MINI PROJECT
(2020-21)

DOWNTUBE

MID-TERM REPORT

Institute of Engineering & Technology

Submitted by-

Mehul Pratap Singh
(181500383)

Supervised By: -

Mr. Piyush Vashisth Sir

Department of Computer Engineering & Applications

Contents

Abstract

In this we will build an android application which can save saomeone’s instagram post , whatsapp status and youtube videos to your internal storage of your android device.

An apk is an android Package is the package file format used by the Android operating system, and a number of other Android-based operating systems for distribution and installation of mobile apps, mobile games and middleware.

Introduction

1.1 General Introduction to the topic

Sometimes while scrolling instagram walls,we found some photos useful and want to save them for future without compromising with the image quality. And with youtube also if we save the videos in offline in youtube only it took some mobile data while in offline tab for changing they thumbnails and sometimes it re download a video, also we can’t take these videos in sd card and watch them on our laptop , for this we have to use mobile data even if we have saved it offline in another device .
While searching for the solution we found that there are some apps which can save whatsapp status , some download instagram posts and videos and different app for youtube videos download so I end up making this app which can do all these things in one app.

DOwN TuBE

all you need at one place

About Downtube: -

- It is a complete solution you need to download your loved files from most loved social media apps..
- You can download whatsapp status , YouTube videos , instagram posts in just one click.
- you do not need three different apps for three different works.
- In next versions we will give access to download videos from other videos streaming platform as well.

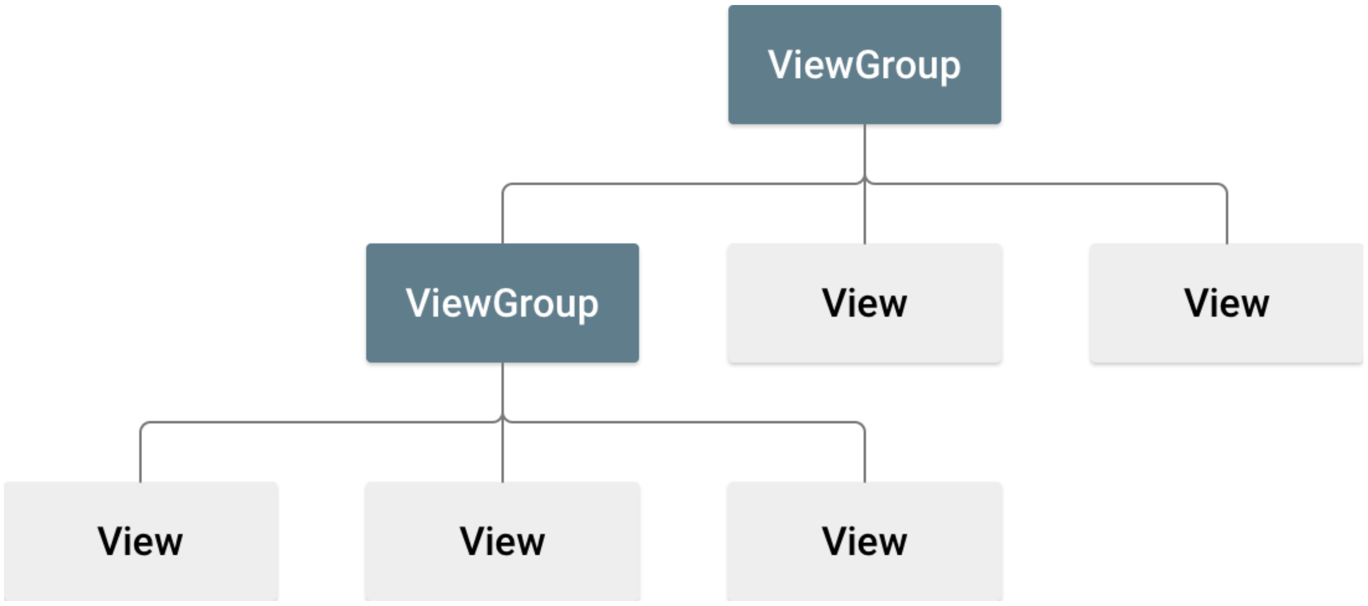
About Android Studio: -

- Android Studio is the official integrated development environment (IDE) for Android application development. It is based on the IntelliJ IDEA, a Java integrated development environment for software, and incorporates its code editing and developer tools.

