  
            R.drawable.recommend  
)  
,  
        IntroSlide(  
            title: "Chords",  
            desc: "This page shows basic guitar chords",  
            R.drawable.chord_image  
)  
)  
)  
)
```

- Thêm sự kiện setOnClickListener cho button Next và textSkipIntro. Khi click button next thì sẽ chuyển sang giới thiệu màn hình kế tiếp cho đến khi màn hình cuối kì thì chuyển đến SignInActivity. Khi ấn textSkipIntro thì chuyển thẳng đến SignInActivity

```

        override fun onCreate(savedInstanceState: Bundle?) {
            super.onCreate(savedInstanceState)
            setContentView(R.layout.activity_intro)
            introSliderViewPaper.adapter = introSlideAdapter
            setupIndicators()
            setCurrentIndicator(0)
            introSliderViewPaper.registerOnPageChangeCallback(object:
                ViewPager2.OnPageChangeCallback(){
                    override fun onPageSelected(position: Int) {
                        super.onPageSelected(position)
                        setCurrentIndicator(position)
                    }
                })
            buttonNext.setOnClickListener{ it: View!
                if(introSliderViewPaper.currentItem + 1 < introSlideAdapter.itemCount){
                    introSliderViewPaper.currentItem +=1
                }else{
                    Intent(applicationContext, SignInActivity::class.java).also { it: Intent
                        startActivity(it)
                        finish()
                    }
                }
            }
            textSkipIntro.setOnClickListener { it: View!
                Intent(applicationContext, SignInActivity::class.java).also { it: Intent
                    startActivity(it)
                    finish()
                }
            }
        }
    }
}

```

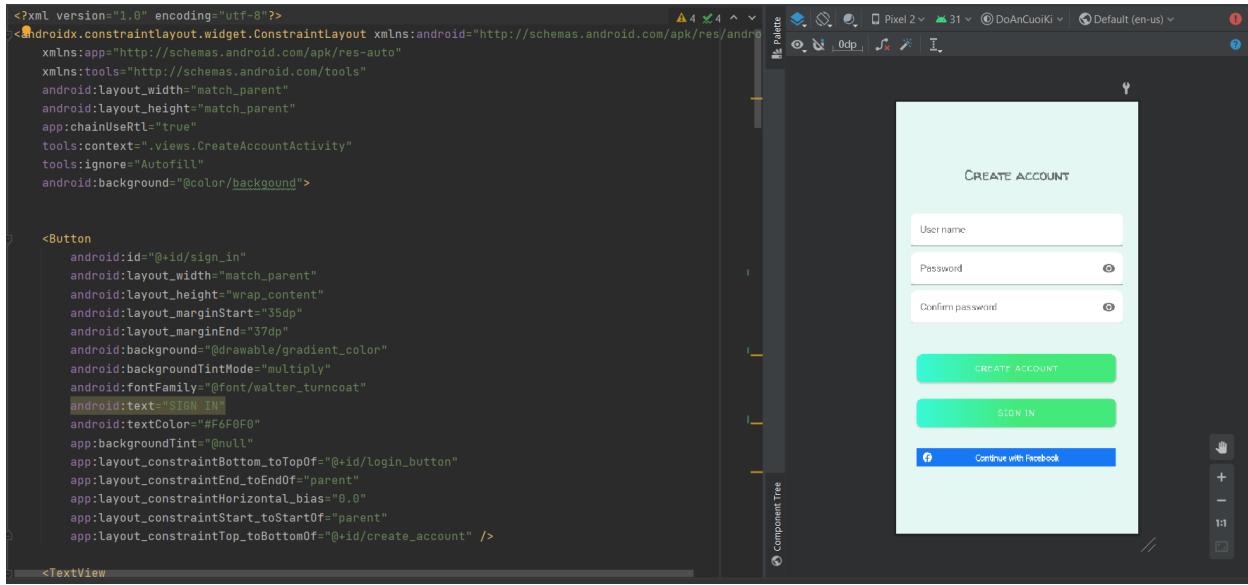
2. Đăng nhập và tạo tài khoản và đăng xuất

- Vì app có chức năng lưu trữ các bài hát người dùng yêu thích và mỗi người dùng sẽ có những bài hát khác nhau nên việc tạo tài khoản là cần thiết.

2.1 Đăng kí tài khoản:

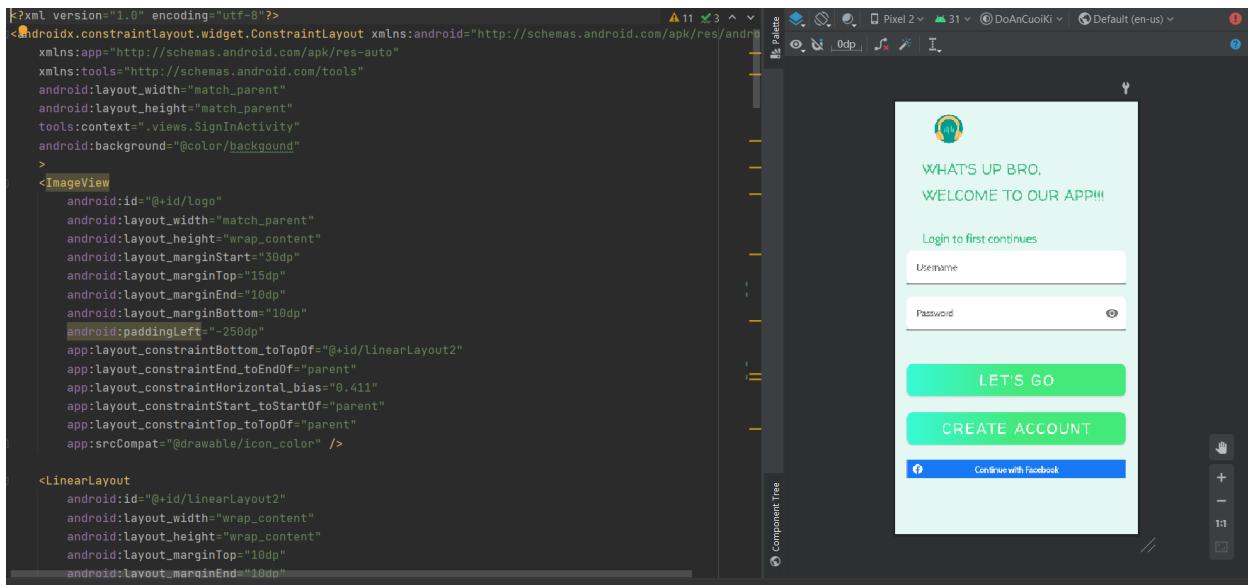
- Giao diện thiết kế nằm ở file activity_create_account.xml và xử lí logic nằm ở .views.CreateAccountActivity.kt
- Đăng kí tài khoản trực tiếp (tên đăng nhập có định dạng email)Lưu trữ trong Authentication của Firebase, trong giao diện đăng kí tài khoản, sau khi người dùng điền đầy đủ thông tin(nếu điền thiếu hoặc sai thì sẽ có

thông báo) người dùng nhấn vào Button “CREATE ACCOUNT” giao diện đăng kí tài khoản của nhóm em như sau:



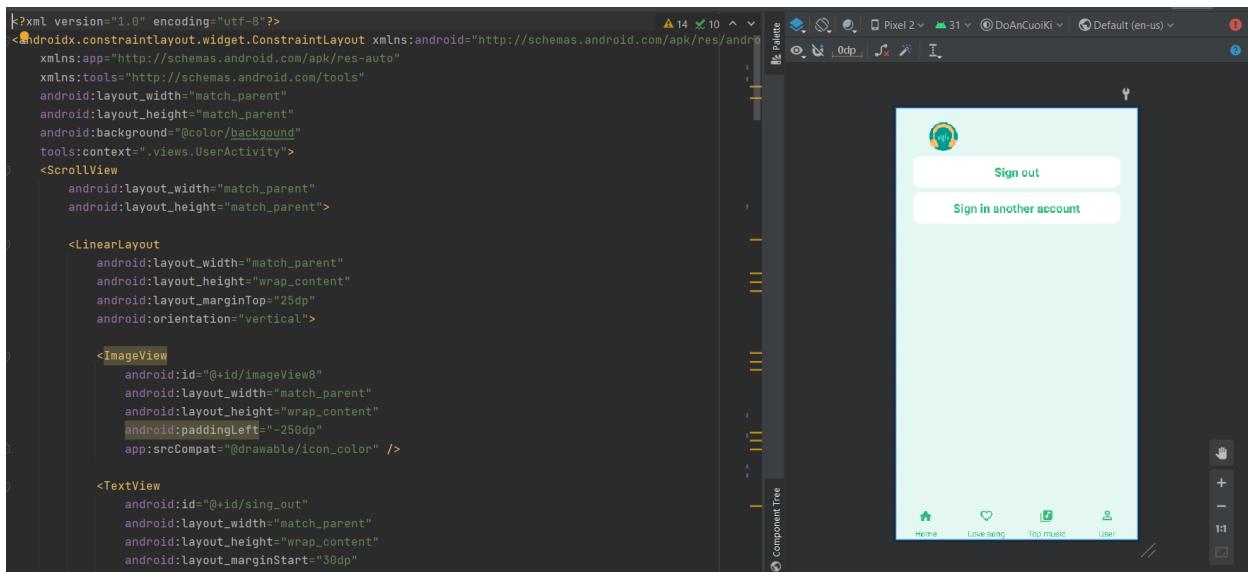
2.2 Đăng nhập:

- Giao diện thiết kế nằm ở file activity_login.xml và xử lí logic nằm ở .views.SignInActivity.kt
- Sử dụng 2 phương thức đăng nhập là sử dụng tài khoản đăng kí Authentication trên và được lưu trữ trong Authentication của Firebase, phương thức đăng nhập thứ 2 là sử dụng đăng nhập bằng Facebook, nếu đăng nhập không thành công sẽ xuất ra dòng trạng thái “sign in failed”, ngược lại, sẽ vào trang chủ của ứng dụng, giao diện của đăng nhập:



2.3 Đăng xuất:

Giao diện đăng xuất gồm 2 button để đăng xuất hoặc đăng nhập tài khoản khác, khi đăng nhập tài khoản khác thì tài khoản cũ sẽ tự động đăng xuất, cả 2 button này đều nhảy về trang đăng nhập, giao diện của trang đăng xuất như sau:



3. Màn hình chờ ứng dụng

Ngoại trừ việc tạo màn hình giới thiệu thì splash screen là màn hình đầu tiên sau khi người dùng truy cập vào ứng dụng. Splash screen sẽ hiện icon gif nhạc đang load. Nếu trải nghiệm đầu tiên kém nó sẽ ảnh hưởng đến sự tin tưởng của người dùng. Thông thường màn hình này sẽ tự động cài đặt mặc định điều này vô tình làm giảm trải nghiệm của người dùng, gây ấn tượng xấu. Đầu tiên tạo 1 Empty

Activity đặt tên là SplashActivity.kt . Ở file activity_splash.xml sửa thành như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".views.SplashActivity">
    <pl.droidsonroids.gif.GifImageView
        android:id="@+id/gif"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:src="@drawable/loading"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Ở file SplashActivity thì cần phải cài đặt thời gian cho tệp gif hiển thị, chúng em để là 2500ms. Sau thời gian này tệp gif hay màn hình chờ biến mất và vào IntroActivity:

```
class SplashActivity : AppCompatActivity() {
    private val splashTimeOut: Long = 2500
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_splash)

