MOBILE APPLICATION DEVELOPMENT

PROJECT PROPOSAL: “Music Player”

Instructions: *Copied or shown assignments will be marked zero. Late submissions are not entertained in any case.*

- Submitted To: Mr. Umair
- Student Name: M. Shahmir Raza, Shamain Anjum, Jawad Haider
- Roll Number: RCF – (50003), (50002), (50005)
- Submission Date: 27th May 2024
- Course & Dept.: BS-IT (6th SEM), Software Engineering

1. MainActivity

MainActivity Class

```
public class MainActivity extends AppCompatActivity {
ListView listView;
String[] items;
Toolbar toolbar;
```

- This defines the MainActivity class, which extends AppCompatActivity. It declares member variables for a ListView, an array of song items, and a Toolbar.

OnCreate Method

```
@Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      toolbar = findViewById(R.id.toolbar);
      setSupportActionBar(toolbar);
      setContentView(R.layout.activity_main);
      listView = findViewById(R.id.listViewSong);

      runtimePermission();

   }
```

- This method is called when the activity is created. It initializes the Toolbar and ListView, sets the content view, and requests runtime permissions.

Runtime Permission Method

```
public void runtimePermission()
   {

Dexter.withContext(this).withPermissions(Manifest.permission.READ_EXTERNAL_STORAG
E, Manifest.permission.RECORD_AUDIO)
            .withListener(new MultiplePermissionsListener() {
               @Override
               public void onPermissionsChecked(MultiplePermissionsReport
multiplePermissionsReport) {
                  displaySongs();
               }

               @Override
               public void onPermissionRationaleShouldBeShown(List<PermissionRequest>
list, PermissionToken permissionToken) {
                  permissionToken.continuePermissionRequest();
               }
            }).check();
    }
```

- This method requests runtime permissions for reading external storage and recording audio using the Dexter library. If permissions are granted, it calls displaySongs(). If the user needs further explanation about the permissions, onPermissionRationaleShouldBeShown handles it.

Find Song Method

```
    public ArrayList<File> findSong(File file) {
        ArrayList<File> arrayList = new ArrayList<>();

        // Check if file is null or if it's not a directory
        if (file == null || !file.isDirectory()) {
            return arrayList;
        }

        File[] files = file.listFiles();
        if (files != null) {
            for (File singlefile : files) {
                if (singlefile.isDirectory() && !singlefile.isHidden()) {
                    arrayList.addAll(findSong(singlefile));
                } else {
                    if (singlefile.getName().endsWith(".mp3") ||
singlefile.getName().endsWith(".wav")) {
                        arrayList.add(singlefile);
                    }
                }
            }
        }
        return arrayList;
    }
```

- This method recursively searches for .mp3 and .wav files in the given directory and its subdirectories. It returns an ArrayList of found song files.

Display Songs Method:

```
  void displaySongs()
{
    final ArrayList<File> mySongs = findSong(Environment.getExternalStorageDirectory());
    items = new String[mySongs.size()];
    for (int i = 0; i<mySongs.size(); i++)
    {
        items[i] = mySongs.get(i).getName().toString().replace(".mp3", "").replace(".wav", "");
    }

    customAdapter customAdapter = new customAdapter();
    listView.setAdapter(customAdapter);

    listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
            String songName = (String) listView.getItemAtPosition(i);
            startActivity(new Intent(getApplicationContext(), PlayerActivity.class)
                    .putExtra("songs", mySongs)
                    .putExtra("songname", songName)
                    .putExtra("pos", i));
        }
    });

}
```

- This method displays the found songs in the ListView. It first calls findSong to get the list of song files, extracts the names of the songs, and then sets up a custom adapter (customAdapter) to display these names. It also sets up an OnItemClickListener to handle clicks on the song items, starting the PlayerActivity with the selected song's details.

Custom Adapter Class

```
class customAdapter extends BaseAdapter
{

    @Override
    public int getCount() {
        return items.length;
    }

    @Override
    public Object getItem(int position) {
        return null;
    }

    @Override
    public long getItemId(int position) {
        return 0;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View myView = getLayoutInflater().inflate(R.layout.list_item, null);
        TextView textSong = myView.findViewById(R.id.txtSongName);
        textSong.setSelected(true);
        textSong.setText(items[position]);

        return myView;
    }
```

```
        }
    }
```

- This inner class extends BaseAdapter to create a custom adapter for the ListView. It overrides necessary methods:
- getCount returns the number of items.
- getItem and getItemId return null and 0 respectively (not used in this example).
- getView inflates a custom layout (list_item) and sets the song name to a TextView.

Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    tools:context=".MainActivity">

  <androidx.appcompat.widget.Toolbar
      android:id="@+id/toolbar"
      android:layout_width="match_parent"
      android:layout_height="?attr/actionBarSize"
      android:background="@color/black">

    <RelativeLayout
        android:layout_width="110dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center">

   <ImageView
       android:id="@+id/logoimage"
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:layout_marginTop="10dp"
       android:src="@drawable/icons1assassins_creed">
   </ImageView>

</RelativeLayout>
  </androidx.appcompat.widget.Toolbar>


  <ListView
      android:id="@+id/listViewSong"
      android:layout_below="@id/toolbar"
      android:layout_marginTop="10dp"
      android:divider="@android:color/transparent"
      android:dividerHeight="10.0sp"
      android:padding="8dp"
      android:layout_width="match_parent"
      android:layout_height="match_parent"/>

</RelativeLayout>
```

2. PlayerActivity

```
public class PlayerActivity extends AppCompatActivity {

   Button btnplay, btnnext, btnprev, btnff, btnfr;
   TextView txtsname, txtsstart, txtsstop;
   SeekBar seekmusic;
   BarVisualizer visualizer;
   ImageView imageView;
   Toolbar toolbar1;

   String sname;
   public static final String EXTRA_NAME = "song_name";
   static MediaPlayer mediaPlayer;
   int position;
   ArrayList<File> mySongs;
   Thread updateseekbar;
```

- Class Declaration: PlayerActivity extends AppCompatActivity, indicating this is an activity class.
- UI Components: Declares buttons (btnplay, btnnext, btnprev, btnff, btnfr), text views (txtsname, txtsstart, txtsstop), a seek bar (seekmusic), an audio visualizer (visualizer), an image view (imageView), and a toolbar (toolbar1).
- Media Player Variables: sname for the song name, mediaPlayer for the media player instance, position for the current song position, mySongs for the list of songs, and updateseekbar for a thread to update the seek bar.

