

MUSIC ONLINE PLAYER APP

BY

**SOURAV PAUL
ID: 183-15-12018**

AND

**INZAMAM-UL-ISLAM
ID: 183-15-11986**

This Report Presented in Partial Fulfillment of the Requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Supervised By

Touhid Bhuiyan
Professor & Head
Department of CSE
Daffodil International University

Co-Supervised By

Md. Juel Mia
Assistant Professor
Department of CSE
Daffodil International University



DAFFODIL INTERNATIONAL UNIVERSITY

DHAKA, BANGLADESH

SEPTEMBER 2022

APPROVAL

This Project/internship titled **Music Online Player App**, submitted by Sourav Paul, ID No: 183-15-12018 and Inzamam-UI-Islam, ID No: 183-15-11986 to the Department of Computer Science and Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Engineering and approved as to its style and contents. The presentation has been held on 13th September, 2022.

BOARD OF EXAMINERS

Chairman



Dr. Touhid Bhuiyan (DTB)

Professor and Head

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

Internal Examiner



Subhenur Latif (SL)

Assistant Professor

Department of Computer Science and Engineering
Faculty of Science & Information Technology

Internal Examiner



Most. Hasna Hena (HH)

Assistant Professor

Department of Computer Science and Engineering
Faculty of Science & Information Technology
Daffodil International University

External Examiner



Dr. Mohammad Shorif Uddin

Professor

Department of Computer Science and Engineering
Jahangirnagar University

DECLARATION

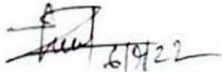
We hereby declare that, this project has been done by us under the supervision of **Touhid Bhuiyan, Professor & Head, Department of CSE Daffodil International University**. We also declare that neither this project nor any part of this project has been submitted elsewhere for award of any degree or diploma.

Supervised by:



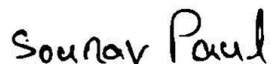
Touhid Bhuiyan
Professor & Head
Department of CSE
Daffodil International University

Co-Supervised by:



Md. Juel Mia
Assistant Professor
Department of CSE
Daffodil International University

Submitted by:



ID: 183-15-12018
Department of CSE
Daffodil International University



ID: 183-15-11986
Department of CSE
Daffodil International University

ACKNOWLEDGEMENT

First, we express our heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the final year project/internship successfully.

We really grateful and wish our profound our indebtedness to **Touhid Bhuiyan, Professor & Head**, Department of CSE Daffodil International University, Dhaka. Deep Knowledge & keen interest of our supervisor in the field of “*CSE*” to carry out this project. His endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

We would like to express our heartiest gratitude to **Sourav Paul, Inzamam-Ul-Islam & Head**, Department of CSE, for his kind help to finish our project and also to other faculty member and the staff of CSE department of Daffodil International University.

We would like to thank our entire course mate in Daffodil International University, who took part in this discuss while completing the course work.

Finally, we must acknowledge with due respect the constant support and patients of our parents.

ABSTRACT

In this paper, we will discuss the usage, approach & implementation of our android application of audio player that plays audio formatted files from an external database server. This app will focus on the extended availability of music streaming in our country. In our country we see many lacking of music usage for entertainment purposes. Although there many alternatives in many developed countries, we're still far behind them in terms of music discoverability. So, we're taking different types of approach to find the definitive solution to decrease the problems that we face in this music populated world. We may seem to be far behind but when we can know the necessity. Nowadays, there are many music lovers around the world right now, if we can solve the issue of music in our country, we will also be able to reach our goal for them. We are also trying to use easy navigation process for accessing the music audios as day by day its demand is also increasing rapidly as time goes on. As we will try to reach our goal with this view in mind, we'll also try to increase our vison for finding the best solution for our goal & enhance the purpose of the application for best user experience.

TABLE OF CONTENTS

CONTENTS	PAGE
Board of examiners	i
Declaration	ii
Acknowledgments	iii
Abstract	iv
CHAPTER	
CHAPTER 1: INTRODUCTION	1-3
1.1 Introduction	1
1.2 Motivation	1-2
1.3 Objective	2
1.4 Expected Outcomes	2
1.5 Report Layout	3
CHAPTER 2: BACKGROUND STUDY	4-7
2.1 Terminologies	4
2.2 Related Works	4-6
2.3 Scope of the Problem	6
2.4 Challenges & Copyright Issues	6-7

CHAPTER 3: REQUIREMENTS SPECIFICATION	8-9
3.1 Design Requirement	8
3.2 Libraries	8-9
CHAPTER 4: DESIGN SPECIFICATION	10-15
4.1 Android Manifest Specs	10
4.2 UI Design	11-17
CHAPTER 5: IMPLEMENTATION AND TESTING	18-29
5.1 Implementation of Functionalities	18-29
5.2 Implementation of Database	30
5.3 Test Results & Reports	31
CHAPTER 6: IMPACT ON SOCIETY, ENVIRONMENT, AND SUSTAINABILITY	32
6.1 Impact on Society	32
6.2 Ethical Aspects	32
6.3 Sustainability Plan	32
CHAPTER 7: CONCLUSION & FUTURE SCOPE	33-34
7.1 Discussion and Conclusion	33
7.2 Scope for Further Developments	34

LIST OF FIGURES

FIGURES	PAGE NO
Figure 2.1.1: Spotify App	4
Figure 2.1.2: Gaana App	5
Figure 2.1.3: SoundCloud App	5
Figure 2.1.4: eMusic App	6
Figure 4.1.1: Android Manifest Specs	10
Figure 4.1.2: Online Fragment of the App	11
Figure 4.1.3: Local Fragment of the App	12
Figure 4.1.4: Album Fragment of the App	13
Figure 4.1.5: Navigation Drawer of the App	14
Figure 4.1.6: Now Playing Layout	15
Figure 4.1.7: Listview of the Songs on the library	16
Figure 4.1.8: Album Fragment Layout	16
Figure 4.1.9: Album View section	17
Figure 4.2.0: Sharing Layout	17
Figure 5.1.1: Methods in Mainactivity class	18-19
Figure 5.1.2: Storage Permission Function	19
Figure 5.1.3: Play & Pause buttons in Mainactivity	20
Figure 5.1.4: Playback function	20
Figure 5.1.5: Methods in AlbumActivities Class	21
Figure 5.1.6: Variables in Album Layout	21
Figure 5.1.7: Using Glide in Albumview	22
Figure 5.1.8: Methods in PlayerActivity class	22
Figure 5.1.9: Pause & Previous function in Player	23
Figure 5.2.0: Next function in Player	23
Figure 5.2.1: Play function in Player	24
Figure 5.2.2: Methods in AlbumAdapter	24
Figure 5.2.3: Variables for ViewHolder	25

Figure 5.2.4: Variables in AlbumAdapter	25
Figure 5.2.5: Methods in SongAdapter	26
Figure 5.2.6: Shuffle Function in SongAdapter	26
Figure 5.2.7: Albumart in SongAdapter	27
Figure 5.2.8: Methods & Variables in SongsFragment	27
Figure 5.2.9: Methods in AlbumsFragment	28
Figure 5.3.0: Variables in AlbumsFragment	28
Figure 5.3.1: OnlineFragment with Adapter	29
Figure 5.3.2: JC Player in Mainactivity	29
Figure 5.3.3: Firebase Database Storage Library	30
Figure 5.3.4: Showing the Source of the Audio from the Database	30

CHAPTER 1: Introduction

1.1 Introduction

Music is all about entertaining the listener's mind & set the mood at ease for the listener. Since from the beginning music has always done the same thing that is setting the mind at ease. As technologies have developed, music has gone through many evolutions. Firstly, there was vinyl plates, electromagnetic charged stripes & CDs. After that we're now using digital formatted audio files to our collection. In today's standard we're now using hundreds of thousands of music audios online or even string in song libraries. Now in this digital world, navigating various types of music content has been made easy for user experience purposes. Day by day the numbers are increasing. There are many genres of music i.e., Rock, Pop, Retro, Metal, Heavy Metal, Jazz, Electronic music, Hip-hop, R&B etc. We now have many digital audio players, video players & even both at once. These players are used for playing the music & enjoying them in different fashions. But every other player apps have common objectives that playing the audio music. So, we're now creating an app that will play any music that is stored in a database storage & play through that in online environment.

Our music player allows user to listen to our audio form the audio storage database that we created & play them through online connectivity. This player app will show the user a collection of audio formatted files in list-view & then the user can select any desired audio file from the collection & play it.

1.2 Motivation

We have seen many apps in other countries that performs well in their music industrial form. Those apps are the main focus of their music platform's promotions. Those app's usage enables them to count the numbers of their music playing that counts towards various charts. So, we're trying to do the same but in our taste.

This is a music player for Android mobile devices. Usually in android devices there can be many media files & various apps for those media files. But the popular streaming apps

like Spotify, Deezer, SoundCloud, Shazam etc. uses many resources from the devices. But this app is very light resourced app unlike any other apps like previously mentioned. Nowadays, the popular apps uses so much features & functionalities that some time it hinders the memory systems from the usage of the devices. This audio player is very user friendly. The UI is very simple to navigate. Although the app is very limited in terms of features, but it has some good features. We also can try to develop the app further in future.

The overall motive for us was to create an app that will act as an alternative to stream music from the external storage of database in android environment that will access through internet. We've seen that in our country the music industry is not on par with western culture in terms of services. The popular apps are being used for the algorithms to work properly. This is the first reason that we're taking in consideration for the creation of the application. We think our creation will be a test subject for the things that will come afterwards. If the solution that we're creating is well received our plans will go smoothly. Our plan will enhance the user experience in our own creative way.