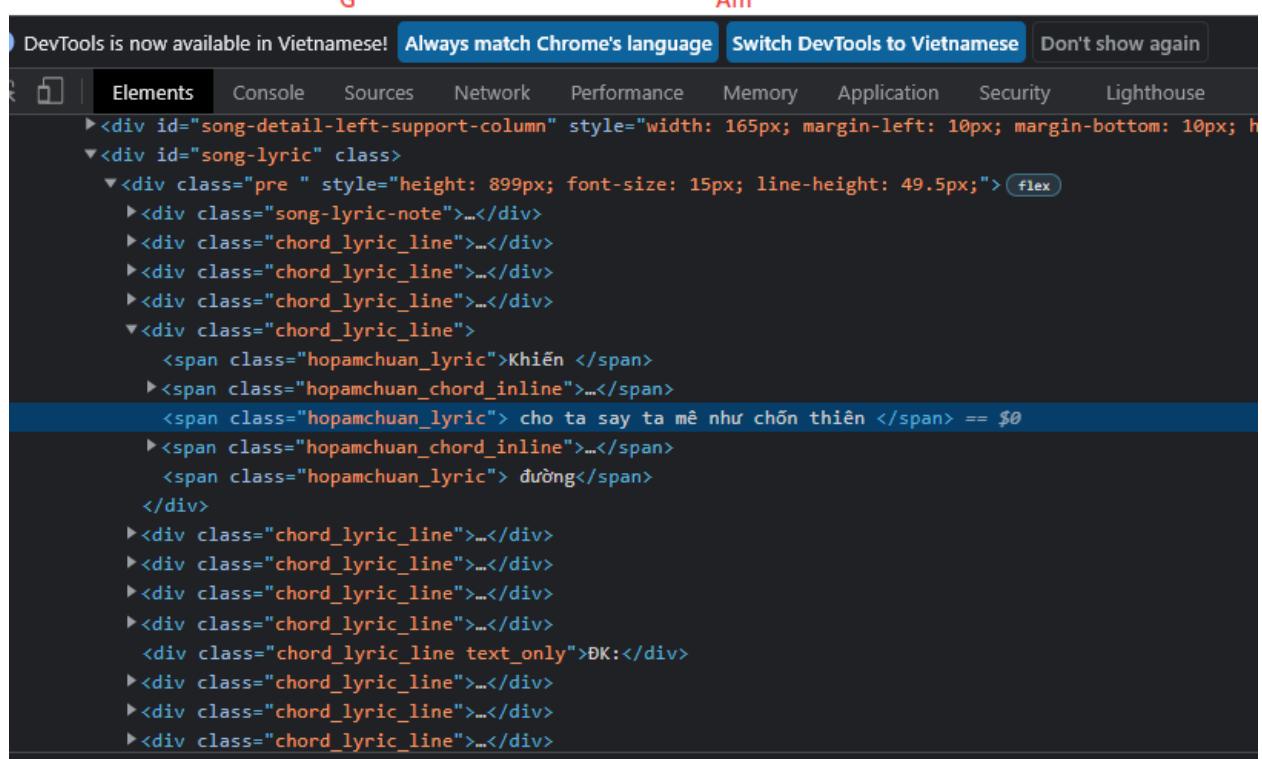
        @Suppress( ...names: "DEPRECATION")
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.R) {
            window.insetsController?.hide(WindowInsets.Type.statusBars())
        } else {
            window.setFlags(
                WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN
            )
        }

        Handler(Looper.getMainLooper()).postDelayed({
            val mainIntent = Intent(baseContext, IntroActivity::class.java)
            startActivity(mainIntent)
            finish()
        }, splashTimeOut)
    }
}
```

4. Kết nối dữ liệu từ firebase

- a. Phần dữ liệu bài hát cho toàn bộ ứng dụng
 - Nguồn data : hopamchuan.com
 - Cấu trúc HTML trang bài hát :

G Am C
Một bậc Quân Vương mang trong con tim hình hài đất nước
G Am C
Ngỡ như dân an ta sẽ chẳng bao giờ buồn
Am C
Nào ngờ một hôm ngao du nhân gian chạm một ánh mắt
G Am C
Khiến cho ta say ta mê như chốn thiên đường
Am C
Trời cao như đang trêu người thân ta khi bông hoa ấy
G Am C
Trót mang con tim trao cho một nam nhân thường
Am C
Giận lòng ta ban cho bông hoa thơm hồn về cung cấm



- Code Crawl : <https://github.com/DoanMinhTruong/Crawl-hopamchuan.com/blob/main/crawl.py>
 - Convert Data to Json và dùng Python đưa dữ liệu lên Cloud Store Firebase:

```

1 import firebase_admin
2 from firebase_admin import credentials
3 from firebase_admin import firestore
4 import json
5 cred = credentials.Certificate('config.json')
6 firebase_admin.initialize_app(cred)
7 db = firestore.client()
8 doc_ref = db.collection('PTUD')
9 data = json.load(open('data.json' , encoding="utf-8"))
10 for i in data:
11     img = i['image']
12     img = img.replace("[ " , "").replace("]" , "")
13     l = len(img)
14     img = img[1 : l-1]
15     i['image'] = img
16     doc_ref.document(str(i['id'])).set(i)

```

- Kết quả sau khi Upload:

doancuoiki-31a1c	PTUD	0
+ Start collection		+ Start collection
+ Add document		+ Add field
PTUD	> 0 >	
users	1 10 11 12 13 14 15 16 17 18 19 20 21 22 23	hopamchuan: "https://hopamchuan.com/song/8889/la-lung/" id: 0 image: "https://avatar-ex-swe.nixcdn.com/song/2018/01/26/1/8/9/0/1516930244148_640.jpg" link_nhac: "https://soundcloud.com/toilavu/lalung" lyrics: "Kia mản [C]đêm hiu hắt [G/B]. Mang tên [A7]em quay về trong ký [Dm7]úc , Của [G]anh qua [C]thời gian Chiều lảng [C]im nghe [G/B]gió Đung đưa [A7]cây . Như là bao nỗi [Dm7]nhớ cuốn [G]anh trôi [C]về đâu ? Nay [F]gió đứng [G]hát , Và [E7]mang nỗi nhớ [Am]chạy đí [F]quên âu lo , Quên [G]hết suy tư [C]một đời . [C7] [F]Mưa trong anh sê [G]vơi , Nhưng [E]đôi môi đang vấn [Am]vương , [F]Chí tình cờ nhìn em rồi mang theo [G]những con đau thét gào.Là lùng em [C]tới...[E]ói , tới [Am]bén anh trong chiều đông xa vắng...mà [F]saoGiờ đây nhìn [G]lại chẳng còn thấy [C] em . [C7]Là lùng em [C]vói...[E]gió hát [Am]lên câu ca làm anh thao thức , Mà [F]bao say mê nồng [G]nản giờ đã phải [C]mau Kia [F]hẳng ngáp [G]tràn , những giấc [Em]mo lại vừa bay [A7]đì , Gat [F]hết cuộc đời lè loi . Thôi mình [G]anh , lại ngồi nhớ [C]em!" name: "Lạ Lùng" rating: 4 singer: "TVQ"

Trong đó, 1 document gồm có 11 field bao gồm: id, hopamchuan(link dẫn đến trang chứa thông tin bài hát), image, link_nhac, lyrics, name, singer, rating, view, status(yêu thích hay không yêu thích), topic và view của bài hát.

Sau đó, kết nối firebase trong android studio để thực hiện tiếp việc truy vấn dữ liệu:

The screenshot shows the Firebase Cloud Firestore 'Get started with Cloud Firestore [KOTLIN]' page. At the top, there are tabs for 'Assistant' and 'Firebase', with 'Firebase' being the active tab. Below the tabs, it says '← Firebase > Cloud Firestore'. The main heading is 'Get started with Cloud Firestore [KOTLIN]'. A descriptive text states: 'Cloud Firestore is a flexible, scalable database from Firebase and Google Cloud. It keeps your data in sync across client apps through realtime listeners and offers offline support so you can build responsive apps that work regardless of network latency or internet connectivity.' Below this, there is a 'Launch in browser' button. The page is divided into two sections: ① Connect your app to Firebase (status: Connected) and ② Add Cloud Firestore to your app (status: Dependencies set up correctly). A note at the bottom says: 'NOTE: After adding the SDK, here are some other helpful configurations to consider:'. Under this note, there is a bullet point: 'Do you want an easier way to manage library versions? You can use the Firebase Android BoM to manage your Firebase library versions and ensure that your app is always using compatible library versions.' At the very bottom, it says: 'To use Cloud Firestore, you need to create the database in the Firebase console.'

5. Chức năng hiện danh sách bài hát

Sau khi lưu trữ dữ liệu trên firebase thì cần tạo 1 object Song lưu khai báo thông tin của một bài hát và 1 Adapter để kết nối dữ liệu bên firebase và ứng dụng.

Ở objects Song ngoại trừ việc khai báo dữ liệu thì còn tạo thêm hàm để cài đặt hiển thị các hình ảnh sử dụng Picasso - một thư viện của android mã nguồn mở phổ biến để load và hiển thị hình ảnh. Và thêm các hàm lấy dữ liệu theo các chức

năng như: lấy dữ liệu sắp xếp tên bài hát giảm dần theo bảng chữ cái, lấy top 10 bài hát có lượng rating cao nhất và top 10 bài hát có lượt xem cao nhất, lấy dữ liệu dựa trên id.

```
class Song() {
    var id : String? = null
    var name : String? = null
    var singer : String? = null
    var topic : String? = null
    var image : String? = null
    var lyrics : String? = null
    var rating : Double? = 0.0
    var view : Double ?= 0.0
    var status : Boolean?=false
    constructor(doc : DocumentSnapshot) : this() {
        id = doc.id
        name = doc.getString( field: "name")
        singer = doc.getString( field: "singer")?.replace( oldValue: "[ ' ", newValue: ""))
            ?.replace( oldValue: " ' ", newValue: "")?.replace( oldValue: " ] ", newValue: "")

        topic = doc.getString( field: "topic")
        image = doc.getString( field: "image")
        rating = doc.getDouble( field: "rating")
        lyrics = doc.getString( field: "lyrics")
        view = doc.getDouble( field: "view")
        status = doc.getBoolean( field: "status")
    }
}
```

```
    fun set() : Task<String> {
        val song = hashMapOf(
            "name" to name,
            "singer" to singer,
            "topic" to topic,
            "image" to image,
            "rating" to rating,
            "lyrics" to lyrics,
            "view" to view,
            "status" to status
        )

        if(id != null){
            return FirebaseUtils.db.collection(collection).document(id!!).set(song)
                .continueWith{ it: Task<Void!>
                    return@continueWith id
                }
        }else{

            return FirebaseUtils.db.collection(collection).add(song)
                .continueWith { task ->
                    return@continueWith task.result?.path ?: ""
                }
        }
    }
}
```

```
    fun setPic(context: Context, imageView: ImageView) {
        if (image != null) {
            // Get download url, and let Picasso load the image url into imageView
            Picasso.get().load(image)
                .placeholder(R.drawable.ic_broken_img)
                .into(imageView)
        } else {
            Picasso.get().load(R.drawable.ic_broken_img).into(imageView)
        }
    }
}
```

```
companion object {
    const val collection = "PTU"

    fun get() : Task<QuerySnapshot> {
        return FirebaseUtils.db.collection(collection).get()
    }
    fun getRecent() : Task<QuerySnapshot> {
        return FirebaseUtils.db.collection(collection)
            .orderBy( field: "name", Query.Direction.DESCENDING).get()
    }
    fun getRating(): Task<QuerySnapshot>{
        return FirebaseUtils.db.collection(collection)
            .orderBy( field: "rating", Query.Direction.DESCENDING).limit( limit: 10).get()
    }
    fun getView(): Task<QuerySnapshot>{
        return FirebaseUtils.db.collection(collection)
            .orderBy( field: "view", Query.Direction.DESCENDING).limit( limit: 10).get()
    }
    fun get(id : String) : Task<DocumentSnapshot> {
        return FirebaseUtils.db.collection(collection).document(id).get()
    }
}
```

Ở MusicsAdapters thì chúng em truyền vào context, layoutId, ArrayList song, bên cạnh đó là khởi tạo các hàm onCreateViewHolder, onBindViewHolder, getItemCount và class ViewHolder. Với hàm onBindViewHolder áp dụng dữ liệu vào để xem. Bên cạnh đó, ứng dụng còn hỗ trợ việc ghi nhận các bài hát yêu thích khi người dùng click vào checkBox favorite nên sẽ được khai báo và triển khai trong adapter này. Khi người dùng click vào checkbox yêu thích thì bài hát này sẽ được thêm vào firebase và hiển thị ở màn hình Favorite của ứng dụng. Sau khi ấn vào 1 bài hát có trong danh sách sẽ được setOnClickListener qua thông tin của bài hát được thể hiện trong MusicDetailsActivity - sẽ trình bày ở phần kế tiếp.