```
   @Override
   public boolean onOptionsItemSelected(@NonNull MenuItem item) {
      if (item.getItemId()==android.R.id.home);
      {
         onBackPressed();
      }
      return super.onOptionsItemSelected(item);
   }
```

- Handles the back button in the toolbar. If the home button is pressed, onBackPressed is called to navigate back.

```
   @Override
   protected void onDestroy() {
      if (visualizer != null)
```

```
        {
            visualizer.release();
        }
        super.onDestroy();
    }
```

- Releases the audio visualizer resources when the activity is destroyed to prevent memory leaks.

onCreate Method

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_player);
```

Initializes the activity and sets the layout to activity_player.

Toolbar Setup

```
        toolbar1 = findViewById(R.id.toolbar1);
        setSupportActionBar(toolbar1);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);
        getSupportActionBar().setDisplayShowHomeEnabled(true);
```

- Sets up the toolbar with home button support.

Initialize UI Components

```
        btnprev = findViewById(R.id.btnprev);
        btnplay = findViewById(R.id.playbtn);
        btnnext = findViewById(R.id.btnnext);
        btnff = findViewById(R.id.btnff);
        btnfr = findViewById(R.id.btnfr);
        txtsname = findViewById(R.id.txtsn);
        txtsstart = findViewById(R.id.txtsstart);
        txtsstop = findViewById(R.id.txtsstop);
        seekmusic = findViewById(R.id.seekbar);
        visualizer = findViewById(R.id.blast);
        imageView = findViewById(R.id.imageview);
```

- Finds and assigns the UI components by their IDs.

Media Player Setup

```
        if (mediaPlayer != null)
```

```
{
    mediaPlayer.stop();
    mediaPlayer.release();
}
```

- Stops and releases any existing media player instance to prevent conflicts.

Get Intent Data

```
Intent i = getIntent();
Bundle bundle = i.getExtras();

mySongs = (ArrayList) bundle.getParcelableArrayList("songs");
String songName = i.getStringExtra("songname");
position = bundle.getInt("pos", 0);
txtsname.setSelected(true);
Uri uri = Uri.parse(mySongs.get(position).toString());
sname = mySongs.get(position).getName();
txtsname.setText(sname);
```

- Retrieves the song list, current song name, and position from the intent extras. Sets the song name to the text view.

Media Player Initialization

```
mediaPlayer = MediaPlayer.create(getApplicationContext(), uri);
mediaPlayer.start();
```

- Creates and starts the media player with the selected song.

Seek Bar Update Thread

```
updateseekbar = new Thread()
{
    @Override
    public void run() {
        int totalDuration = mediaPlayer.getDuration();
        int currentposition = 0;

        while (currentposition<totalDuration)
        {
            try {
                sleep(500);
                currentposition = mediaPlayer.getCurrentPosition();
                seekmusic.setProgress(currentposition);
            }
            catch (InterruptedException | IllegalStateException e)
            {
                e.printStackTrace();
                }
            }
        }
    };
    seekmusic.setMax(mediaPlayer.getDuration());
    updateseekbar.start();
```

- Sets up a thread to update the seek bar every 500 milliseconds based on the media player's current position.

Seek Bar Color and Listener

```
seekmusic.getProgressDrawable().setColorFilter(getResources().getColor(R.color.purple_200),
PorterDuff.Mode.MULTIPLY);
        seekmusic.getThumb().setColorFilter(getResources().getColor(R.color.purple_200),
PorterDuff.Mode.SRC_IN);

        seekmusic.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener() {
            @Override
            public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            }

            @Override
            public void onStartTrackingTouch(SeekBar seekBar) {

            }

            @Override
            public void onStopTrackingTouch(SeekBar seekBar) {
                mediaPlayer.seekTo(seekBar.getProgress());

            }
        });
```

- Sets the color for the seek bar and its thumb. Adds a listener to update the media player position when the seek bar is moved.

Update Start and Stop Time

```
        String endTime = createTime(mediaPlayer.getDuration());
        txtsstop.setText(endTime);

        final Handler handler = new Handler();
        final int delay = 1000;

        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
```

```
        String currentTime = createTime(mediaPlayer.getCurrentPosition());
        txtsstart.setText(currentTime);
        handler.postDelayed(this, delay);
    }
}, delay);
```

- Sets the end time and updates the start time every second using a handler.

Play/Pause Button

```
btnplay.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        if (mediaPlayer.isPlaying())
        {
            btnplay.setBackgroundResource(R.drawable.ic_play);
            mediaPlayer.pause();
        }
        else
        {
            btnplay.setBackgroundResource(R.drawable.ic_pause);
            mediaPlayer.start();
        }
    }
});
```

- Toggles play and pause state of the media player and updates the button icon accordingly.

Media Player Completion Listener

```
mediaPlayer.setOnCompletionListener(new MediaPlayer.OnCompletionListener() {
    @Override
    public void onCompletion(MediaPlayer mp) {
        btnnext.performClick();
    }
});
```

- Plays the next song when the current song finishes.

Audio Visualizer

```
int audiosessionId = mediaPlayer.getAudioSessionId();
if (audiosessionId != -1)
{
    visualizer.setAudioSessionId(audiosessionId);
}
```

- Sets the audio session ID for the visualizer to visualize the audio.

