THỰC HÀNH LẬP TRÌNH DI ĐỘNG 2022

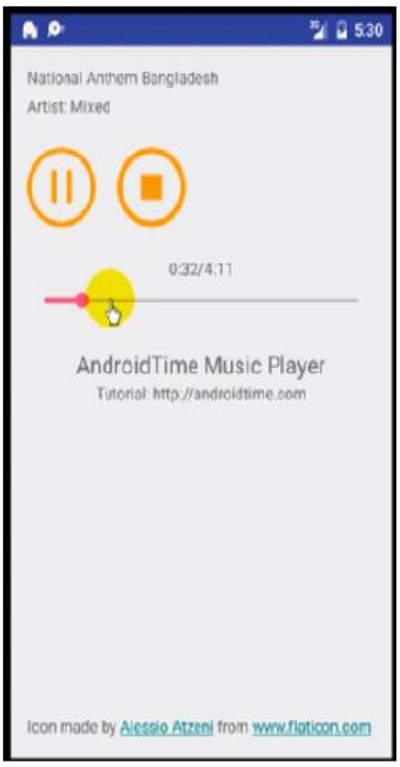
Lab 4: Notification and Service

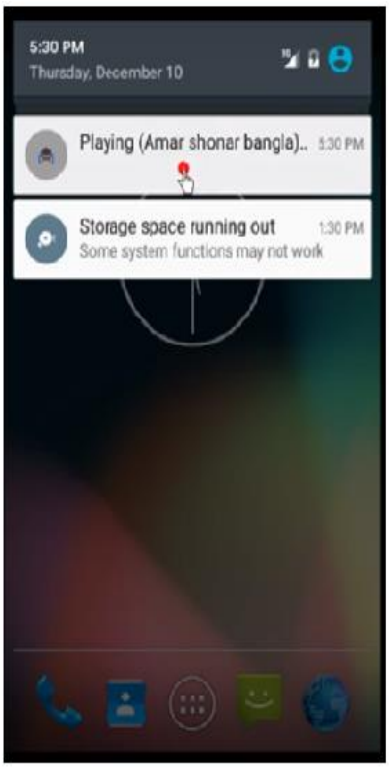
Yêu cầu:

- Thực hiện theo **nhóm thực hành 2-3 người** trong danh sách chia nhóm
- Nộp **project** có tên theo cú pháp *Lab4_NhomX* và file **video demo** cũng đặt tên theo cú pháp như trên, thêm **mô tả nhóm và thành viên** nhóm trong app

Nội dung:

- Xây dựng ứng dụng nghe nhạc theo gợi ý bên dưới, setup file nhạc sẵn trong điện thoại
- Thông tin logo dưới SeekBar thay thế bằng thông tin của bạn (Họ Tên / MSSV / Demo Service); icon nút play/pause/Stop tìm từ nguồn Internet
- Có thêm một nút hiển thị lời bài hát. Khi nhấn nút này thì toàn lời bài hát sẽ hiển thị phía dưới (Gợi ý dùng Text View và thực hiện không hiển thị lời bài hát bằng textviewobj.setVisibility(View.INVISIBLE); khi nhấn nút Show Lyrics thì hiển thi lời bài hát bằng textviewobj.setVisibility(View.VISIBLE);
- Có Nofitication cho biết bài hát đó đang chạy





Gợi ý:

A. REVIEW

1. Notification Example

Android **Toast** class provides a handy way to show users alerts but problem is that these alerts are not persistent which means alert flashes on the screen for a few seconds and then disappears.

Android notification message fills up the void in such situations. Android notification is a message that we can display to the user outside of our application's normal UI. Notifications in android are built using NotificationCompat library.

Creating Android Notification

A Notification is created using the NotificationManager class as shown below:

```
NotificationManager notificationManager = (NotificationManager)
  getSystemService(NOTIFICATION_SERVICE);
```

The Notification.Builder provides an builder interface to create an Notification object as shown below:

```
NotificationCompat.Builder mBuilder = new
NotificationCompat.Builder(this);
```

Android Notification Methods

We can set the notification properties on this builder object. Some of the frequently used methods and there descriptions are given below.