1.3 Objective

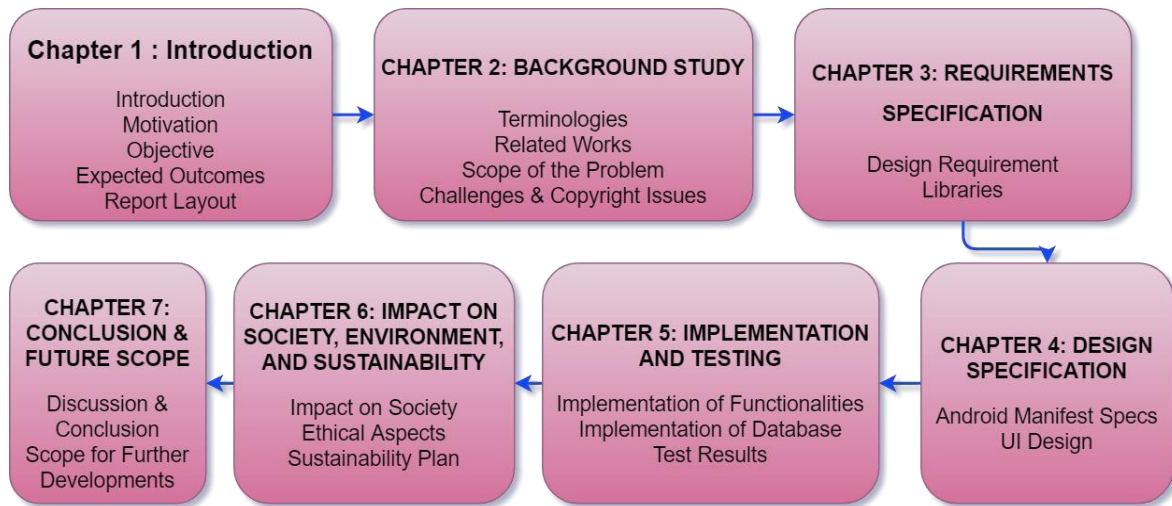
The main objective of our mobile application is:

- To be able to stream audio files that are stored in a database storage & access them through our app & play it on the go.
- With online section we're also aiming to add offline section of the audio files that will be able to play stored files in the device.

1.4 Expected Outcomes

- It will use online connectivity for streaming services
- App will have a very user-friendly UI that will make the app very reliable for constant usage
- App will be able to play audio file contents from the device
- App will contain large library of Bengali songs
- Music lovers will benefit from the app

1.5 Report Layout



Chapter 2: BACKGROUND STUDY

2.1 Terminologies

Firstly, we need to discuss that no matter how far we're taking the app for development, we will always try to make the ground concepts always intact. So, no matter what complexity of concepts and methods we use in the future we'll always try to set our mind in one objective only. Our approach is very simple. In the concept of our application, there's a music player in our mind that will play music. Naturally, what we're aiming to do is that those music files will be stored in a database storage & play those through online connectivity.

2.2 Related Works: Similar Apps

Spotify

Spotify is well known streaming app worldwide. It's the most popular app as a music streaming service provider. It provides digitally copyright restricted music & podcasts.

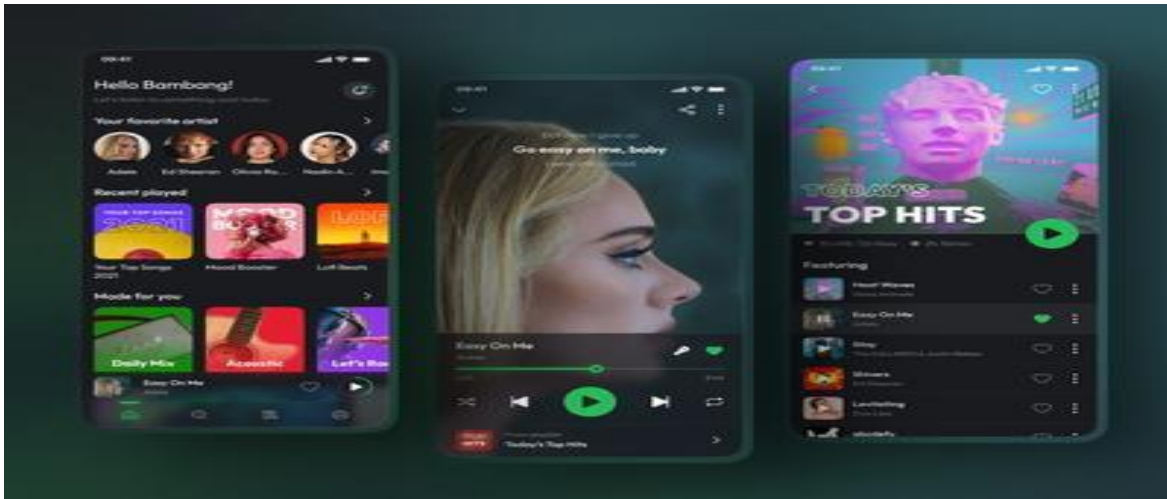


Figure 2.1.1: Spotify App

Gaana

Gaana is an Indian streaming app. This app's all Indian content catalog is available for users around the world. This app's all Indian content catalog is available for users around the world.

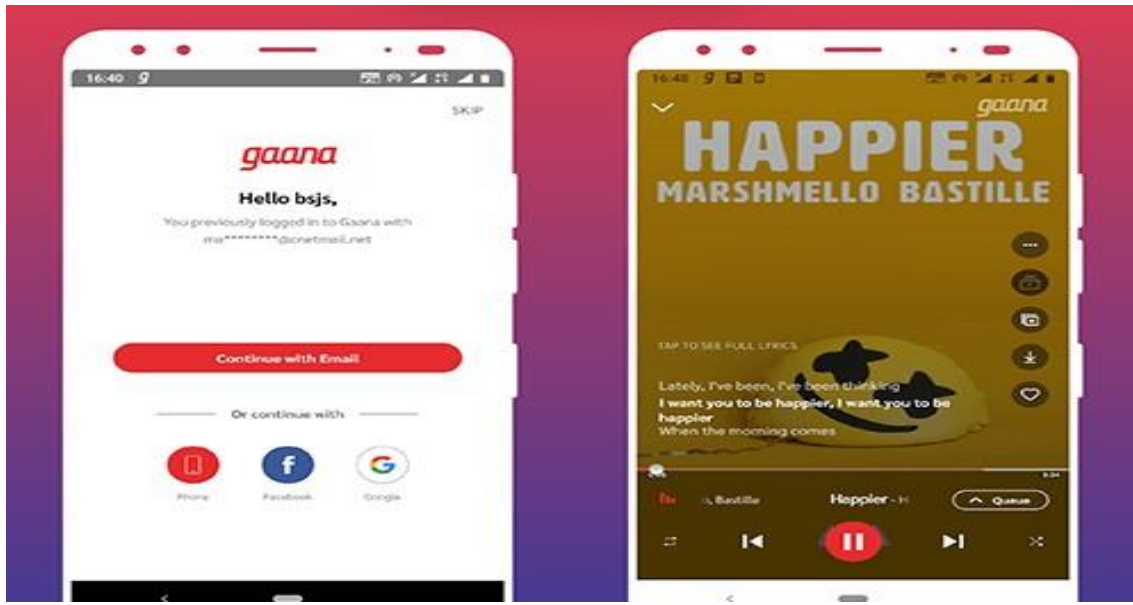


Figure 2.1.2: Gaana App

SoundCloud

SoundCloud is an application for sharing musical content. Users are able to upload their own audio & music contents on this platform. Users are able to stream those audio contents on this app. It helps new artists to make music to spread their music's visibility.

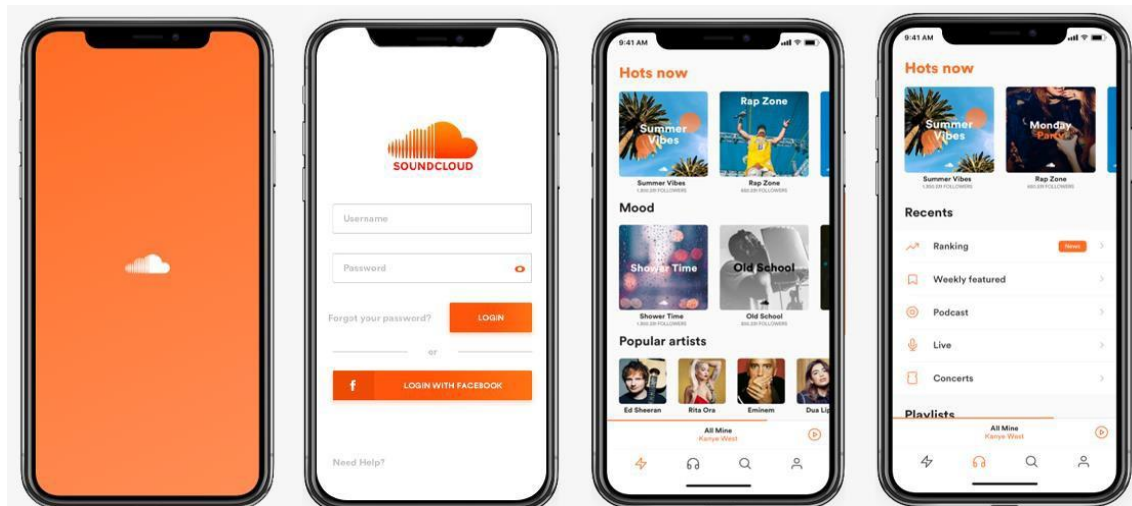


Figure 2.1.3: SoundCloud App

eMUSIC

emusic is an app of online music & audiobook type store. It has both free & subscription streaming option. Subscriber users can download & listen to unlimited number of MP3 files, but free users can download a fixed number of MP3 audio files.