Next Button

```
btnnext.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayer.stop();
        mediaPlayer.release();
        position = ((position+1)%mySongs.size());
        Uri u = Uri.parse(mySongs.get(position).toString());
        mediaPlayer = MediaPlayer.create(getApplicationContext(), u);
        sname = mySongs.get(position).getName();
        txtsname.setText(sname);
        mediaPlayer.start();
        btnplay.setBackgroundResource(R.drawable.ic_pause);
        startAnimation(imageView);
        int audiosessionId = mediaPlayer.getAudioSessionId();
        if (audiosessionId != -1)
        {
            visualizer.setAudioSessionId(audiosessionId);
        }
        String endTime = createTime(mediaPlayer.getDuration());
        txtsstop.setText(endTime);
        seekmusic.setMax(mediaPlayer.getDuration());
    }
});
```

- Plays the next song, updates the media player, UI components, and starts an animation.

Previous Button

```
btnprev.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        mediaPlayer.stop();
        mediaPlayer.release();
        position = ((position-1)<0)?(mySongs.size()-1):(position-1);
        Uri u = Uri.parse(mySongs.get(position).toString());
        mediaPlayer = MediaPlayer.create(getApplicationContext(), u);
        sname = mySongs.get(position).getName();
        txtsname.setText(sname);
        mediaPlayer.start();
        btnplay.setBackgroundResource(R.drawable.ic_pause);
        startAnimation(imageView);
        int audiosessionId = mediaPlayer.getAudioSessionId();
        if (audiosessionId != -1)
        {
            visualizer.setAudioSessionId(audiosessionId);
        }
```

```
            String endTime = createTime(mediaPlayer.getDuration());
            txtsstop.setText(endTime);
            seekmusic.setMax(mediaPlayer.getDuration());
        }
    });
```

- Plays the previous song, updates the media player, UI components, and starts an animation.

Fast Forward and Rewind Buttons

```
    btnff.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (mediaPlayer.isPlaying())
            {
                mediaPlayer.seekTo(mediaPlayer.getCurrentPosition()+10000);
            }
        }
    });

    btnfr.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (mediaPlayer.isPlaying())
            {
                mediaPlayer.seekTo(mediaPlayer.getCurrentPosition()-10000);
            }
        }
    });
}
```

- Fast forwards and rewinds the song by 10 seconds if the media player is playing.

Animation Method

```
  public  void  startAnimation(View view)
  {
    ObjectAnimator animator = ObjectAnimator.ofFloat(imageView, "rotation", 0f,360f);
    animator.setDuration(1000);
    AnimatorSet animatorSet = new AnimatorSet();
    animatorSet.playTogether(animator);
    animatorSet.start();
  }
```

- Starts a rotation animation for the image view.

Create Time Method

```java
    public String createTime(int duration)
    {
        String time = "";
        int min = duration/1000/60;
        int sec = duration/1000%60;

        time+=min+":";

        if (sec<10)
        {
            time+="0";
        }
        time+=sec;

        return time;

    }
}
```

- Converts milliseconds to a formatted time string (minutes).

Layout:

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/black"
    android:orientation="vertical"
    android:weightSum="10"
    tools:context=".PlayerActivity">

    <androidx.appcompat.widget.Toolbar
        android:id="@+id/toolbar1"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="@color/black">

        <RelativeLayout
            android:layout_width="110dp"
            android:layout_height="wrap_content"
            android:layout_gravity="center">

            <TextView
                android:id="@+id/title2"
                android:text=" Now Playing"
            android:textStyle="normal"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">

        </TextView>

    </RelativeLayout>
</androidx.appcompat.widget.Toolbar>

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="7"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:id="@+id/txtsn"
        android:layout_margin="20dp"
        android:ellipsize="marquee"
        android:marqueeRepeatLimit="marquee_forever"
        android:padding="10dp"
        android:singleLine="true"
        android:text="Song Name"
        android:textColor="#FFF"
        android:textSize="22sp"
        android:textAlignment="center"
        android:textStyle="bold"
        android:layout_width="match_parent"
        android:layout_height="wrap_content">
    </TextView>

    <ImageView
        android:id="@+id/imageview"
        android:layout_marginBottom="8dp"
        android:src="@drawable/icons1assassins_creed"
        android:layout_width="250dp"
        android:layout_height="250dp">
    </ImageView>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="60dp">
        <SeekBar
            android:id="@+id/seekbar"
            android:layout_centerInParent="true"
            android:layout_alignParentBottom="true"
            android:layout_margin="20dp"
            android:layout_marginBottom="40dp"
            android:layout_width="250dp"
            android:layout_height="wrap_content">

        </SeekBar>

        <TextView
            android:id="@+id/txtsstart"
            android:layout_toLeftOf="@+id/seekbar"
            android:layout_centerInParent="true"
            android:layout_alignParentLeft="false"
            android:layout_marginLeft="10dp"
            android:text="0:10"
            android:textColor="@color/white"
            android:textSize="14sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </TextView>

        <TextView
            android:id="@+id/txtsstop"
            android:layout_toRightOf="@+id/seekbar"
            android:layout_centerInParent="true"
            android:layout_alignParentRight="false"
            android:layout_marginRight="10dp"
            android:text="4:10"
            android:textColor="@color/white"
            android:textSize="14sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
        </TextView>

    </RelativeLayout>
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="3">

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:ignore="MissingClass">
        <Button
            android:id="@+id/playbtn"
            android:layout_centerHorizontal="true"
            android:background="@drawable/ic_pause"
            android:layout_width="70dp"
            android:layout_height="70dp">
```

```xml
</Button>

<Button
    android:id="@+id/btnnext"
    android:layout_toRightOf="@id/playbtn"
    android:layout_marginTop="15dp"
    android:background="@drawable/ic_next"
    android:layout_width="50dp"
    android:layout_height="50dp">
</Button>