1. **Notification build()** : Combines all of the options that have been set and returns a new Notification object
2. **NotificationCompat.Builder setAutoCancel (boolean autoCancel)** : Setting this flag will make it such that the notification is automatically canceled when the user clicks it in the panel
3. **NotificationCompat.Builder setContent (RemoteViews views)** : Supplies a custom RemoteViews to use instead of the standard one
4. **NotificationCompat.Builder setContentInfo (CharSequence info)** : Sets the large text at the right-hand side of the notification

5. **NotificationCompat.Builder setContentIntent (PendingIntent intent)** : Supplies a PendingIntent to send when the notification is clicked

6. **NotificationCompat.Builder setContentText (CharSequence text)** : Sets the text (second row) of the notification, in a standard notification

7. **NotificationCompat.Builder setContentTitle (CharSequence title)** : Sets the text (first row) of the notification, in a standard notification

8. **NotificationCompat.Builder setDefaults (int defaults)** : Sets the default notification options that will be used. An example is;

9. mBuilder.setDefaults(Notification.DEFAULT_LIGHTS | Notification.DEFAULT_SOUND)

10. **NotificationCompat.Builder setLargeIcon (Bitmap icon)** : Sets the large icon that is shown in the ticker and notification

11. **NotificationCompat.Builder setNumber (int number)** : Sets the large number at the right-hand side of the notification

12. **NotificationCompat.Builder setOngoing (boolean ongoing)** : Sets whether this is an ongoing notification

13. **NotificationCompat.Builder setSmallIcon (int icon)** : Sets the small icon to use in the notification layouts

14. **NotificationCompat.Builder setStyle (NotificationCompat.Style style)** : Adds a rich notification style to be applied at build time

15. **NotificationCompat.Builder setTicker (CharSequence tickerText)** : Sets the text that is displayed in the status bar when the notification first arrives

16. **NotificationCompat.Builder setVibrate (long[] pattern)** : Sets the vibration pattern to use

17. **NotificationCompat.Builder setWhen (long when)** : Sets the time that the event occurred. Notifications in the panel are sorted by this time

Android Notification Button and Styles

The Notification.Builder allows you to add up to three buttons with definable actions to the notification.

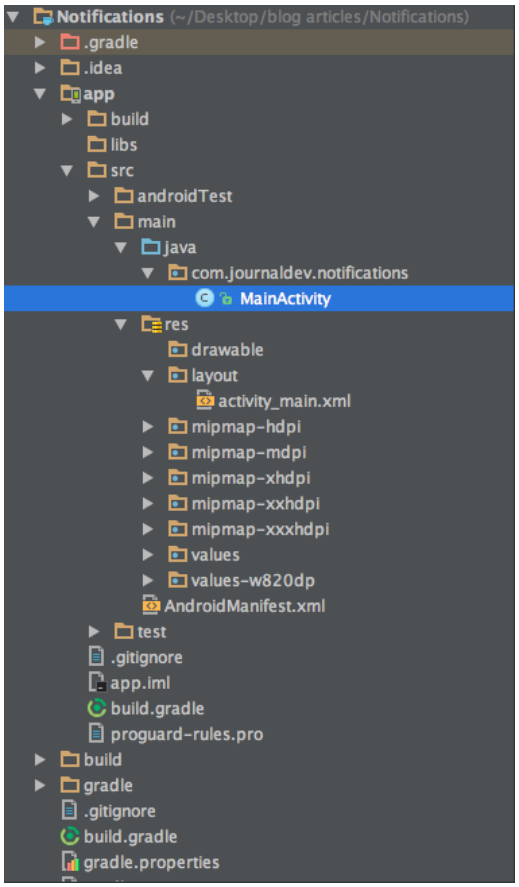
Android 4.1 and above support expandable notifications which shows a big view of the notification when it is expanded. There are three styles that can be used with the big view: **big picture style**, **big text style**, **Inbox style**.