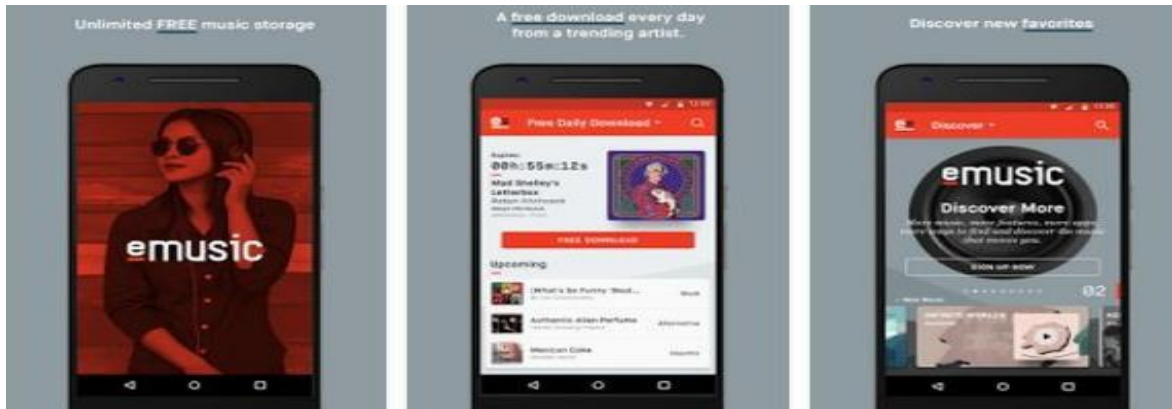


Figure 2.1.4: eMusic App

These apps are the most trending apps on today's music app areas. They use large libraries & database of their own to stream the songs. They're mostly providing free music.

2.3 Scope of the Problem:

Music is not mostly free in the current modern world. So, our plan is to make good progress for our app. In today's standard apps have reached highest level in terms of creativity. In general, we think that music library management & app's vision will have to be connected.

2.4 Challenges & Copyright Issues:

- Building the best easy user interface for the app
- Featuring app with the latest design of development as we progress
- Optimizing the app will have to perfect
- Setting a good number of functionalities for the ease of access
- Sharing the app for vast usages for the large visibility of the app
- **Copyright issues:**

So, by our country's law in copyright, it holds like "Under section 15 of the 2000 Act, copyright protection is conferred on original¹⁰ literary works, dramatic works, musical works, artistic works, cinematograph films and sound recording. It extends to the computer program also. Copyright refers to a bundle of exclusive rights vested in the

owner of copyright.” We are working related to this issue to identify and for now we are using some Non copyright music in our streaming app.

How we plan to deal with it:

1. It’s a starting app which was created by us for free, and we are hoping to make more changes to it.
2. Copyrighted music won’t play here for now.
3. We are on the process to make sure that we can add some of copyrighted music.
4. On the conversation with companies and now if we plan for something big, they already to help us over that.
5. The songs will be legal and it will take some times to prevail all the issues.
6. With the market behaves we will also conduct to release some originals things like podcast, Audiobook with different stories for our people.
7. This app will be very interesting asset for our country and we will make sure to work for it in longer time.

Chapter 3: REQUIREMENTS SPECIFICATIONS

Here we will share the tools & firmware used for the development of the app.

3.1 Design Requirements:

App's Functional language	Java
Design's Language	XML
Framework	Android
Minimum SDK Requirement	Android 4.0.3 (API 15)
Maximum SDK Requirement	Android 11 (API 30)
Supported Language	English
Supported DPI's	120dpi, 160dpi, 240dpi, 320dpi, 480dpi, 640dpi
App UI Layout	Material layout
Interface Design's Language	XML
Framework	Android
Supported Audio Formats	Any Music Format
Supported CPU Of Android	x86, ARMv7, ARMv8
Usage Of Layouts	Constraint, Linear & Relative Layout
Usage Of Google services	YES
Usage Of Firebase	YES
Supported Audio Formats	Any Music Format

3.2 Libraries:

JC-Player: A simple audio player for Android that you can plugin to your apps quickly get audio playback working. We used this player for our online playback of the app

Firebase: It is the essential platform used by android studio for the backend app development

Design Support Library: The Design package provides APIs to add material design components and patterns for the app

Recycler-View: A flexible view of the app for providing a limited window into a large data set

Glide: An image loading and caching library for Android applications, focused on smooth scrolling

Palette: A helper class to extract prominent colors from an image to the layout of the app

Chapter 4: Design Specifications

Here we will share the tools & firmware used for the development of the app.

4.1 Android Manifest Specs:

Firstly, we have the AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mediaplayer">
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <application
        android:name=".notification.NoificationCenter"
        android:allowBackup="true"
        android:icon="@drawable/applogo"
        android:label="@string/EXtremeMusicPlayerOnline"
        android:roundIcon="@drawable/applogo"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        tools:replace="android:label">
        <activity
            android:name=".activities.PlayerActivity"
            android:theme="@style/Theme.AppCompat.NoActionBar" />
        <activity android:name=".activities.AlbumActivity"/>
        <activity android:name=".activities.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Figure 4.1.1: Android Manifest Specs

Here we can see the app getting the user permission of external storage of the device that it's being installed on and also the use of the internet for the online connectivity.