```
class MusicAdapters(val context: Context, val layoutId: Int, val songs: ArrayList<Song>):
    RecyclerView.Adapter<MusicAdapters.ViewHolder>(){
    public val fav_songs = ArrayList<String>();
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ViewHolder {
        var view : View = LayoutInflater.from(parent.context)
            .inflate(layoutId, parent, attachToRoot: false)
        return ViewHolder(layoutId, view)
    }
```

```
override fun onBindViewHolder(holder: ViewHolder, position: Int) {
    // Apply data into each view
    val song = songs[position]
    holder.name.text = song.name
    holder.singer.text = song.singer
    song.setPic(context, holder.image)
    if(song.status == true) holder.btnFavorite.isChecked = true

    holder.btnFavorite.setOnCheckedChangeListener { compoundButton, isChecked ->
        if(isChecked) {
            fav_songs.add(song.id.toString())
            FirebaseUtils.db
                .collection( collectionPath: "users")
                .document(FirebaseUtils.firebaseioAuth.currentUser!!.uid)
                .update(
                    field: "fav_songs",
                    FieldValue.arrayUnion(song.id.toString())
                )
            FirebaseUtils.db
                .collection( collectionPath: "PTUD")
                .document(song.id.toString())
                .update(
                    field: "status",
                    value: true)
            Toast.makeText(context, text: "add to favorite songs successfully!!", Toast.LENGTH_SHORT).show()
        }else{
            Toast.makeText(context, text: "remove favorite song hahaha", Toast.LENGTH_SHORT).show()
        }
    }
}
```

```
    }
}

holder.itemView.setOnClickListener { it: View!
    val intent = Intent(context, MusicDetailsActivity::class.java)
    val bundle = Bundle()
    bundle.putString("song", song.id)
    intent.putExtras(bundle)
    context.startActivity(intent)
}
```

```

        }

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

    class ViewHolder(layoutId: Int, itemView: View) : RecyclerView.ViewHolder(itemView) {
        var name : TextView
        var singer : TextView
        var image : ImageView
        var btnFavorite : CheckBox
        var status : Boolean ?= false

        init {
            name = itemView.findViewById(R.id.name)
            singer = itemView.findViewById(R.id.singer)
            image = itemView.findViewById(R.id.image)
            btnFavorite = itemView.findViewById(R.id.cbHeart)
        }
    }
}

```

6. Chức năng tìm kiếm bài hát theo tên, lyrics của bài hát

Ý tưởng: như tên của chức năng, ở chức năng này em xây dựng công cụ tìm kiếm bài hát dựa trên một đoạn văn bản và bọn em sẽ lấy đoạn văn bản này tìm kiếm những bài hát có tên và lyric phù hợp với đoạn văn bản đó.

- Giải quyết: trong giao diện HomeActivity sẽ tạo 1 editText để người dùng nhập nội dung tìm kiếm vào và khi này chức năng sẽ gọi hàm filter ở FilterAdapters để thực hiện tìm kiếm và trả kết quả tìm kiếm về trực tiếp cho RecyclerView hiển thị ở Front - end. Chức năng cụ thể của từng phần:
 - + Thừa kế lại đối tượng Song đã nói ở trên.
 - + HomeActivity là nơi kết nối giữa FE và BE:
 - Trong onCreate, có find.addTextChangedListener dùng object TextWatcher để tạo function before, on và after Textchanged tương ứng với từng thời điểm khác nhau ở textbox searchView này.

```

        find.addTextChangedListener(object : TextWatcher {
            override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {
                adapter?.filter?.filter(constraint: "")
            }

            override fun onTextChanged(cs: CharSequence?, p1: Int, p2: Int, p3: Int) {
                adapter?.filter?.filter(cs)
            }

            override fun afterTextChanged(s: Editable?) {
            }
        })
    }
}

```

- Function getFindSong để lấy số bài hát từ database và đẩy vào adapter.

```

    adapter = FilterAdapters(getFindSong(), context: this)
    songList.layoutManager = LinearLayoutManager(context: this, RecyclerView.VERTICAL, reverseLayout: false)
    adapterQuickView = MusicAdapters(context: this, R.layout.song_information, songs)
    songList.adapter = adapterQuickView
    songList.adapter = adapter
    find.addTextChangedListener(object : TextWatcher {
        override fun beforeTextChanged(p0: CharSequence?, p1: Int, p2: Int, p3: Int) {
            adapter?.filter?.filter(constraint: "")
        }

        override fun onTextChanged(cs: CharSequence?, p1: Int, p2: Int, p3: Int) {
        }

        override fun afterTextChanged(s: Editable?) {
            adapter?.filter?.filter(s)
        }
    })
}

```

- + FilterAdapter: là nơi phân phối giữa dữ liệu và BE, và là nơi đẩy các data từ BE lên FE. Chức năng nổi bật nhất của FilterAdapter là chính là Filter để lọc dữ liệu, thay kế từ Filterable.

```
class FilterAdapters(var countryList: MutableList<Song>, val context: Context) :  
    RecyclerView.Adapter<RecyclerView.ViewHolder>(), Filterable {  
  
    var countryFilterList = ArrayList<Song>()  
  
    init {  
        countryFilterList = countryList as ArrayList<Song>  
    }  
  
    override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {  
        (holder as ViewHolder).bind(countryFilterList.get(position))  
    }  
  
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): RecyclerView.ViewHolder {  
        val layoutInflater = LayoutInflater.from(parent.context)  
        return ViewHolder(layoutInflater.inflate(R.layout.song_information, parent, attachToRoot: false))  
    }  
  
    override fun getItemCount(): Int {  
        return countryFilterList.size  
    }  
}
```

```
inner class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {  
    fun bind(model: Song): Unit {  
        itemView.singer.text = model.singer  
        itemView.name.text = model.name  
        Picasso.get().load(model.image)  
            .placeholder(R.drawable.ic_broken_img)  
            .into(itemView.image)  
        itemView.setOnClickListener { it: View!  
            val intent = Intent(context, MusicDetailsActivity::class.java)  
            val bundle = Bundle()  
            bundle.putString("song", model.id)  
            intent.putExtras(bundle)  
            context.startActivity(intent)  
        }  
    }  
}
```

```

        override fun getFilter(): Filter {
            return object : Filter() {
                override fun performFiltering(constraint: CharSequence?): FilterResults {
                    val charSearch = constraint.toString()
                    if (charSearch.isEmpty()) {
                        countryFilterList = countryList as ArrayList<Song>
                    } else {
                        val resultList = ArrayList<Song>()
                        for (row in countryList) {
                            if (row.name?.toLowerCase(Locale.ROOT)
                                ?.contains(constraint.toString().toLowerCase()) == true ||
                                row.lyrics?.toLowerCase(Locale.ROOT)
                                    ?.contains(constraint.toString().toLowerCase()) == true ||
                                row.singer?.toLowerCase(Locale.ROOT)
                                    ?.contains(constraint.toString().toLowerCase()) == true
                            ) {
                                resultList.add(row)
                            }
                        }
                        countryFilterList = resultList
                    }
                }
                override fun publishResults(constraint: CharSequence?, results: FilterResults?) {
                    countryFilterList = results?.values as ArrayList<Song>
                    notifyDataSetChanged()
                }
            }
        }
    }
}

```

7. Tạo danh sách yêu thích của người dùng
- Ý tưởng giải quyết : lưu trữ thông tin bài hát yêu thích của người dùng trong 1 collection là “users” trên Firestore, mỗi document trong collection này đại diện cho 1 người dùng(sử dụng id của user hiện tại để làm id của document), mỗi document sẽ có 1 thuộc tính có kiểu mảng là “fav_songs” chứa thông tin id bài hát mà người đó đã bấm “yêu thích”. Sau đó phần BE sẽ lấy dữ liệu này(mảng id bài hát yêu thích) truy vấn sang collection “PTUD” và hiển thị bài hát yêu thích trong layout *Love Song* của ứng dụng.

The screenshot shows the Firebase Firestore interface. On the left, there's a sidebar with a user icon, 'users', and a specific document ID 'NI57EmI5HiSKbBCVabt9FdFPNYw1'. The main area displays three tabs: 'doancuoiki-31a1c' (with '+ Start collection' and 'PTUD' sub-collections), 'users' (with '+ Add document' and a sub-document 'NI57EmI5HiSKbBCVabt9FdFPNYw1' selected), and 'NI57EmI5HiSKbBCVabt9FdFPNYw1' (with '+ Start collection' and '+ Add field' options). Under '+ Add field', there's a 'fav_songs' array field containing four elements: 0 "1", 1 "10", 2 "86", and 3 "12".

Cấu trúc lưu trữ dữ liệu của người dùng trên Firestore

- Code xử lí: xử lí trong file FavActivity.kt và MusicAdapter.kt:

```
holder.btnAddFavorite.setOnCheckedChangeListener { compoundButton, isChecked ->
    if(isChecked) {
        fav_songs.add(song.id.toString())
        FirebaseUtils.db
            .collection( collectionPath: "users")
            .document(FirebaseUtils.firebaseioAuth.currentUser!! .uid)
            .update(
                field: "fav_songs",
                FieldValue.arrayUnion(song.id.toString())
            )
        FirebaseUtils.db
            .collection( collectionPath: "PTUD")
            .document(song.id.toString())
            .update(
                field: "status",
                value: true)
        Toast.makeText(context, text: "add to favorite songs successfully!!", Toast.LENGTH_SHORT).show()
    }else{
        fav_songs.remove(song.id.toString())
        FirebaseUtils.db
            .collection( collectionPath: "users")
            .document(FirebaseUtils.firebaseioAuth.currentUser!! .uid)
            .update(
                field: "fav_songs",
                FieldValue.arrayRemove(song.id.toString())
            )
        FirebaseUtils.db
            .collection( collectionPath: "PTUD")
            .document(song.id.toString())
            .update(
                field: "status",
                value: false)
    }
}
```

Thay đổi trạng thái bài hát là bài yêu thích hay không, thêm/xóa id bài hát trên firestore

```

FirebaseFirestore.getInstance().collection( collectionPath: "users")
    .document(FirebaseUtils.firebaseioAuth.currentUser!!.uid).get()
    .addOnSuccessListener { querySnapshot ->
        val documents = querySnapshot.data
        if (documents != null) {
            for (doc in documents) {
                string_id.add(doc.toString())
                val regex = Regex( pattern: "[0-9]+")
                val matches = regex.findAll(doc.toString() as CharSequence).map{it.value}.toList()
                for(id in matches){
                    list_id.add(id)
                    Song.get(id)
                        .addOnSuccessListener { documentSnapshot ->
                            val songa = Song(documentSnapshot!!)
                            songs.add(songa)
                            Log.d( tag: "name: ", songa.name.toString())
                            adapterQuickView!!.notifyItemInserted( position: songs.size - 1)
                        }
                        .addOnFailureListener { e ->
                            Log.e(
                                tag: "",
                                msg: "fromCloudFirestore: Error loading ContactInfo data from Firestore - " + e.message
                            );
                        };
                }
            }
        }
    }
}