Cancelling Android Notification

We can also call the cancel() for a specific notification ID on the NotificationManager. The cancelAll()method call removes all of the notifications you previously issued.

In this tutorial we'll create an application that wraps an intent that would view a webpage, into a PendingIntent. That PendingIntent would fire when the notification is tapped upon. Also we'll add the feature that cancels the notification programmatically.

Android Notification Example Project Structure



Android Notification Example

The activity_main.xml is a basic relative layout with two buttons : One to create a notification and the other to cancel it.

activity_main.xml:

```
<?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:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.journaldev.notifications.MainActivity">

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CREATE NOTIFICATION"
        android:id="@+id/button"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="CANCEL NOTIFICATION"
        android:id="@+id/button2"
        android:layout_below="@+id/button"
        android:layout_alignRight="@+id/button"
        android:layout_alignEnd="@+id/button"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />
</RelativeLayout>
```

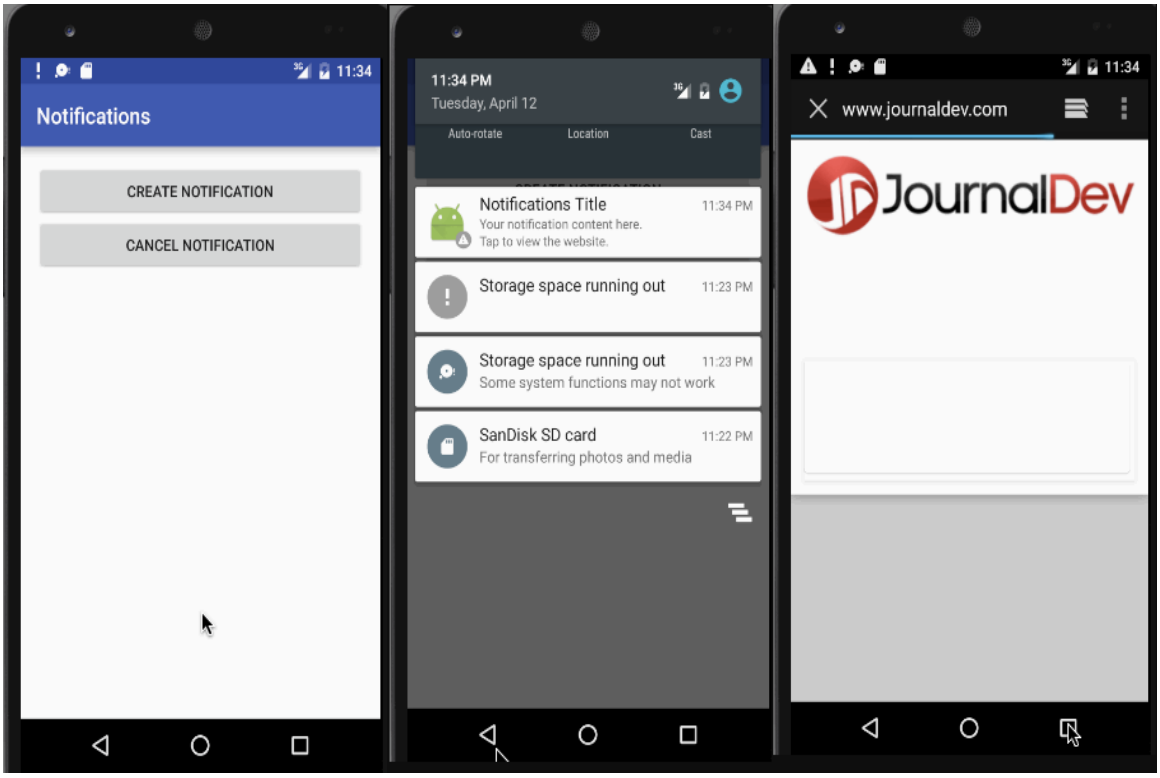
The MainActivity.java is given below.

```
package com.journaldev.notifications;

import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.app.NotificationCompat;
import android.widget.Toast;

import butterknife.ButterKnife;
import butterknife.OnClick;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ButterKnife.inject(this);
    }

    @OnClick(R.id.button)
    public void sendNotification() {

        NotificationCompat.Builder builder = new
NotificationCompat.Builder(this);
        builder.setSmallIcon(android.R.drawable.ic_dialog_alert);
        Intent intent = new Intent(Intent.ACTION_VIEW,
Uri.parse("https://www.journaldev.com/"));
        PendingIntent pendingIntent =
PendingIntent.getActivity(this, 0, intent, 0);
        builder.setContentIntent(pendingIntent);
```

```
builder.setLargeIcon(BitmapFactory.decodeResource(getResources(),
R.mipmap.ic_launcher));
        builder.setContentTitle("Notifications Title");
        builder.setContentText("Your notification content here.");
        builder.setSubText("Tap to view the website.");

        NotificationManager notificationManager =
(NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        // Will display the notification in the notification bar
        notificationManager.notify(1, builder.build());
    }

    @OnClick(R.id.button2)
    public void cancelNotification() {

        String ns = Context.NOTIFICATION_SERVICE;
        NotificationManager nMgr = (NotificationManager)
getApplicationContext().getSystemService(ns);
        nMgr.cancel(1);
    }
}
```

