MINI PROJECT
(2020-21)

File Sharing Android Application- "Easy Share"

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by-

1) **Prachi Bansal (181500459)**
2) **Neha Adnekar (181500421)**
3) **Mani Bansal (181500364)**
4) **Vivek Goyal (181500817)**
5) **Jitendra Singh (181500300)**

Supervised By: -
Mr. Neeraj khanna
Technical Trainer
Department of Computer Engineering & Applications

Contents

Abstract

In this we will build an android application that will transfer data from one device to another device using Wi-fi Direct. Our Team named this app "Easy Share". Easy Share is an open source Wi-fi Direct file sharing application that will enable sharing of data between Android devices running Android 4.0 or higher using a Wi-Fi direct connection without the use of a separate Wi-Fi access point. This will enable data transfer between devices without relying on any existing network infrastructure. This application is intended to provide a much higher speed alternative to Bluetooth file transfer.

There will be a client and a server i.e. the two devices that are connected through Wi-Fi direct. Once the connection is established, you can transfer data. Easy Share involves creating and registering a broadcast receiver for the application itself, discovering peers, connecting to a peer, and transferring data to a peer.

In this project comparative study and analysis of file sharing applications has been performed and presented. These file sharing applications enable the users for faster sharing rate among them through WLAN. They provide ease of access among multiple compatible devices. Several tests have been conducted including transferring several types of files and results have been gathered and these results are used in the analysis. A comparative analysis has also been done and the challenges or issues that are present in the concerned file sharing applications have been mentioned. The work primarily focuses on the commonly used file sharing applications in Smartphone's worldwide namely Xender and SHARE it.

Introduction

1.1 General Introduction to the topic

Today's computers are capable of sharing all types of files, including documents, songs, videos, and full applications. File sharing is a key function for many businesses and other use cases.

Easy Share is an open source Wi-fi Direct file sharing application that will enable sharing of data
between Android devices. File sharing is transferring or sharing the digital data such as image, audio, video, document etc. stored on an electronic device (mobile, laptop etc.) with similar compatible electronic devices. In this project we are using a Wi-Fi direct connection without the use of a separate Wi-Fi access point for sharing the files.
Wi-Fi Direct, initially called Wi-Fi P2P, is a Wi-Fi standard enabling devices to easily connect with each other without requiring a wireless access point. It is usable for everything from internet browsing to file transfer, and to communicate with more than one device simultaneously at typical Wi-Fi speeds. One advantage of Wi-Fi Direct is the ability to connect devices even if they are from different manufacturers. Only one of the Wi-Fi devices needs to be compliant with Wi-Fi Direct to establish a peer-to-peer connection that transfers data directly between them with greatly reduced setup.

Wi-fi Direct negotiate the link with a Wi-fi Protected Setup System that assigns each device a limited wireless access point. The "pairing"of Wi-fi direct devices can be set up to require the proximity of a near field communication, a Bluetooth, signal, or a button press one or a all Devices. Wi-fi Direct may not only the need for Routers, but may also replace the need of Bluetooth for Application that do not rely on low battery.

File sharing is essential, and we are now having many ways to perform this but every way has its merits and demerits. For storing the use of dedicated server is used which has various demerits and also the cloud server is used which is complicated for deployment and also require more systems. The file sharing has become very popular now a day for its higher data secure and utilisation properties.

They should be able to send and receive the files from mobile, through Wi-fi Direct and which is designed as this project and the product of the project is named as "Easy Share".

1.2.1 Working:

The basic steps for a Wi-fi direct application are:

- Initial Setup
- Discovering Peers
- Connecting to Peers
- Transferring Data

We followed these steps in our application and registered a Broadcast receiver to listen to the intents. The way Wi-fi direct works is that we can call certain methods from the main activity to perform certain Wi-fi direct functions like search peers and connect. The methods will interact with the hardware. When hardware is ready, it will broadcast some intents to the OS and software. The broadcast receiver then can catch these intents and do the follow up works accordingly.

1.2.2 Application Area:

- **Cyber Security field**- File sharing apps are a common tool used in cyber-attacks, both by professional hackers and ordinary individuals.
- **Education field**- Study stuff & business documents instantly shared.
- **Government**- Government use file sharing to share the important information on websites with limited rights.
- **IT Industries**-With file sharing app, employee effort and time–become efficiently utilized.

1.3.1 Hardware Requirements:

- Processor: i3
- Hard Disk: 5GB
- Memory: 1GB RAM
- Android Phone with KitKat and higher

1.3.2 Software Requirements:

- Windows 7 (ultimate, enterprise, Windows 8,Windows10
- Android Studio

1.4 Development Challenges

• **Scalability**: The aim of scaling is to allow the system to expand in scale without disrupting the activities of the users. The enormous data should not lead to corrupt or any other issues.
• **Heterogeneity**: the system should be available and accessible from different plat- form without any difference in performance.
• **Real-Time Data Delivery**: the system should be able to deliver data in real-time when-where ever data is requested.
• **Limited Bandwidth**: Although Internet is available to all devices(within the targeted), but many of devices have very slow connection speed and for better costing Limited Bandwidth is to be taken care of.

Problem Statement

To design File storing and sharing system having following features

1. To able to implement Peer-to-Peer server :
   Peer-to-peer (**P2P**) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

2. To create Data base as back end to store data on the servers:
   A **database server** is a server which uses a database application that provides database services to other computer programs or to computers, as defined by the client–server model.

3. Create a web-platform as user interface for downloading, uploading and sharing of files:
   In its simplest form, a peer-to-peer (**P2P**) **network** is **created** when two or more PCs are connected and share resources without going through a separate server computer. A **P2P network** can be an ad hoc connection—a couple of computers connected via a Universal Serial Bus to transfer files
4. Create a desktop application which facilitate to synchronise a folder with server ensuring the files in that folder has its copy on server :
   File synchronization ensures that two or more data storage devices, computers, or programs have the same information at a given time. You would agree to this fact that organizing your business data across various devices is not as simple as it sounds.
5. Designing an Android application, so that user can store of send any file from the mobile to server :
   Once you have set up your app to share files using content URIs, you can respond to other apps' requests for those files. One way to respond to these requests is to provide a file selection interface from the server app that other applications can invoke. This approach allows a client application to let users select a file from the server app and then receive the selected file's content URI.
6. Ensuring the encryption of data on the server for batter security:

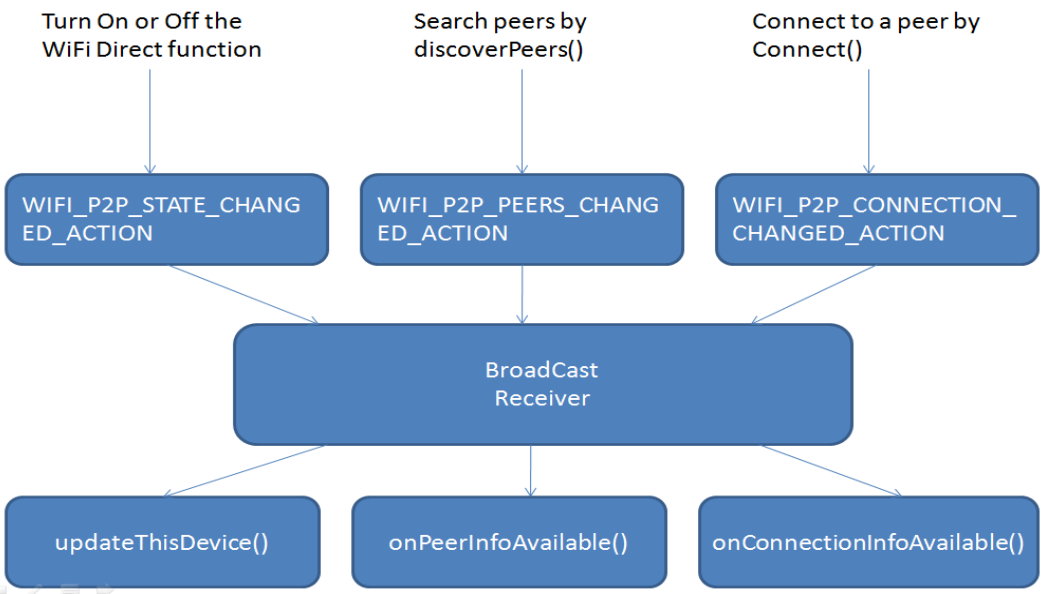
   To transfer files to and from a server without danger of hackers compromising or stealing data, it is vital to use **File Transfer Protocol Secure (FTPS)**. It encrypts data files and your authentication information.FTPS uses both a command channel and a data channel, and the user can encrypt both. Bear in mind that it only protects files during transfer. As soon as they reach the server, the data is no longer encrypted. For this reason, encrypting the files before sending them adds another layer of security.

7. Appling several business concepts like advertisement and Entrepreneurship skills:
   There are different processes in a single business - for example, operations, marketing, recruiting, customer support, finance, etc., and ensuring that the same data is available to all the team members working in these different processes is itself a challenge.

Objective

The main objectives of the creating this android Application is to enable easy access and transfer of personal content between local storage devices using P2P networks and FTP(file transfer protocol) without turning on the data .

Implementation Details



1. Classes :-

WifiP2pManager: This is a class which is used to interact with the Wi-Fi hardware on our device which helps in connecting and discovering the peers.

2. Methods :-

WifiP2pManager class provides some methods for peer–to–peer connection.

- **initialize()**: This method registers the application within a Wi-Fi framework, and it is called before calling any other Wi-Fi P2p methods.
- **connect()**: This method is used to start a Wi-Fi pair connection with a device having a specific
- **discoverPeers()**: This method is used to detect all the available peers that are in the range.
- **cancelConnect()**: This method is used to cancel all the connected Wi-Fi pair connections.
- **removeGroup()**: This method is used to remove the current peer–to–peer group.

3. Intents

When using the Wi-Fi pair connection, we need to listen for broadcast intents and handle the events occurred.

1WIFI_P2P_STATE_CHANGED_ACTION: It indicates a change in the Wi-Fi P2P status
WIFI_P2P_PEERS _CHANGED_ACTION: It indicates a change in the list of available peers
WIFI_P2P_CONNECTION _CHANGED_ACTION: It indicates the state of Wi-Fi P2P

connectivity is changed
WIFI_P2P_THIS_DEVICE _CHANGED_ACTION: It indicates the device details are changed

EasyShare with Source code

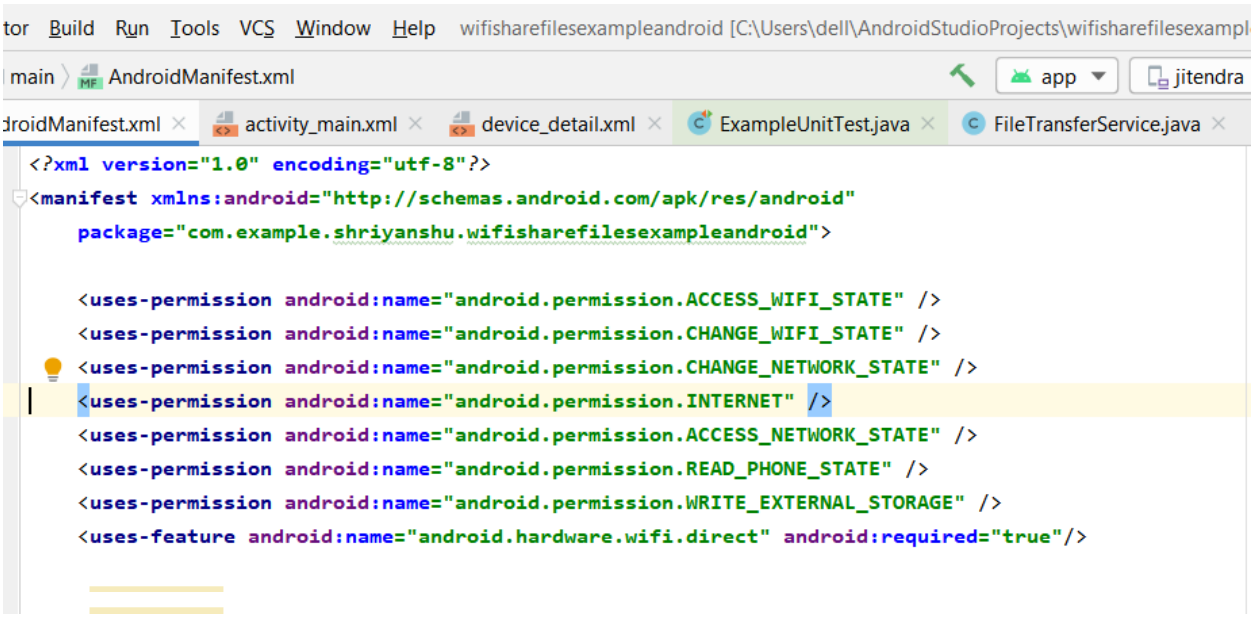
This Android Application which transfers files by creating a Wi-Fi pair connection in Android.

- Create an Application **WiFiShareFilesExampleAndroid** in Android Studio
- Add the permissions in the **AndroidManifest.xml** file.

AndroidManifest.xml

To use the Wi-Fi pair connection, we have to add some necessary permissions in this file.