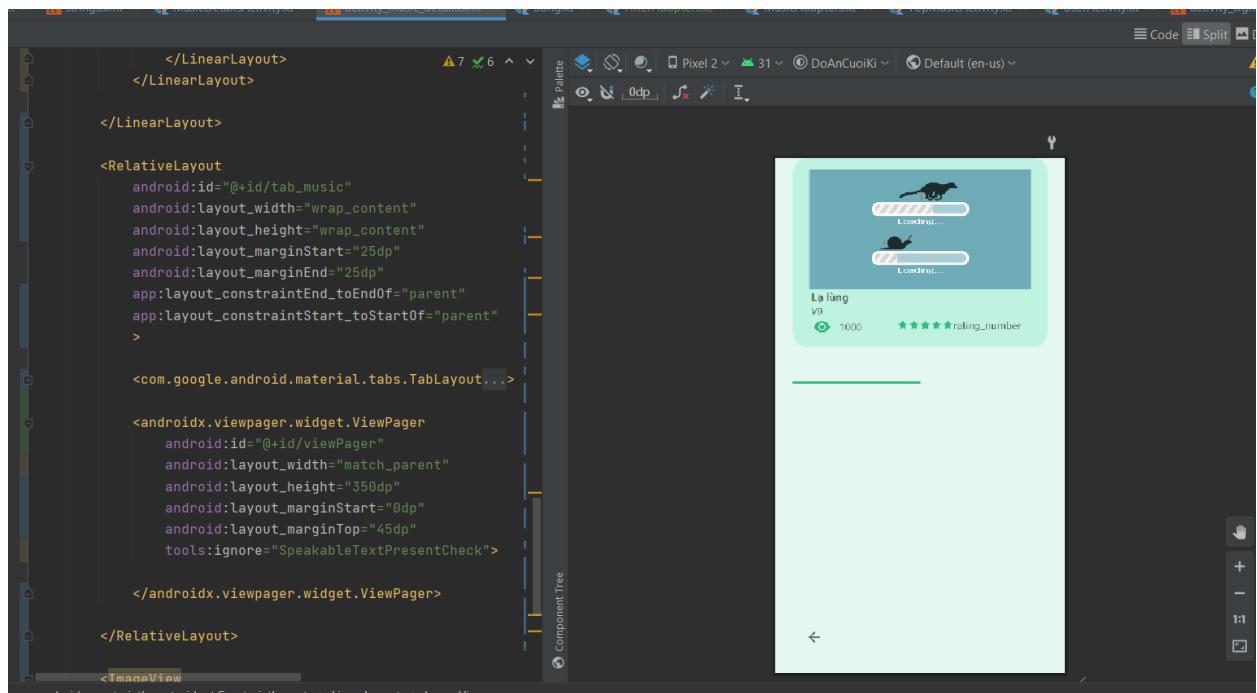
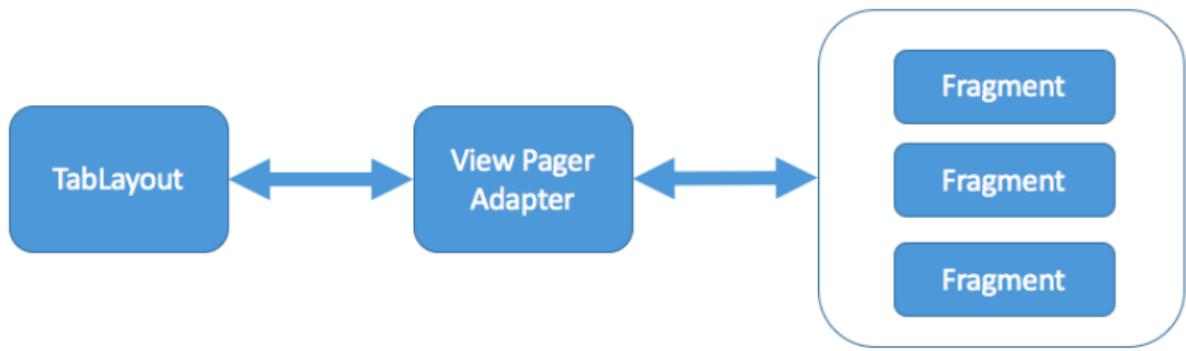
```

Thực hiện lấy mảng id bài hát yêu thích và hiển thị lên màn hình

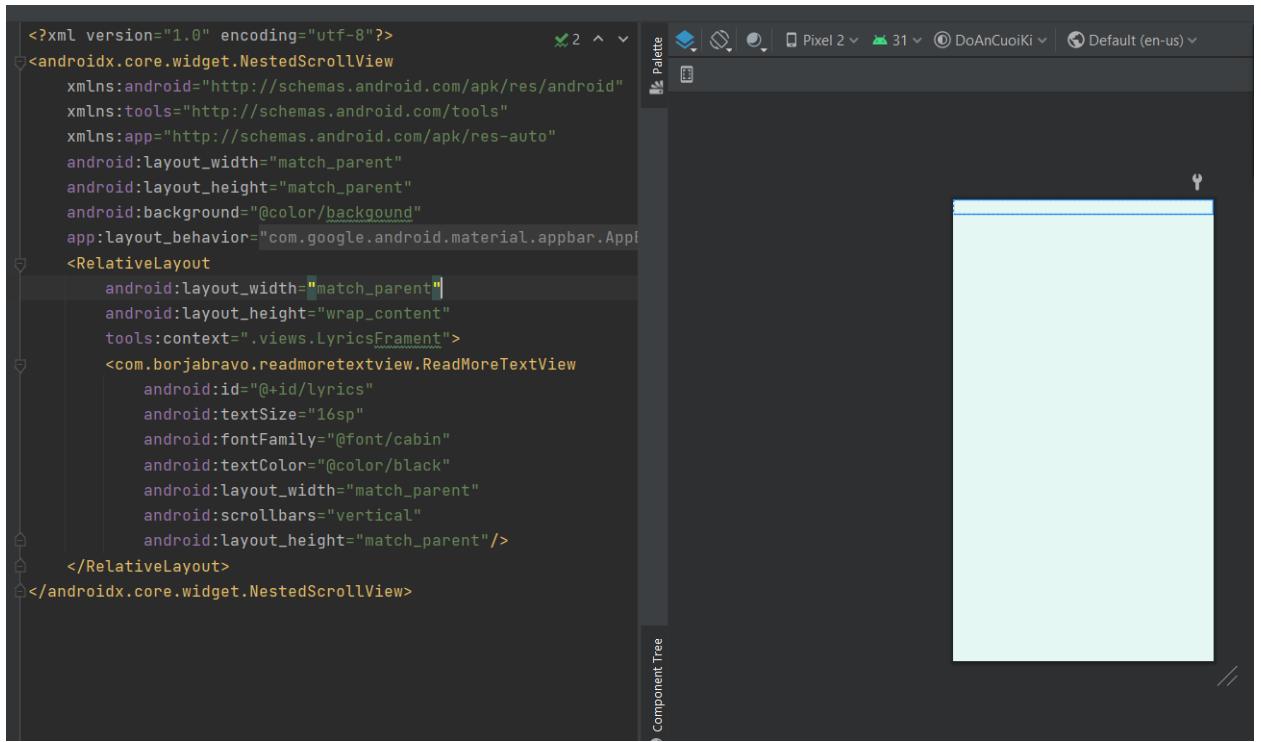
8. Hiển thị thông tin của bài hát đang xem và gợi ý những bài tương tự sử dụng Machine Learning

a. *Hiển thị thông tin của bài hát:*

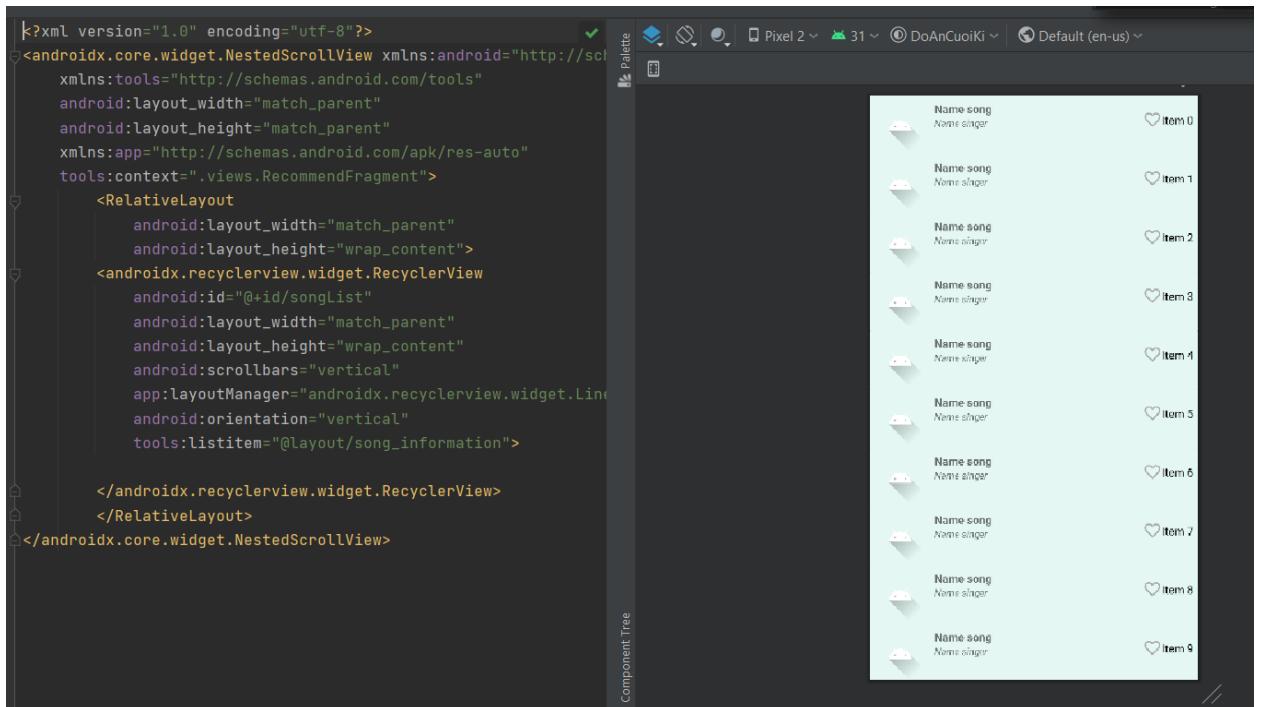
Sau khi ấn vào một bài hát bất kì trên ứng dụng, sẽ chuyển đến màn hình thông tin của bài hát đó - MusicDetailsActivity. Ở đây, chúng em sử dụng viewpager cộng với 2 fragment. LyricsFragment để hiển thị lyrics của bài hát và RecommendFragment để hiển thị các bài hát tương tự được gợi ý. Đầu tiên, tạo 1 Empty Activity và đặt tên là MusicDetailActivity. Ở file xml, ngoại trừ tạo các TextView và ImageView để hiển thị thông tin của bài hát như tên bài hát, tên ca sĩ, lượt xem và lượng rating, thì còn tạo thêm tablayout để tạo thanh kéo qua lại cho hai fragment và viewpager để cho phép người dùng vuốt tay qua trái hay qua phải để chuyển qua lại giữa hai nội dung một cách đồng thời. Nó có thể trượt và chuyển đổi một cách nhẹ nhàng giữa hai giao diện. ViewPager không phải là một view chuẩn trong android mà là một thành phần nằm trong gói android.support:design:xxx



Kế đến là tạo 2 fragment blank và đặt tên lần lượt là LyricsFragment và RecommendFragment. Ở file fragment_lyrics.xml sẽ cho hiển thị lời bài hát



Và file fragment_recommend.xml:



Để có thể hiển thị như mong muốn, về phần xử lý cụ thể như sau:

LyricsFragment: Chúng em gọi lại object Song, sử dụng hàm lấy dữ liệu theo id của bài hát tương ứng. Sau đó xử lý chuỗi lyrics sau đó biểu diễn màu hợp âm khác với màu chữ phần lời trong bài hát. Việc xử lý này sử dụng regex trong kotlin và để đổi màu, sử dụng SpannableString

```

class LyricsFragment : Fragment() {
    var _song : Song?=null
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val id = requireActivity().intent.extras!!.getString( key: "song")
        Song.get(id!!)
        .addOnSuccessListener{ documentSnapshot ->
            _song = Song(documentSnapshot)
            val lyrics = _song!.lyrics
            val regex = Regex( pattern: "[\\u00c1[A-Z|a-z][0-9]*[/]*\\u00c3]")
            val spannable = SpannableString(lyrics)
            val matches = regex.findAll(lyrics as CharSequence)
            matches.forEach {match ->
                match.range.forEach { it:int -
                    val startl = it
                    val endl = it + match.value.length/match.range.count()
                    spannable.setSpan(ForegroundColorSpan(Color.rgb( red: 70, green: 166, blue: 25)),startl,endl, Spannable.SPAN_EXCLUSIVE_EXCLUSIVE)
                }
            }
            lyrics.setText(spannable, TextView.BufferType.SPANNABLE)
        }
        return inflater.inflate(R.layout.fragment_lyrics, container, attachToRoot: false)
    }
}

```

- **RecommendFragment:** Để hiển thị danh sách các bài hát nên ở đây ta cần gọi lại MusicAapters, sau đó lấy list id các bài hát tương tự từ Recomment_API từ id của bài hát đang xem bằng cách ở MusicDetailsActivity tạo thêm một hàm trả về id của bài đang xem

```

class RecommendFragment : Fragment() {
    val songs = ArrayList<Song>()
    var _adapterQuickView: MusicAdapters? = null
    val recommendList = ArrayList<Int>()
    private lateinit var _viewModel : APIViewModel
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val repository = Repository()
        var _view = inflater.inflate(R.layout.fragment_recommend, container, attachToRoot: false)
        _adapterQuickView = MusicAdapters(requireContext(), R.layout.song_information, songs)
        _view.findViewById<RecyclerView>(R.id.songList).adapter = _adapterQuickView
        val viewModelFactory = APIViewModelFactory(repository)
        _viewModel = ViewModelProvider( owner: this , viewModelFactory).get(APIViewModel::class.java)
        _viewModel.getRecommend()
        _viewModel.myResponse.observe(requireActivity(), Observer { response ->

```