<Button
    android:id="@+id/btnprev"
    android:layout_toLeftOf="@id/playbtn"
    android:layout_marginTop="15dp"
    android:background="@drawable/ic_previous"
    android:layout_width="50dp"
    android:layout_height="50dp">
</Button>

<Button
    android:id="@+id/btnff"
    android:layout_toRightOf="@id/btnnext"
    android:layout_marginTop="20dp"
    android:layout_marginLeft="15dp"
    android:background="@drawable/ic_fast_forward"
    android:layout_width="40dp"
    android:layout_height="40dp">
</Button>

<Button
    android:id="@+id/btnfr"
    android:layout_toLeftOf="@id/btnprev"
    android:layout_marginTop="20dp"
    android:layout_marginRight="15dp"
    android:background="@drawable/ic_fast_rewind"
    android:layout_width="40dp"
    android:layout_height="40dp">
</Button>

<com.gauravk.audiovisualizer.visualizer.BarVisualizer
    xmlns:custom="http://schemas.android.com/apk/res-auto"
    android:id="@+id/blast"
    android:layout_width="match_parent"
    android:layout_height="70dp"
    android:layout_alignParentBottom="true"
    custom:avDensity="0.5"
    custom:avType="outline"
    custom:avWidth="4dp"
```

```
            custom:avColor="@color/purple_200"
            custom:avSpeed="normal"/>
    </RelativeLayout>
  </LinearLayout>

</LinearLayout>
```

3. SplashscreenActivity (Launcher Activity)

This Splashscreen activity is a full-screen splash screen that displays the app's version name and animates two text views with a ZoomIn effect using the YoYo animation library. The splash screen displays these animations sequentially with slight delays, and after a total of 3.2 seconds, it starts the MainActivity and finishes itself.

Imports

```
package com.example.srmusic;
import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.WindowManager;
import android.widget.TextView;
import com.daimajia.androidanimations.library.Techniques;
import com.daimajia.androidanimations.library.YoYo;
```

- AppCompatActivity from AndroidX library for activity support.
- Intent, Bundle, Handler, View, WindowManager, and TextView from Android framework for managing intents, UI, and handling delayed tasks.
- Techniques and YoYo from the daimajia animation library for animations.

Class Declaration

```
public class Splashscreen extends AppCompatActivity {
```

onCreate Method

```
  @Override
  protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      getWindow().setFlags(WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS,
WindowManager.LayoutParams.FLAG_LAYOUT_NO_LIMITS);

      setContentView(R.layout.activity_splashscreen);
```

- super.onCreate(savedInstanceState);: Calls the superclass's onCreate method to perform any setup needed for the activity.

- getWindow().setFlags: Sets the window flags to make the layout extend to full screen by using FLAG_LAYOUT_NO_LIMITS.
- setContentView(R.layout.activity_splashscreen);: Sets the content view to the layout file activity_splashscreen.

Initializing UI Components

```
TextView textView = findViewById(R.id.textView);
TextView textView2 = findViewById(R.id.textView2);
TextView version = findViewById(R.id.version);

String ver = BuildConfig.VERSION_NAME;

version.setText(ver);
```

- TextView textView = findViewById(R.id.textView);: Finds and initializes the textView by its ID.
- TextView textView2 = findViewById(R.id.textView2);: Finds and initializes the textView2 by its ID.
- TextView version = findViewById(R.id.version);: Finds and initializes the version TextView by its ID.
- String ver = BuildConfig.VERSION_NAME;: Retrieves the app version name from the build configuration.
- version.setText(ver);: Sets the version name to the version TextView.

Handlers for Delayed Tasks and Animations

```
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        textView.setVisibility(View.VISIBLE);
        YoYo.with(Techniques.ZoomIn).duration(400).playOn(findViewById(R.id.textView));
    }
}, 2200);
```

- new Handler().postDelayed(new Runnable() { ... }, 2200);: Creates a new handler that will execute the provided Runnable after 2200 milliseconds (2.2 seconds).
- textView.setVisibility(View.VISIBLE);: Sets the visibility of textView to VISIBLE.
- YoYo.with(Techniques.ZoomIn).duration(400).playOn(findViewById(R.id.textView));: Applies a ZoomIn animation to textView with a duration of 400 milliseconds using the YoYo animation library

```
new Handler().postDelayed(new Runnable() {
    @Override
    public void run() {
        textView2.setVisibility(View.VISIBLE);
        YoYo.with(Techniques.ZoomIn).duration(400).playOn(findViewById(R.id.textView));
```

```
            }
        }, 2600);
```

- new Handler().postDelayed(new Runnable() { ... }, 2600);: Creates a new handler that will execute the provided Runnable after 2600 milliseconds (2.6 seconds).
- textView2.setVisibility(View.VISIBLE);: Sets the visibility of textView2 to VISIBLE.
- YoYo.with(Techniques.ZoomIn).duration(400).playOn(findViewById(R.id.textView));: Applies a ZoomIn animation to textView2 with a duration of 400 milliseconds using the YoYo animation library.

```
        new Handler().postDelayed(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(Splashscreen.this, MainActivity.class);
                startActivity(intent);
                finish();
            }
        }, 3200);