Screenshot-



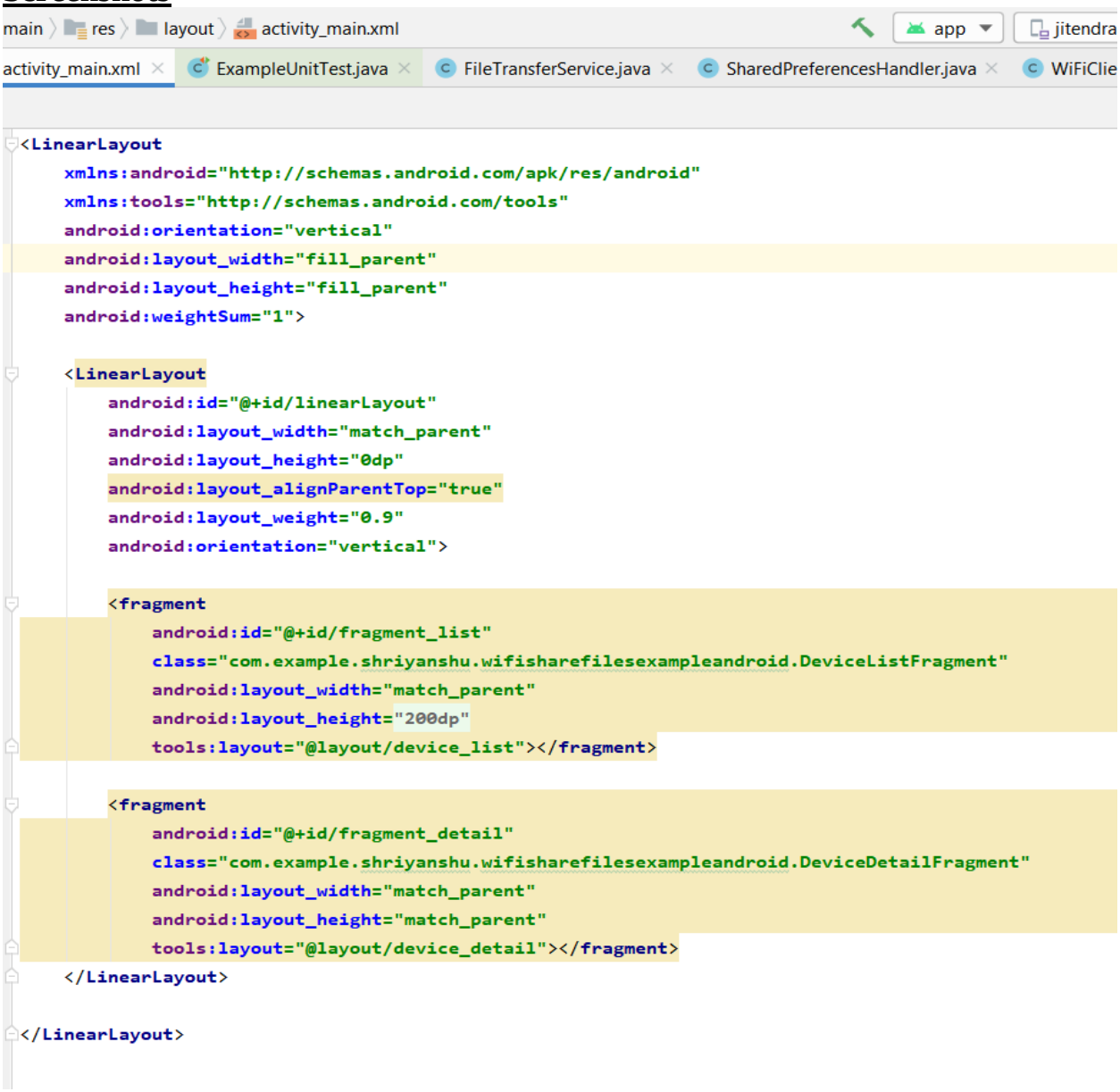
Now do it programmatically by creating some Java files and XML files in our project.

Layout Files :-

1. activity_main.xml

This is our main layout file. Here, we add two fragments inside LinearLayout.

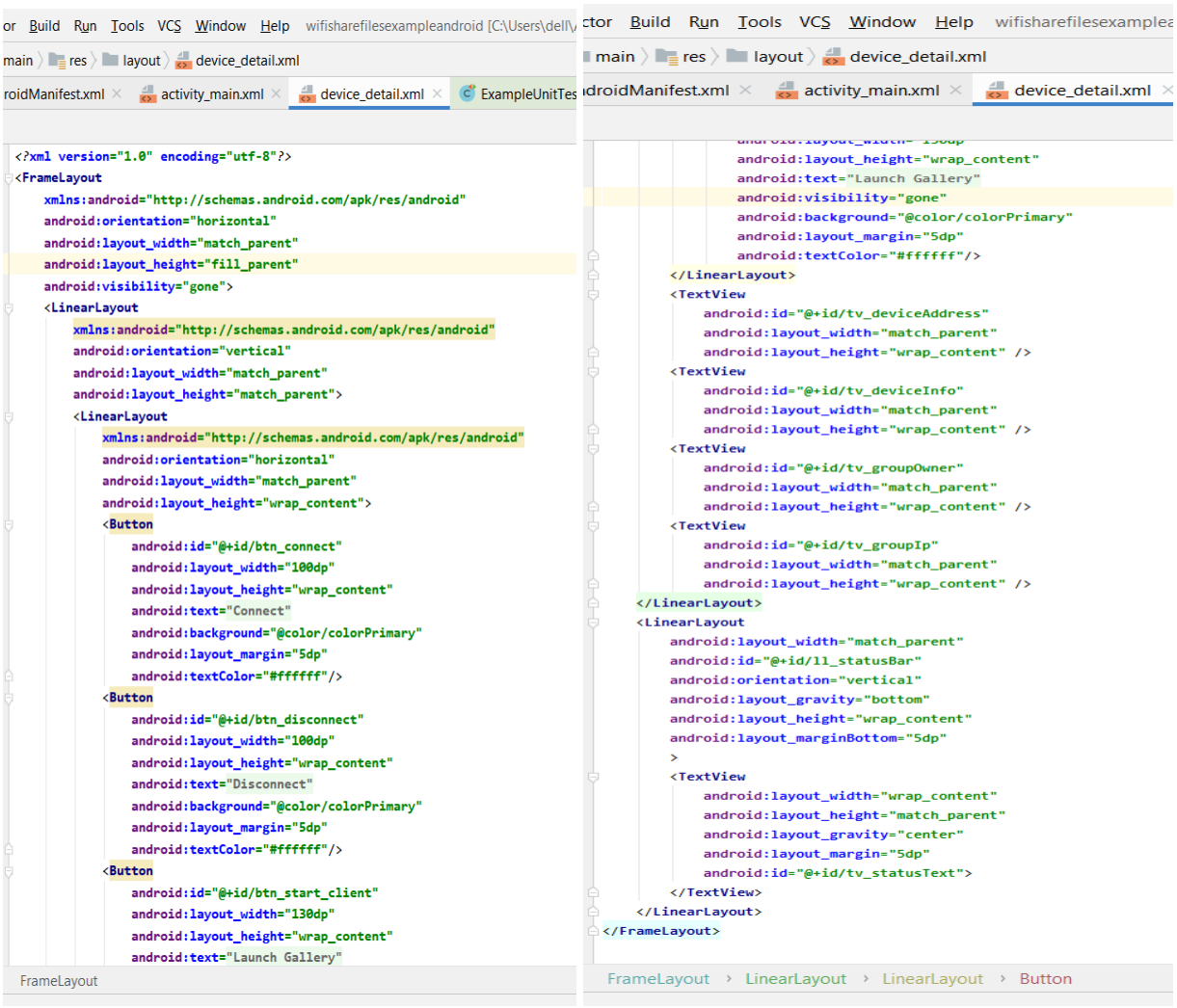
Screenshots



2. device_detail.xml

Create this XML file to add the layout for device details.

Screenshots



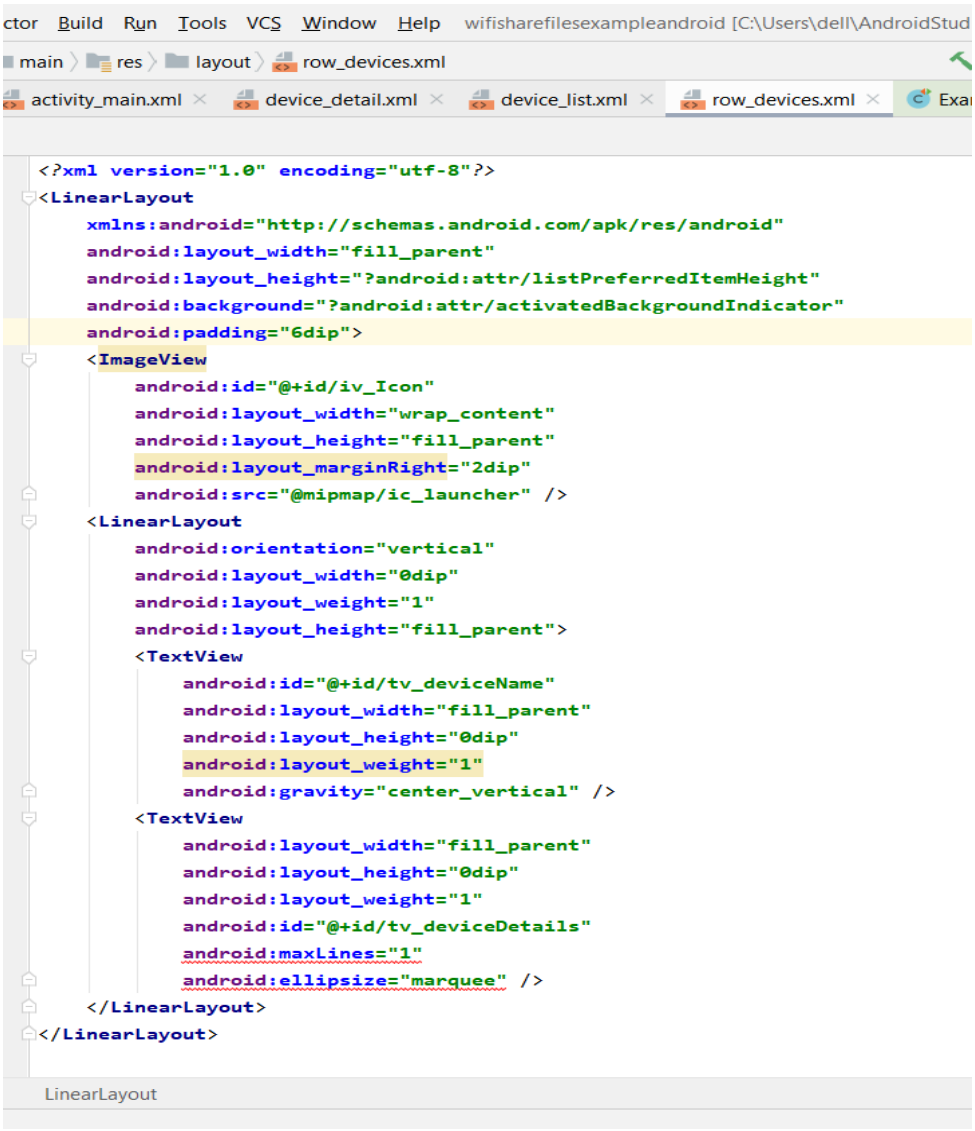
3. device_list.xml

Create this XML file to add the layout for device lists.

Screenshots

4. row_devices.xml

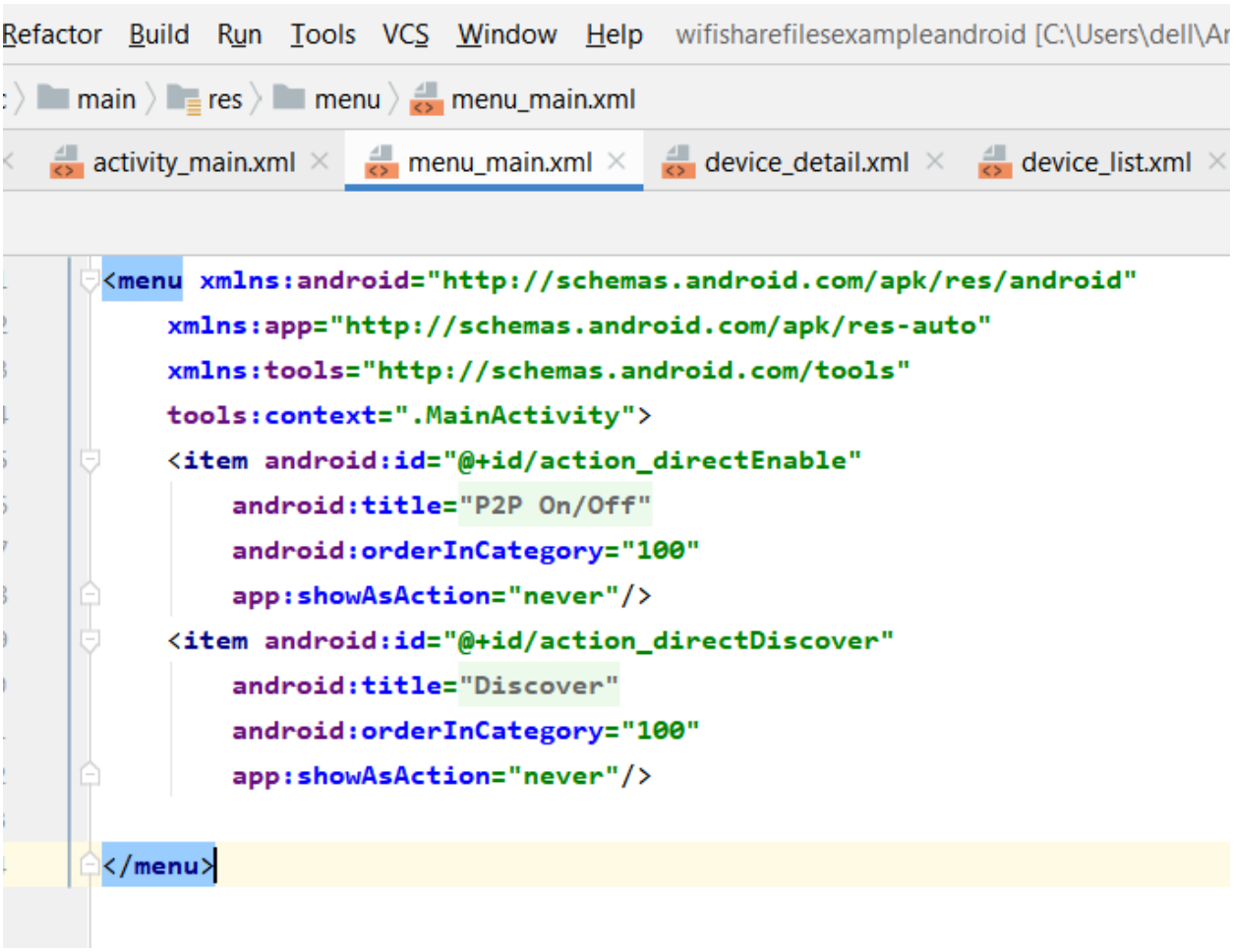
Create this XML file and add the code here.



5. menu_main.xml

Create this XML file inside res/menu folder. Here we add the two action items.