```
viewModel.myResponse.observe(requireActivity(), Observer { response ->
    if(response.isSuccessful){
        var recommends : Recommend?
        recommends = response.body()?.get((requireContext() as MusicDetailsActivity).getSong()!!._id!!).toInt()
        recommendList.addAll(recommends!!.recommend)
        for(id in recommendList){
            Song.get(id.toString())
                .addOnSuccessListener { documentSnapshot ->
                    val songa = Song(documentSnapshot!!)
                    songs.add(songa)
                    Log.d( tag: "name: ", songa.name.toString())
                    adapterQuickView!!.notifyItemInserted( position: songs.size - 1)
                }
                .addOnFailureListener { e ->
                    Log.e(
                        tag: "",
                        msg: "fromCloudFirestore: Error loading ContactInfo data from Firestore - " + e.message
                    );
                }
        }
        Log.d( tag: "Response" , response.body()?.get(2)?.recommend.toString())
    }else{
        Log.d( tag: "Response" , response.errorBody().toString())
    }
})]
return view
}
```

- Sau đó, để hai fragment này có thể sử dụng được, ở MusicDetailsActivity mình cần xử lý phần trả về id bài hát hiện đang xem và tạo hàm setUp cho hiển thị Tablayout và fragment tương ứng:

```
class MusicDetailsActivity : AppCompatActivity() {
    var song : Song ?=null
    override fun onCreate(savedInstanceState: Bundle?){
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_music_details)
        ic_back.setOnClickListener { it: View!
            startActivity(Intent(packageContext: this,HomeActivity::class.java))
        }
        val bundle = intent.extras
        val id = bundle!!.getString(key: "song")
        Song.get(id!!)
        .addOnSuccessListener{ documentSnapshot ->
            song = Song(documentSnapshot)
            namesong.text = song!!.name
            singer.text = song!!.singer
            text_view.text = song!!.view.toString()
            song!!.setPic(context: this,imageSong)
            ratingBar.rating = song!!.rating!!.toFloat()
            rating_number.text = song!!.rating!!.toFloat().toString()
        }
        setUpTabs()
    }
    private fun setUpTabs() {
        val viewPager = findViewById<ViewPager>(R.id.viewPager)
        val tabs = findViewById<TabLayout>(R.id.tab_layout)
        val adapter = ViewPagerAdapter(supportFragmentManager)
        adapter.addFragment(LyricsFragment(), title: "Lyrics")
        adapter.addFragment(RecommendFragment(), title: "Recommend")
        viewPager.adapter = adapter
        tabs.setupWithViewPager(viewPager)
    }
}
```

```

    }
    @JvmName(name: "getSong1")
    public fun getSong(): Song? {
        return song
    }
}
```

b. Gợi ý những bài tương tự với bài hát đã xem:

- Xử lý model:
- + Load data

```
[ ] df = pd.read_csv("data.csv")
df.head(3)
```

	id	hopamchuan
0	0	https://hopamchuan.com/song/8889/la-lung/
1	1	https://hopamchuan.com/song/3032/chieu-nay-kho... Chiều nay khó
2	2	https://hopamchuan.com/song/4674/ba-ke-cong-he/ba-ke-cong-he/

```
[ ] X = df[['id','name','singer','view','rating','topic']]
corpus = df['topic']
```

- + Chuẩn hóa dữ liệu và biến đổi về dạng có cấu trúc sử dụng TF-IDF

```
▶ from nltk.tokenize.toktok import ToktokTokenizer
nltk.download('stopwords')
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
    # lower case and remove special characters\whitespaces
    doc = re.sub(r'[^a-zA-Z\s]', ' ', doc, re.I|re.A)
    doc = doc.lower()
    doc = doc.strip()
    # tokenize document
    tokens = wpt.tokenize(doc)
    # filter stopwords out of document
    filtered_tokens = [token for token in tokens if token not in stop_words]
    # re-create document from filtered tokens
    doc = ' '.join(filtered_tokens)
    return doc

normalize_corpus = np.vectorize(normalize_document)

[ nltk_data] Downloading package stopwords to C:\Users\Bao_Bao
[ nltk_data]      ne\AppData\Roaming\nltk_data...
[ nltk_data]  Package stopwords is already up-to-date!
```

```
[ ] corpus_df = X
norm_corpus = normalize_corpus(corpus)
```

```
[ ] from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix = cv_matrix.toarray()
cv_matrix

[ ] vocab = cv.get_feature_names()
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)

[ ] # you can set the n-gram range to 1,2 to get unigrams as well as bigrams
bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = bv_matrix.toarray()
vocab = bv.get_feature_names()
pd.DataFrame(bv_matrix, columns=vocab)

[ ] from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., use_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()

vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

+ Dùng Cluster để phân cụm, thu được kết quả và lưu lại:

```

▶ similarity_matrix = cosine_similarity(tv_matrix)
similarity_df = pd.DataFrame(similarity_matrix)
similarity_df
from scipy.cluster.hierarchy import dendrogram, linkage

z = linkage(similarity_matrix, 'ward')
# pd.DataFrame(z, columns=['name', 'Document\Cluster 2',
#                         'Distance', 'Cluster Size'], dtype='object')
from scipy.cluster.hierarchy import fcluster
max_dist = 5.0

cluster_labels = fcluster(z, max_dist, criterion='distance')
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
dflabel = pd.concat([corpus_df, cluster_labels], axis=1)

[ ] dflabel.to_csv("dataPTUD.csv")

```

```
[ ] df = pd.read_csv("dataPTUD.csv")
df.head(10)
```

	Unnamed: 0	id	name	singer	view	rating	topic	ClusterLabel
0	0	0	Lạ Lùng	['Vũ']	2068970	4.5	buồn, thất tình, cô đơn	1
1	1	1	Chiều Nay Không Có Mưa Bay	['Trung Quân Idol', 'Thái Tuyết Trâm', 'Vũ Đinh...']	1322261	4.0	buồn, thất tình, cô đơn	1
2	2	2	Ba Ké Con Nghe	['Nguyễn Hải Phong', 'Dương Trần Nghĩa', 'Bập ...']	1236608	4.0	gia đình	6
3	3	3	Có Chàng Trai Viết Lên Cây	['Phan Mạnh Quỳnh']	1212524	4.5	buồn, thất tình, cô đơn	1
4	4	4	Một Nhà	['Da Lab', 'Vicky Nhunny']	1206893	4.5	sôi nổi, tình yêu	4
5	5	5	Em gái mưa	['Hương Tràm', 'Mr. Siro', 'Anh Khang', 'Thế P...']	1094696	4.5	buồn, thất tình, cô đơn	1

- Sử dụng kết quả thu được ở dataPTUD.csv và FastAPI để xây dựng 1 API nhỏ để thực hiện get kết quả để xuất từ id của 1 bài hát bất kỳ:

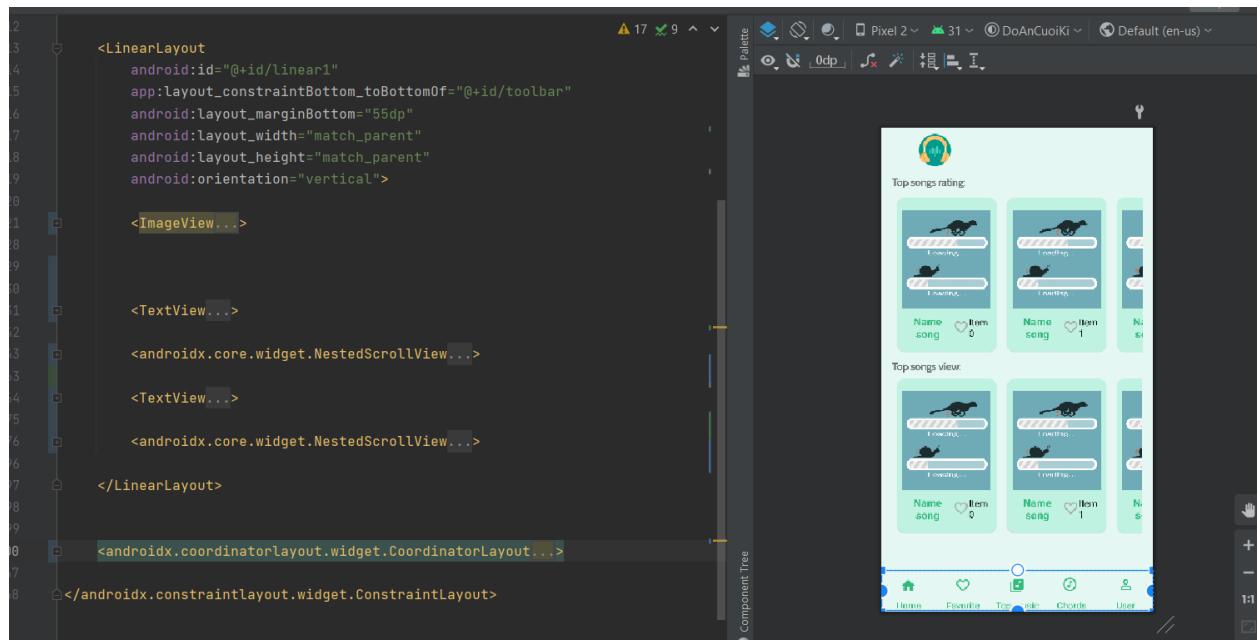
```

25  @app.on_event("startup")
26  def predict(id_music):
27      cluster = df['ClusterLabel'].loc[df.id == id_music]
28      return int(cluster)
29  def nameclass(cluster):
30      ids = df.id.loc[df.ClusterLabel==cluster]
31      print(ids)
32      ids = list(ids)
33      ids = [int(x) for x in ids]
34      return {
35          "recommend" : ids
36      }
37  @app.get('/')
38  async def index():
39      res = [nameclass(predict(int(id))) for id in range(99)]
40      return res
41  @app.get('/{id}')
42  async def get(id : int):
43      return (nameclass(predict(int(id))))

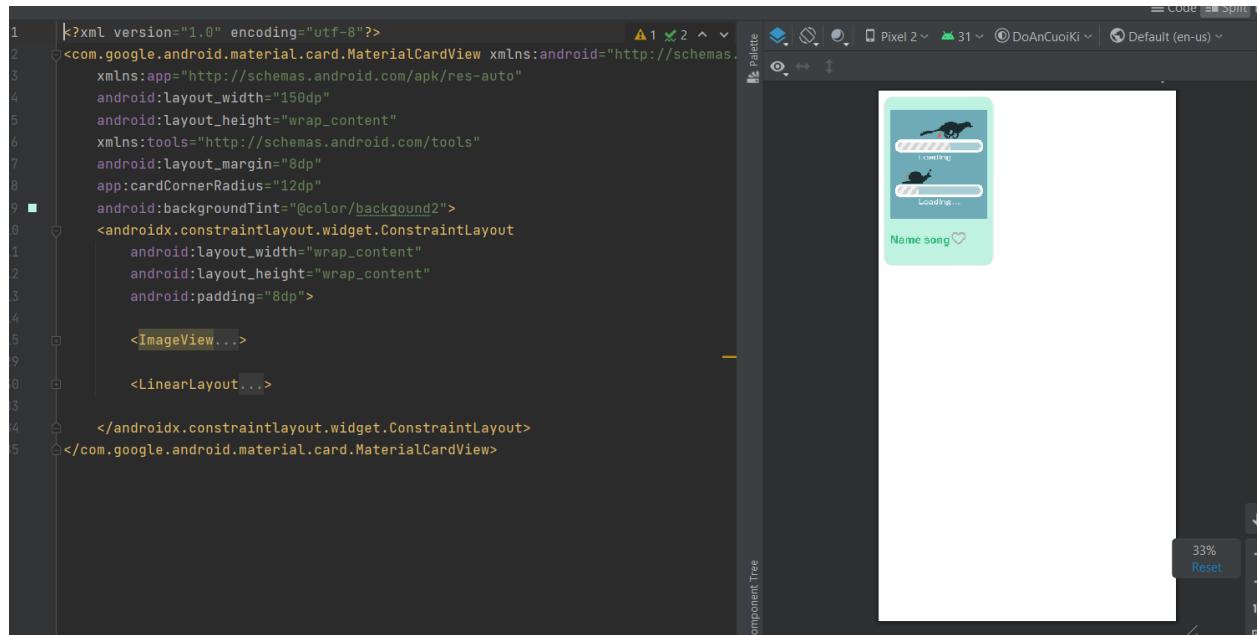
```