    }
}
```

- new Handler().postDelayed(new Runnable() { ... }, 3200);: Creates a new handler that will execute the provided Runnable after 3200 milliseconds (3.2 seconds).
- Intent intent = new Intent(Splashscreen.this, MainActivity.class);: Creates a new intent to start MainActivity.
- startActivity(intent);: Starts the MainActivity.
- finish();: Finishes the current Splashscreen activity, removing it from the back stack so the user cannot navigate back to it.

Layout:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    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/black"
    tools:context=".Splashscreen">

    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/animationView"
        android:layout_width="230dp"
        android:layout_height="230dp"
        android:layout_marginTop="40dp"
        android:layout_centerHorizontal="true"
        app:lottie_rawRes="@raw/animation1"
        app:lottie_autoPlay="true"
        app:lottie_loop="false"/>

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="SUPER PLAYER"
        android:layout_centerHorizontal="true"
        android:layout_below="@id/animationView"
        android:textColor="@color/white"
        android:textSize="36sp"
        android:visibility="gone"
        android:textStyle="bold"/>

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="5dp"
        android:text="Let the Music Play ♪♫"
        android:textSize="20sp"
        android:gravity="center"
        android:textColor="@color/white"
        android:layout_below="@+id/textView"
        android:visibility="gone"
        android:layout_centerHorizontal="true"
        android:textAppearance="@style/TextAppearance.Compat.Notification.Info" />

    <TextView
        android:id="@+id/version"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:paddingBottom="60dp"
        android:text="L.O.A.D.I.N.G>>>"
        android:textAlignment="center"
        android:textColor="@color/white"
        android:textSize="20sp"
        android:textAppearance="@style/TextAppearance.Compat.Notification.Info" />

</RelativeLayout>
```

4. Manifest File

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
```

```
xmlns:tools="http://schemas.android.com/tools">
```

Permissions

```
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECORD_AUDIO" />
```

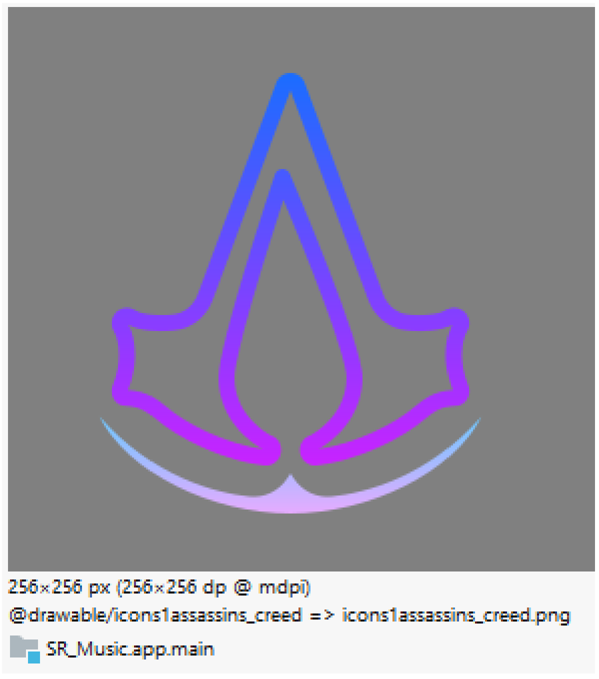
- READ_EXTERNAL_STORAGE: Allows the app to read from external storage.
- RECORD_AUDIO: Allows the app to record audio

```
<application
   android:allowBackup="true"
   android:dataExtractionRules="@xml/data_extraction_rules"
   android:fullBackupContent="@xml/backup_rules"
```

App Icon:

```
    android:icon="@drawable/icons1assassins_creed"
    android:label="@string/app_name"
    android:roundIcon="@drawable/icons1assassins_creed"
```

- android:icon: Defines the launcher icon for the app.
- android:label: Specifies the app's name (label).
- android:roundIcon: Specifies the round version of the launcher icon.



256×256 px (256×256 dp @ mdpi)
@drawable/icons1assassins_creed => icons1assassins_creed.png
SR_Music.app.main

```
    android:supportsRtl="true"
    android:theme="@style/Theme.SRMusic"
    tools:targetApi="31">
    <activity
       android:name=".Splashscreen"
```

```
        android:exported="true">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>

        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
    <activity
        android:name=".PlayerActivity"
        android:exported="false">
        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
    <activity
        android:name=".MainActivity"
        android:exported="true">


        <meta-data
            android:name="android.app.lib_name"
            android:value="" />
    </activity>
</application>

</manifest>
```

5. Build.gradle (:app):

```
dependencies {

    implementation 'androidx.appcompat:appcompat:1.6.1'

    implementation 'com.karumi:dexter:6.2.2'
```

- com.karumi:dexter:6.2.2: Dexter simplifies the process of requesting permissions at runtime, handling the complex logic of the Android permission system.

```
    implementation 'com.google.android.material:material:1.12.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
    testImplementation 'junit:junit:4.13.2'

    implementation 'com.gauravk.audiovisualizer:audiovisualizer:0.9.2'
```

- com.gauravk.audiovisualizer:audiovisualizer:0.9.2: Provides visualizations for audio in Android applications, allowing the display of audio waveforms, bars, and other visual effects synchronized with audio playback.

```
androidTestImplementation 'androidx.test.ext:junit:1.1.5'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'

def lottieVersion = "3.4.0"
implementation "com.airbnb.android:lottie:6.0.0"

implementation 'com.daimajia.androidanimations:library:2.4@aar'
```

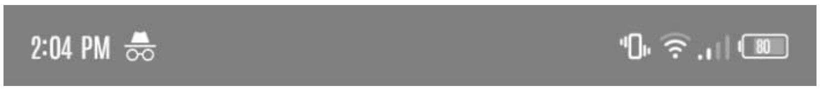
- com.daimajia.androidanimations:library:2.4@aar: This library provides a set of pre-defined animations, like bounce, fade, flip, etc., for Android views, which can be applied with minimal effort.