Screenshot



Java Files

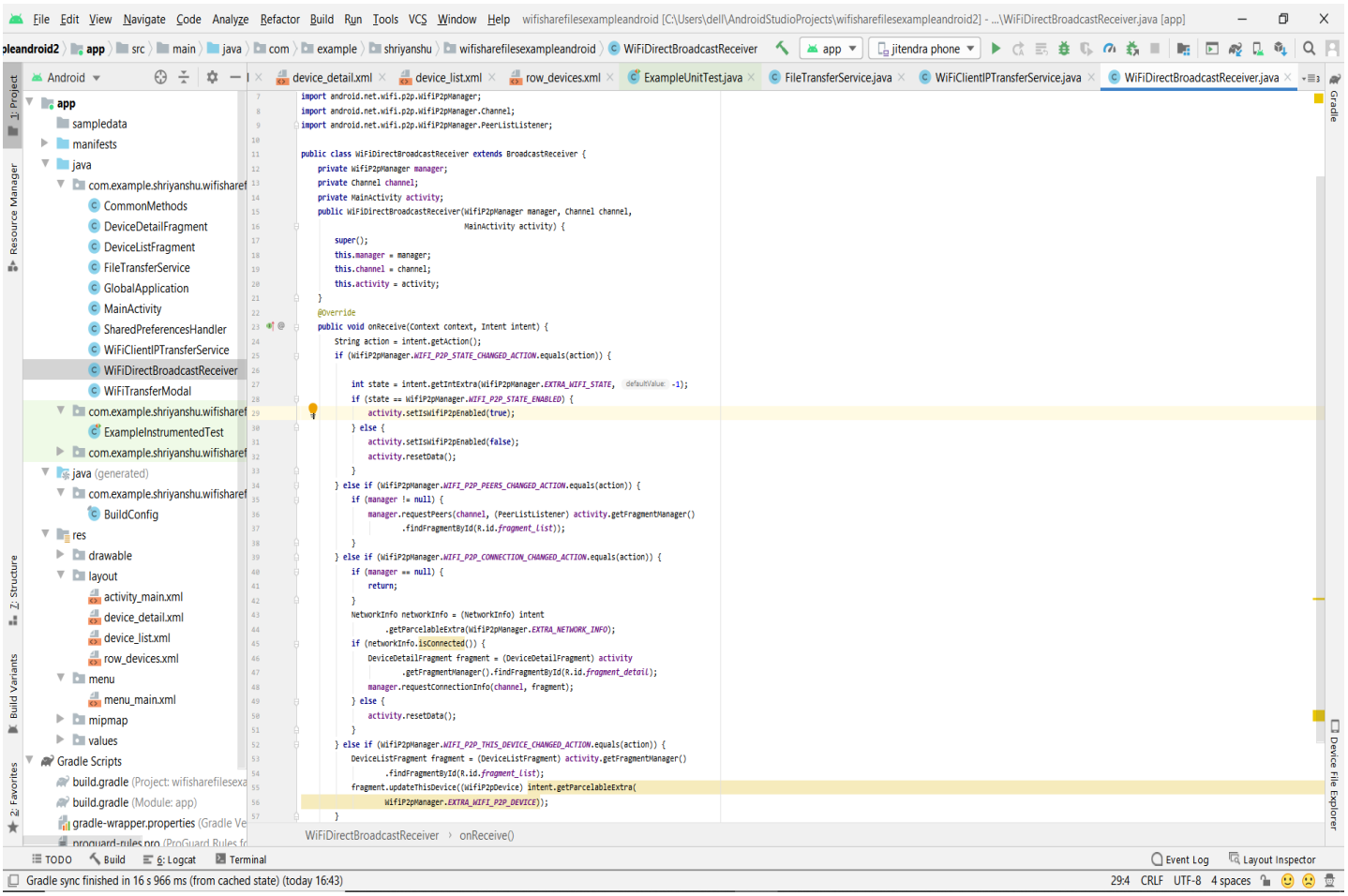
WifiDirectBroadcastReceiver.java

Create this class and extends with BroadcastReceiver. Here, this class notifies important Wi-Fi P2p events. The events are discussed above.

Initialize

```
private WifiP2pManager manager;
private Channel channel;
private MainActivity activity;
```

Add the Code in onReceive() Method:



MainActivity.java

This is our Main Activity. Here, it adds the necessary intent values by creating an intent-filter, registers the broadcast receiver, obtains the instance of WifiP2pManager class, registers the application by using initialize() method, also discover peers and connects with the available devices.

Initialize

```
public class MainActivity extends AppCompatActivity implements ChannelListener, DeviceActionListener {


    public static final String TAG = "WiFiShareFilesExampleAndroid";
    private WifiP2pManager manager;
    private boolean isWifiP2pEnabled = false;
    private boolean retryChannel = false;
    private final IntentFilter intentFilter = new IntentFilter();
    private Channel channel;
    private BroadcastReceiver receiver = null;
```

Add the code in onCreate(…){..}:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // add necessary intent values to be matched.
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
    intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

    manager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
    channel = manager.initialize( srcContext: this, getMainLooper(),  listener: null);
}
```

Now, register the receiver in onResume() method

```
@Override
public void onResume() {
    super.onResume();
    receiver = new WiFiDirectBroadcastReceiver(manager, channel,  activity: this);
    registerReceiver(receiver, intentFilter);
}

@Override
public void onPause() {
    super.onPause();
    unregisterReceiver(receiver);
}
```

Add the code for discover peers

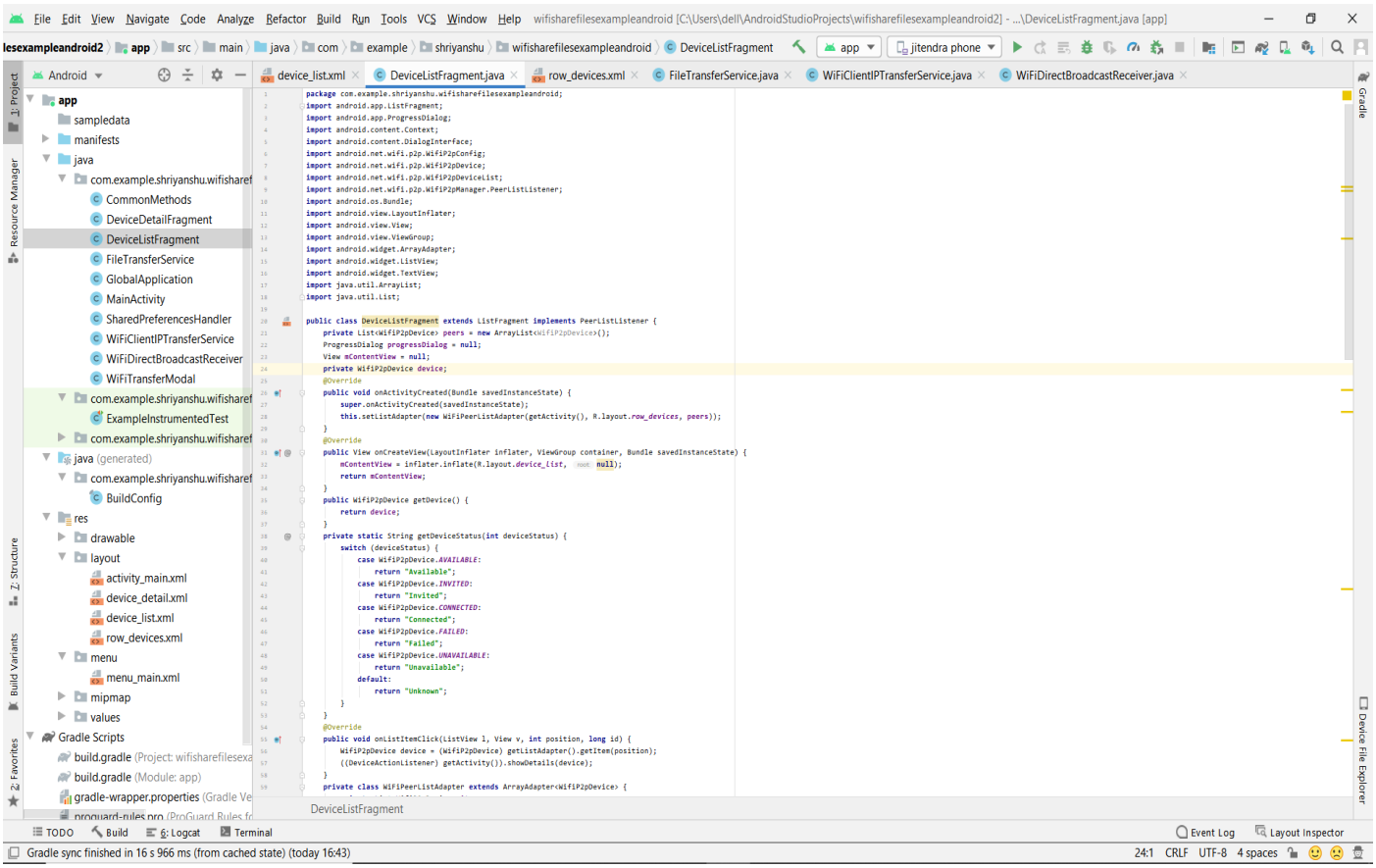
```
        final DeviceListFragment fragment = (DeviceListFragment) getFragmentManager()
                .findFragmentById(R.id.fragment_list);
        fragment.onInitiateDiscovery();
        manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {

            @Override
            public void onSuccess() {
                Toast.makeText( context: MainActivity.this, text: "Discovery Initiated",
                        Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onFailure(int reasonCode) {
                Toast.makeText( context: MainActivity.this, text: "Discovery Failed : " + reasonCode,
                        Toast.LENGTH_SHORT).show();
            }
        });
        return true;
    default:
        return super.onOptionsItemSelected(item);
}
```

Now call connect() method for connecting with an available peer.

```
    @Override
    public void connect(WifiP2pConfig config) {
        manager.connect(channel, config, new ActionListener() {

            @Override
            public void onSuccess() {
                // WiFiDirectBroadcastReceiver will notify us. Ignore for now.
            }