9. Chức năng hiển thị top bài hát sắp xếp theo rating và view

Đầu tiên, tạo 1 Empty Activity và đặt tên là TopMusicActivity. Ở file xml cần sửa như sau:



Do sử dụng RecyclerView nên chúng em tạo thêm 1 layout hiển thị listitem trên recyclerView này. Tạo một layout tên songs_top.xml và sửa lại như sau:



Về cơ bản hai danh sách bài hát này hiển thị như nhau, nên chỉ cần tạo 1 layout là songs_top để hiện thông tin. Ở file TopMusicActivity cần phải truyền MusicAdapter để có thể hiển thị dữ liệu. Và tạo 2 biến lấy dữ liệu từ adapter này và gọi dữ liệu từ hàm getView() - lấy top 10 bài hát có view cao nhất và hàm getRating() - lấy top 10 bài hát có lượng rating cao nhất.

```
class TopMusicActivity : AppCompatActivity() {
    val songs = ArrayList<Song>()
    val songviews = ArrayList<Song>()
    var adapterQuickView: MusicAdapters ?= null
    var adapterViewSong : MusicAdapters ?=null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_top_music)
```

```

    adapterQuickView = MusicAdapters( context: this, R.layout.songs_top, songs)
    songList.adapter = adapterQuickView

    Song.getRating()
        .addOnSuccessListener { querySnapshot ->
            val documents = querySnapshot.documents
            for (doc in documents) {
                val songa = Song(doc)
                songs.add(songa)
                adapterQuickView!!.notifyDataSetChanged()
            }
        }
        .addOnFailureListener { e ->
            Log.e(
                tag: "",
                msg: "fromCloudFirestore: Error loading ContactInfo data from Firestore - " + e.message
            );
        };
    }

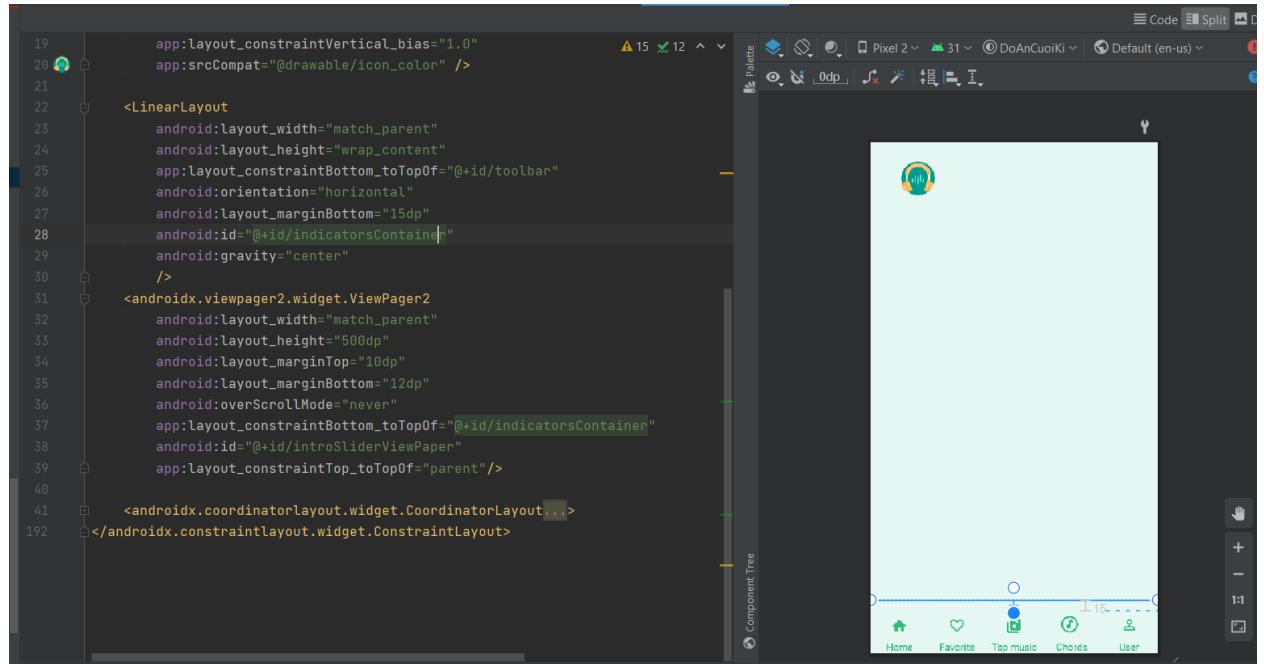
    adapterViewSong = MusicAdapters( context: this, R.layout.songs_top, songviews)
    songList_view.adapter = adapterViewSong
    Song.getView()
        .addOnSuccessListener { querySnapshot ->
            val documents = querySnapshot.documents
            for (doc in documents) {
                val songa = Song(doc)
                songviews.add(songa)
                adapterViewSong!!.notifyDataSetChanged()
            }
        }
        .addOnFailureListener { e ->
            Log.e(
                tag: "",
                msg: "fromCloudFirestore: Error loading ContactInfo data from Firestore - " + e.message
            );
        };
    }
}

```

10. Chức năng xem các hợp âm có sẵn

Đầu tiên, tạo 1 Emty Activity và đặt tên là ChordActivity. Ý tưởng của phần này dựa trên màn hình giới thiệu của ứng dụng. Hay nói cách khác là sử dụng Viewpager2 và tạo 1 file slide_item_chord để hiện layout của từng màn hình. Để có thể hiện thông tin của các hợp âm cần truyền dữ liệu vào nên chúng em đã khởi tạo 1 object Chord lưu tên của hợp âm, và hình ảnh của hợp âm để người dùng có thể dễ nhìn để học theo thao tác và tập luyện đánh đàn.

Phần activity_chord.xml sửa lại như sau:



Ở ChordAdapter, sửa như sau:

```

class ChordAdapter(private val chord: List<Chord>): RecyclerView.Adapter<ChordAdapter.IntroSlideViewHolder>() {
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): IntroSlideViewHolder {
        return IntroSlideViewHolder(
            LayoutInflater.from(parent.context).inflate(
                R.layout.slide_item_chord,
                parent,
                attachToRoot: false
            )
        )
    }

    override fun onBindViewHolder(holder: IntroSlideViewHolder, position: Int) {
        holder.bind(chord[position])
    }
    override fun getItemCount(): Int {
        return chord.size
    }

    inner class IntroSlideViewHolder(view: View): RecyclerView.ViewHolder(view){
        private val textTitle = view.findViewById<TextView>(R.id.textTitle)
        private val imageIcon = view.findViewById<ImageView>(R.id.imageSlideIcon)
        fun bind(chord: Chord){
            textTitle.text = chord.title
            imageIcon.setImageResource(chord.icon)
        }
    }
}

```

Ở ChordActivity, truyền một list dữ liệu hợp âm có sẵn sau đó setupIndicator để hiện sự thay đổi chuyển giao giữa các màn hình bằng cách vuốt sang trái hoặc sang phải.

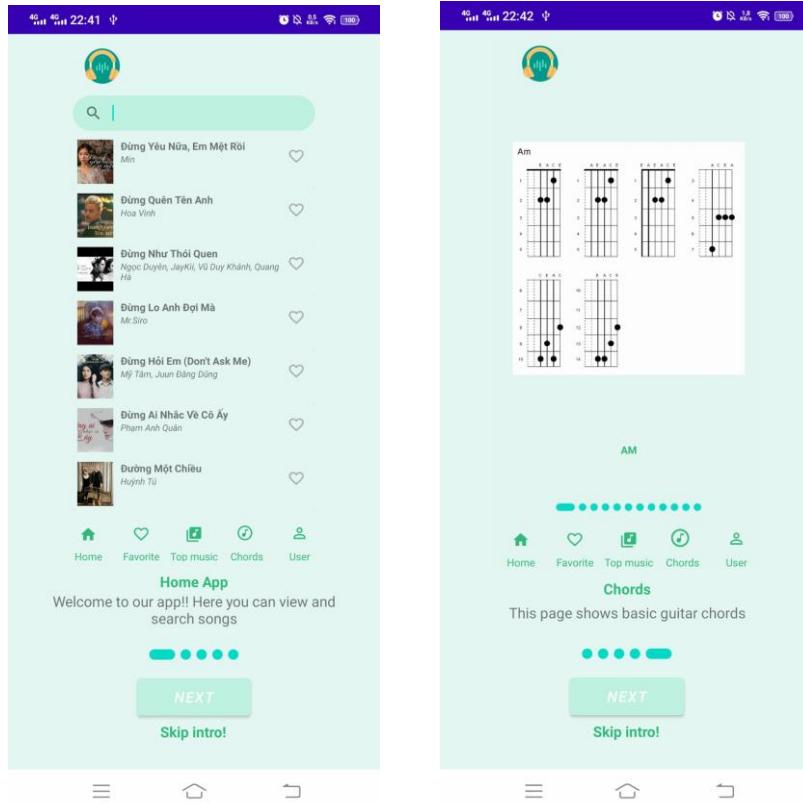
```
private fun setupIndicators(){
    val indicator = arrayOfNulls<ImageView>(chordAdapter.itemCount)
    val layoutParams: LinearLayout.LayoutParams =
        LinearLayout.LayoutParams(
            ViewGroup.LayoutParams.WRAP_CONTENT,
            ViewGroup.LayoutParams.WRAP_CONTENT
        )
    layoutParams.setMargins( left: 8, top: 0, right: 8, bottom: 0)
    for(i in indicator.indices){
        indicator[i] = ImageView(applicationContext)
        indicator[i].apply { this: ImageView?
            this?.setImageDrawable(
                ContextCompat.getDrawable(
                    applicationContext,
                    R.drawable.indicator_2
                )
            )
            this?.layoutParams = layoutParams
        }
        indicatorsContainer.addView(indicator[i])
    }
}
```

```
52     private fun setCurrentIndicator(index:Int){
53         val childCount = indicatorsContainer.childCount
54         for ( i in 0 until childCount){
55             val imageView = indicatorsContainer[i] as ImageView
56             if (i == index){
57                 imageView.setImageDrawable(
58                     ContextCompat.getDrawable(
59                         applicationContext,
60                         R.drawable.indicator
61                     )
62                 )
63             } else{
64                 imageView.setImageDrawable(
65                     ContextCompat.getDrawable(
66                         applicationContext,
67                         R.drawable.indicator_2
68                     )
69                 )
70             }
71         }
72     }
73 }
```

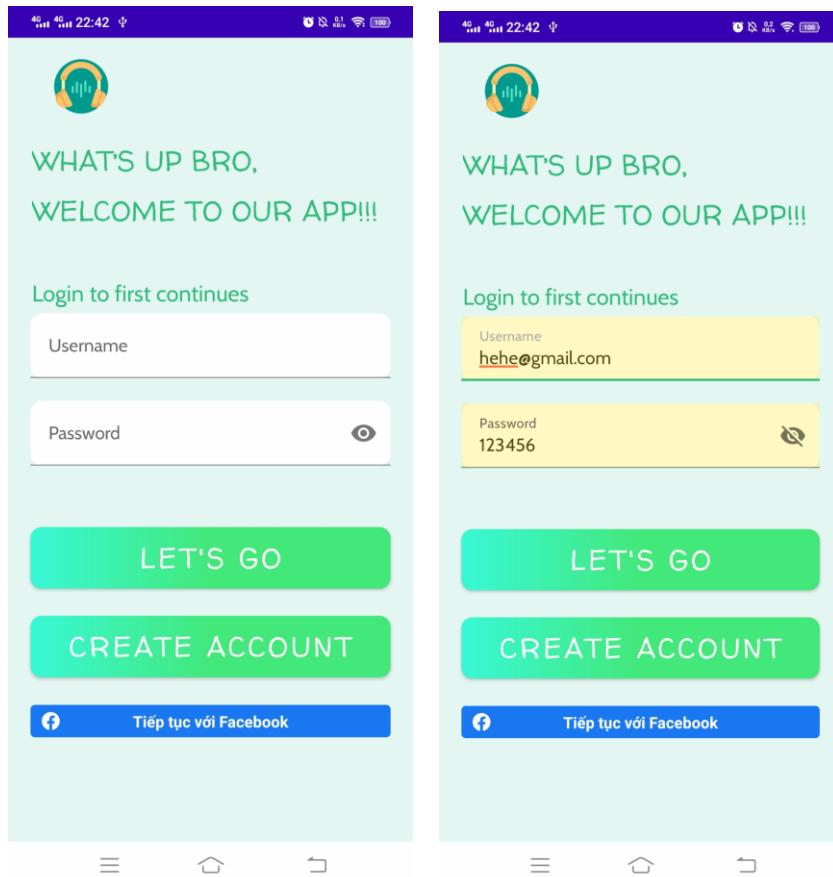
III. DEMO

Giao diện của ứng dụng:

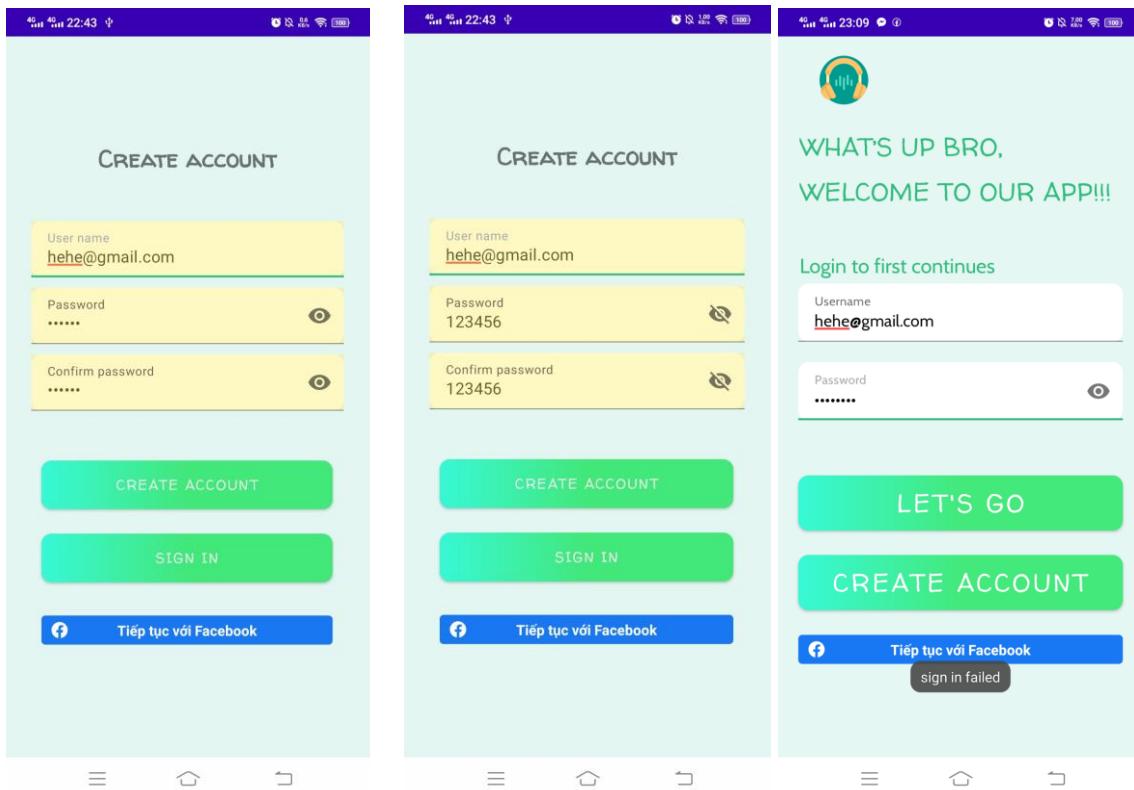
- Giao diện giới thiệu ứng dụng:



- Giao diện đăng nhập:



- Giao diện đăng ký:



- Hình ảnh đầu tiên nhìn thấy khi đăng nhập thành công:

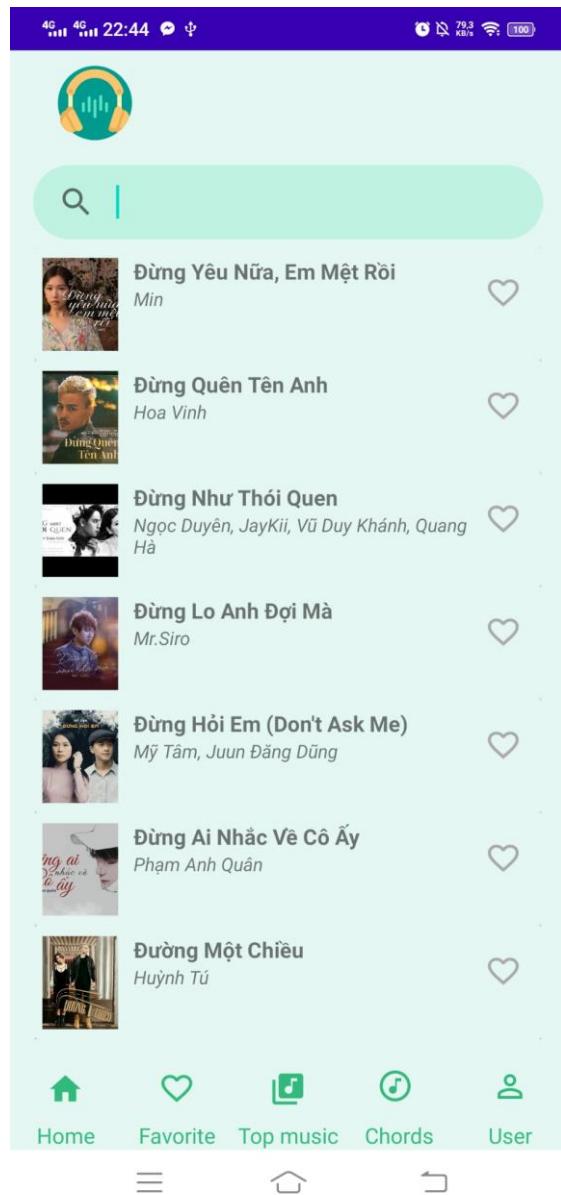


Loading Music.

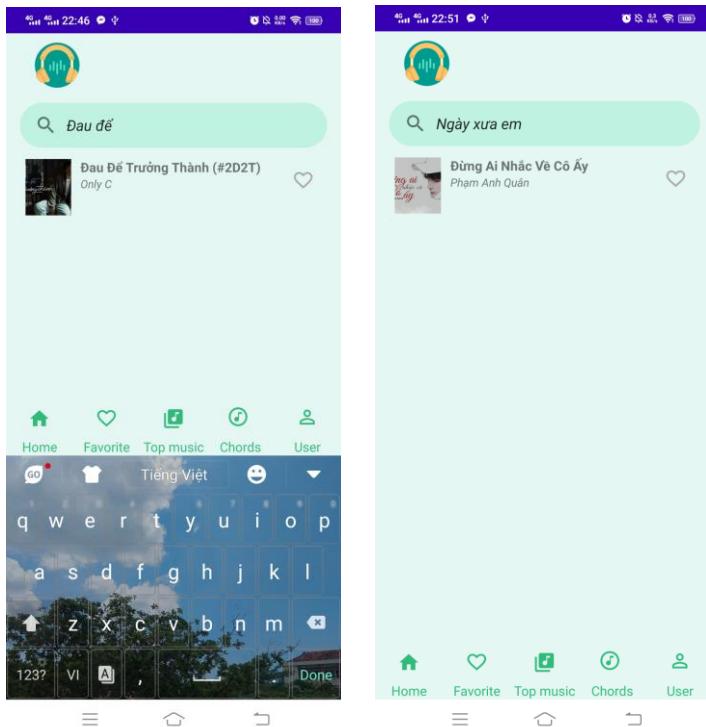
signed in successfully



- Trang chủ của ứng dụng:



- Giao diện khi tìm kiếm bài hát:



1. Lời Bài Hát Đừng Ai Nhắc Về Cô Ấy

Ngày xưa em nhớ không em?
Em hứa là:
"Ngày thành đôi sẽ không xa đâu anh à"
Rồi sau đó chẳng biết ai sai
Lỡ mất tương lai
Cứ thế xa nhau thôi
Vì đâu nồng nỗi đến hôm nay?
Điều gì ngăn li chuyện tình ta nhanh đến vậy?
Nặng nợ yêu thương nên anh cứ nhớ điều ngọt ngào nhất
Về một người về một quá khứ
[Chorus:]
Đừng ai nhắc đến em một lời
Để anh bớt chơi vơi khi nghĩ tới
Em đã xa anh mà anh còn lo lắng cho em?
Nụ cười cố giấu đôi mi buồn
Nhưng mơ thấy em nước mắt lại tuôn
Vòng tay yêu ớt này em đã không cần
Tự ôm nỗi nhớ trong im lặng
Tại sao nỡ buông tay anh thật nhanh
Giờ lẻ anh đã xóa nhòa niềm tin hóa đá
Thời gian trôi sao vết thương chẳng lành
Ngày hôm qua đã khác hôm nay
Chỉ là không yêu nhau thôi sau đau đớn vậy

- Giao diện hiển thị thông tin bài hát:

4G 4G 22:46 ⏱ 1.8 KB/s 100%

Đau Để Trưởng Thành (#2D2T)
Only C
405964.0 ★★★★★ 4.5

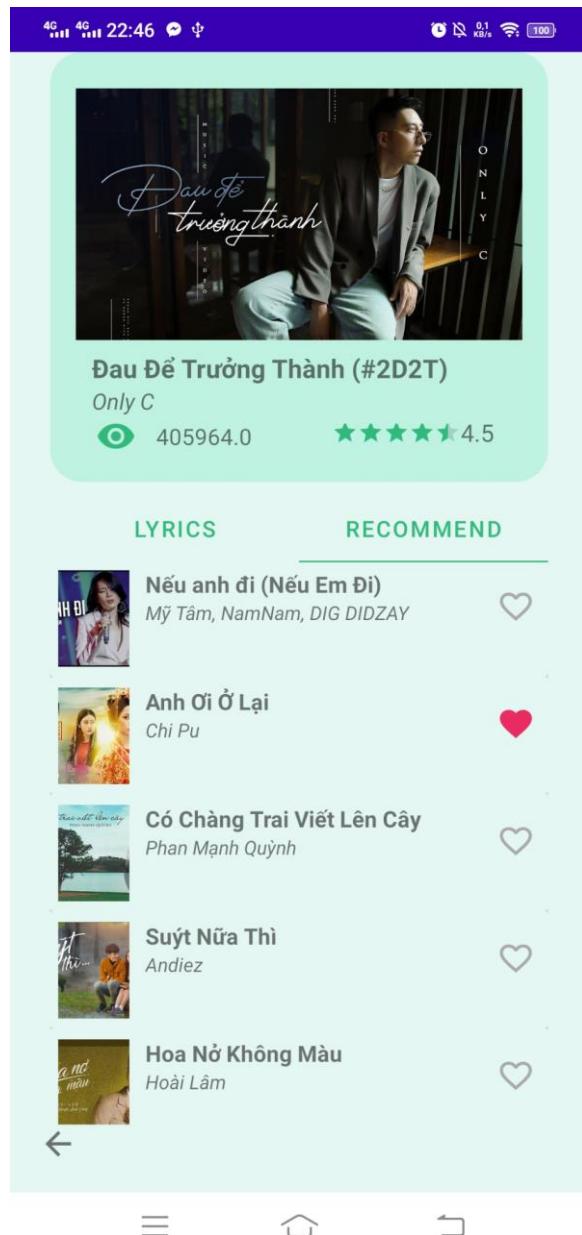
LYRICS **RECOMMEND**

[C]Yêu ai mà chẳng có [G]lúc buồn,
[Am]Thương ai rồi cũng đến [Em]lúc
buôngÙ thì [F]anh phải đứng dậy [G]thôi,
[Em]nhật ký cất đi [Am]rồi[Dm]Thôi nói ra
làm gì lại càng thêm [G]đau[C]Nếu quay
thời gian đến [G]lúc đầuĐể nói [Am]rằng
này chàng trai hãy [Em]nhớ choNgười con
[F]gái cô đơn lầm [G]đấy, [Em]tại sao không
nín [Am]lấy[Dm]Để mất đi một người chỉ vì
vô [G]tâmCó những [C]điều mà anh muốn
nói, không kịp [G]đâu đã quá muộn rồiCó
những [Am]điều mà anh từng ngó lơ, lại
[Em]là ước mơ bao ngườiLà tại [F]anh nỗi
[G]đau đi đến suốt [Em]đời, khi [Am]yêu
chỉ cần như [Dm]thế, Chia tay có khi làm
[G]mình lớn lênMỗi khi [C]buồn thì em
cứ khóc, khóc thật [G]to mỗi anh nghe
thôiMỗi khi [Am]đau niềm đau cũng xé đôi