```
}
```

6. Themes.xml:

- Changed the status bar color:

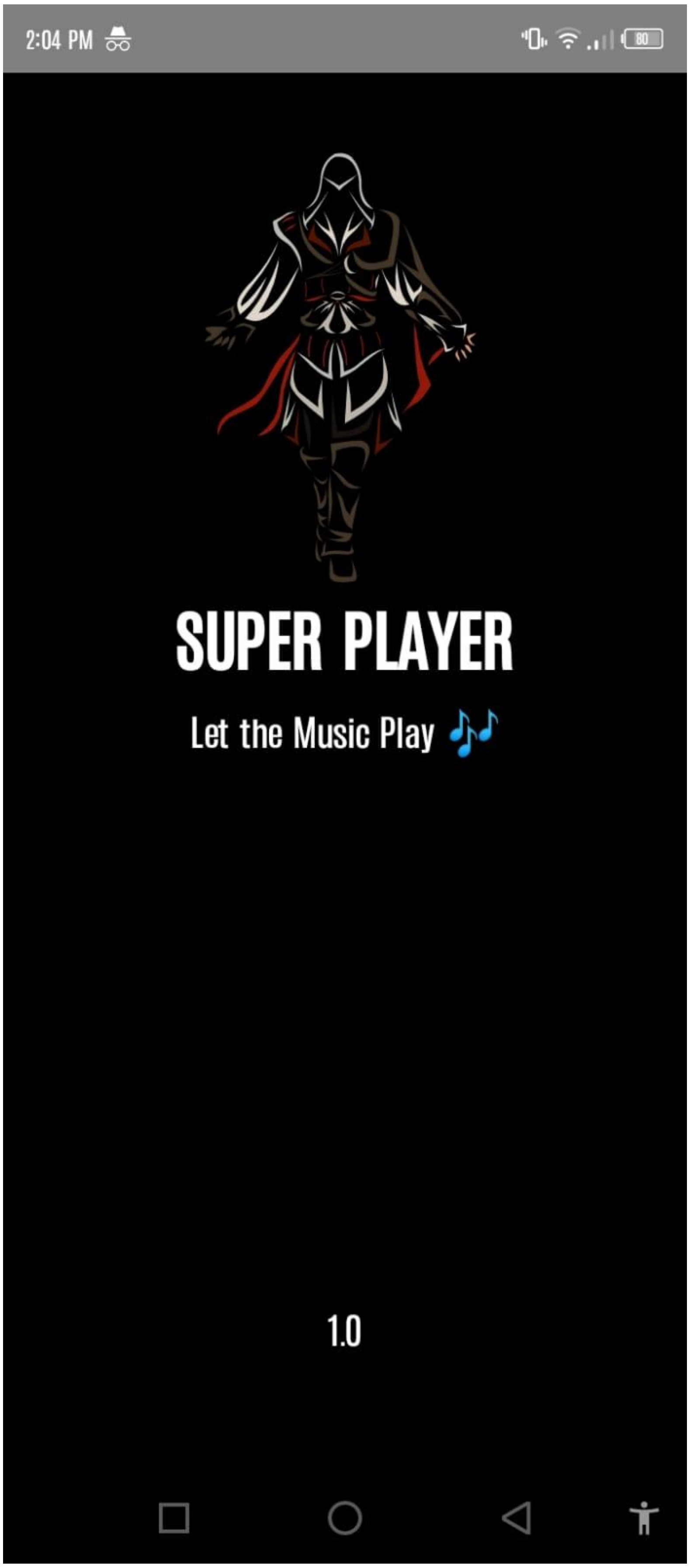
```
  <item name="android:windowLightStatusBar">true</item>
   <item name="android:statusBarColor">@android:color/tertiary_text_light</item>
</style>
```



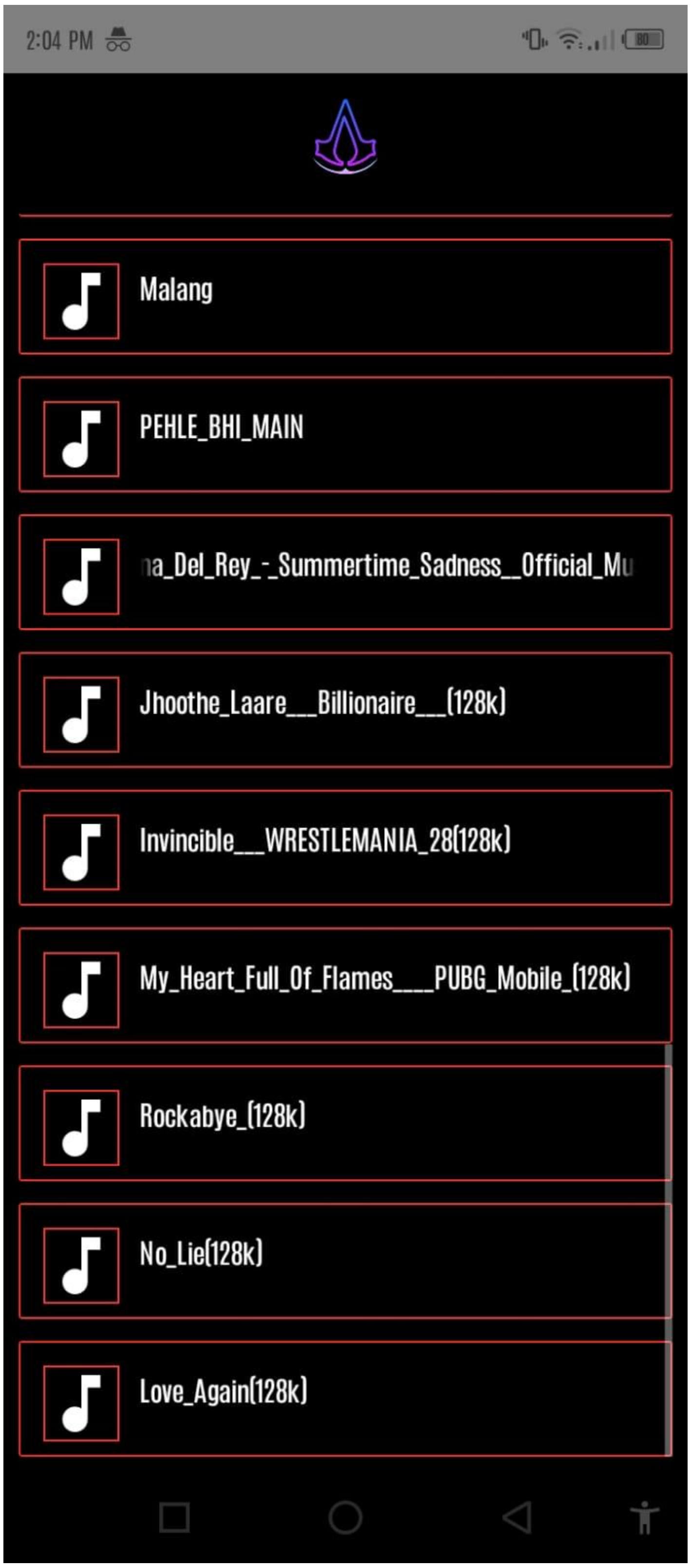
7. Output of the App:

On next page-------------→