In the above code we've passed an intent of this website into the PendingIntent. The notificationId is set to 1 and it's used to build the notification and cancel it.

The output of the app in action is given below.



2. Android Service Example

Android user interface is restricted to perform long running jobs to make user experience smoother. A typical long running tasks can be periodic downloading of data from internet, saving multiple records into database, perform file I/O, fetching your phone contacts list, etc. For such long running tasks, Service is the alternative.

Bound Service

Service which call indefinitely in between activity. An Android component may bind itself to a Service using bindservice (). A bound service would run as long as the other application components are bound to it. As soon as they unbind, the service destroys itself.

Unbound Service

Service which call at the life span of calling activity. In this case, an application component starts the service, and it would continue to run in the background, even if the original component that initiated it is destroyed. For instance, when started, a service would continue to play music in the background indefinitely.

A service can be run by the system, If someone calls Context.startService() or bindService() method.

onStartCommand()

This method is called when the service be started, by calling startService(). Once this method executes, the service is started and can run in the background indefinitely. If you implement this, it is your responsibility to stop the service when its work is done, by calling stopSelf() or stopService(). If you are defining your service as, bounded service then you don't need to implement this method.

onBind()

You need to override this method, only if you are defining your service as bounded service. This method is called, when another component wants to bind with the service by calling bindService(). In your implementation of this method, you must provide an interface that clients use to communicate with the service, by returning an IBinder. You must always implement this method, but if you don't want to allow binding, then you should return null.

onCreate()

This method is called while the service is first created. Here all the service initialization is done. This method is never called again.

onDestroy()

The system calls this method when the service is no longer used and is being destroyed. This method is used to, clean up any resources such as threads, registered listeners, receivers, etc. This is the last call the service receives.