4.2 UI Design:

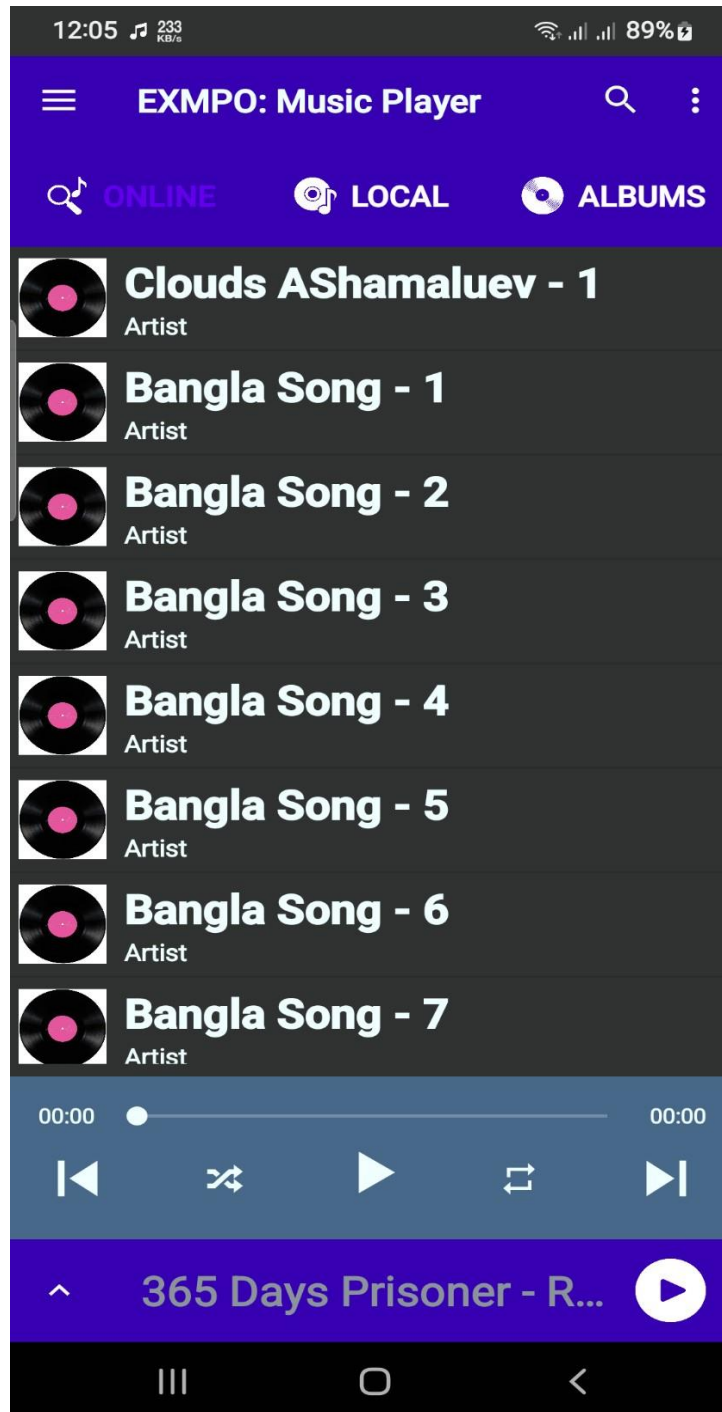


Figure 4.1.2: Online Fragment of the App

Here, we see the Online Fragment section of the app

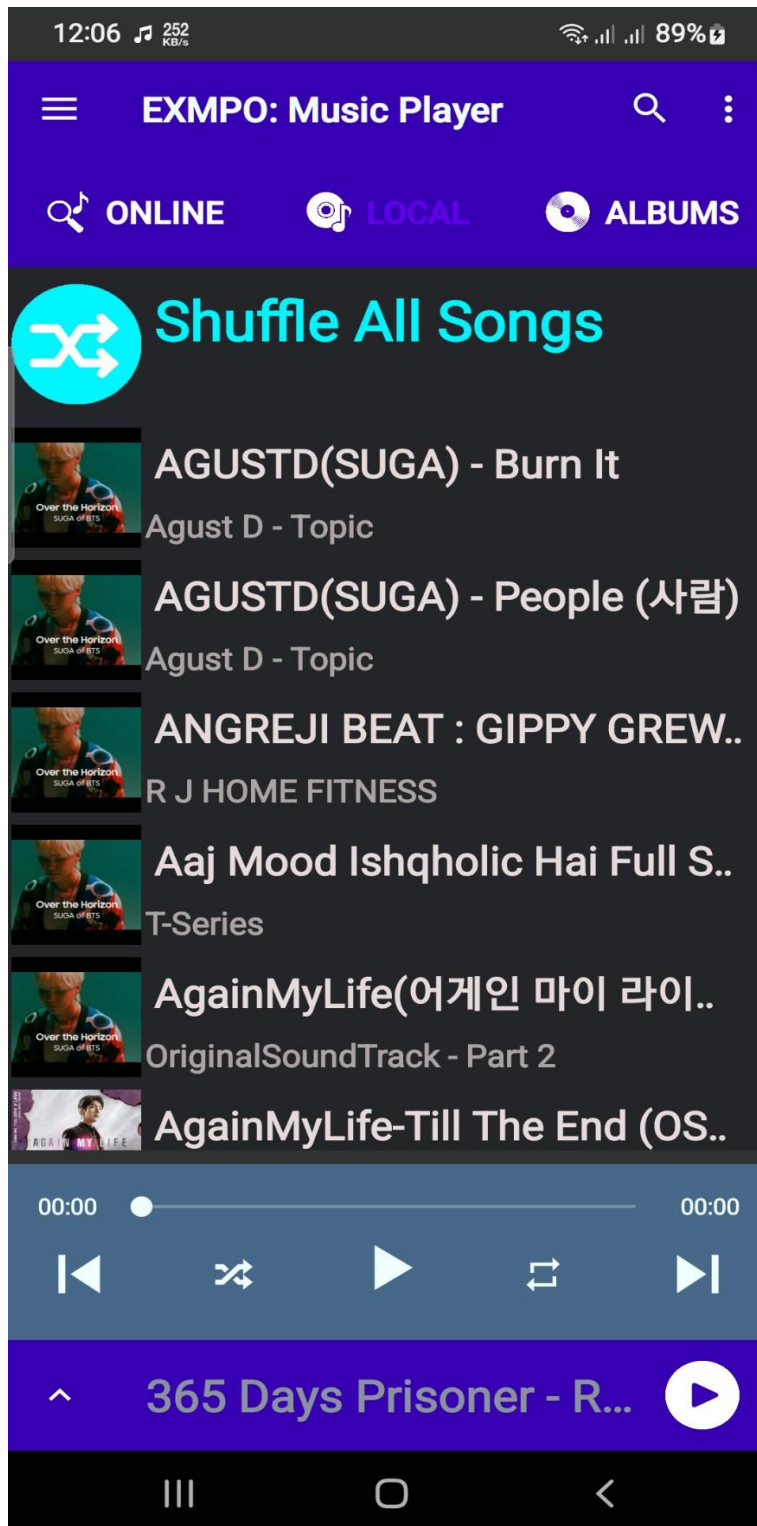


Figure 4.1.3: Local Fragment of the App

Here, we see the Local Fragment section of the app

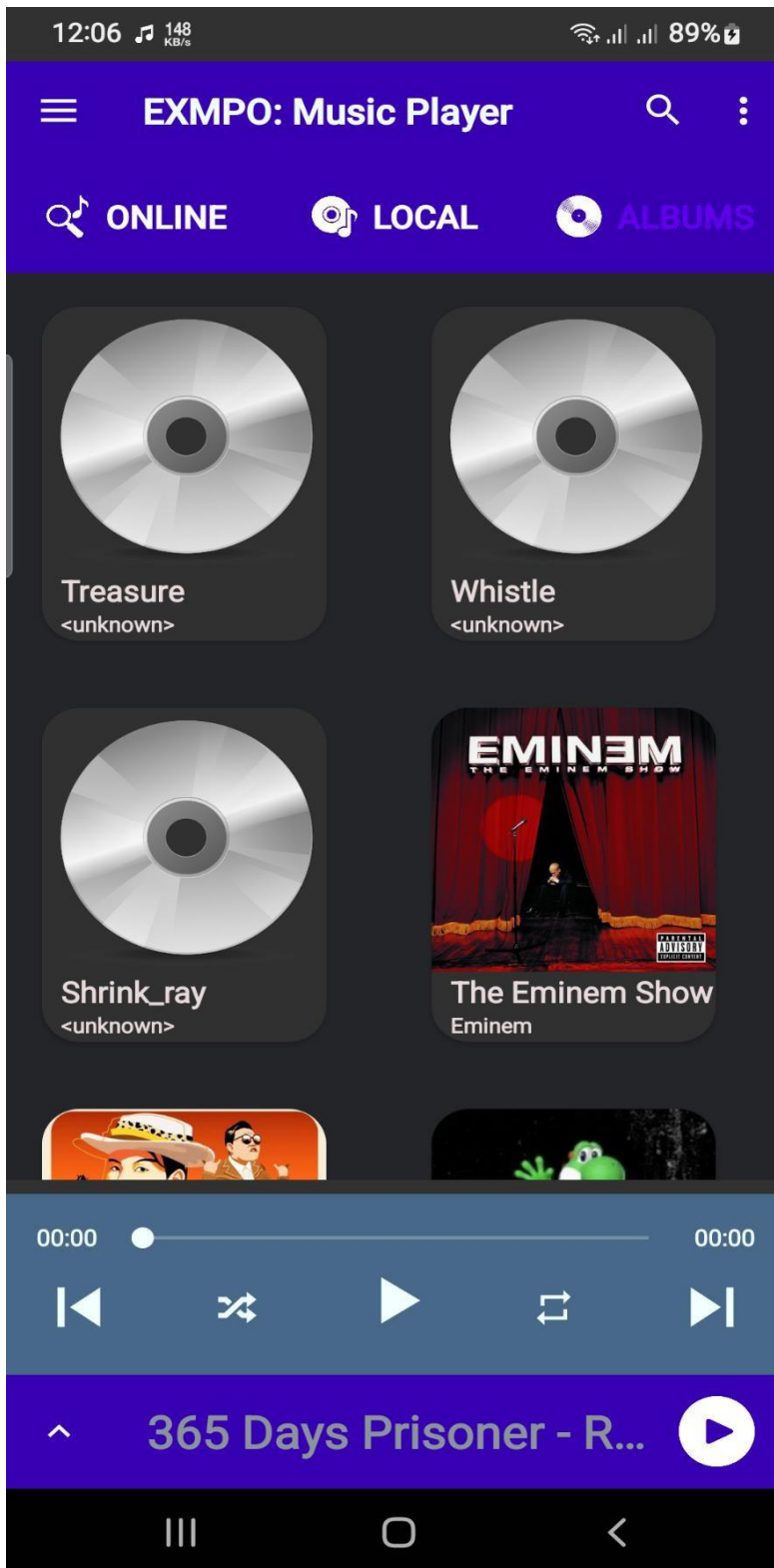


Figure 4.1.4: Album Fragment of the App

Here, we see the Album Fragment section of the app

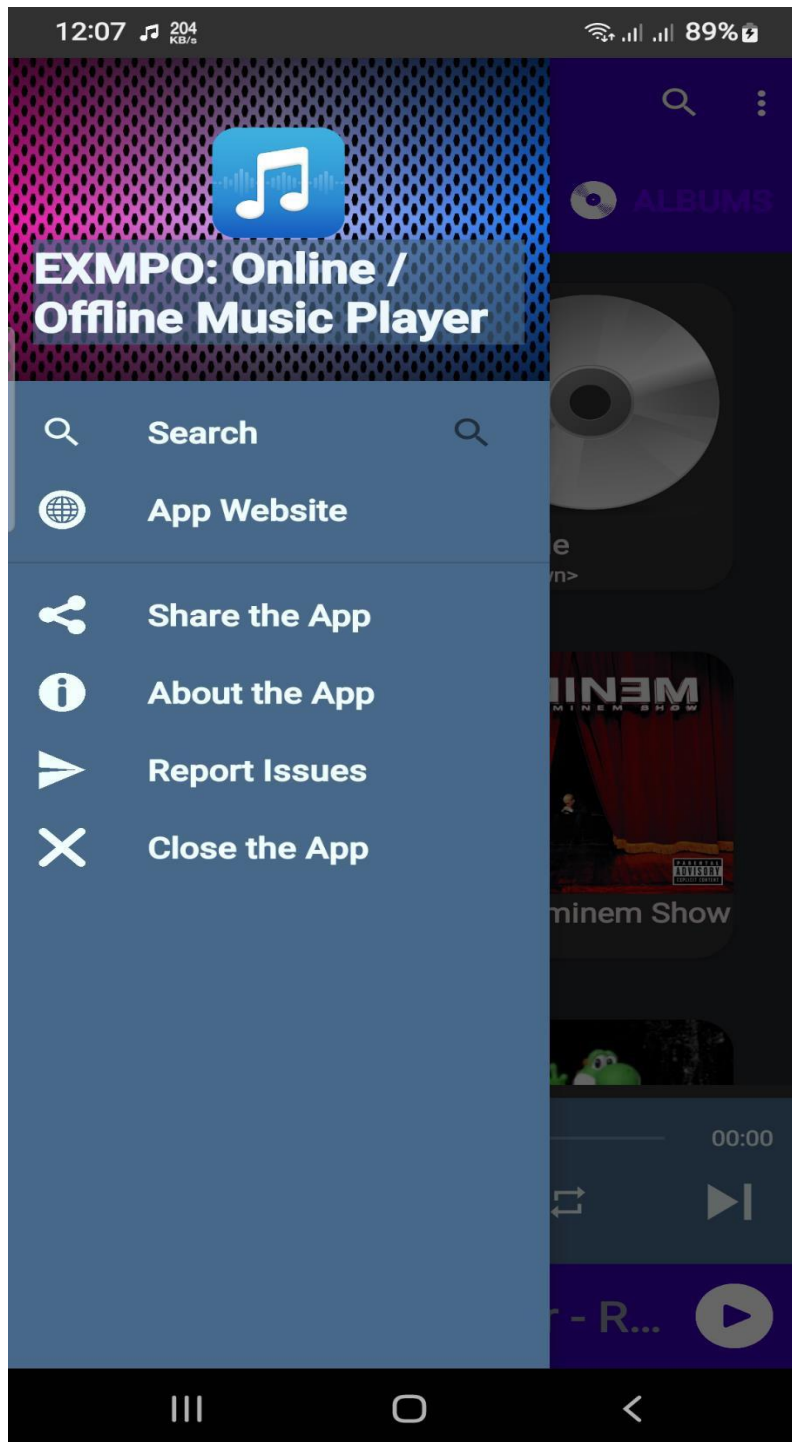


Figure 4.1.5: Navigation Drawer of the App

Here we can see that we have a Navigation drawer which has “Search”, “App Website”, “Share the App”, “About the App”, “Report Issues” & “Close the App” actions.