- <u>Splashscreen:</u>



- <u>MainActivity:</u>


2:04 PM

Malang

PEHLE_BHI_MAIN

ıa_Del_Rey_-_Summertime_Sadness__Official_Mu

Jhoothe_Laare___Billionaire___(128k)

Invincible___WRESTLEMANIA_28(128k)

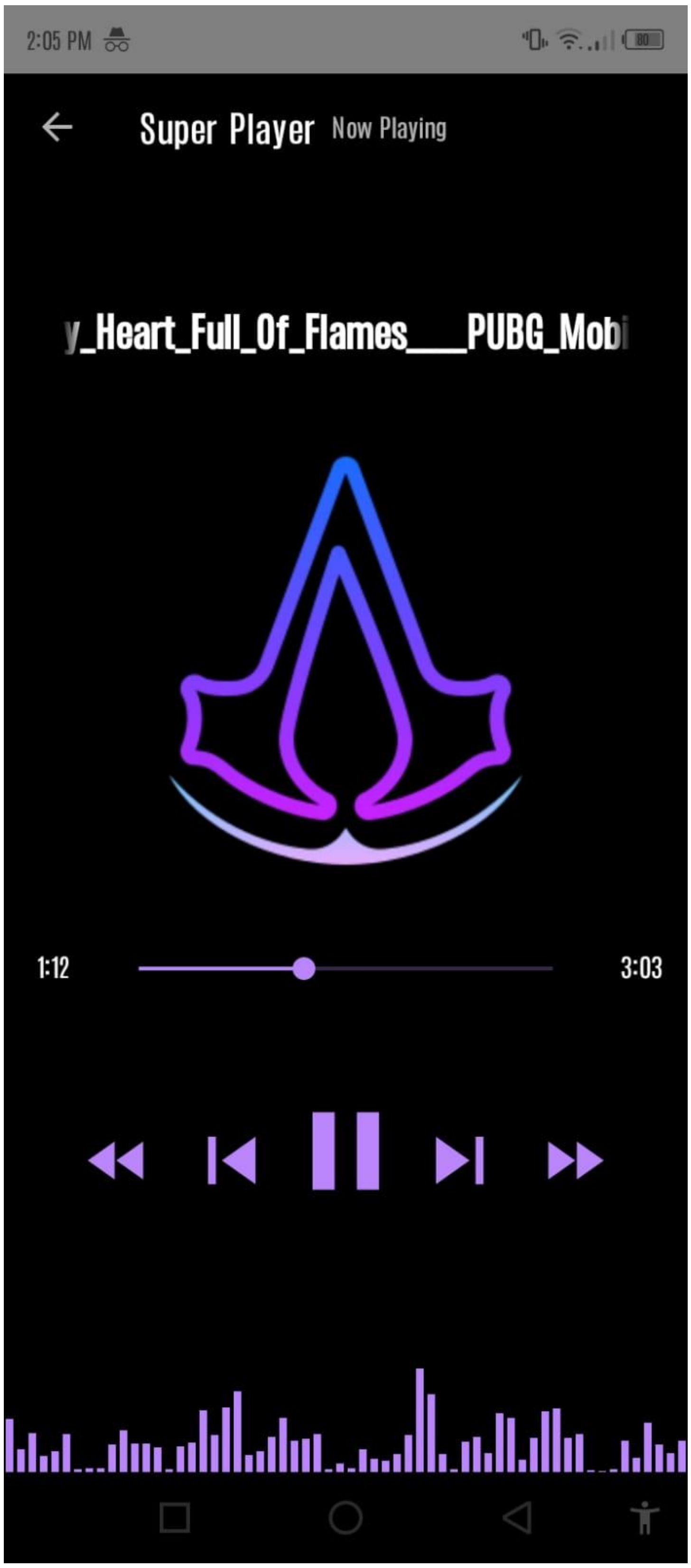
My_Heart_Full_Of_Flames____PUBG_Mobile_(128k)

Rockabye_(128k)

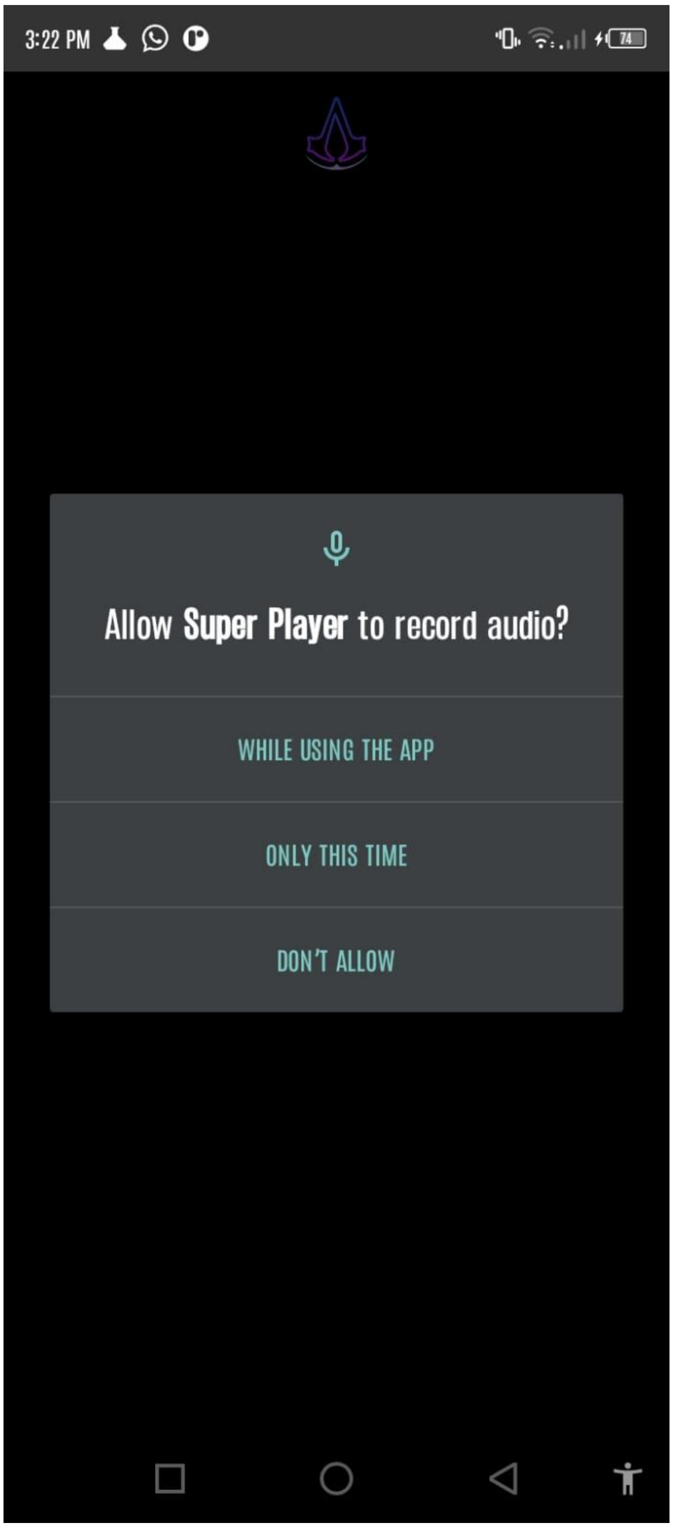