Creating a Android Service

Create a new class and extend it from android.app.Service. You need to override onStartCommand(), onBind(), onCreate() and onDestroy() method to handle the service lifecycle.

```
package com.javatechig.serviceexample;
import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.util.Log;
```

```
public class HelloService extends Service {

    private static final String TAG = "HelloService";
    private boolean isRunning  = false;
    @Override
    public void onCreate() {
        Log.i(TAG, "Service onCreate");
        isRunning = true;
    }
    @Override
    public int onStartCommand(Intent intent, int flags, int startId)
{
        Log.i(TAG, "Service onStartCommand");
        //Creating new thread for my service
        //Always write your long running tasks in a separate thread,
to avoid ANR
        new Thread(new Runnable() {
            @Override
            public void run() {
                //Your logic that service will perform will be placed
here
                //In this example we are just looping and waits for
1000 milliseconds in each loop.
                for (int i = 0; i < 5; i++) {
                    try {
                        Thread.sleep(1000);
                    } catch (Exception e) {
                    }
                    if(isRunning){
                        Log.i(TAG, "Service running");
                    }
                }
                //Stop service once it finishes its task
                stopSelf();
            }
        }).start();
        return Service.START_STICKY;
```

```
    }
    @Override
    public IBinder onBind(Intent arg0) {
        Log.i(TAG, "Service onBind");
        return null;
    }
    @Override
    public void onDestroy() {
        isRunning = false;
        Log.i(TAG, "Service onDestroy");
    }
}
```

Service Manifest Declaration

In theory, A service can be called from other application unless it is restricted. You can ensure that your service is available to only your app by including the android:exported attribute and setting it to “false”. This effectively stops other apps from starting your service, even when using an explicit intent.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javatechig.serviceexample" >
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".HelloActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

```
        <!--Service declared in manifest -->
        <service android:name=".HelloService"
            android:exported="false"/>
    </application>

</manifest>
```

Starting Android Service

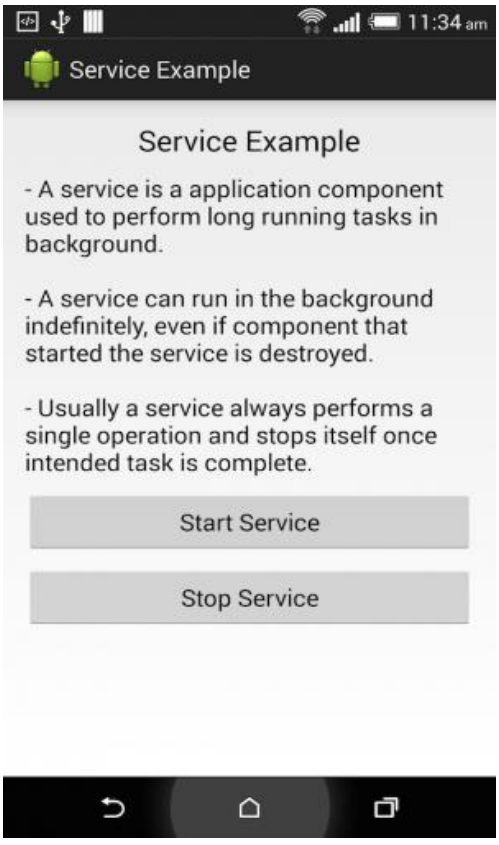
You can start a service from an activity or other application component by passing an Intent to startService(). The Android system calls the service's onStartCommand() method and passes it the Intent.

In our example, we will start service by calling startService() method while start service button is clicked.

```
Intent intent = new Intent(this, HelloService.class);
startService(intent);
```

Stop Running Android Service

- A service must be stopped itself by calling stopSelf() method, once it is finishes execution. However, you can also stop a service yourself by calling stopService() method.
- A call to stopService method will call onDestroy() callback in your service. You have to manually stop your operation being performed by your application. To do this in the above example, we have taken a boolean variable to control the execution of service.