            @Override
            public void onFailure(int reason) {
                Toast.makeText( context: MainActivity.this, text: "Connect failed. Retry.",
                        Toast.LENGTH_SHORT).show();
            }
        });
    }
```

DeviceDetailFragment.java

Create this fragment class. Here, it manages with a particular peer and “allows the interaction with the device” means when a connection is established successfully, we can transfer the data to the device by setting up a network connection.
Add the code here by creating a server socket that accepts the connection and writes the data.



Progress [70%]

Completed Parts of Project

1- User Interface Of Easy Share is created.
2- Layout files are created.
3- Mostly Java files created.

Remaining parts of Project

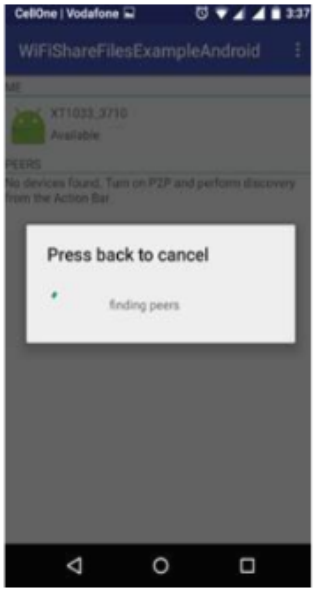
1- Some Java files are left to create.
2- Sending & Receiving Threads for larger files.
3-Manipulation of Received files.

6.1 Screenshots

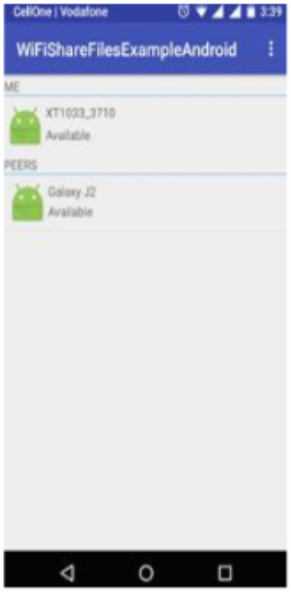
6.1.1 Screenshot Of User interface of Android Application.

Before Discover

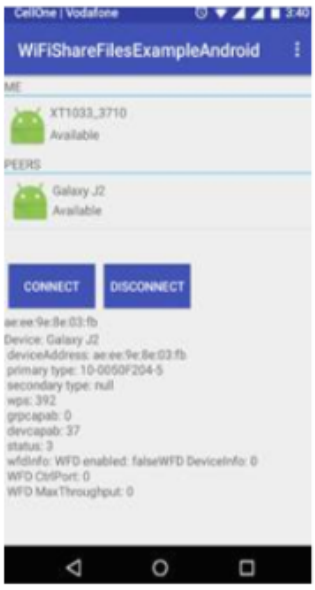
Perform Discover



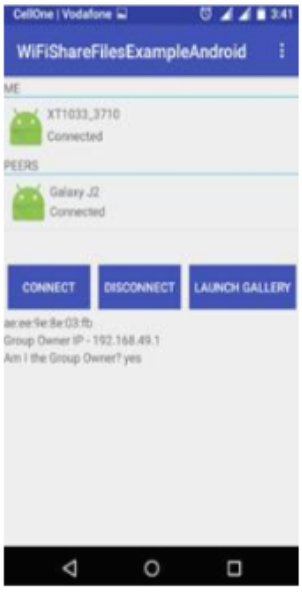
Finding Peers



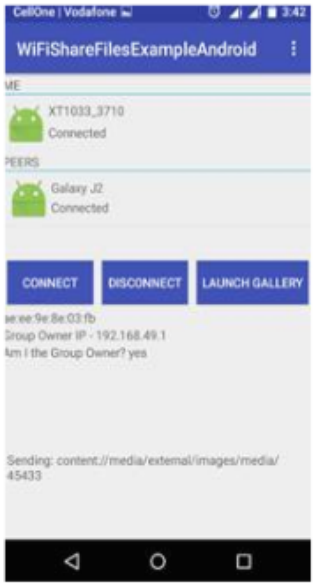
Peer Available



Connecting Device

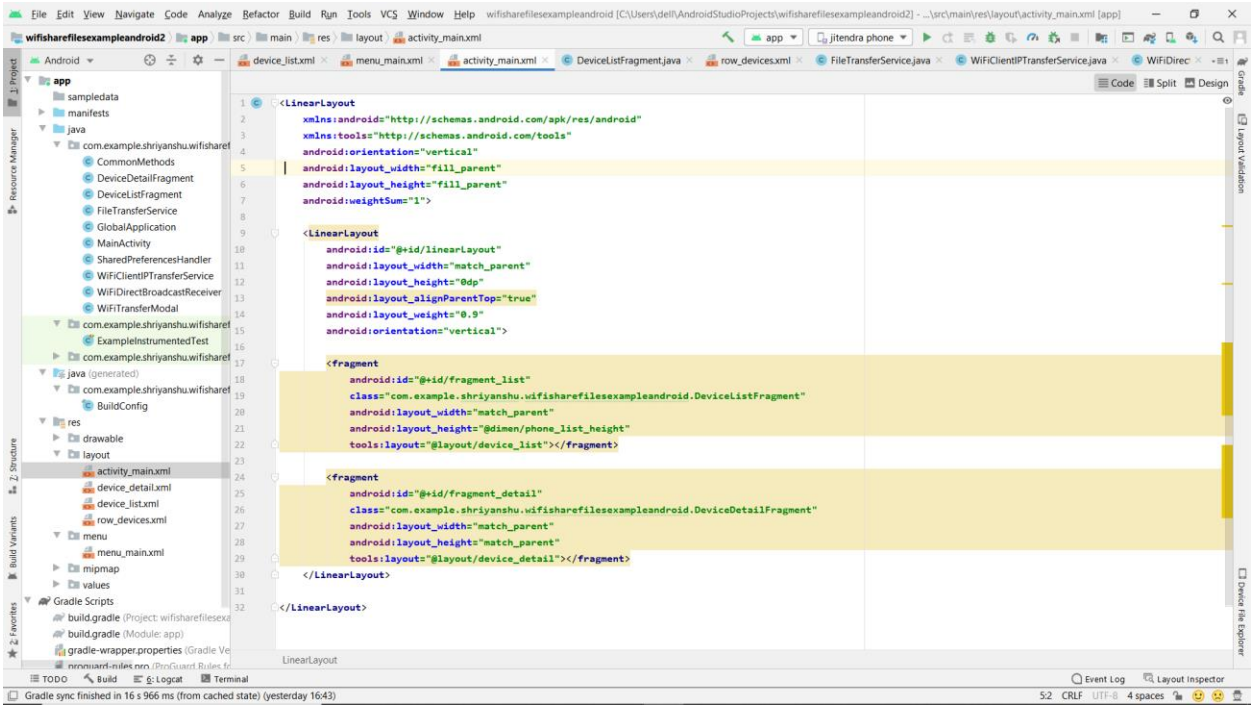


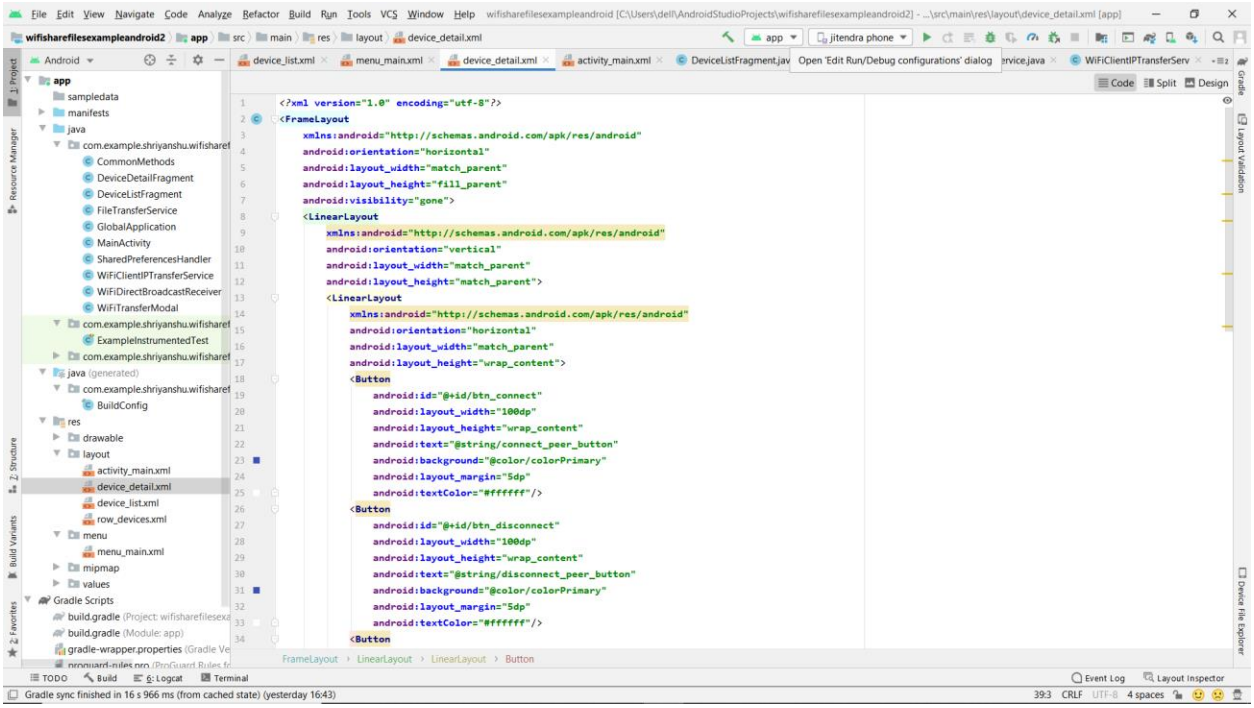
Device Connected

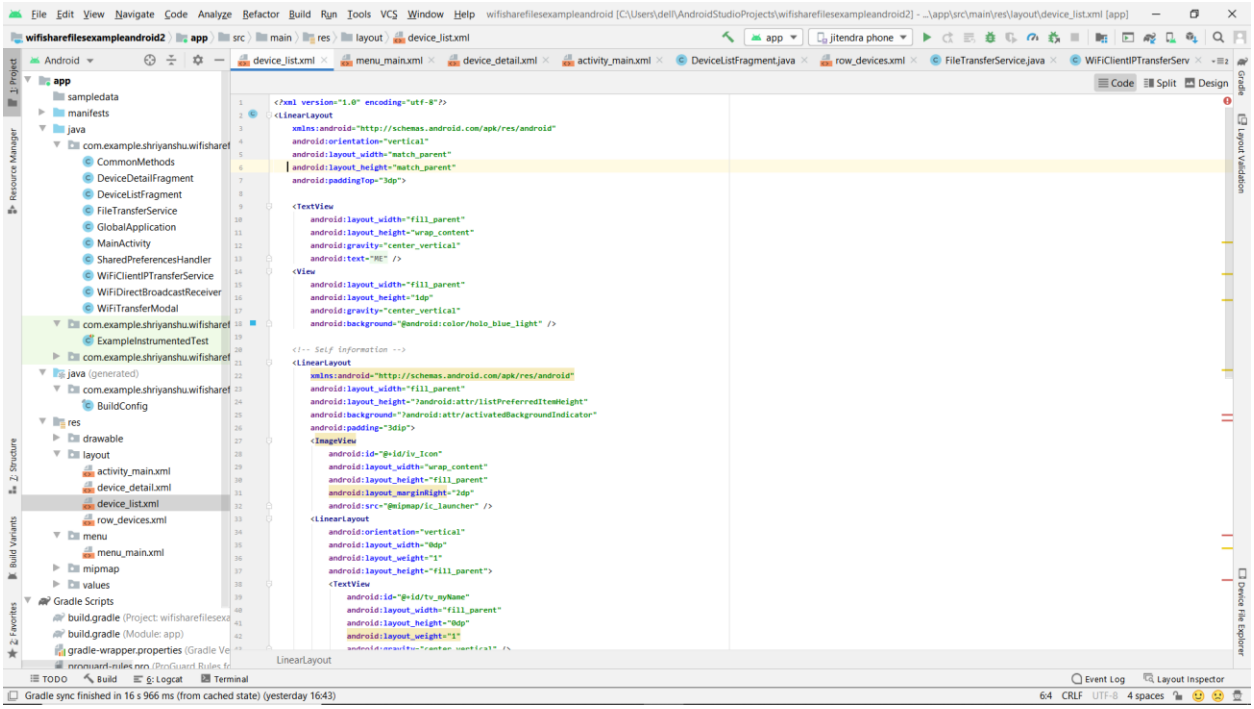


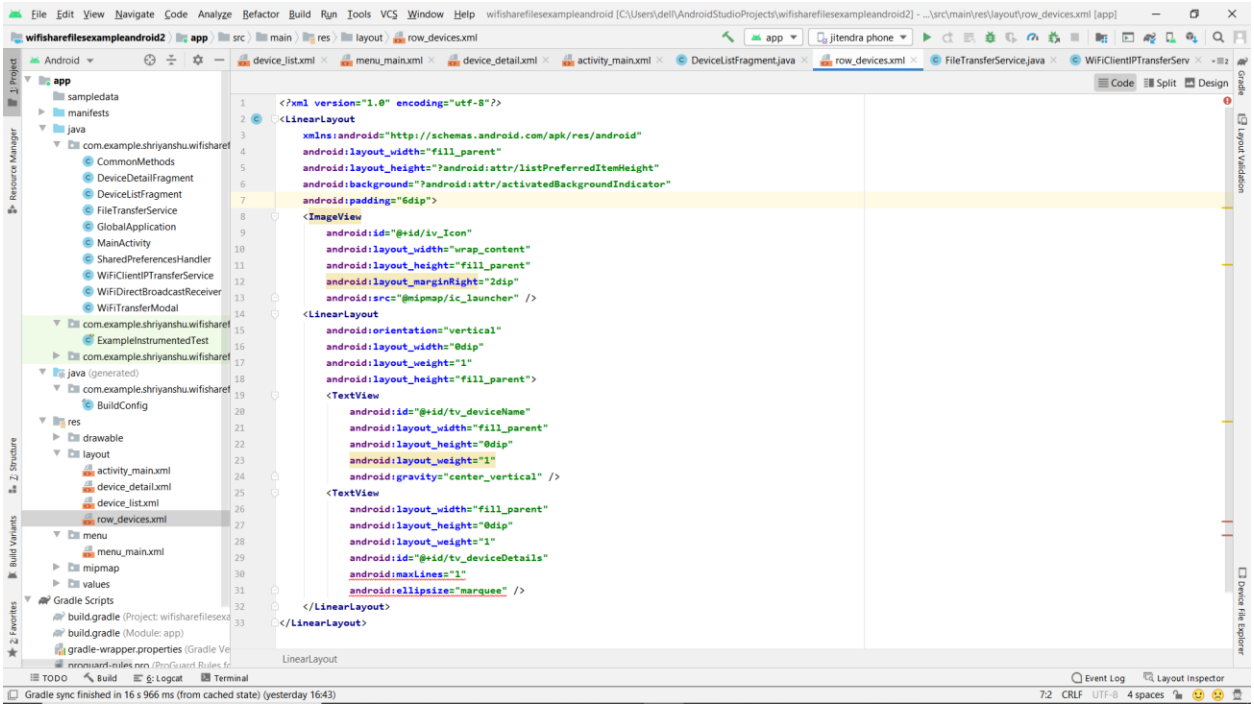
Status after Sending the data successfully

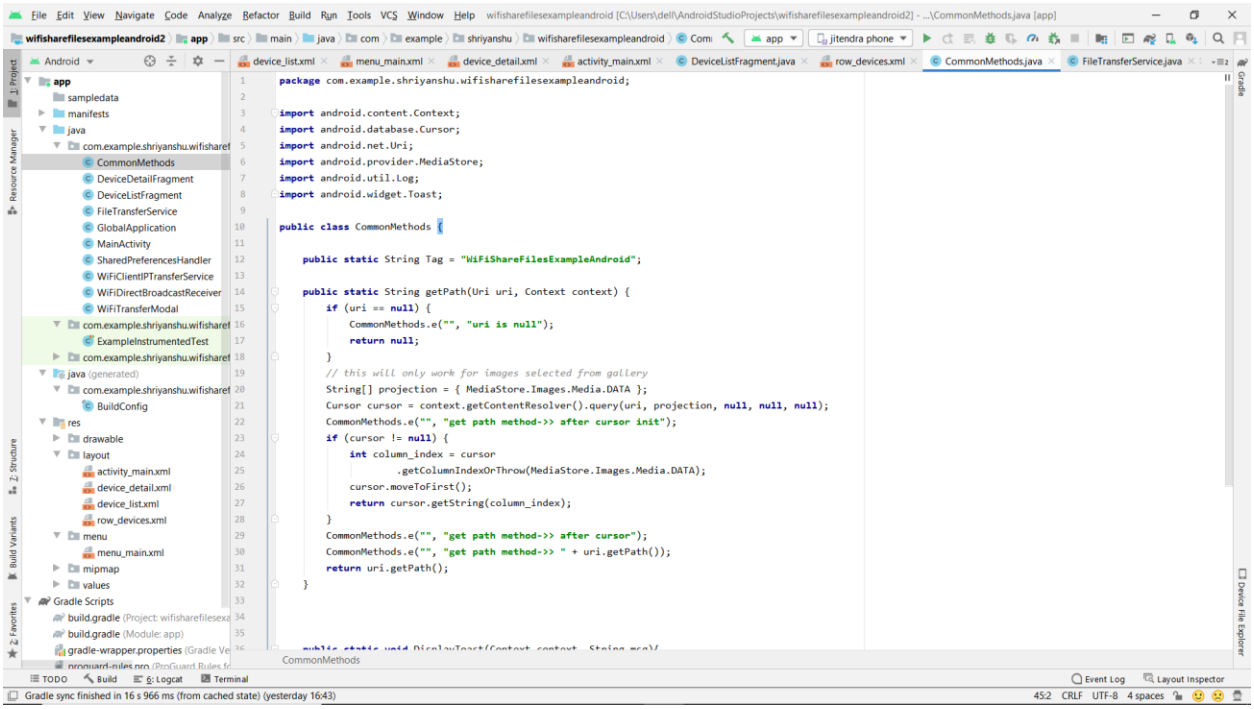
6.1.2 Screenshots of Source code

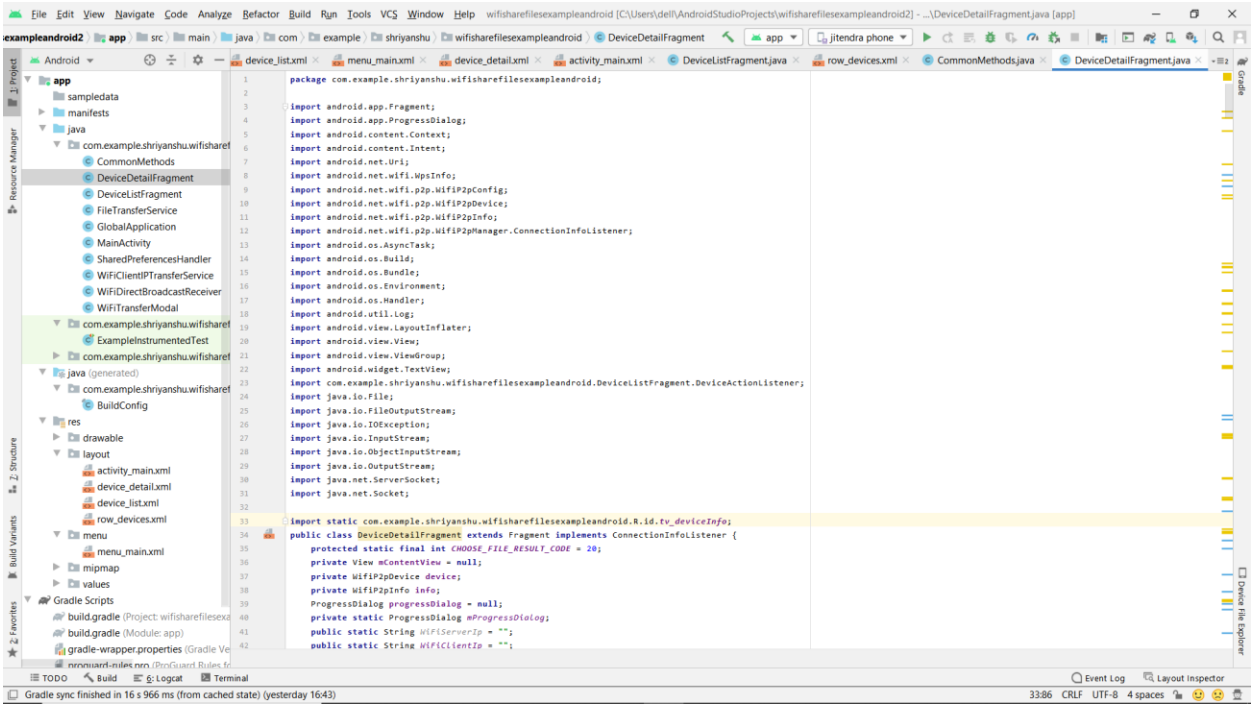


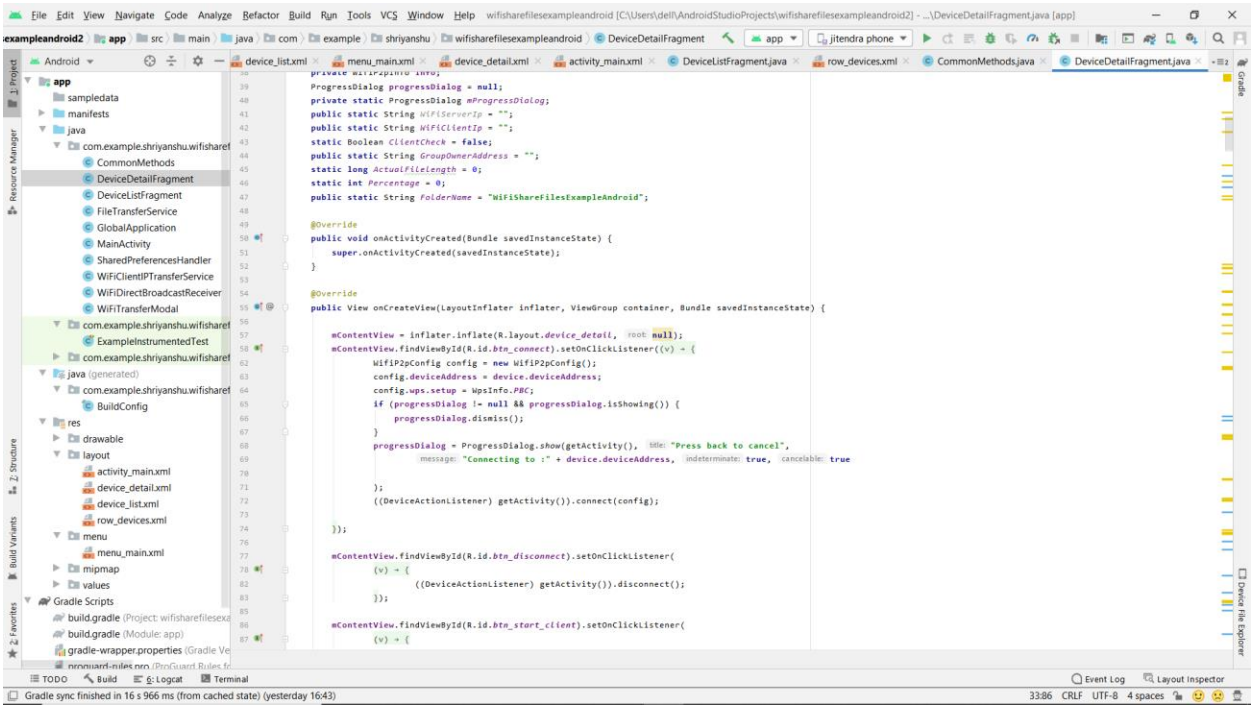

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
wifisharefilesexampleandroid [C:\Users\dell\AndroidStudioProjects\wifisharefilesexampleandroid2] - ...\src\main\res\layout\device_detail.xml [app]
wifisharefilesexampleandroid2 app src main res layout device_detail.xml
app
jitendra phone
Android
device_list.xml
menu_main.xml
device_detail.xml
activity_main.xml
DeviceListFragment.jav
Open 'Edit Run/Debug configurations' dialog
rvice.java
WiFiClientIPTransferServ
Code
Split
Design
app
sampledata
manifests
java
com.example.shriyanshu.wifisharef
CommonMethods
DeviceDetailFragment
DeviceListFragment
FileTransferService
GlobalApplication
MainActivity
SharedPreferencesHandler
WiFiClientIPTransferService
WiFiDirectBroadcastReceiver
WiFiTransferModal
com.example.shriyanshu.wifisharef
ExampleInstrumentedTest
com.example.shriyanshu.wifisharef
java (generated)
com.example.shriyanshu.wifisharef
BuildConfig
res
drawable
layout
activity_main.xml
device_detail.xml
device_list.xml
row_devices.xml
menu
menu_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: wifisharefilesexa
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Ve
proguard-rules.pro (ProGuard Rules fo
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="fill_parent"
android:visibility="gone">
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent">
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="wrap_content">
<Button
android:id="@+id/btn_connect"
android:layout_width="100dp"
android:layout_height="wrap_content"
android:text="@string/connect_peer_button"
android:background="@color/colorPrimary"
android:layout_margin="5dp"
android:textColor="#ffffff"/>
<Button
android:id="@+id/btn_disconnect"
android:layout_width="100dp"
android:layout_height="wrap_content"
android:text="@string/disconnect_peer_button"
android:background="@color/colorPrimary"
android:layout_margin="5dp"
android:textColor="#ffffff"/>
<Button
FrameLayout › LinearLayout › LinearLayout › Button
TODO
Build
6: Logcat
Terminal
Event Log
Layout Inspector
Gradle sync finished in 16 s 966 ms (from cached state) (yesterday 16:43)
39:3
CRLF
UTF-8
4 spaces

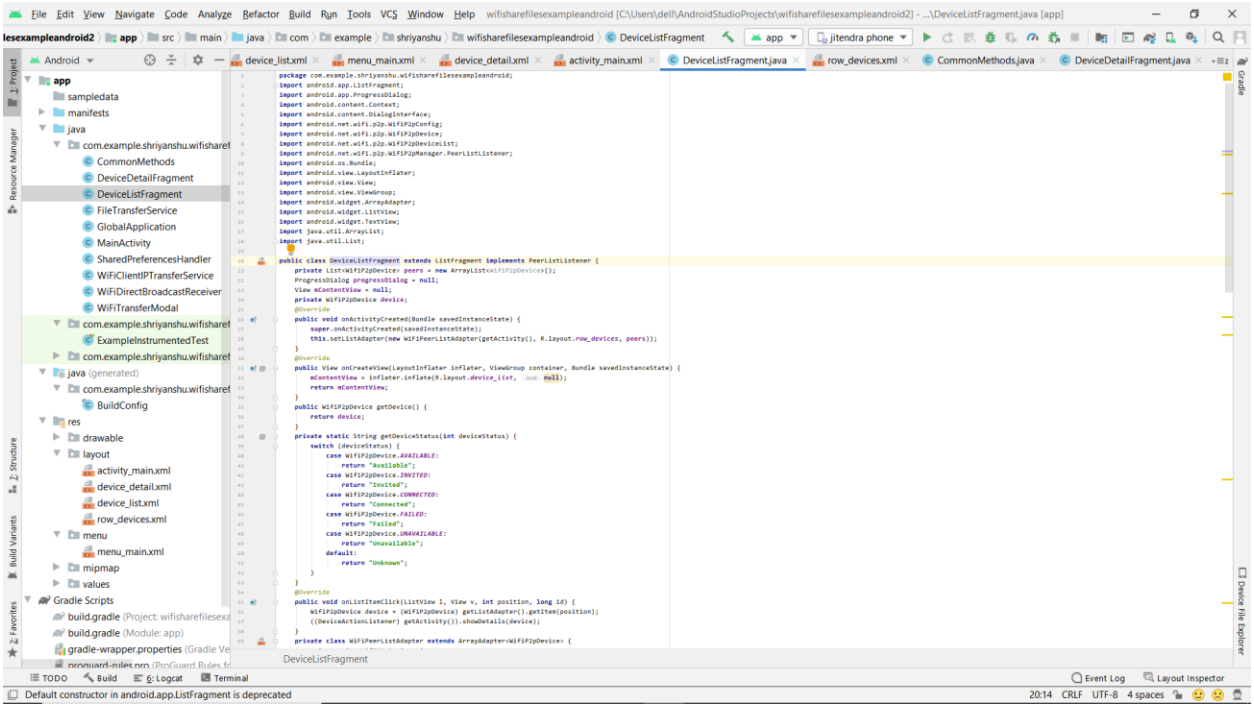

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
wifisharefilesexampleandroid [C:\Users\dell\AndroidStudioProjects\wifisharefilesexampleandroid2] - ...\app\src\main\res\layout\device_list.xml [app]
wifisharefilesexampleandroid2 app src main res layout device_list.xml
app
jitendra phone
Android
device_list.xml
menu_main.xml
device_detail.xml
activity_main.xml
DeviceListFragment.java
row_devices.xml
FileTransferService.java
WiFiClientIPTransferServ
Code
Split
Design
app
sampledata
manifests
java
com.example.shriyanshu.wifisharef
CommonMethods
DeviceDetailFragment
DeviceListFragment
FileTransferService
GlobalApplication
MainActivity
SharedPreferencesHandler
WiFiClientIPTransferService
WiFiDirectBroadcastReceiver
WiFiTransferModal
com.example.shriyanshu.wifisharef
ExampleInstrumentedTest
com.example.shriyanshu.wifisharef
java (generated)
com.example.shriyanshu.wifisharef
BuildConfig
res
drawable
layout
activity_main.xml
device_detail.xml
device_list.xml
row_devices.xml
menu
menu_main.xml
mipmap
values
Gradle Scripts
build.gradle (Project: wifisharefilesexa
build.gradle (Module: app)
gradle-wrapper.properties (Gradle Ve
proguard-rules.pro (ProGuard Rules fo
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingTop="3dp">
<TextView
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:gravity="center_vertical"
android:text="ME" />
<View
android:layout_width="fill_parent"
android:layout_height="1dp"
android:gravity="center_vertical"
android:background="@android:color/holo_blue_light" />
<!-- Self information -->
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
android:background="?android:attr/activatedBackgroundIndicator"
android:padding="3dip">
<ImageView
android:id="@+id/iv_icon"
android:layout_width="wrap_content"
android:layout_height="fill_parent"
android:layout_marginRight="2dp"
android:src="@mipmap/ic_launcher" />
<LinearLayout
android:orientation="vertical"
android:layout_width="0dp"
android:layout_weight="1"
android:layout_height="fill_parent">
<TextView
android:id="@+id/tv_myName"
android:layout_width="fill_parent"
android:layout_height="0dp"
android:layout_weight="1"
LinearLayout
TODO
Build
6: Logcat
Terminal
Event Log
Layout Inspector
Gradle sync finished in 16 s 966 ms (from cached state) (yesterday 16:43)
6:4
CRLF
UTF-8
4 spaces


File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
wifisharefilesexampleandroid [C:\Users\dell\AndroidStudioProjects\wifisharefilesexampleandroid2] - ...\src\main\res\layout\row_devices.xml [app]
wifisharefilesexampleandroid2 app src main res layout row_devices.xml
app jitendra phone
device_list.xml menu_main.xml device_detail.xml activity_main.xml DeviceListFragment.java row_devices.xml FileTransferService.java WiFiClientIPTransferServ
Code Split Design
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="?android:attr/listPreferredItemHeight"
android:background="?android:attr/activatedBackgroundIndicator"
android:padding="6dip">
<ImageView
android:id="@+id/iv_Icon"
android:layout_width="wrap_content"
android:layout_height="fill_parent"
android:layout_marginRight="2dip"
android:src="@mipmap/ic_launcher" />
<LinearLayout
android:orientation="vertical"
android:layout_width="0dip"
android:layout_weight="1"
android:layout_height="fill_parent">
<TextView
android:id="@+id/tv_deviceName"
android:layout_width="fill_parent"
android:layout_height="0dip"
android:layout_weight="1"
android:gravity="center_vertical" />
<TextView
android:layout_width="fill_parent"
android:layout_height="0dip"
android:layout_weight="1"
android:id="@+id/tv_deviceDetails"
android:maxLines="1"
android:ellipsize="marquee" />
</LinearLayout>
</LinearLayout>
LinearLayout
TODO Build 6: Logcat Terminal
Event Log Layout Inspector
Gradle sync finished in 16 s 966 ms (from cached state) (yesterday 16:43)
7:2 CRLF UTF-8 4 spaces

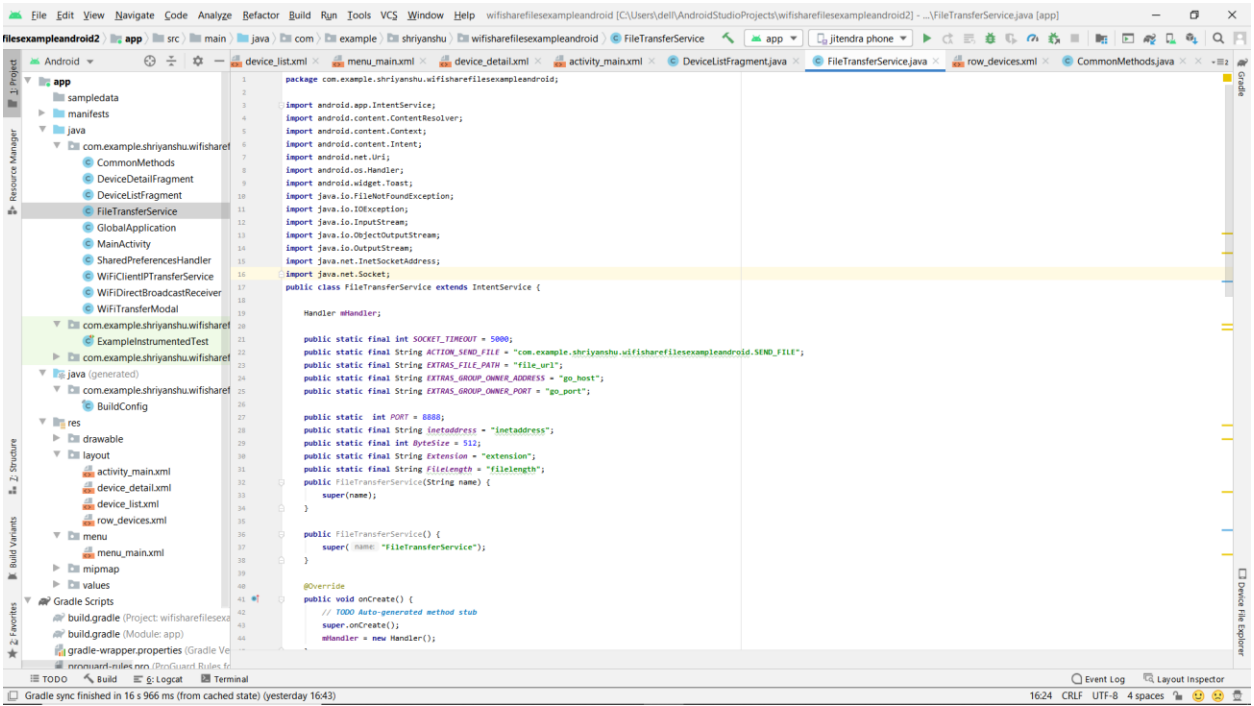

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
wifisharefilesexampleandroid [C:\Users\dell\AndroidStudioProjects\wifisharefilesexampleandroid2] - ...\CommonMethods.java [app]
wifisharefilesexampleandroid2 app src main java com example shriyanshu wifisharefilesexampleandroid Comi
app jitendra phone
device_list.xml menu_main.xml device_detail.xml activity_main.xml DeviceListFragment.java row_devices.xml CommonMethods.java FileTransferService.java
package com.example.shriyanshu.wifisharefilesexampleandroid;
import android.content.Context;
import android.database.Cursor;
import android.net.Uri;
import android.provider.MediaStore;
import android.util.Log;
import android.widget.Toast;
public class CommonMethods {
public static String Tag = "WiFiShareFilesExampleAndroid";
public static String getPath(Uri uri, Context context) {
if (uri == null) {
CommonMethods.e("", "uri is null");
return null;
}
// this will only work for images selected from gallery
String[] projection = { MediaStore.Images.Media.DATA };
Cursor cursor = context.getContentResolver().query(uri, projection, null, null, null);
CommonMethods.e("", "get path method->> after cursor init");
if (cursor != null) {
int column_index = cursor
.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
cursor.moveToFirst();
return cursor.getString(column_index);
}
CommonMethods.e("", "get path method->> after cursor");
CommonMethods.e("", "get path method->> " + uri.getPath());
return uri.getPath();
}
CommonMethods
TODO Build 6: Logcat Terminal
Event Log Layout Inspector
Gradle sync finished in 16 s 966 ms (from cached state) (yesterday 16:43)
45:2 CRLF UTF-8 4 spaces



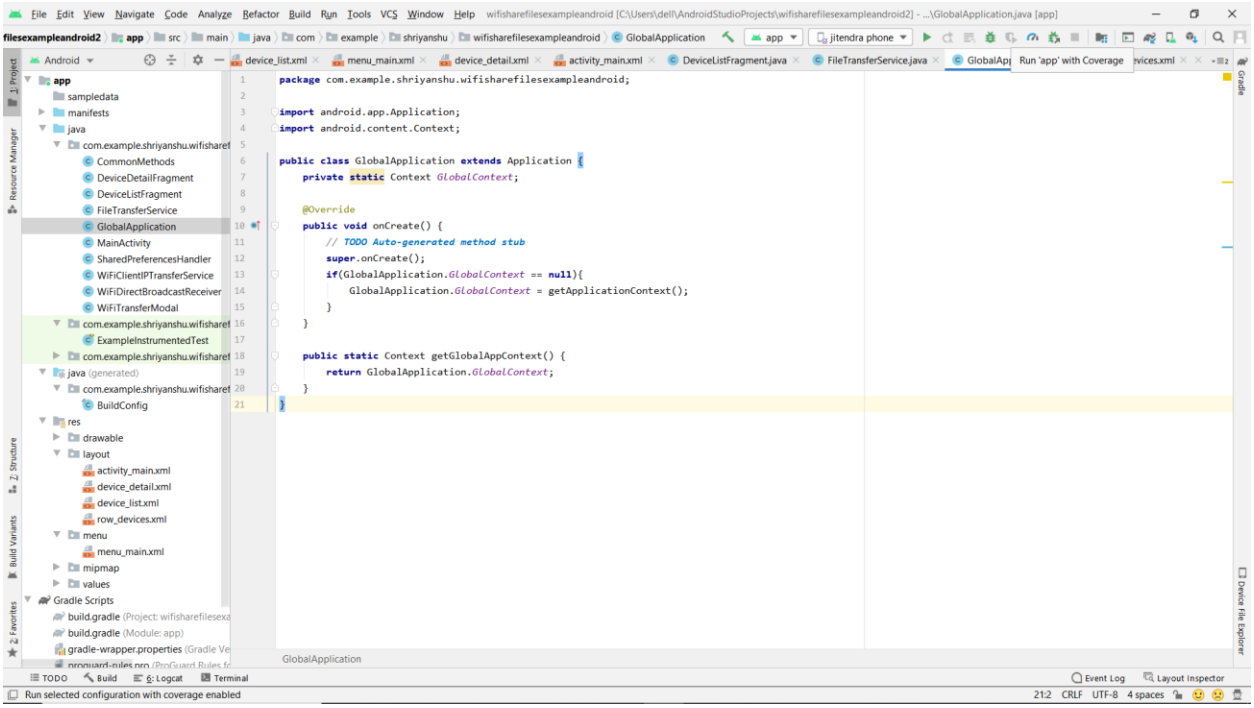




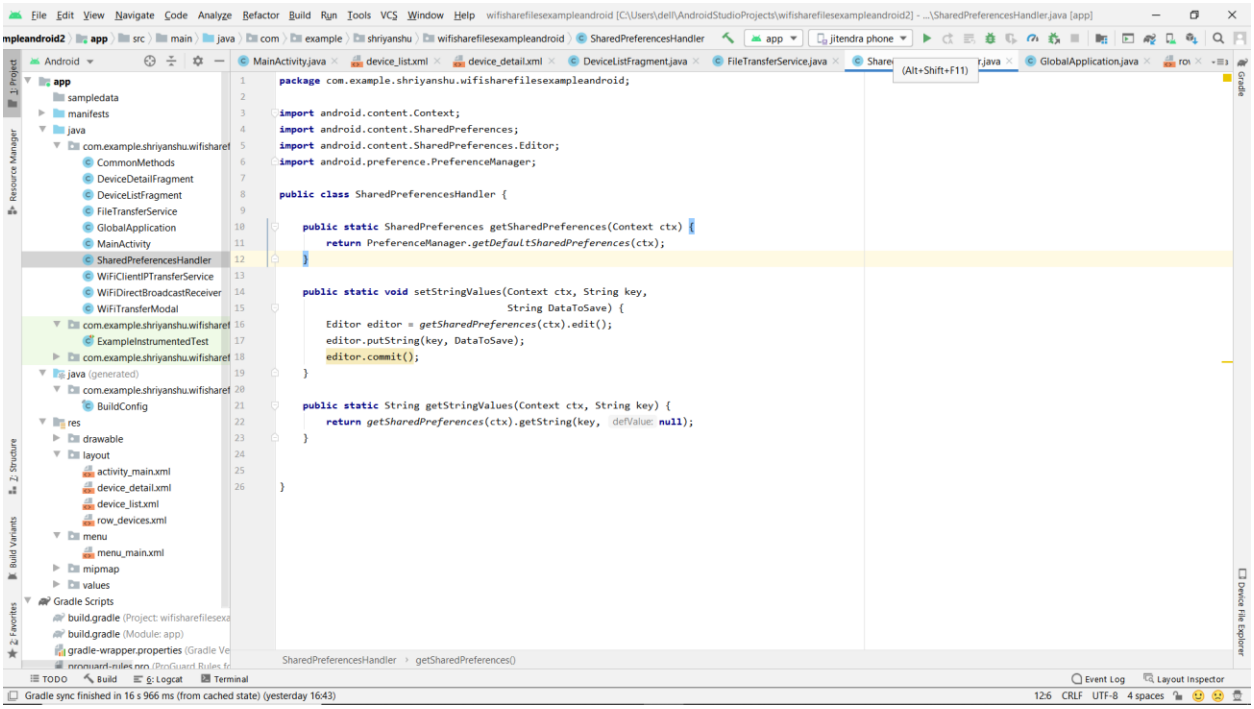
```
package com.example.shriyanshu.wifisharefilesexampleandroid;
import android.app.ListFragment;
import android.app.ProgressDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.net.wifi.p2p.WifiP2pConfig;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pDeviceList;
import android.net.wifi.p2p.WifiP2pManager.PeerListListener;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import java.util.ArrayList;
import java.util.List;