No_Lie(128k)

Love_Again(128k)

- <u>PlayerActivity:</u>



- **Permissions:**

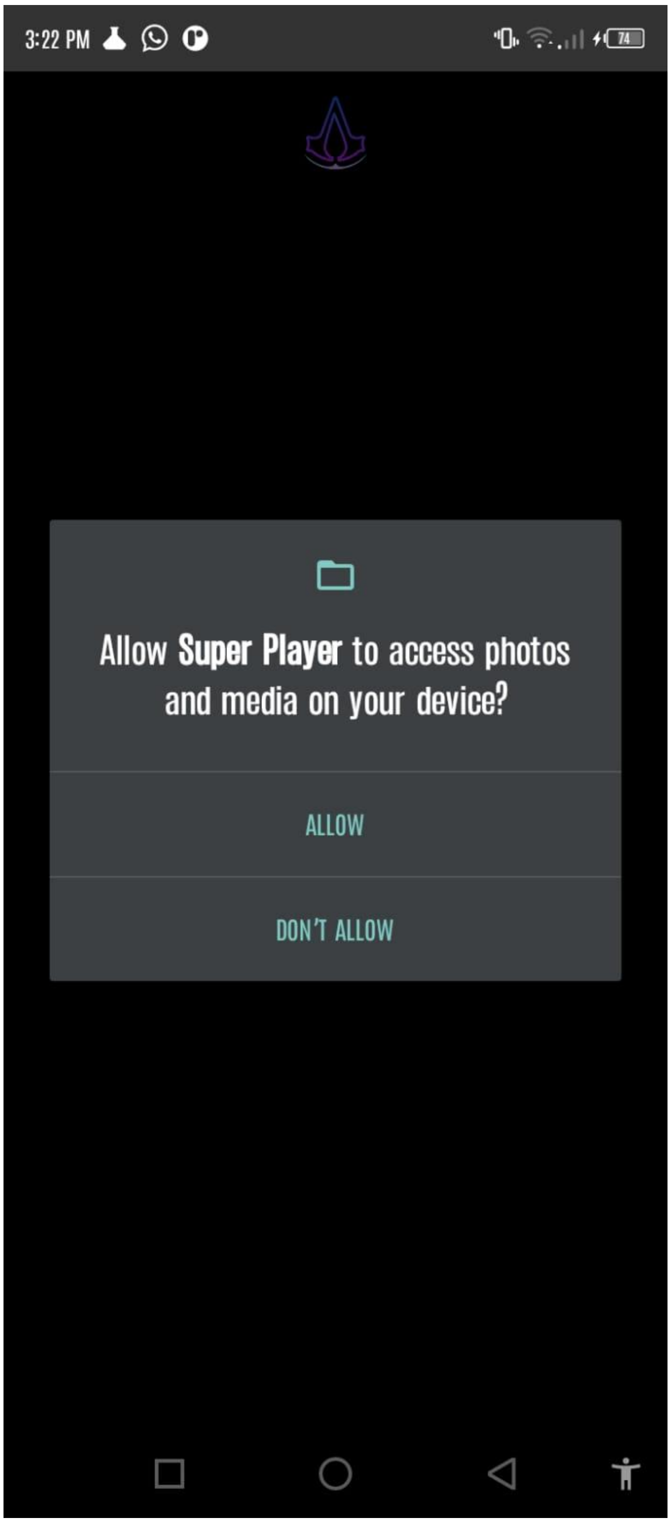

3:22 PM

Allow **Super Player** to record audio?

WHILE USING THE APP

ONLY THIS TIME

DON'T ALLOW


3:22 PM

Allow **Super Player** to access photos and media on your device?

ALLOW

DON'T ALLOW

THE END