Figure: Now Playing view page & structure

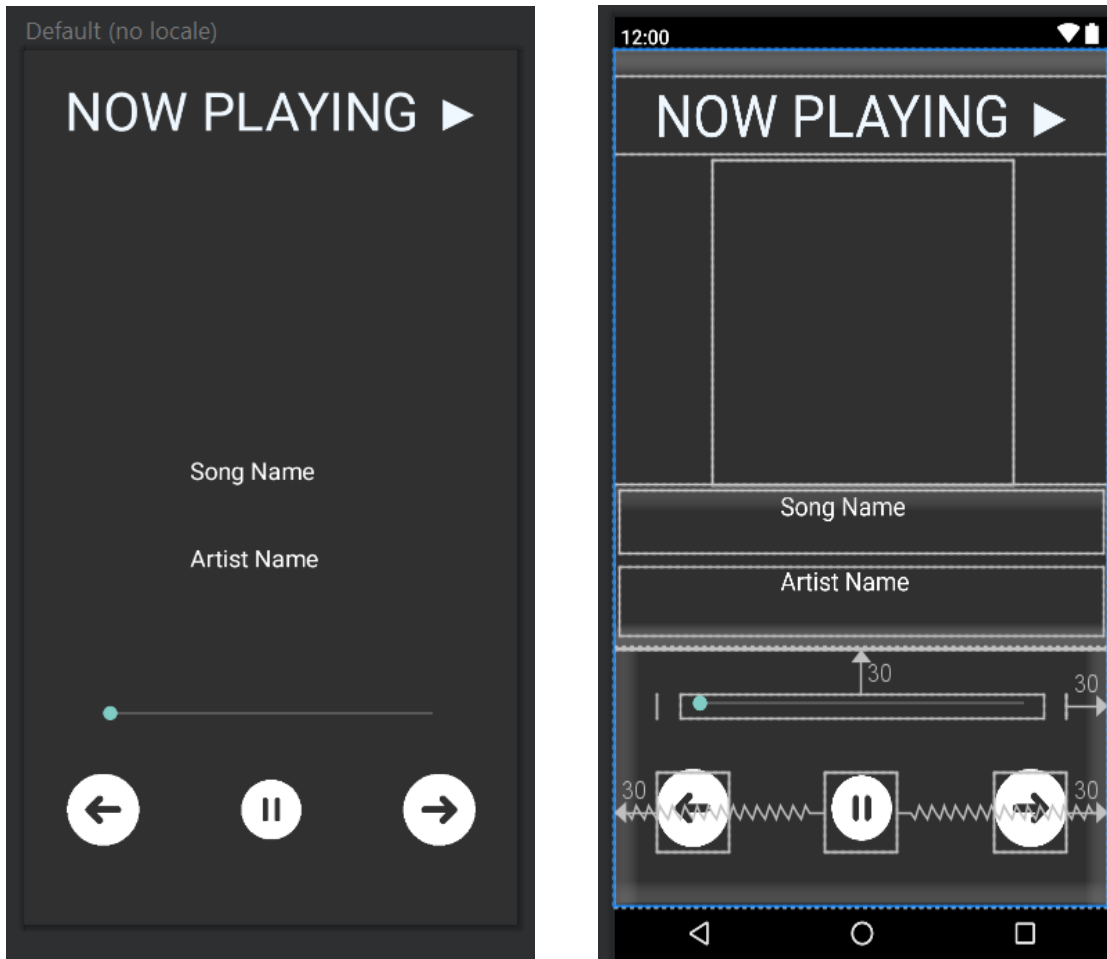


Figure 4.1.6: Now Playing Layout

In this Now playing section we see the Now Playing text at the top of the screen, in middle section we see an adapter view that is the album art of the song/audio file playing the player. The retrieves the meta data & tags from the file & the adapter gets the album to show on the middle screen of the now playing layout. We can see the “Song Name” & “Artist Name” then the seek bar of the audio file, Pause/Play/Next/Previous buttons.

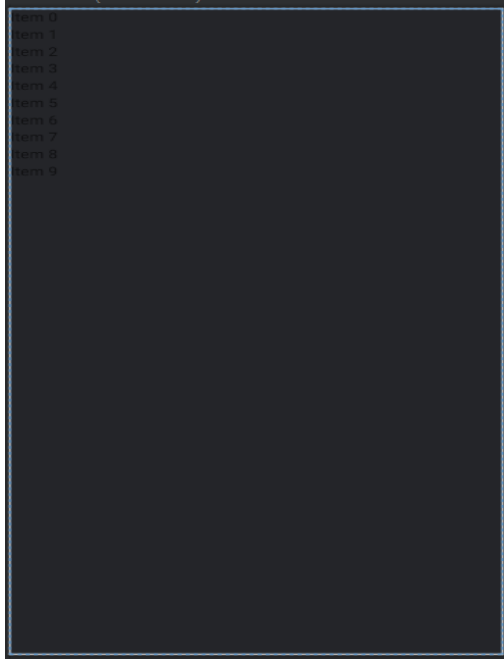


Figure 4.1.7: Listview of the Songs on the library (Both in Explore & Song Library)
Here we can see the Item1, Item2, Item3..... and so on for the song library. Same thing will appear for explore music section where it'll generate the item song file from the firebase storage & display it like this.



Figure 4.1.8: Album Fragment Layout

Here in this album fragment layout section, we'll see the Song cover from the audio file on the left top section of the layout. Then on the right we'll see song name & artist name of the file. These data are from the tag of the audio files that selected.



Figure 4.1.9: Album View section

Here we'll see the albums one by one from the top in Grid-view from the layout & display the cover art, song name, artist name from the file to the screen.

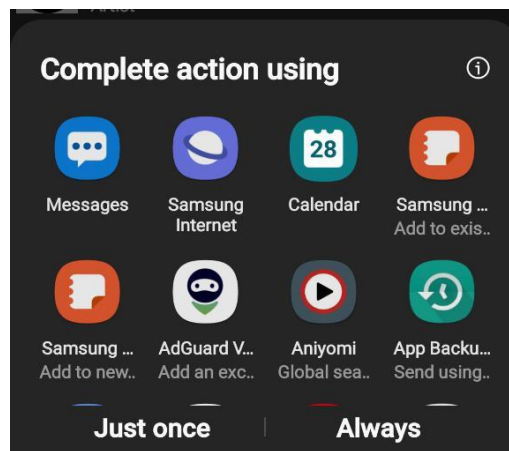


Figure 4.2.0: Sharing Layout

Here users will be able to share the app at different social platforms of their choice.

CHAPTER 5: IMPLEMENTATION AND TESTING

5.1 Implementation of Functionalities:

There are five sections for java functions of our app:

1. Activities
2. Adapters
3. Fragments
4. Models
5. Notifications

Before the functions showing we'd like share that these are always the most up-to date functions & we're showing the only offline parts and services. We're still working on the online services functions. We'll show the snippets of the java code parts.

Activities are in section (3 parts)

Mainactivity.java

```
package com.example.mediaplayer.activities;

import ...

public class MainActivity extends AppCompatActivity {
    private int Storage_Permission_code=1;
    private static final String TAG = "MainActivity";
    DataReading dataReading;
    protected static MainActivity instance;
    private byte arts[];

    private HashMap<String, List<Song> >albums=new HashMap<>();
    public static ArrayList<ArrayList<Song>>al=new ArrayList<>();

    private NotificationManagerCompat notificationManager;
    MediaMetadataRetriever metadataRetriever;
    private MediaSessionCompat mediaSession;
```

```

private NotificationManagerCompat notificationManager;
MediaMetadataRetriever metadataRetriever;
private MediaSessionCompat mediaSession;

TabLayout tableLayout;|
ViewPager viewPager;
protected ViewPagerAdapter viewPagerAdapter;
LinearLayout miniplayer;
TextView textView;
public static ImageView imageView;
public static Notification notification;

public DrawerLayout drawerLayout;
public ActionBarDrawerToggle actionBarDrawerToggle;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

```

Figure 5.1.1: Methods in Mainactivity class

In this mainactivity java class we're declaring storage permission, data reading, arts, albums, notifications, metadata retriever, table layout, view pager, miniplayer etc.

```

public static MainActivity getInstance() { return instance; }

private void requestStoragePermission(){
    if(ActivityCompat.shouldShowRequestPermissionRationale( activity: this,Manifest.permission.READ_EXTERNAL_STORAGE)){
        new AlertDialog.Builder( context: this).setTitle("Permission Needed").setMessage("Need to read songs from your stor
            @Override
            public void onClick(DialogInterface dialog, int which) {
                ActivityCompat.requestPermissions( activity: MainActivity.this,new String[]{Manifest.permission.READ_EXTERNAL
            }
        }).setNegativeButton( text: "Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) { dialog.dismiss(); }
        }).create().show();
    }else{
        ActivityCompat.requestPermissions( activity: this,new String[]{Manifest.permission.READ_EXTERNAL_STORAGE},Storage
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    if(requestCode==Storage_Permission_code){
        if(grantResults.length>0&&grantResults[0]==PackageManager.PERMISSION_GRANTED){
            Toast.makeText( context: this, text: "Permission Granted",Toast.LENGTH_SHORT).show();
            start();
        }else{
            Toast.makeText( context: this, text: "Permission Denied",Toast.LENGTH_SHORT).show();
        }
    }
}
}