public class DeviceListFragment extends ListFragment implements PeerListListener {
    private List<WifiP2pDevice> peers = new ArrayList<WifiP2pDevice>();
    ProgressDialog progressDialog = null;
    View mContentView = null;
    private WifiP2pDevice device;
    @Override
    public void onActivityCreated(Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
        this.setListAdapter(new WiFiPeerListAdapter(getActivity(), R.layout.row_devices, peers));
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        mContentView = inflater.inflate(R.layout.device_list, null);
        return mContentView;
    }
    public WifiP2pDevice getDevice() {
        return device;
    }
    private static String getDeviceStatus(int deviceStatus) {
        switch (deviceStatus) {
            case WifiP2pDevice.AVAILABLE:
                return "Available";
            case WifiP2pDevice.INVITED:
                return "Invited";
            case WifiP2pDevice.CONNECTED:
                return "Connected";
            case WifiP2pDevice.FAILED:
                return "Failed";
            case WifiP2pDevice.UNAVAILABLE:
                return "Unavailable";
            default:
                return "Unknown";
        }
    }
    @Override
    public void onListItemClick(ListView l, View v, int position, long id) {
        WifiP2pDevice device = (WifiP2pDevice) getListAdapter().getItem(position);
        ((DeviceActionListener) getActivity()).showDetails(device);
    }
    private class WiFiPeerListAdapter extends ArrayAdapter<WifiP2pDevice> {
```



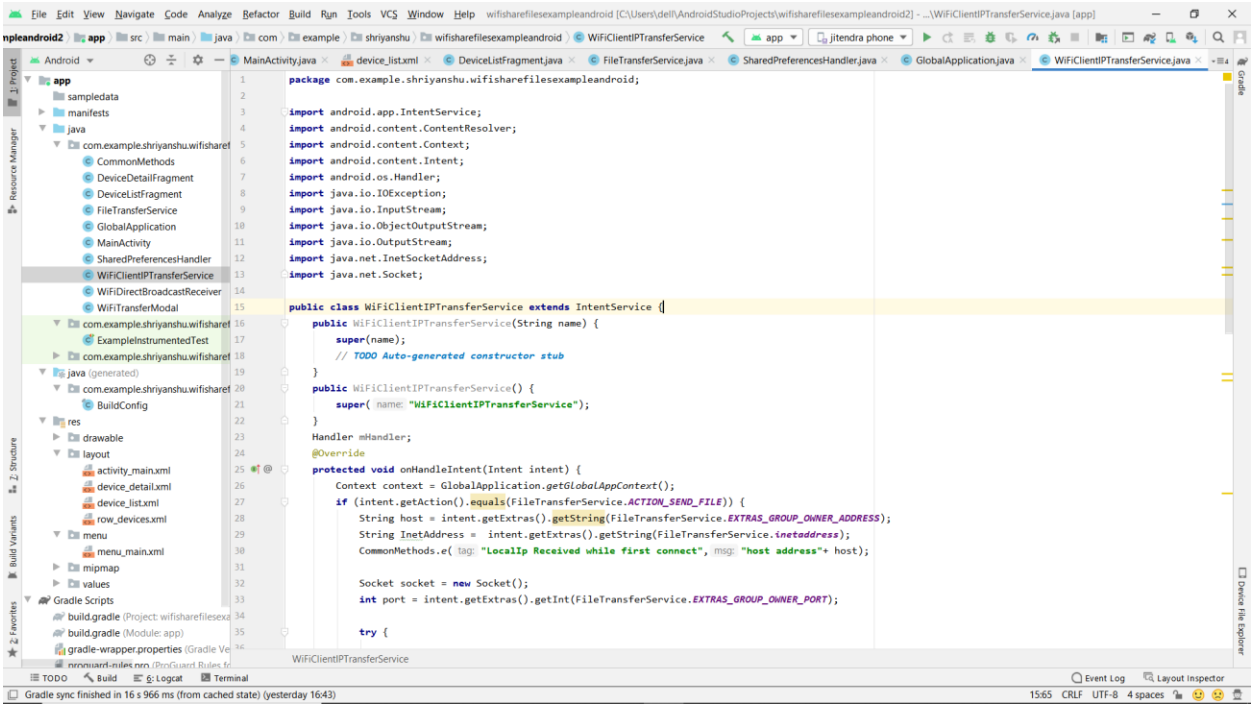
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.app.IntentService;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.os.Handler;
import android.widget.Toast;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;
public class FileTransferService extends IntentService {

    Handler mHandler;

    public static final int SOCKET_TIMEOUT = 5000;
    public static final String ACTION_SEND_FILE = "com.example.shriyanshu.wifisharefilesexampleandroid.SEND_FILE";
    public static final String EXTRAS_FILE_PATH = "file_url";
    public static final String EXTRAS_GROUP_OWNER_ADDRESS = "go_host";
    public static final String EXTRAS_GROUP_OWNER_PORT = "go_port";

    public static  int PORT = 8888;
    public static final String inetaddress = "inetaddress";
    public static final int ByteSize = 512;
    public static final String Extension = "extension";
    public static final String Filelength = "filelength";
    public FileTransferService(String name) {
        super(name);
    }

    public FileTransferService() {
        super( name: "FileTransferService");
    }

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        mHandler = new Handler();
```