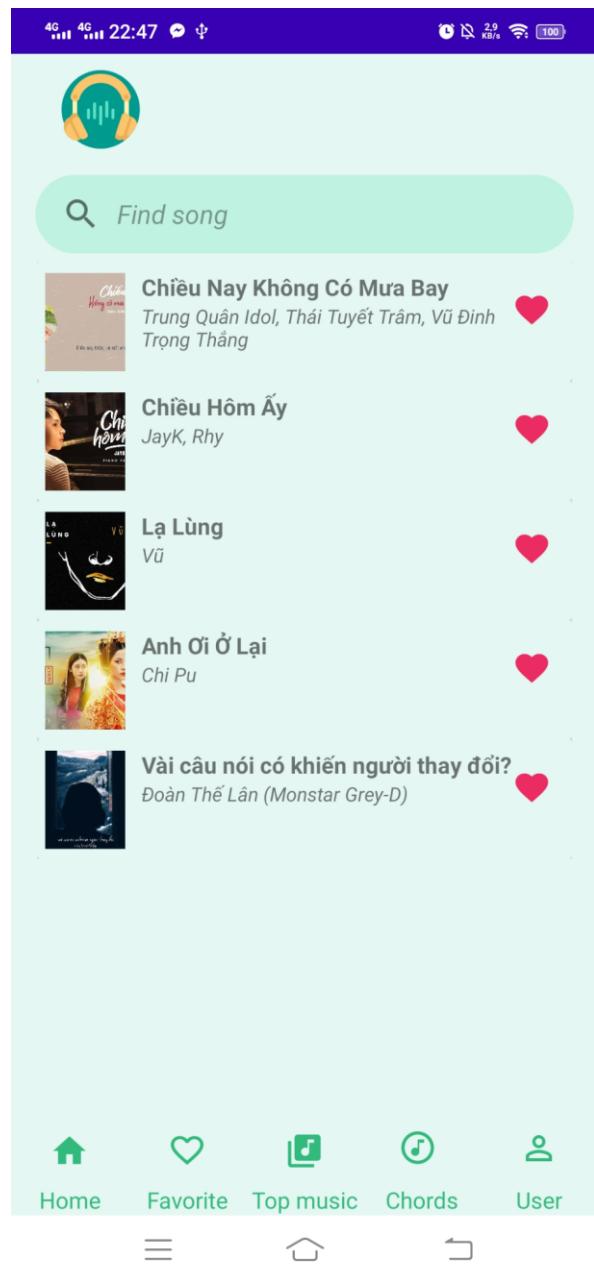
←

≡ ⌂ ⌂

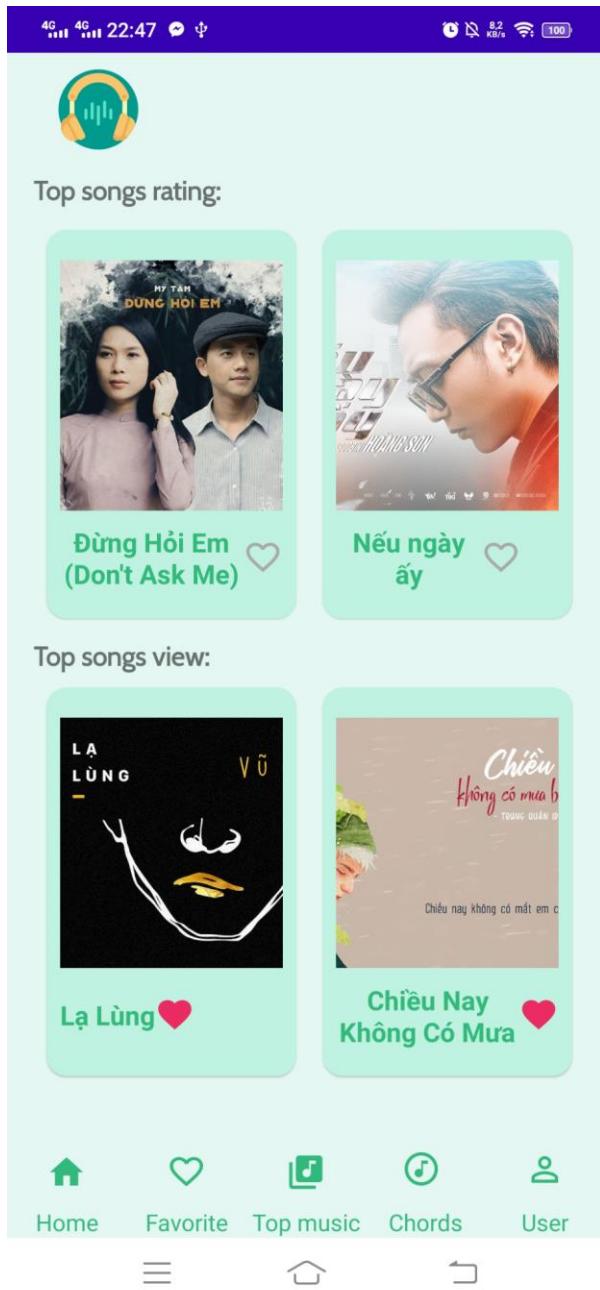
- Giao diện hiện các bài hát tương tự được đề xuất:



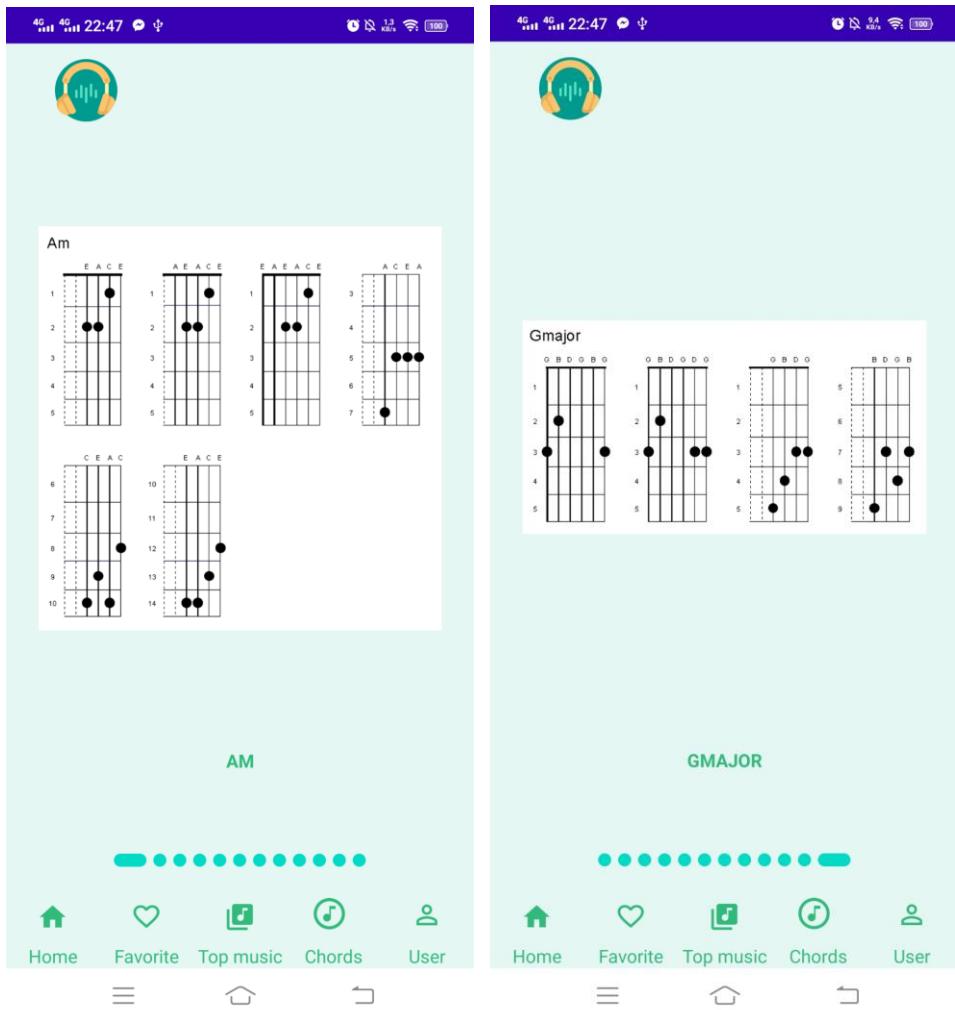
- Giao diện hiện các bài hát yêu thích:



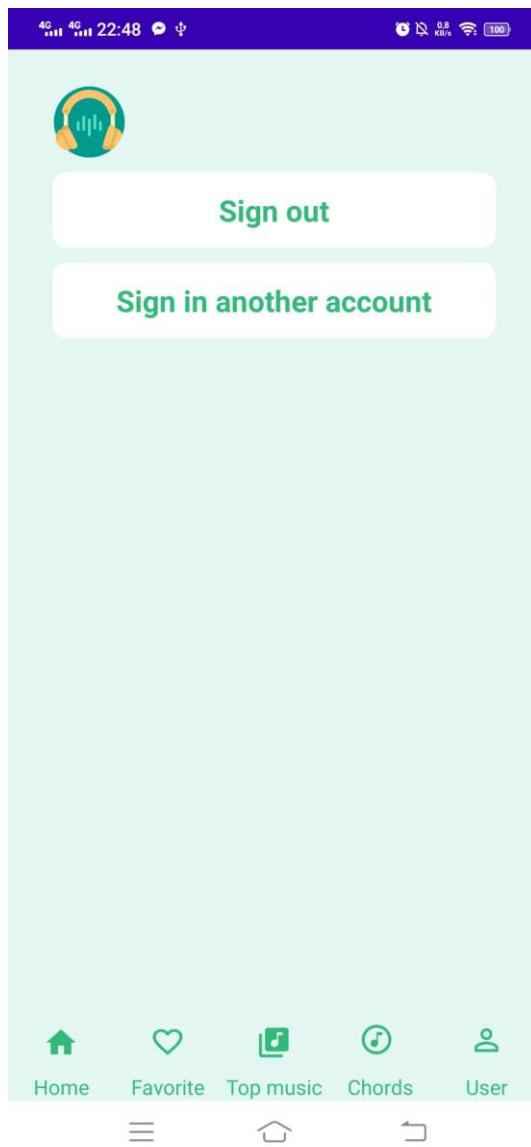
- Giao diện hiển thị các bài hát có lượng rating cao nhất hoặc lượt xem nhiều nhất(mỗi phần hiển thị 5 bài)



- Giao diện xem hợp âm chuẩn:



- Giao diện sign out:



IV. ƯU VÀ NHƯỢC ĐIỂM CỦA ỨNG DỤNG

1. Ưu điểm:

- + Giao diện dễ thương, thân thiện
- + Ứng dụng dễ thao tác, có thể thêm/xóa bài hát yêu thích bằng 1 cái click
- + Có thể gợi ý các bài hát tương tự bài người dùng đang xem (dựa vào chủ đề)
- + Hiển thị các bài hát có lượt xem cao và lượt đánh giá cao
- + Mỗi bài hát có hiển thị số lượng lượt xem và lượng rating

- + Có thể tìm kiếm bài hát dựa trên lời bài hát, tên bài hát và tên ca sĩ.

2. Nhược điểm:

- + Chưa xem được hợp âm 1 cách trực tiếp và chưa tự động cuộn trang, có thể gây bất tiện cho người chơi nhạc
- + Còn giới hạn người dùng
- + Cách search còn chưa tối ưu, độ phức tạp cao, dễ bị văng ứng dụng khi nhấn quá nhanh(ứng dụng load không kịp)

V. HƯỚNG PHÁT TRIỂN TRONG TƯƠNG LAI:

- Clean lại code nhằm dễ bảo trì và phát triển.
- Mở rộng data các bài hát sang nhiều thể loại khác nhau (trong ứng dụng chỉ Nhạc Trẻ).
- Tăng cường thiết kế lại giao diện nhằm gây ấn tượng cho người dùng(tạo hiệu ứng tự cuộn trang khi người dùng sử dụng để chơi nhạc cụ, xem được ảnh hợp âm trực tiếp trong lời bài hát)
- Chính sửa lại cách lưu trữ trạng thái yêu thích bài hát để có thể sử dụng cho nhiều người dùng.
- Tối ưu hóa khả năng search để có thể search trên lượng dữ liệu lớn.
- Xây dựng thêm các chức năng như Rating, Kho nhạc yêu thích, nghe nhạc trên app...