```

Figure 5.1.2: Storage permission function

Here, we're requesting permission results after granting the storage permission

Play/Pause

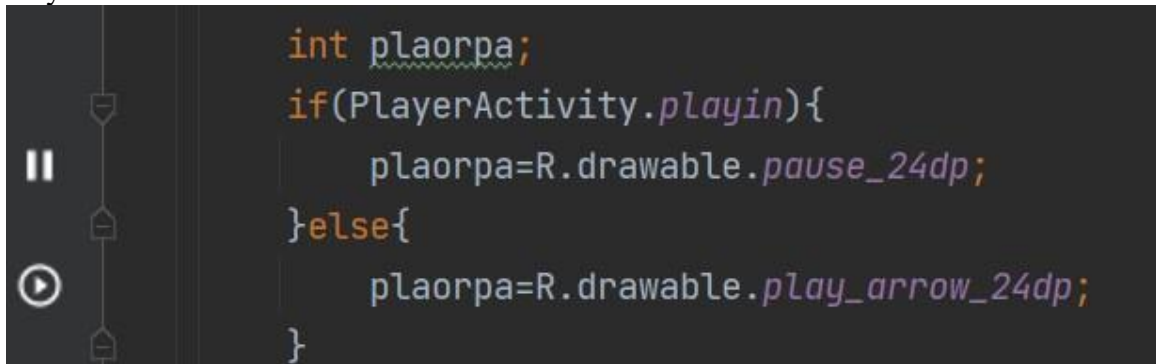


Figure 5.1.3: Play & Pause buttons in mainactivity

Here we're seeing the pause and play buttons for the actions of the playback in java class

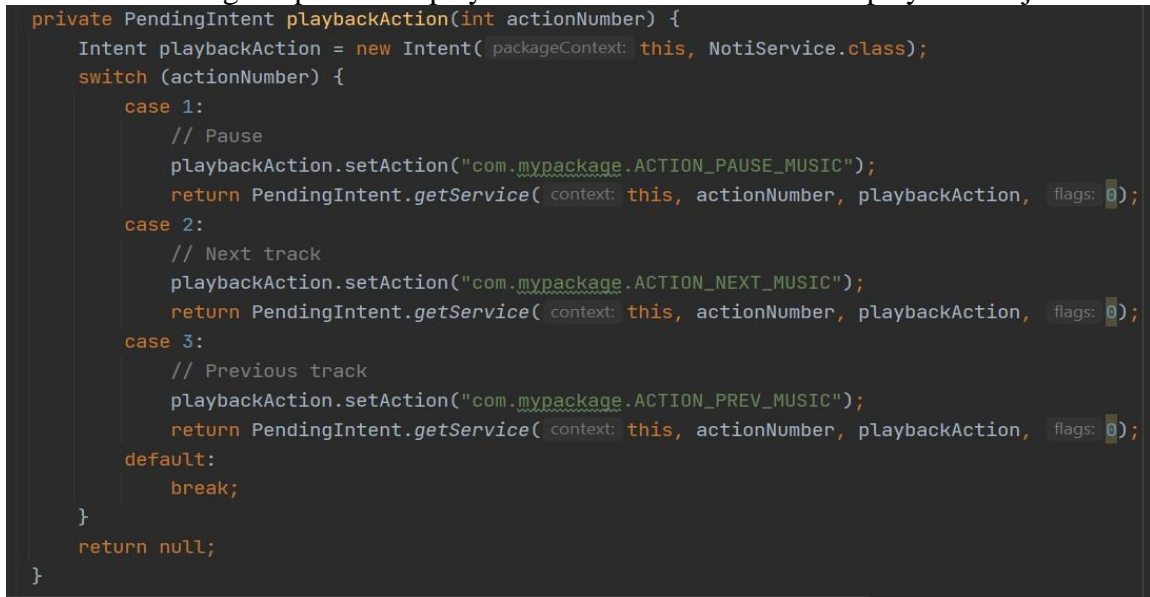


Figure 5.1.4: Playback function

Here we see the playback function for the player's playback function in 3 cases: pause, next track, previous track. This function will control the playback section of the app.

AlbumActivity.java

```

public class AlbumActivity extends AppCompatActivity implements OnClickListener {

    protected ImageView imageView;
    protected int position;
    private TextView textView1, textView2;
    protected RecyclerView recyclerView;
    protected RecyclerView.LayoutManager mmanager;
    private LinearLayout linearLayout;
    static SongAlbumAdapter songalbumAdapter;
    private Palette.Swatch lightVibrantSwatch;
    private Palette.Swatch darkMutedSwatch;

```

Figure 5.1.5: Methods in AlbumActivities Class

Here we see imageview, recyclerview, songalbumadapter & palette specifying on the albumactivity class.

Album's variables

```

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_album);
    recyclerView = findViewById(R.id.album_recycler);
    imageView=findViewById(R.id.albumimage);
    linearLayout=findViewById(R.id.linear);
    recyclerView.setHasFixedSize(true);
    textView1=findViewById(R.id.text1);
    textView2=findViewById(R.id.text2);
    mmanager=new LinearLayoutManager( context: this);
    Intent i = getIntent();
    Bundle bundle = i.getExtras();
    position = bundle.getInt( key: "index");
    songalbumAdapter = new SongAlbumAdapter(al.get(position), onclicklisten: this);
    System.out.println(al.get(position).get(0).getArtist());
    textView1.setEllipsize(TextUtils.TruncateAt.MARQUEE);
    textView1.setText(al.get(position).get(0).getArtist());
    String size=(al.get(position).size()+"");
    textView2.setText(size);

```

Figure 5.1.6: Variables in Album Layout

Here declaring recyclerview, imageview, textview, songalbumadapter, mmanger, for album's fragment layout.

Glide implementation

```
Glide.with( activity: this) RequestManager
    .load(ContentUri.withAppendedId(Uri.parse("content://media/external/audio/albumart"),al.get(position).get(0).getAlbu
    .thumbnail(0.2f)
    .centerCrop()
    .placeholder(R.drawable.track)
    .skipMemoryCache(true)
    .diskCacheStrategy(DiskCacheStrategy.NONE)
    .listener(new RequestListener<Drawable>() {
        @Override
        public boolean onLoadFailed(@Nullable GlideException e, Object model, Target<Drawable> target, boolean isFirstRes
            return false;
        }

        @Override
        public boolean onResourceReady(Drawable resource, Object model, Target<Drawable> target, DataSource dataSource, b
            setBack();
            return false;
        }
    })
})
```

Figure 5.1.7: Using Glide in Albumview

Here we see glide implementation for album to extract the image from the audio file to a thumbnail view on the album fragment.

PlayerActivity.java

```
public static boolean playin;
private ArrayList<Song> Asongs;
private static SeekBar mSeekBar;
private static PlayerActivity instance;
int position;
private static TextView curTime,totTime;
    private TextView songTitle,artistname;
private static ImageView pause,prev,next;
private ImageView imageView;
protected int val;
protected boolean place;
private Palette.Swatch DarkVibrantSwatch;
private Palette.Swatch darkMutedSwatch;

    protected NotificationCenter nofiticationCenter;
    protected LinearLayout linearLayout,linear1;
private static MediaPlayer mMediaPlayer;

public int getPosition() { return position; }
```

Figure 5.1.8: Methods in PlayerActivity class

Here we see all methods used in player activity class like seekbar and its position, duration times (current – to), songtitle textview, imagview for pause/previous/next in the player, palette, notification etc.

```
public void Buttons(){
    pause.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) { play(); }
    });
    prev.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (position == 0) {
                position = Asongs.size() - 1;
            } else {
                position--;
            }
            initPlayer(position);
        }
    });
};
```

Figure 5.1.9: Pause & Previous function in Player

Here in the playeractivity, we see functions for pause & previous of the audio after playing.

```
next.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        position=(position+1)% Asongs.size();
        initPlayer(position);
    }
});
}
```

Figure 5.2.0: Next function in Player

Here we see the next function for the audio playing in playeractivity.


```

public void play() {

    if (mMediaPlayer != null && !mMediaPlayer.isPlaying()) {
        playin=true;
        mMediaPlayer.start();
        pause.setBackgroundResource(R.drawable.pause_24dp);
        MainActivity.getInstance().sendOnChannel( Asongs.get(position).getName(), Asongs.get(position).getArtist(), posi
        MainActivity.imageView.setBackgroundResource(R.drawable.pause_24dp);
    } else {
        pause();
    }
}

public void pause() {
    if (mMediaPlayer.isPlaying()) {
        playin=false;
        mMediaPlayer.pause();
        pause.setBackgroundResource(R.drawable.play_arrow_24dp);
        MainActivity.getInstance().sendOnChannel( Asongs.get(position).getName(), Asongs.get(position).getArtist(), posi
        MainActivity.imageView.setBackgroundResource(R.drawable.play_arrow_24dp);
    }
}
}

```

Figure 5.2.1: Play function in Player

Here we see play function for the player where we see what happens if we click the play button & pause button on the player.

AlbumAdapter.java

```

public class AlbumAdapter extends RecyclerView.Adapter<AlbumAdapter.ViewHolder> implements Filterable {
    private OnClickListener alclicklisten;
    //for search
    //ArrayList<ArrayList<Song>>filtered=new ArrayList<>();

    private LayoutInflater inflater;

    public AlbumAdapter(OnClickListener alclicklisten) {
        this.alclicklisten = alclicklisten;
        inflater=(LayoutInflater) MainActivity.getInstance().getSystemService(MainActivity.getInstance().LAYOUT_INFLATER_SER
    }

    @NonNull
    @Override
    public ViewHolder onCreateViewHolder(@NonNull ViewGroup viewGroup, int i) {

        View view=LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.adapter_album_layout,viewGroup, attachToRoot: fa
        return new ViewHolder(view,alclicklisten);
    }
}

```

Figure 5.2.2: Methods in AlbumAdapter

In this class we see inflater method for filtering implementation (LayoutInflater), viewholder in album adapter class.

```

public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {

    Song song=MainActivity.al.get(i).get(0);
    viewHolder.album.setText(song.getAlbum());
    viewHolder.artist.setText(song.getArtist());
    Glide

        .with(MainActivity.getInstance()) RequestManager
        .load(ContentUri.withAppendedId(Uri.parse("content://media/external/audio/albumart"), song.getAlbumID()).to
        .apply(new RequestOptions()
            .placeholder(R.drawable.xd)
            .diskCacheStrategy(DiskCacheStrategy.NONE)
            .skipMemoryCache(true)
        )
        .thumbnail(0.1f)
        .transition(new DrawableTransitionOptions()
            .crossFade()
        )
        .into(viewHolder.imageView);
}

```

Figure 5.2.3: Variables for ViewHolder

Here we see song, album, artist with the glide library for the image view to display on the album adapter.

```

@Override
public int getItemCount() { return MainActivity.al.size(); }

@Override
public Filter getFilter() { return null; }

public static class ViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener{
    private TextView album,artist;
    private ImageView imageView;
    private OnClickListener onClickListen;
    public ViewHolder(@NonNull View itemView,OnClickListener onClickListen) {
        super(itemView);
        album=itemView.findViewById(R.id.album_name);
        artist=itemView.findViewById(R.id.album_artist);
        imageView=itemView.findViewById(R.id.album_image);
        this.onClickListen=onClickListen;
        itemView.setOnClickListener(this);
    }
}

```

Figure 5.2.4: Variables in AlbumAdapter

Here the variables are textview, imageview, album, artist.