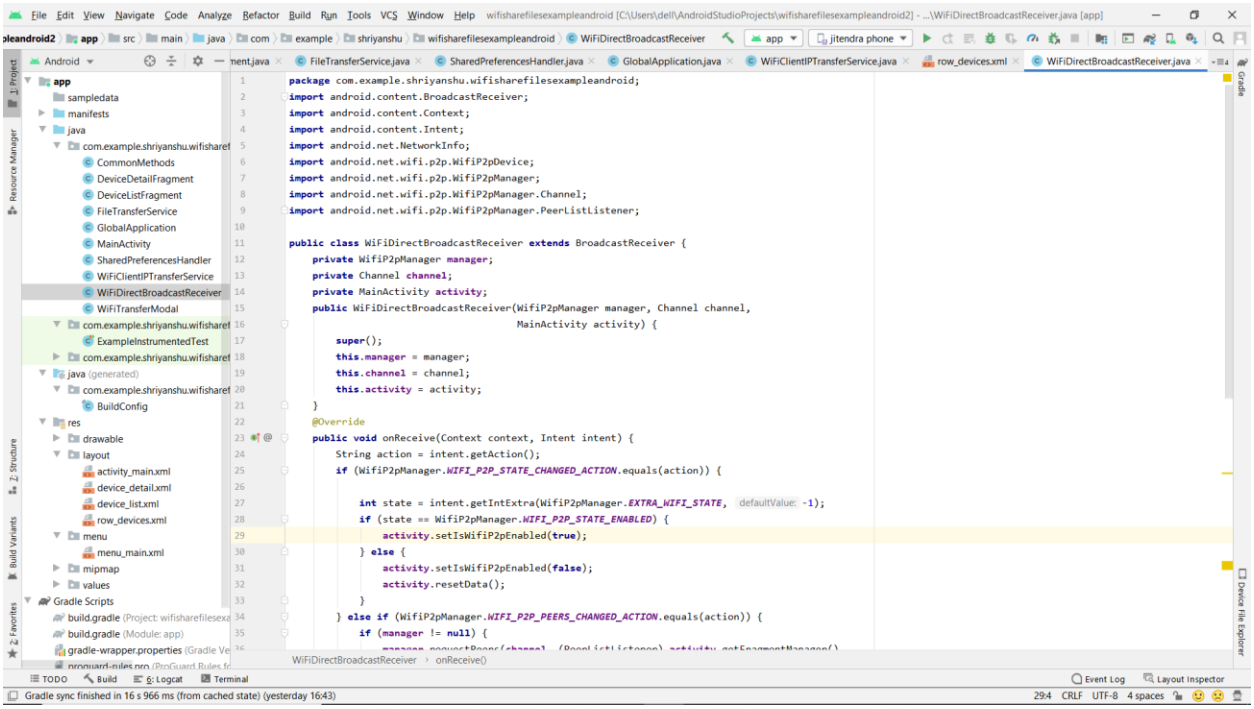
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.app.Application;
import android.content.Context;

public class GlobalApplication extends Application {
    private static Context GlobalContext;