B. INSTRUCTION

1. Create a class named **MainActivity.class** under the main package like **com.androidtime.net** then write the below code into the MainActivity.class

```
package com.andriodtime.net;

/**
 * Author Md. Nazmul Hasan Masum
 * Background media player using service
 */

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.text.method.LinkMovementMethod;
import android.util.Log;
import android.widget.ImageButton;
import android.widget.SeekBar;
import android.widget.TextView;

public class MainActivity extends Activity {
    public static ImageButton btnPlay, btnStop;
    public static TextView textViewSongTime;
    private Intent playerService;
    public static SeekBar seekBar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
        playerService = new Intent(MainActivity.this, PlayerInService.class);
        startService(playerService);

    }
```

```
    private void initView() {
        btnPlay = (ImageButton) findViewById(R.id.btnPlay);
        btnStop = (ImageButton) findViewById(R.id.btnStop);
        seekBar = (SeekBar) findViewById(R.id.seekBar);
        textViewSongTime = (TextView) findViewById(R.id.textViewSongTime);

        TextView textViewIconCopyright = (TextView) findViewById(R.id.textViewIconCopyright);
        textViewIconCopyright.setText(Html.fromHtml("Icon made by " +
                "<a href="\&quot;http://www.flaticon.com/authors/alessio-atzeni\&quot;">Alessio Atzeni</a> " +
                "from " + "<a href="\&quot;http://www.flaticon.com\&quot;">www.flaticon.com</a> "));
        textViewIconCopyright.setMovementMethod(LinkMovementMethod.getInstance());
    }

    @Override
    protected void onDestroy() {
        if (!PlayerInService.mp.isPlaying()) {
            PlayerInService.mp.stop();
            stopService(playerService);
        } else {
            btnPlay.setBackgroundResource(R.drawable.pause);
        }
        super.onDestroy();
    }
```

```
    @Override
    protected void onResume() {
        try {
            if (!PlayerInService.mp.isPlaying()) {
                btnPlay.setBackgroundResource(R.drawable.player);

            } else {
                btnPlay.setBackgroundResource(R.drawable.pause);
            }
        } catch (Exception e) {
            Log.e("Exception", "" + e.getMessage() + e.getStackTrace() + e.getCause());
        }

        super.onResume();

    }


}
```

2. Create a class named **Utility.class** under the main package **com.androidtime.net** then write the below code into the Utility.class

```
package com.andriodtime.net;

import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;

/**
 * Created by masum on 09/12/2015.
 */
public class Utility {
    //Notification
    // Set up the notification ID
    public static final int NOTIFICATION_ID = 1;
    @SuppressWarnings("unused")
    public static NotificationManager mNotificationManager;
```

```
    // Create Notification
    @SuppressWarnings("deprecation")
    public static void initNotification(String songTitle, Context mContext) {
        try {
            String ns = Context.NOTIFICATION_SERVICE;
            mNotificationManager = (NotificationManager) mContext.getSystemService(ns);
            int icon = R.drawable.music_showcase;
            CharSequence tickerText = "Playing your song";
            long when = System.currentTimeMillis();

            Notification notification = new Notification(icon, tickerText, when);
            notification.flags = Notification.FLAG_ONGOING_EVENT;
            Context context = mContext.getApplicationContext();
            CharSequence songName = "" + songTitle;

            Intent notificationIntent = new Intent(mContext, MainActivity.class);
            PendingIntent contentIntent = PendingIntent.getActivity(context, 0, notificationIntent, 0);
            notification.setLatestEventInfo(context, songName, null, contentIntent);
            mNotificationManager.notify(NOTIFICATION_ID, notification);

        } catch (Exception e) {
            e.printStackTrace();
        }

    }
```

```
    public static int getProgressPercentage(long currentDuration, long totalDuration) {
        Double percentage = (double) 0;
        long currentSeconds = (int) (currentDuration / 1000);
        long totalSeconds = (int) (totalDuration / 1000);

        // calculating percentage
        percentage = (((double) currentSeconds) / totalSeconds) * 100;

        // return percentage
        return percentage.intValue();
    }

    public static int progressToTimer(int progress, int totalDuration) {
        int currentDuration = 0;
        totalDuration = (int) (totalDuration / 1000);
        currentDuration = (int) ((((double) progress) / 100) * totalDuration);

        // return current duration in milliseconds
        return currentDuration * 1000;
    }
```

```
    public static String milliSecondsToTimer(long milliseconds) {
        String finalTimerString = "";
        String secondsString = "";

        //Convert total duration into time
        int hours = (int) (milliseconds / (1000 * 60 * 60));
        int minutes = (int) (milliseconds % (1000 * 60 * 60)) / (1000 * 60);
        int seconds = (int) ((milliseconds % (1000 * 60 * 60)) % (1000 * 60) / 1000);
        // Add hours if there
        if (hours &gt; 0) {
            finalTimerString = hours + ":";
        }

        // Pre appending 0 to seconds if it is one digit
        if (seconds &lt; 10) {
            secondsString = "0" + seconds;
        } else {
            secondsString = "" + seconds;
        }

        finalTimerString = finalTimerString + minutes + ":" + secondsString;

        // return timer string
        return finalTimerString;
    }

}
```