SongAdapter.java

```

public class SongAdapter extends RecyclerView.Adapter<SongAdapter.ViewHolder> implements Filterable {
    protected static Typeface myfont;
    protected OnClickListener monclicklisten;
    private Activity context;
    public static ArrayList<Song>songs;
    private ArrayList<Song>allSongs;
    private static LayoutInflater inflater;

```

Figure 5.2.5: Methods in SongAdapter
Variables in songadapter are listed in the above figure.

Shuffle Function

```

@Override
public void onBindViewHolder(@NonNull ViewHolder viewHolder, int i) {
    if(i==0){
        viewHolder.textView1.setText("Shuffle All");
        viewHolder.textView2.setText("");

        viewHolder.textView1.setTextSize(25);
        viewHolder.textView1.setTextColor(Color.parseColor("colorString: "#FF1105"));
        Glide
            .with(context) RequestManager
            .load(R.drawable.shuffle) RequestBuilder <Drawable>
            .apply(new RequestOptions()
                .diskCacheStrategy(DiskCacheStrategy.NONE)
                .skipMemoryCache(true)
            )
            .thumbnail(0.1f)
            .transition(new DrawableTransitionOptions()
                .crossFade()
            )
            .into(viewHolder.mImageView);

```

Figure 5.2.6: Shuffle Function in SongAdapter
Here for the shuffle function, we using the text “Shuffle All”, the color, textsize, also glide for the suffle transition.

```

        Glide
            .with(context) RequestManager
            .load(ContentUris.withAppendedId(Uri.parse("content://media/external/audio/albumart")),
            .apply(new RequestOptions()
                .placeholder(R.drawable.album_art)
                .diskCacheStrategy(DiskCacheStrategy.NONE)
                .skipMemoryCache(true))
            )
            .thumbnail(0.1f)
            .transition(new DrawableTransitionOptions()
                .crossFade()
            )
            .into(viewHolder.mImageView);
        return;
    } catch (Exception e) {
    }
}
}
}

```

Figure 5.2.7: Albumart in SongAdapter

Here for albumart of the song it'll request with glide crossfade transition in songadapter class.

SongsFragment.java

```

public class SongsFragment extends Fragment implements OnClickListener {
    private View v;
    private RecyclerView recyclerView;
    private RecyclerView.LayoutManager mmanager;
    private static SongAdapter songAdapter;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
        v=inflater.inflate(R.layout.songs_fragment,container, attachToRoot: false);
        recyclerView = v.findViewById(R.id.recycleview);
        recyclerView.setHasFixedSize(true);
        mmanager=new LinearLayoutManager(getContext());
        songAdapter = new SongAdapter(MainActivity.getInstance(), songs, onClickListen: this);
        recyclerView.setLayoutManager(mmanager);
        recyclerView.setAdapter(songAdapter);
        return v;
    }
}

```

Figure 5.2.8: Methods & Variables in SongsFragment

Here we see the view, recyclerview, songadapter methods, variables of inflater view, recyclerview, mmanager layout, songadapter etc.

AlbumsFragment.java

```
public class AlbumsFragment extends Fragment implements OnClickListener {
    protected View v;
    protected RecyclerView recyclerView;

    ////////////////HERE
    protected RecyclerView.LayoutManager mmanager;
    protected static AlbumAdapter albumAdapter;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
}
```

Figure 5.2.9: Methods in AlbumsFragment

The methods that are used in albumsfragment are listed above in the figure.

```
@Override
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {
    v=inflater.inflate(R.layout.albums_fragment,container, attachToRoot: false);
    recyclerView = v.findViewById(R.id.albums_recycleview);
    recyclerView.setHasFixedSize(true);
    mmanager=new GridLayoutManager(getContext(), spanCount: 2);
    albumAdapter = new AlbumAdapter( alclicklisten: this);
    recyclerView.setLayoutManager(mmanager);
    recyclerView.setAdapter(albumAdapter);
    return v;
}

@Override
public void onClick(int position) {
    Intent intent=new Intent(MainActivity.getInstance(), AlbumActivity.class).putExtra( name: "index",position);
    startActivity(intent);
}
}
```

Figure 5.3.0: Variables in AlbumsFragment

Here we see the variables like recyclerview, albumadppter etc.

OnlineFragment.java:

```
public class OnlineFragment extends Fragment {

    private ListView listArea;
    ArrayList<String> AreaArrayList = new ArrayList<>();
    ArrayList<String> AreaArrayList1 = new ArrayList<>();

    @Override
    public void onCreate(Bundle savedInstanceState) { super.onCreate(savedInstanceState); }

    @Override
    public View onCreateView (LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState){
        final View view = inflater.inflate(R.layout.onLine_fragment, container, attachToRoot: false);

        listArea=(ListView)view.findViewById(R.id.listview);
    }
}
```

```

        AreaListAdapter SongAdapter = new AreaListAdapter(getActivity(), AreaArrayList);
        listArea.setAdapter(SongAdapter);
        return view;
    }

    //adapter for the listview
    public class AreaListAdapter extends BaseAdapter {
        ArrayList<String> AreaList;
        ArrayList<String> AreaArrayList1;

        Context context;
        LayoutInflater inflater;

        public AreaListAdapter(Context context, ArrayList<String> AreaArrayList) {
            this.context = context;

            inflater = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            if (AreaArrayList != null) {

                this.AreaList = AreaArrayList;
                this.AreaArrayList1 = AreaArrayList;
            }
        }
    }

```

Figure 5.3.1: OnlineFragment with Adapter

The Model functions are for Data Reading & Song path finder functions.

The Notification functions are for the background playback service for the playing section.

We'll use JC player plugin for the online section of the app.

JC Player implementation in Mainactivity:

```

    /// Online Songs Source
    String music1 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Clouds%20-%20AShamaL...";
    String music2 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-1.mp3?alt=medi...";
    String music3 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-2.mp3?alt=medi...";
    String music4 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-3.mp3?alt=medi...";
    String music5 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-4.mp3?alt=medi...";
    String music6 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-5.mp3?alt=medi...";
    String music7 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-6.mp3?alt=medi...";
    String music8 = "https://firebasestorage.googleapis.com/v0/b/extreme-musicplayer.appspot.com/o/Bangla-7.mp3?alt=medi...";

    JcPlayerView jcPlayerView = (JcPlayerView) findViewById(R.id.jcplayer);

    ArrayList<JcAudio> jcAudios = new ArrayList<>();
    jcAudios.add(JcAudio.createFromURL( title: "Song 1",music1));
    jcAudios.add(JcAudio.createFromURL( title: "Song 2",music2));
    jcAudios.add(JcAudio.createFromURL( title: "Song 3",music3));
    jcAudios.add(JcAudio.createFromURL( title: "Song 4",music4));
    jcAudios.add(JcAudio.createFromURL( title: "Song 5",music5));
    jcAudios.add(JcAudio.createFromURL( title: "Song 6",music6));
    jcAudios.add(JcAudio.createFromURL( title: "Song 7",music7));
    jcAudios.add(JcAudio.createFromURL( title: "Song 8",music8));

    jcPlayerView.initPlaylist(jcAudios, jcPlayerManagerListener: null);
    jcPlayerView.createNotification();

```

Figure 5.3.2: JC Player in Mainactivity

5.2 Database Storage of Songs:

Storage section for our online firebase database:

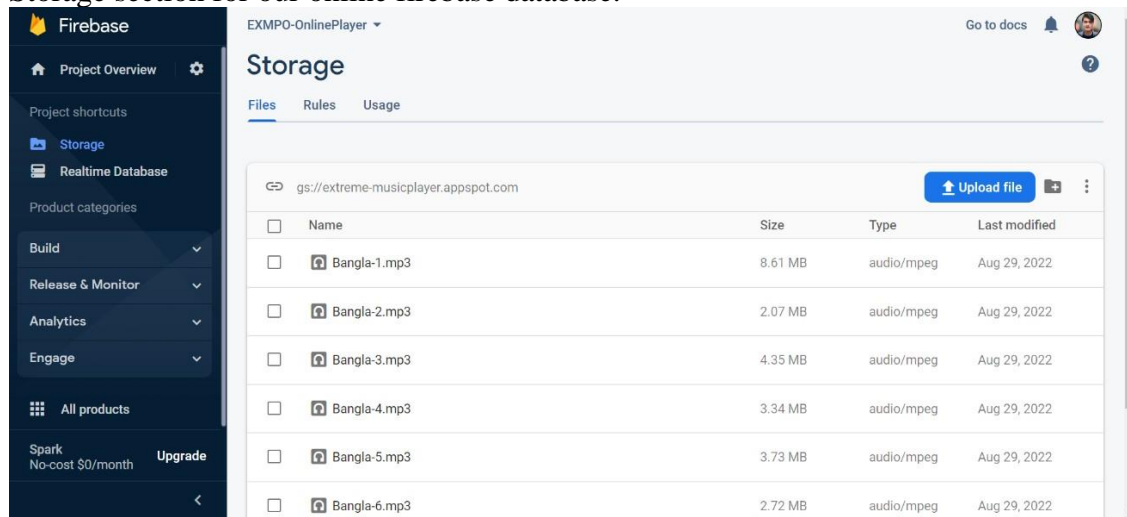


Figure 5.3.3: Firebase Database Storage Library

We're showing the database that we'll use for the song library for the app. We will continuously update the library with the app's advancements.

