**Code:**

**activity\_main.xml**

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center\_horizontal"

android:orientation="vertical">

<TextView

android:id="@+id/textView\_FileName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginTop="50dp"

android:text="Please select file"

android:textSize="18sp" />

<Button

android:id="@+id/button\_GetFile"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="getFile"

android:text="Get file from device" />

<Button

android:id="@+id/button\_SendFile"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:onClick="sendViaBluetooth"

android:text="Send via Bluetooth" />

</LinearLayout>

**MainActivity.java**

package com.example.bluetoothtransfer;

import android.Manifest;

import android.app.Activity;

import android.bluetooth.BluetoothAdapter;

import android.content.ContentUris;

import android.content.Context;

import android.content.Intent;

import android.content.pm.PackageManager;

import android.content.pm.ResolveInfo;

import android.database.Cursor;

import android.net.Uri;

import android.os.Build;

import android.os.Bundle;

import android.os.Environment;

import android.provider.DocumentsContract;

import android.provider.MediaStore;

import android.support.v4.content.ContextCompat;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.TextView;

import android.widget.Toast;

import java.io.File;

import java.util.List;

public class MainActivity extends AppCompatActivity {

private static final int *DISCOVER\_DURATION* = 300;

private static final int *REQUEST\_BLU* = 1;

String path;

private static final String[] *INITIAL\_PERMS* = {Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*,

Manifest.permission.*READ\_EXTERNAL\_STORAGE*,

Manifest.permission.*READ\_CONTACTS*,

Manifest.permission.*WRITE\_CONTACTS*,

Manifest.permission.*CAMERA*,

Manifest.permission.*ACCESS\_FINE\_LOCATION*};

private static final int *INITIAL\_REQUEST* = 1337;

private static final int *REQUEST\_WRITE\_STORAGE* = *INITIAL\_REQUEST* + 4;

TextView textView\_FileName;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.*activity\_main*);

textView\_FileName = (TextView) findViewById(R.id.*textView\_FileName*);

if (!canAccessLocation() || !canAccessCamera() || !canAccessWriteStorage() || !canAccessReadStorage() || !canAccessReadContacts() || !canAccessWriteContacts()) {

if (Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*M*) {

requestPermissions(*INITIAL\_PERMS*, *INITIAL\_REQUEST*);

}

}

}

@Override

public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {

switch (requestCode) {

case REQUEST\_WRITE\_STORAGE:

if (canAccessWriteStorage()) {

//reload my activity with permission granted or use the features what required the permission

System.*out*.println("permission grantedddd");

} else {

Toast.*makeText*(this, "The app was not allowed to write to your storage. Hence, it cannot function properly. Please consider granting it this permission", Toast.*LENGTH\_LONG*).show();

}

break;

}

}

public void sendViaBluetooth(View v) {

if (path == null) {

Toast.*makeText*(this, "Please select file first", Toast.*LENGTH\_SHORT*).show();

return;

}

BluetoothAdapter btAdapter = BluetoothAdapter.*getDefaultAdapter*();

if (btAdapter == null) {

Toast.*makeText*(this, "Bluetooth is not supported on this device", Toast.*LENGTH\_LONG*).show();

} else {

enableBluetooth();

}

}

public void getFile(View v) {

Intent mediaIntent = new Intent(Intent.*ACTION\_GET\_CONTENT*);

mediaIntent.setType("\*/\*"); //set mime type as per requirement

startActivityForResult(mediaIntent, 1001);

}

public void enableBluetooth() {

Intent discoveryIntent = new Intent(BluetoothAdapter.*ACTION\_REQUEST\_DISCOVERABLE*);

discoveryIntent.putExtra(BluetoothAdapter.*EXTRA\_DISCOVERABLE\_DURATION*, *DISCOVER\_DURATION*);

startActivityForResult(discoveryIntent, *REQUEST\_BLU*);

}

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (resultCode == *DISCOVER\_DURATION* && requestCode == *REQUEST\_BLU*) {

Intent intent = new Intent();

intent.setAction(Intent.*ACTION\_SEND*);

intent.setType("\*/\*");

File f = new File(path);

intent.putExtra(Intent.*EXTRA\_STREAM*, Uri.*fromFile*(f));

PackageManager pm = getPackageManager();

List<ResolveInfo> appsList = pm.queryIntentActivities(intent, 0);

if (appsList.size() > 0) {

String packageName = null;

String className = null;

boolean found = false;

for (ResolveInfo info : appsList) {

packageName = info.activityInfo.packageName;

if (packageName.equals("com.android.bluetooth")) {

className = info.activityInfo.name;

found = true;

break;

}

}

if (!found) {

Toast.*makeText*(this, "Bluetooth havn't been found",

Toast.*LENGTH\_LONG*).show();