3. Create a class named **PlayerInService.class** under the main package **com.androidtime.net** then write the below code into the PlayerInService.class

```
package com.andriodtime.net;

import android.app.Service;
import android.content.Intent;
import android.media.AudioManager;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnPreparedListener;
import android.net.Uri;
import android.os.Handler;
import android.os.IBinder;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ImageButton;
import android.widget.SeekBar;
import android.widget.TextView;

import java.lang.ref.WeakReference;

public class PlayerInService extends Service implements OnClickListener, MediaPlayer.OnCompletionListener, SeekBar.OnSeekBarChangeListener {
        private WeakReference btnPlay;
        private WeakReference btnStop;
        public static WeakReference textViewSongTime;
        public static WeakReference songProgressBar;
        static Handler progressBarHandler = new Handler();

        public static MediaPlayer mp;
        private boolean isPause = false;
```

```
        @Override
        public void onCreate() {
                mp = new MediaPlayer();
                mp.reset();
                mp.setAudioStreamType(AudioManager.STREAM_MUSIC);
                super.onCreate();
        }

        @SuppressWarnings("deprecation")
        @Override
        public int onStartCommand(Intent intent, int flags, int startId) {
                initUI();
                super.onStart(intent, startId);
                return START_STICKY;
        }

        @Override
        public IBinder onBind(Intent intent) {
                return null;
        }

        private void initUI() {
                btnPlay = new WeakReference&lt;&gt;(MainActivity.btnPlay);
                btnStop = new WeakReference&lt;&gt;(MainActivity.btnStop);
        textViewSongTime = new WeakReference&lt;&gt;(MainActivity.textViewSongTime);
                songProgressBar = new WeakReference&lt;&gt;(MainActivity.seekBar);
                songProgressBar.get().setOnSeekBarChangeListener(this);
                btnPlay.get().setOnClickListener(this);
                btnStop.get().setOnClickListener(this);
        mp.setOnCompletionListener(this);

        }
```

```
       public void onClick(View v) {
              switch (v.getId()) {
              case R.id.btnPlay:
                     if (mp.isPlaying()) {
                            mp.pause();
                            isPause = true;
              progressBarHandler.removeCallbacks(mUpdateTimeTask);
                            btnPlay.get().setBackgroundResource(R.drawable.player);
                            return;
                     }

                     if (isPause) {
                            mp.start();
                            isPause = false;
              updateProgressBar();
                            btnPlay.get().setBackgroundResource(R.drawable.pause);
                            return;
                     }

                     if (!mp.isPlaying()) {
                            playSong();
                     }

                     break;
              case R.id.btnStop:
                     mp.stop();
          onCompletion(mp);
          textViewSongTime.get().setText("0.00/0.00"); // Displaying time completed playing
                     break;

              }
       }
```

```
	public void updateProgressBar(){
		try{
			progressBarHandler.postDelayed(mUpdateTimeTask, 100);
		}catch(Exception e){

		}
	}

	static Runnable mUpdateTimeTask = new Runnable() {
		public void run(){
			long totalDuration = 0;
			long currentDuration = 0;

			try {
				totalDuration = mp.getDuration();
				currentDuration = mp.getCurrentPosition();
		textViewSongTime.get().setText(Utility.milliSecondsToTimer(currentDuration) + "/" + Utility.milliSecondsToTimer(totalDuration)); /
				int progress = (int)(Utility.getProgressPercentage(currentDuration, totalDuration));
				songProgressBar.get().setProgress(progress);    /* Running this thread after 100 milliseconds */
		progressBarHandler.postDelayed(this, 100);

			} catch(Exception e){
				e.printStackTrace();
			}

		}
	};

	@Override
	public void onDestroy() {

	}
```

```
	  // Play song
	  public void playSong() {
			Utility.initNotification("Playing (Amar shonar bangla)...", this);
			try {
				mp.reset();
				Uri myUri = Uri.parse("android.resource://" + this.getPackageName() + "/" + R.raw.bangla);
				mp.setDataSource(this, myUri);
				mp.prepareAsync();
				mp.setOnPreparedListener(new OnPreparedListener() {
			public void onPrepared(MediaPlayer mp) {
			    try {
				mp.start();
				updateProgressBar();
				btnPlay.get().setBackgroundResource(R.drawable.pause);
			    } catch (Exception e) {
				Log.i("EXCEPTION", "" + e.getMessage());
			    }
			}
		    });

			} catch (Exception e) {
				e.printStackTrace();
			}
	    }
```

```
	@Override
	public void onCompletion(MediaPlayer mp){
	    songProgressBar.get().setProgress(0);
	    progressBarHandler.removeCallbacks(mUpdateTimeTask); /* Progress Update stop */
	    btnPlay.get().setBackgroundResource(R.drawable.player);
	}

	    @Override
	    public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {

	    }

	    @Override
	    public void onStartTrackingTouch(SeekBar seekBar) {
	    progressBarHandler.removeCallbacks(mUpdateTimeTask);
	    }

	    @Override
	    public void onStopTrackingTouch(SeekBar seekBar) {
	    progressBarHandler.removeCallbacks(mUpdateTimeTask);
	    int totalDuration = mp.getDuration();
	    int currentPosition = Utility.progressToTimer(seekBar.getProgress(),totalDuration);
	    mp.seekTo(currentPosition);
	    updateProgressBar();
	    }
}
```

