EXPERIMENT: 9

BINCY .C B

MCA S4

ROLL NO:15

AIM :Create an app to implement Android Service for any Background Processes

Create a background service The IntentService class provides a straightforward structure for running an operation on a single background thread. This allows it to handle long-running operations without affecting your user interface's responsiveness. Also, an IntentService isn't affected by most user interface lifecycle events, so it continues to run in circumstances that would shut down an AsyncTask **An IntentService has a few limitations:**

• It can't interact directly with your user interface. To put its results in the UI, you have to send them to an Activity.

• Work requests run sequentially. If an operation is running in an IntentService, and you send it another request, the request waits until the first operation is finished.

• An operation running on an IntentService can't be interrupted. However, in most cases an IntentService is the preferred way to perform simple background operations. This shows you how to do the following things:

• Create your own subclass of IntentService.

• Create the required callback method onHandleIntent().

• Define the IntentService in your manifest file.

Handle incoming intents

To create an IntentService component for your app, define a class that extends IntentService, and within it, define a method that overrides onHandleIntent(). For example:

```
public class RSSPullService extends IntentService {
@Override
protected void onHandleIntent(Intent workIntent) {
// Gets data from the incoming Intent
String dataString = workIntent.getDataString();
...
// Do work here, based on the contents of dataString
...
}
}
```

Notice that the other callbacks of a regular Service component, such as onStartCommand() are automatically invoked by IntentService. In an IntentService, you should avoid overriding these callbacks. Define the intent service in the manifest

An IntentService also needs an entry in your application manifest. Provide this entry as a <service>element that's a child of the <application> element:

<application

android:icon="@drawable/icon"

android:label="@string/app_name">

...

<!--

Because android:exported is set to "false",

the service is only available to this app.

-->

<service

android:name=".RSSPullService"

android:exported="false"/>

...

</application>

The attribute android:name specifies the class name of the IntentService.Notice that the element doesn't contain an intent filter. The Activity that sends work requests to the service uses an explicit Intent, so no filter is needed. This also means that only components in the same app or other applications with the same user ID can access the service.

Now that you have the basic IntentService class, you can send work requests to it with Intent objects. The procedure for constructing these objects and sending them to your IntentService is described in Send work requests to the background service.

Service is going to do background operation without interacting with UI and it works even after activity destroy.ss

This example demonstrates what is Android background music service.

Step 1 – Create a new project in Android Studio, go to File ⇒ New Project and fill all required

details to create a new project.

Step 2 – Add the following code to res/layout/activity_main.xml.

```
<?xml version = "1.0" encoding = "utf-8"?>

<android.support.constraint.ConstraintLayout 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"

   tools:context = ".MainActivity">

   <TextView

      android:id = "@+id/text"

      android:layout_width = "wrap_content"

      android:layout_height = "wrap_content"

      android:text = "Start Service"

      android:textSize = "25sp"

      app:layout_constraintBottom_toBottomOf = "parent"

      app:layout_constraintLeft_toLeftOf = "parent"

      app:layout_constraintRight_toRightOf = "parent"

      app:layout_constraintTop_toTopOf = "parent" />

</android.support.constraint.ConstraintLayout>
```

In the above code, we have taken text view, when user click on text view, it will start music service and stop music service.

Step 3 – Add the following code to src/MainActivity.java

```
package com.example.andy.myapplication;

import android.app.ActivityManager;

import android.content.Context;
```

```
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final TextView text = findViewById(R.id.text);
    text.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        if (isMyServiceRunning(service.class)) {
          text.setText("Stoped");
          stopService(new Intent(MainActivity.this, service.class));
        } else {
```

```
            text.setText("Started");

            startService(new Intent(MainActivity.this, service.class));

          }

        }

    });

  }

  private boolean isMyServiceRunning(Class<?> serviceClass) {

    ActivityManager manager = (ActivityManager)
getSystemService(Context.ACTIVITY_SERVICE);

    for (ActivityManager.RunningServiceInfo service :
manager.getRunningServices(Integer.MAX_VALUE)) {

      if (serviceClass.getName().equals(service.service.getClassName())) {

        return true;

      }

    }

    return false;

  }

}
```