    @Override
    public void onCreate() {
        // TODO Auto-generated method stub
        super.onCreate();
        if(GlobalApplication.GlobalContext == null){
            GlobalApplication.GlobalContext = getApplicationContext();
        }
    }

    public static Context getGlobalAppContext() {
        return GlobalApplication.GlobalContext;
    }
}
```



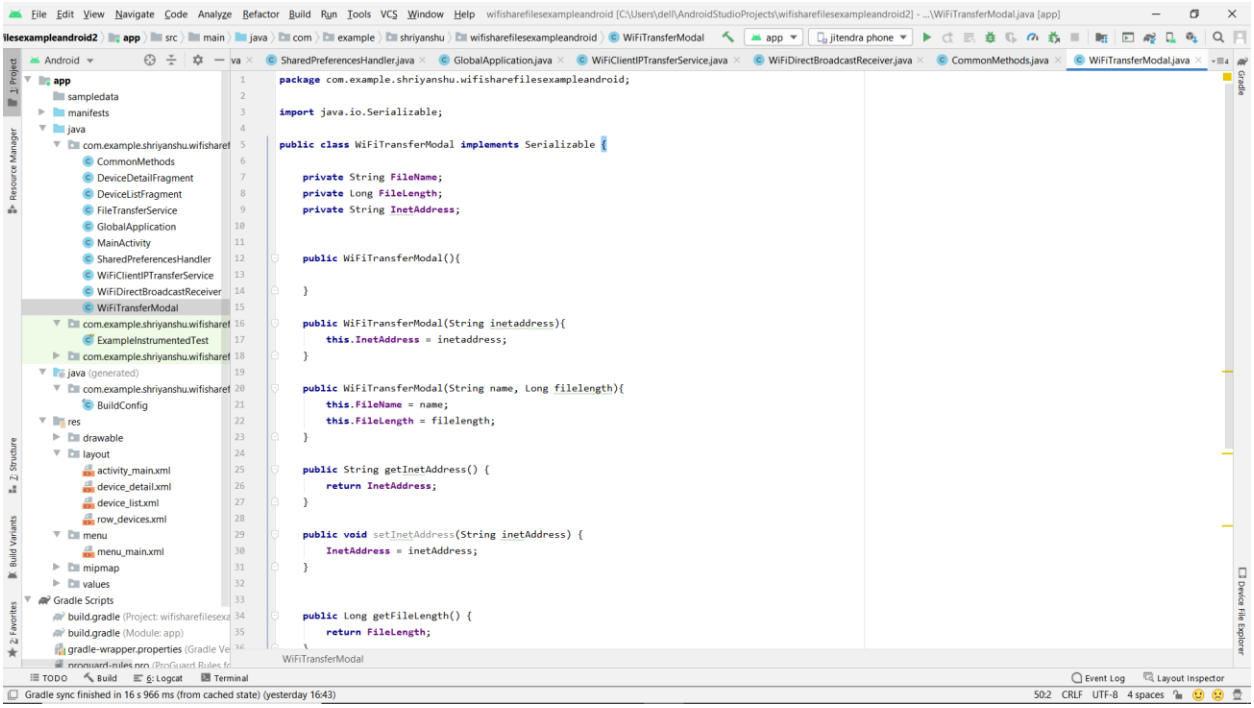
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.content.Context;
import android.content.SharedPreferences;
import android.content.SharedPreferences.Editor;
import android.preference.PreferenceManager;

public class SharedPreferencesHandler {

    public static SharedPreferences getSharedPreferences(Context ctx) {
        return PreferenceManager.getDefaultSharedPreferences(ctx);
    }

    public static void setStringValues(Context ctx, String key,
                                       String DataToSave) {
        Editor editor = getSharedPreferences(ctx).edit();
        editor.putString(key, DataToSave);
        editor.commit();
    }

    public static String getStringValues(Context ctx, String key) {
        return getSharedPreferences(ctx).getString(key, defValue: null);
    }


}
```



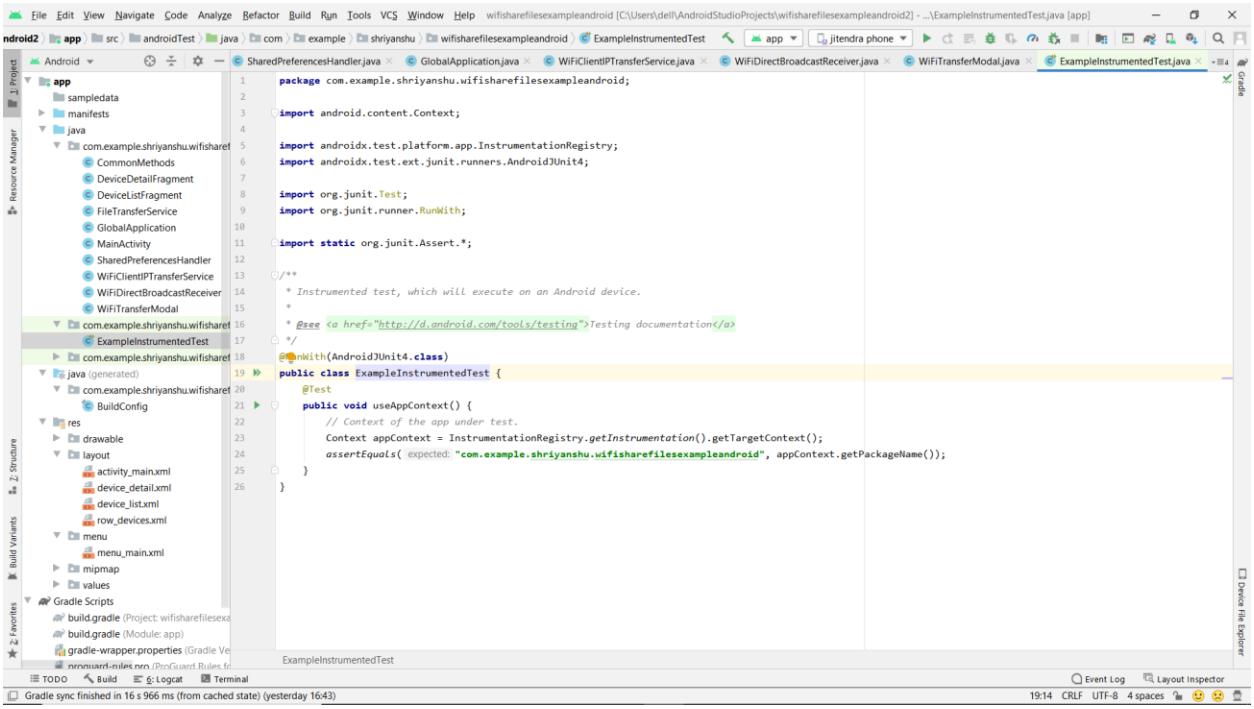
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.app.IntentService;
import android.content.ContentResolver;
import android.content.Context;
import android.content.Intent;
import android.os.Handler;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectOutputStream;
import java.io.OutputStream;
import java.net.InetSocketAddress;
import java.net.Socket;

public class WiFiClientIPTransferService extends IntentService {
    public WiFiClientIPTransferService(String name) {
        super(name);
        // TODO Auto-generated constructor stub
    }
    public WiFiClientIPTransferService() {
        super( name: "WiFiClientIPTransferService");
    }
    Handler mHandler;
    @Override
    protected void onHandleIntent(Intent intent) {
        Context context = GlobalApplication.getGlobalAppContext();
        if (intent.getAction().equals(FileTransferService.ACTION_SEND_FILE)) {
            String host = intent.getExtras().getString(FileTransferService.EXTRAS_GROUP_OWNER_ADDRESS);
            String InetAddress =  intent.getExtras().getString(FileTransferService.inetaddress);
            CommonMethods.e( tag: "LocalIp Received while first connect", msg: "host address"+ host);

            Socket socket = new Socket();
            int port = intent.getExtras().getInt(FileTransferService.EXTRAS_GROUP_OWNER_PORT);

            try {
```



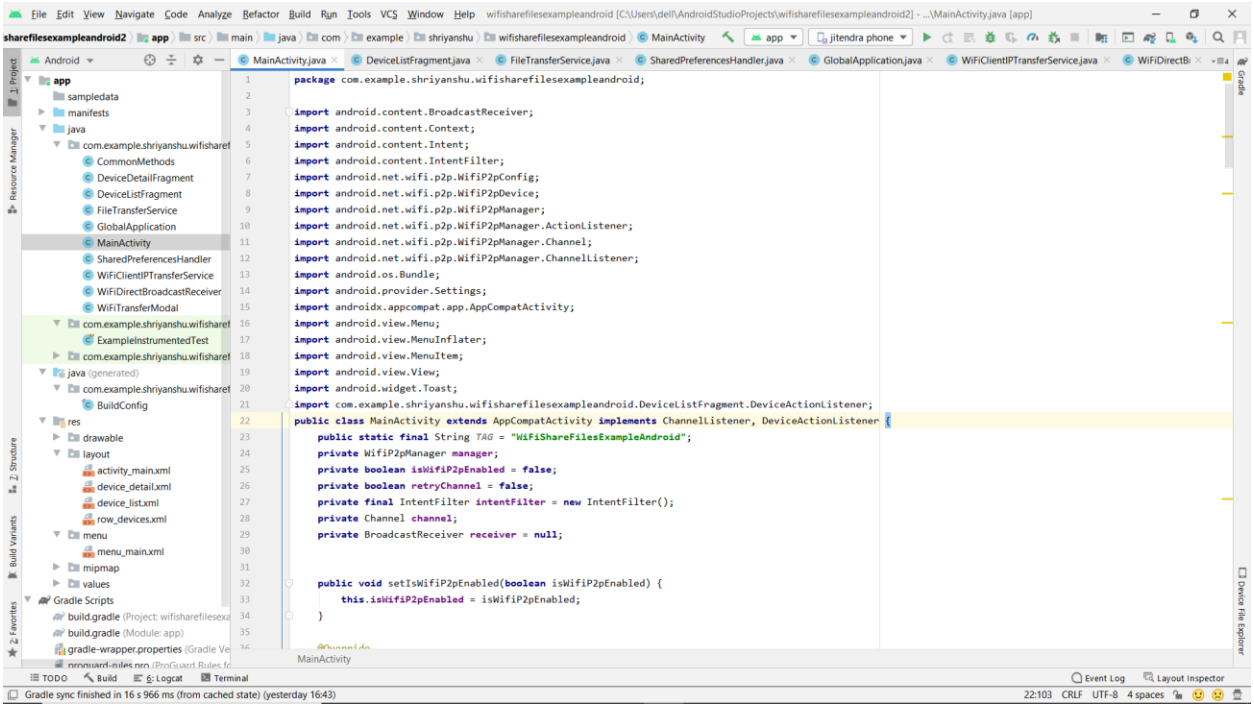
```
package com.example.shriyanshu.wifisharefilesexampleandroid;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.net.NetworkInfo;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import android.net.wifi.p2p.WifiP2pManager.Channel;
import android.net.wifi.p2p.WifiP2pManager.PeerListListener;

public class WiFiDirectBroadcastReceiver extends BroadcastReceiver {
    private WifiP2pManager manager;
    private Channel channel;
    private MainActivity activity;
    public WiFiDirectBroadcastReceiver(WifiP2pManager manager, Channel channel,
                                       MainActivity activity) {
        super();
        this.manager = manager;
        this.channel = channel;
        this.activity = activity;
    }
    @Override
    public void onReceive(Context context, Intent intent) {
        String action = intent.getAction();
        if (WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)) {

            int state = intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE,  defaultValue: -1);
            if (state == WifiP2pManager.WIFI_P2P_STATE_ENABLED) {
                activity.setIsWifiP2pEnabled(true);
            } else {
                activity.setIsWifiP2pEnabled(false);
                activity.resetData();
            }
        } else if (WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)) {
            if (manager != null) {
```



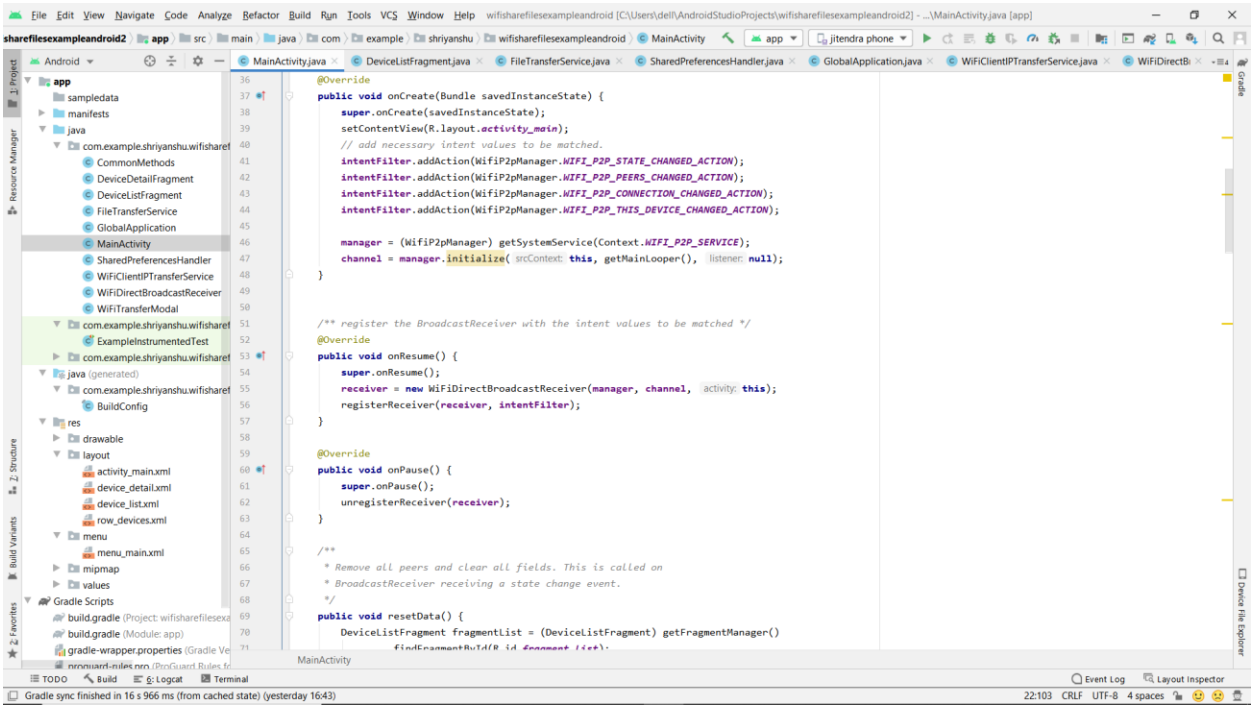
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import java.io.Serializable;

public class WiFiTransferModal implements Serializable {

    private String FileName;
    private Long FileLength;
    private String InetAddress;


    public WiFiTransferModal(){

    }

    public WiFiTransferModal(String inetaddress){
        this.InetAddress = inetaddress;
    }

    public WiFiTransferModal(String name, Long filelength){
        this.FileName = name;
        this.FileLength = filelength;
    }

    public String getInetAddress() {
        return InetAddress;
    }

    public void setInetAddress(String inetAddress) {
        InetAddress = inetAddress;
    }


    public Long getFileLength() {
        return FileLength;
```