4. Create **activity_main.xml** file under the res>layout folder

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="16dp">

    <ImageButton
        android:id="@+id/btnPlay"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="46dp"
        android:background="@drawable/player"
        android:layout_below="@+id/textView"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <ImageButton
        android:id="@+id/btnStop"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/btnPlay"
        android:layout_marginLeft="21dp"
        android:layout_marginStart="21dp"
        android:layout_toEndOf="@+id/btnPlay"
        android:layout_toRightOf="@+id/btnPlay"
        android:background="@drawable/stop" />

    <SeekBar
        android:id="@+id/seekBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_below="@+id/textViewSongTime"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="National Anthem Bangladesh"
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Artist: Mixed"
        android:id="@+id/textView2"
        android:layout_marginTop="4dp"
        android:layout_below="@+id/textView"
        android:layout_alignLeft="@+id/textView"
        android:layout_alignStart="@+id/textView"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="0.00/0.00"
        android:id="@+id/textViewSongTime"
        android:layout_marginTop="23dp"
        android:gravity="center"
        android:layout_below="@+id/btnStop"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true" />
```

```
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Tutorial: http://androidtime.com"
        android:id="@+id/textView3"
        android:layout_below="@+id/textView4"
        android:layout_centerHorizontal="true" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="AndroidTime Music Player"
        android:id="@+id/textView4"
        android:textSize="20sp"
        android:layout_below="@+id/seekBar"
        android:layout_centerHorizontal="true"
        android:layout_marginTop="21dp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Icon made by Alessio Atzeni from www.flaticon.com"
        android:id="@+id/textViewIconCopyright"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true" />

</RelativeLayout>
```

5. Finally **manifest.xml** file look like this with adding service **PlayerInService**

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <service android:name=".PlayerInService" />

    </application>

</manifest>
```