In the above code to start and stop service. We have used intent and passed context and service class. Now create a service class in package folder as service.class and add the following code –

```
package com.example.andy.myapplication;
```

```
import android.app.Service;
import android.content.Intent;
import android.media.MediaPlayer;
import android.os.IBinder;
import android.widget.Toast;
public class service extends Service {
  MediaPlayer musicPlayer;
  @Override
  public IBinder onBind(Intent intent) {
    return null;
  }
  @Override
  public void onCreate() {
    super.onCreate();
    musicPlayer = MediaPlayer.create(this, R.raw.abc);
    musicPlayer.setLooping(false);
  }
  @Override
  public int onStartCommand(Intent intent, int flags, int startId) {
```

```
    Toast.makeText(this, "Music Service started by user.", Toast.LENGTH_LONG).show();
    musicPlayer.start();
    return START_STICKY;
  }
  @Override
  public void onDestroy() {
    super.onDestroy();
    musicPlayer.stop();
    Toast.makeText(this, "Music Service destroyed by user.", Toast.LENGTH_LONG).show();
  }
}
```

In the above code we have music Player class and started music player and stopped music player as shown below –

```
@Override
public void onCreate() {
  super.onCreate();
  musicPlayer = MediaPlayer.create(this, R.raw.abc);
  musicPlayer.setLooping(false);
}
@Override
```

```
public int onStartCommand(Intent intent, int flags, int startId) {
  Toast.makeText(this, "Music Service started by user.", Toast.LENGTH_LONG).show();
  musicPlayer.start();
  return START_STICKY;
}
@Override
public void onDestroy() {
  super.onDestroy();
  musicPlayer.stop();
  Toast.makeText(this, "Music Service destroyed by user.", Toast.LENGTH_LONG).show();
}
```

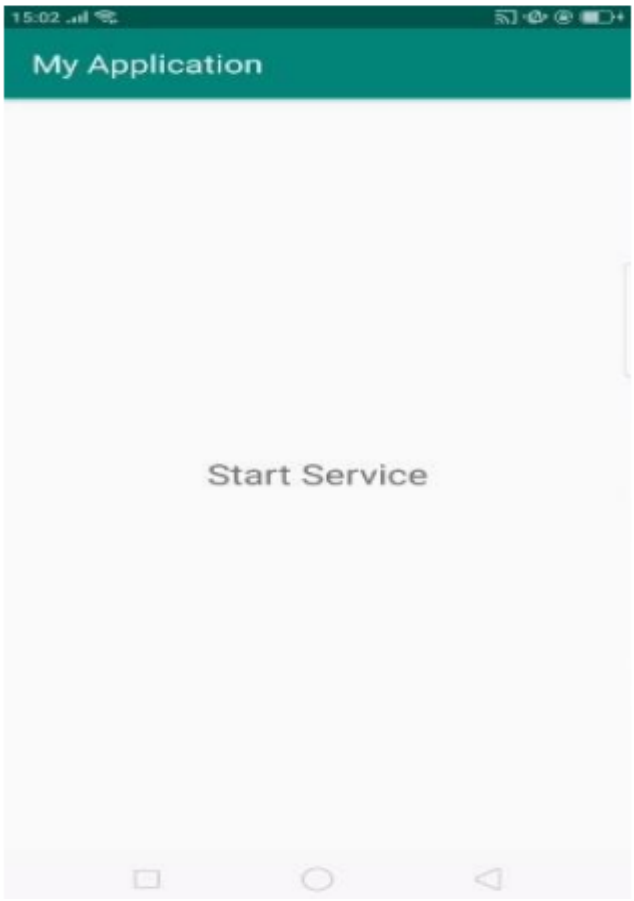
Step 4 − Add the following code to manifest.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<manifest xmlns:android = "http://schemas.android.com/apk/res/android"
  package = "com.example.andy.myapplication">
  <application
    android:allowBackup = "true"
    android:icon = "@mipmap/ic_launcher"
    android:label = "@string/app_name"
```

```
    android:roundIcon = "@mipmap/ic_launcher_round"
    android:supportsRtl = "true"
    android:theme − "@style/AppTheme">
    <activity android:name = ".MainActivity">
      <intent-filter>
        <action android:name = "android.intent.action.MAIN" />
        <category android:name = "android.intent.category.LAUNCHER" />
      </intent-filter>
    </activity>
    <service android:name = ".service"/>
  </application>
</manifest>
```

Let's try to run your application. I assume you have connected your actual Android Mobile device with your computer. To run the app from android studio, open one of your project's activity files and click Run icon from the toolbar. Select your mobile device as an option and then check your mobile device which will display your default screen –

OUTPUT



In the above result is an initial screen, Click on Text view, it will start music service as shown below-

In the above result, service is startd now click on text view, it will stop music service as shown below-