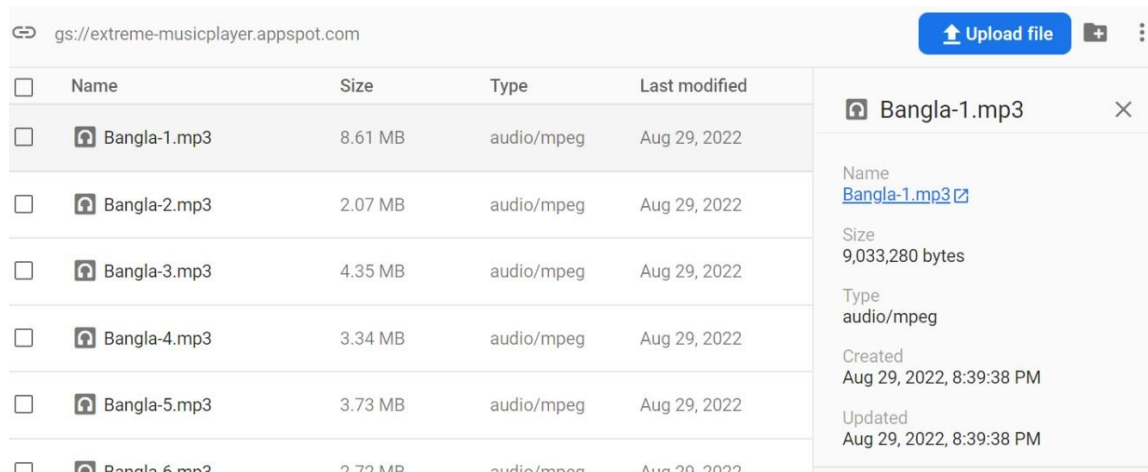


Figure 5.3.4: Showing the Source of the Audio from the Database

Image is showing the source that we'll link with the player for the online playing of the audio files from the storage to the app itself.

In future days, we'll be uploading a big collection of song libraries in this storage section & expand our library as much as possible.

5.3 TEST RESULTS & REPORTS:

Features

- Supports any kind of audio formats
- Users can randomize & repeat songs on the online section & shuffle for local songs
- Online player and local player both has notification background playback option while an audio is playing
- Uses Material theme & design of AndroidX
- Has a search function for the local section of the app
- Has an interactive Navigation Drawer that has multiple actions
- There's a mini-player that can be toggled by pressing the arrow or any place of the bottom section of the app
- Online player and local player both has notification background playback option while an audio is playing
- Play/Pause/Next/Previous buttons on the player for both online & local sections. Both players have duration time of seek-bar from where we can change the state of the song playing by tapping

Reports

Merits

- It uses online connectivity for better streaming services
- User friendly UI has made the app very reliable for constant usage
- Has many areas for improvement in terms of UI
- Very basic architecture for functions usage

Demerits

- Very basic design may be a bit overwhelming
- Has many areas for improvement in terms of UI
- Time is needed for more planned implementations

CHAPTER 6: IMPACT ON SOCIETY, ENVIRONMENT, AND SUSTAINABILITY

6.1 Impact on society:

Music carries a lot of influence these days. We know the sound any messages through those music as it is an art with giving a Nobel reason to allocate. From this app we are rebuilding a sector where our own country people can listen so many OWN language music without any purchase. So how it will benefit and giving a superior in society? Let's find out:

1. No expenses to hear music.
2. Boundless unknown songs will be feature which many of our generational people want to listen
3. After the final breakdown of app and if we good response we will always try attract people for so many musical contests through our app. Where they will learn our culture
4. Adding lot of audiobooks platform which established social messages books over there to learn different things and makes studies easier by listening.

6.2 Ethical Aspects:

We created this app moral values with regarding not our own benefit. We know music has a great influence in human life. Our work determines to persuade and very relevant for everyone. This music online player app is absolutely based on without any purchase where we think many people of our entertainment purpose will just be benefited by that. Our goal is set to be done for better ethical music version with no affecting things for us.

6.3 Sustainability plan:

Our motto is to take this app for next level as we are also going to work for that hiring different people to make it better. But first we want to see some public response. The things we are planning is We will take it more languages after Bangla, adding some key features, Will include the web version later. Also, our plan is most cases to get a better result and it's absolutely free for people which is a key aspect in this sustainability.

Chapter 7: CONCLUSION & FUTURE SCOPE

7.1 Discussion & Conclusion:

This app project is an ambitious project for us as we're trying to do something that is very unique & useful for our entertainment usage purposes. As we've said before this is just the start. We're trying to do our best to develop such mobile application. In these recent years, mobile app development areas have improved so much. We can utilize our time and effort to develop this app & achieve our goal. We want to give the users of the app, the best experience possible. We will expand our knowledge and try to make a better music system for our country. In our country as we know, is not well approached in this music services. As our app is still not progressed far yet, we'll try to develop our app to fit in with the current advancements of android application services. We'll then implement as much features as we can to make the app more appealing to the user base.

Now let's discuss the pros and cons & what can we do in the future as we progress in the development.

The Pros:

1. The app will be used for audio playback services
2. It'll have the audio files which can be played online
3. We can shuffle the songs in offline song library section
4. We can select songs in different album's choices
5. It has background playing option in the notification area

The Cons:

1. It doesn't have too many features
2. It needs more optimizations for less load time
3. Needs more storage capability to store big song library

7.2 Scope for Further Developments:

The future plans are very ambitious for the app. We're thinking to make the app as free of cost as possible. Because in our country the cost is a big issue. In foreign countries that may be not a big issue. For example, in the Spotify, Deezer Soundcloud etc. they have many plans for premium features to stream the songs.

After that we need divide the works of the app to some developers of our knowing, because to maintain a good app it needs more updates, testing & fine-tuning. As we the 2 members may not be able to get the works done before the deadlines that we'll do.

We are thinking not to immediately implement the paid features as we haven't thought that far yet. So, if we see that our app is well received by the users of the apps. Then we can think of that as an optional side plan.

As we'll develop the app further and beyond, we haven't quite got everything in sight yet. Because that's way ahead of the development process. So, we need to work well in the basic parts of our app. Then we can go from there as we approach with the planning. Also, the planning of the app will have to be simpler for the users. Because in our country the overall people are not well educated enough to know complex details about services that we're planning to provide with the application. To overcome that problem we'll make the UI layout of the app very simple & basic, even in the future works.

References:

1. Kang, Byungseok, et al. "CloudIoT-based Jukebox Platform: A Music Player for Mobile Users in Café." *Journal of Internet Technology* 21.5 (2020): 1363-1374.
2. Leischner, Vojtech, and Zdenek Mikovec. "Spatial Audio Music Player for Web." *RoCHI*. 2021.
3. Inouye, Jon, et al. "System support for mobile multimedia applications." *Proceedings of 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'97)*. IEEE, 1997.
4. Huber, Sebastian, Markus Schedl, and Peter Knees. "nepDroid: An intelligent mobile music player." *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval*. 2012.
5. Cui, Yi, Klara Nahrstedt, and Dongyan Xu. "Seamless user-level handoff in ubiquitous multimedia service delivery." *Multimedia Tools and Applications* 22.2 (2004): 137-170.
6. Wang, Zhong-Rong, and Zhao Liu. "Implementation of Mobile Streaming Media Player Based on BREW." *Journal of Electronic Science and Technology of China* 4.3 (2006): 243-248.
7. Lee, Minhyung, et al. "Cannibalizing or complementing?. The impact of online streaming services on music record sales." *Procedia Computer Science* 91 (2016): 662-671.
8. Schwind, Anika, et al. "QoE analysis of Spotify audio streaming and app browsing." *Proceedings of the 4th internet-QoE workshop on QoE-based analysis and management of data communication networks*. 2019.
9. Hofmann, Markus, and Krishan Sabnani. "Streaming and broadcasting over the Internet." *ATM 2000. Proceedings of the IEEE Conference on High Performance Switching and Routing (Cat. No. 00TH8485)*. IEEE, 2000.
10. Simon, Christian, Matteo Torcoli, and Jouni Paulus. "MPEG-H Audio for Improving Accessibility in Broadcasting and Streaming." *arXiv preprint arXiv:1909.11549* (2019).

MUSIC ONLINE PLAYER APP

ORIGINALITY REPORT

21%	20%	3%	17%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	dspace.daffodilvarsity.edu.bd:8080 Internet Source	9%
2	Submitted to Daffodil International University Student Paper	5%
3	www.freepatentsonline.com Internet Source	1%
4	www.scirp.org Internet Source	1%
5	link.springer.com Internet Source	1%
6	Submitted to Baylor University Student Paper	<1%
7	p.codekk.com Internet Source	<1%
8	opus.lib.uts.edu.au Internet Source	<1%
9	Usir.salford.ac.uk Internet Source	<1%

10	www.journal.uestc.edu.cn Internet Source	<1%
11	Submitted to Kingston University Student Paper	<1%
12	onlinelibrary.wiley.com Internet Source	<1%
13	mindorks.com Internet Source	<1%
14	adoc.pub Internet Source	<1%
15	Submitted to St. Petersburg High School Student Paper	<1%
16	anthonygarvan.github.io Internet Source	<1%

Exclude quotes Off
Exclude bibliography Off

Exclude matches Off