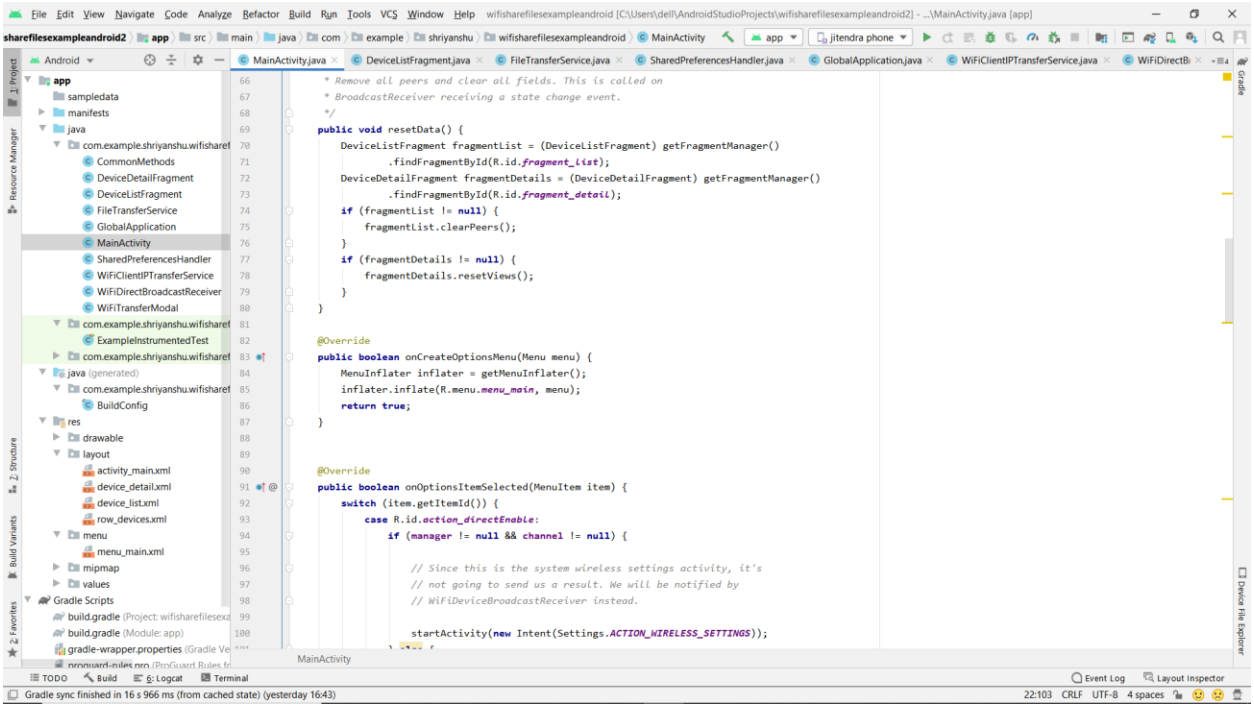
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.content.Context;

import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals( expected: "com.example.shriyanshu.wifisharefilesexampleandroid", appContext.getPackageName());
    }
}
```



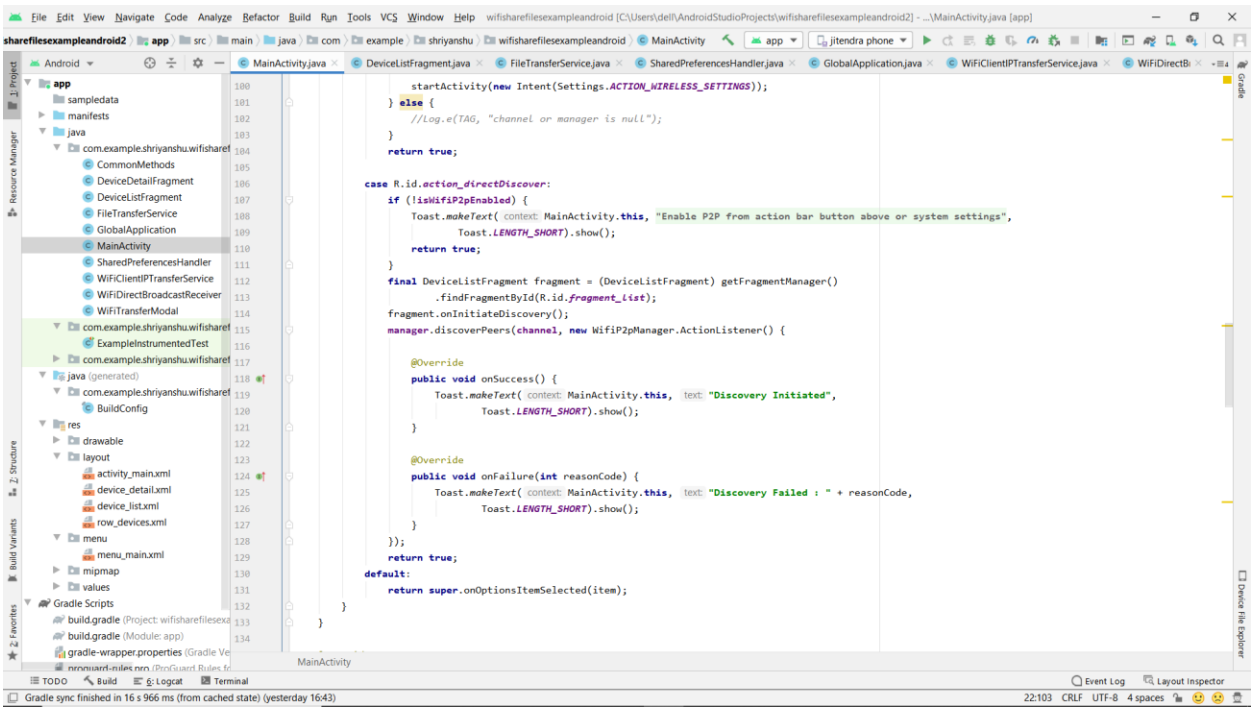
```
package com.example.shriyanshu.wifisharefilesexampleandroid;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.net.wifi.p2p.WifiP2pConfig;
import android.net.wifi.p2p.WifiP2pDevice;
import android.net.wifi.p2p.WifiP2pManager;
import android.net.wifi.p2p.WifiP2pManager.ActionListener;
import android.net.wifi.p2p.WifiP2pManager.Channel;
import android.net.wifi.p2p.WifiP2pManager.ChannelListener;
import android.os.Bundle;
import android.provider.Settings;
import androidx.appcompat.app.AppCompatActivity;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;
import com.example.shriyanshu.wifisharefilesexampleandroid.DeviceListFragment.DeviceActionListener;
public class MainActivity extends AppCompatActivity implements ChannelListener, DeviceActionListener {
    public static final String TAG = "WiFiShareFilesExampleAndroid";
    private WifiP2pManager manager;
    private boolean isWifiP2pEnabled = false;
    private boolean retryChannel = false;
    private final IntentFilter intentFilter = new IntentFilter();
    private Channel channel;
    private BroadcastReceiver receiver = null;


    public void setIsWifiP2pEnabled(boolean isWifiP2pEnabled) {
        this.isWifiP2pEnabled = isWifiP2pEnabled;
    }
```



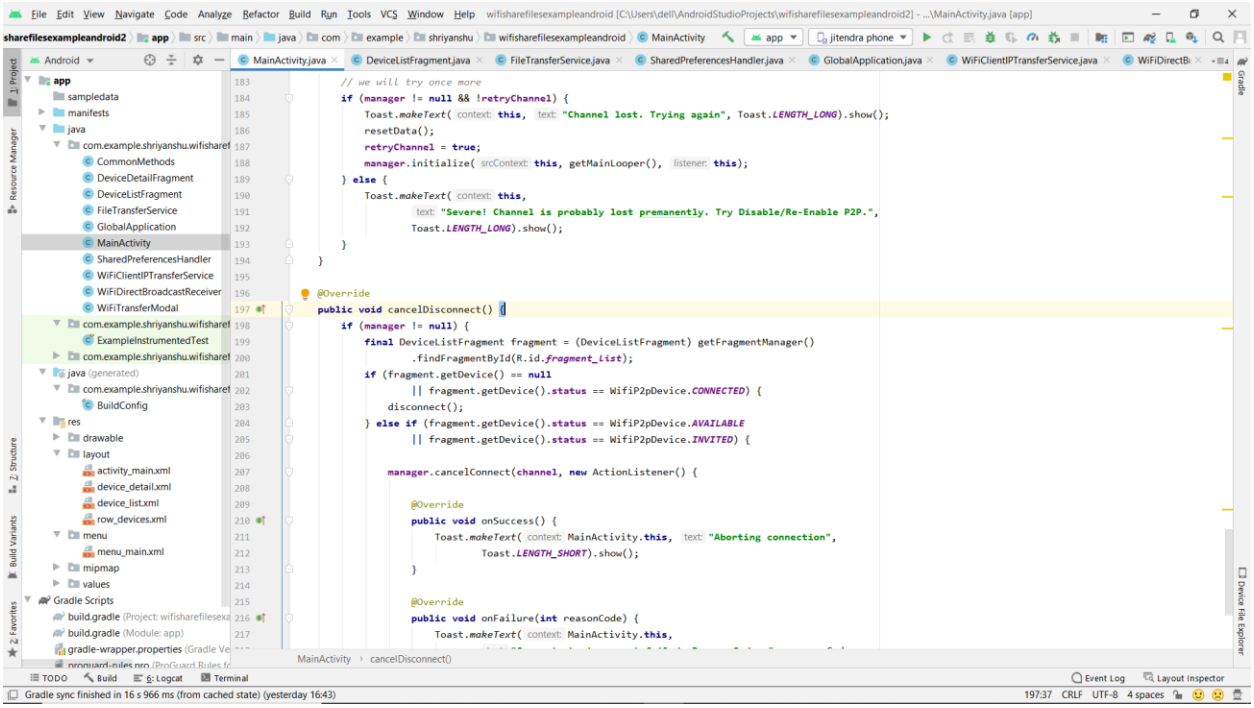
```
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // add necessary intent values to be matched.
        intentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
        intentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
        intentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
        intentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);

        manager = (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
        channel = manager.initialize( srcContext: this, getMainLooper(), listener: null);
    }


    /** register the BroadcastReceiver with the intent values to be matched */
    @Override
    public void onResume() {
        super.onResume();
        receiver = new WiFiDirectBroadcastReceiver(manager, channel, activity: this);
        registerReceiver(receiver, intentFilter);
    }

    @Override
    public void onPause() {
        super.onPause();
        unregisterReceiver(receiver);
    }

    /**
     * Remove all peers and clear all fields. This is called on
     * BroadcastReceiver receiving a state change event.
     */
    public void resetData() {
        DeviceListFragment fragmentList = (DeviceListFragment) getFragmentManager()
```



```
 * Remove all peers and clear all fields. This is called on
 * BroadcastReceiver receiving a state change event.
 */
public void resetData() {
    DeviceListFragment fragmentList = (DeviceListFragment) getFragmentManager()
            .findFragmentById(R.id.fragment_list);
    DeviceDetailFragment fragmentDetails = (DeviceDetailFragment) getFragmentManager()
            .findFragmentById(R.id.fragment_detail);
    if (fragmentList != null) {
        fragmentList.clearPeers();
    }
    if (fragmentDetails != null) {
        fragmentDetails.resetViews();
    }
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu_main, menu);
    return true;
}


@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.action_directEnable:
            if (manager != null && channel != null) {

                // Since this is the system wireless settings activity, it's
                // not going to send us a result. We will be notified by
                // WiFiDeviceBroadcastReceiver instead.

                startActivity(new Intent(Settings.ACTION_WIRELESS_SETTINGS));
```



```
                startActivity(new Intent(Settings.ACTION_WIRELESS_SETTINGS));
            } else {
                //Log.e(TAG, "channel or manager is null");
            }
            return true;

        case R.id.action_directDiscover:
            if (!isWifiP2pEnabled) {
                Toast.makeText( context: MainActivity.this, "Enable P2P from action bar button above or system settings",
                        Toast.LENGTH_SHORT).show();
                return true;
            }
            final DeviceListFragment fragment = (DeviceListFragment) getFragmentManager()
                    .findFragmentById(R.id.fragment_list);
            fragment.onInitiateDiscovery();
            manager.discoverPeers(channel, new WifiP2pManager.ActionListener() {

                @Override
                public void onSuccess() {
                    Toast.makeText( context: MainActivity.this,  text: "Discovery Initiated",
                            Toast.LENGTH_SHORT).show();
                }

                @Override
                public void onFailure(int reasonCode) {
                    Toast.makeText( context: MainActivity.this,  text: "Discovery Failed : " + reasonCode,
                            Toast.LENGTH_SHORT).show();
                }
            });
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```



GitHub Link : https://github.com/Nehaadnekar/File-Sharing-App-MINI-PROJECT

c

References

1) https://nevonprojects.com/secure-file-sharing-using-access-control/
2) https://freeprojectsforall.com/an046-secure-file-sharing-using-access-control/
3) https://ijarcce.com/wp-content/uploads/2012/03/IJARCCE5G_a_shailes_secured.pdf