} else {

intent.setClassName(packageName, className);

startActivity(intent);

}

}

} else if (requestCode == 1001

&& resultCode == Activity.*RESULT\_OK*) {

Uri uriPath = data.getData();

Log.*d*("", "Video URI= " + uriPath);

path = *getPath*(this, uriPath);// "/mnt/sdcard/FileName.mp3"

System.*out*.println("pathhhh " + path);

textView\_FileName.setText(path);

} else {

Toast.*makeText*(this, "Bluetooth is cancelled", Toast.*LENGTH\_LONG*)

.show();

}

}

private boolean canAccessWriteStorage() {

return (hasPermission(Manifest.permission.*WRITE\_EXTERNAL\_STORAGE*));

}

private boolean canAccessReadStorage() {

return (hasPermission(Manifest.permission.*READ\_EXTERNAL\_STORAGE*));

}

private boolean canAccessReadContacts() {

return (hasPermission(Manifest.permission.*READ\_CONTACTS*));

}

private boolean canAccessWriteContacts() {

return (hasPermission(Manifest.permission.*WRITE\_CONTACTS*));

}

private boolean canAccessCamera() {

return (hasPermission(Manifest.permission.*CAMERA*));

}

private boolean canAccessLocation() {

return (hasPermission(Manifest.permission.*ACCESS\_FINE\_LOCATION*));

}

private boolean hasPermission(String perm) {

return (PackageManager.*PERMISSION\_GRANTED* == ContextCompat.*checkSelfPermission*(this, perm));

}

public static String getPath(final Context context, final Uri uri) {

final boolean isKitKatOrAbove = Build.VERSION.*SDK\_INT* >= Build.VERSION\_CODES.*KITKAT*;

// DocumentProvider

if (isKitKatOrAbove && DocumentsContract.*isDocumentUri*(context, uri)) {

// ExternalStorageProvider

if (isExternalStorageDocument(uri)) {

final String docId = DocumentsContract.*getDocumentId*(uri);

final String[] split = docId.split(":");

final String type = split[0];

if ("primary".equalsIgnoreCase(type)) {

return Environment.*getExternalStorageDirectory*() + "/" + split[1];

}

// TODO handle non-primary volumes

}

// DownloadsProvider

else if (*isDownloadsDocument*(uri)) {

final String id = DocumentsContract.*getDocumentId*(uri);

final Uri contentUri = ContentUris.*withAppendedId*(

Uri.*parse*("content://downloads/public\_downloads"), Long.*valueOf*(id));

return *getDataColumn*(context, contentUri, null, null);

}

// MediaProvider

else if (*isMediaDocument*(uri)) {

final String docId = DocumentsContract.*getDocumentId*(uri);

final String[] split = docId.split(":");

final String type = split[0];

Uri contentUri = null;

if ("image".equals(type)) {

contentUri = MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI*;

} else if ("video".equals(type)) {

contentUri = MediaStore.Video.Media.*EXTERNAL\_CONTENT\_URI*;

} else if ("audio".equals(type)) {

contentUri = MediaStore.Audio.Media.*EXTERNAL\_CONTENT\_URI*;

}

final String selection = "\_id=?";

final String[] selectionArgs = new String[]{

split[1]

};

return *getDataColumn*(context, contentUri, selection, selectionArgs);

}

}

// MediaStore (and general)

else if ("content".equalsIgnoreCase(uri.getScheme())) {

return *getDataColumn*(context, uri, null, null);

}

// File

else if ("file".equalsIgnoreCase(uri.getScheme())) {

return uri.getPath();

}

return null;

}

public static String getDataColumn(Context context, Uri uri, String selection,

String[] selectionArgs) {

Cursor cursor = null;

final String column = "\_data";

final String[] projection = {

column

};

try {

cursor = context.getContentResolver().query(uri, projection, selection, selectionArgs,

null);

if (cursor != null && cursor.moveToFirst()) {

final int column\_index = cursor.getColumnIndexOrThrow(column);

return cursor.getString(column\_index);

}

} finally {

if (cursor != null)

cursor.close();

}

return null;

}

public static boolean isExternalStorageDocument(Uri uri) {

return "com.android.externalstorage.documents".equals(uri.getAuthority());

}

public static boolean isDownloadsDocument(Uri uri) {

return "com.android.providers.downloads.documents".equals(uri.getAuthority());

}

public static boolean isMediaDocument(Uri uri) {

return "com.android.providers.media.documents".equals(uri.getAuthority());

}

}

**AndroidManifest.xml**

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="com.example.bluetoothtransfer">

<uses-permission android:name="android.permission.BLUETOOTH\_ADMIN" />

<uses-permission android:name="android.permission.BLUETOOTH" />

<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: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>

</application>

</manifest>