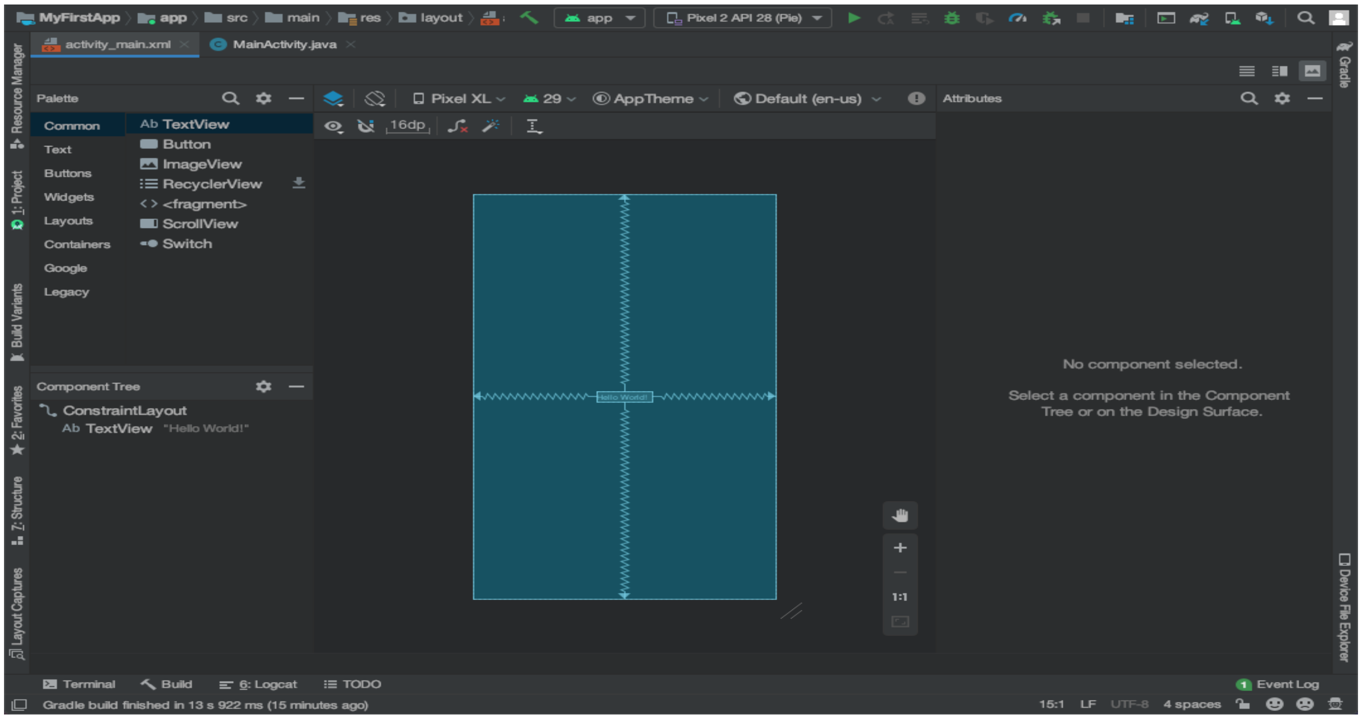
- To support application development within the Android operating system, Android Studio uses a Gradle-based build system, emulator, code templates, and Github integration. Every project in Android Studio has one or more modalities with source code and resource files. These modalities include Android app modules, Library modules, and Google App Engine modules.
- Android Studio uses an Instant Push feature to push code and resource changes to a running application. A code editor assists the developer with writing code and offering

code completion, refraction, and analysis. Applications built in Android Studio are then compiled into the APK format for submission to the Google Play Store.

About front end in android studio: -



It is very easy to design front end of an app in android studio.



No component selected.

Select a component in the Component Tree or on the Design Surface.

Gradle build finished in 13 s 922 ms (15 minutes ago)

Drag and drop makes our work very easy and simple and saves a lot of time.

Support devices: -

This app will support on any android devices working on android version 4.0 and above

<u>Tested devices</u>

DOwN TuBE can work in almost all android devices but we personally checked the app on these devices , except some features all are working , which are implemented till now

1. redmi note 5 pro
2. Pixel 3
3. Pixel 3xl
4. coolpad note 3
5. poco x3
6. redmi 3s prime

1.3 Hardware Requirements

- Memory [4GB RAM (or higher)]
- Intel core i3 64-bit Processor (or higher)
- Hard disk (for backup)

1.3 Software requirements

- JDK 11
- Operating system(windows or linux)
- Android studio
- Language (JAVA)

Objective

Our objective is to create an app

1. which is user friendly.

2. Can help a user in multiple ways.

3. Have no adds.

4. No paid features everything is free.

5. Open source so no trouble for security.

Implementation Details

Part1: building the main page of the application

1. In the Project window, open app > res > layout > activity_main.xml.

2. To make room for the Layout Editor, hide the Project window. To do so, select View > Tool Windows > Project, or just click Project on the left side of the Android Studio screen.

3. If your editor shows the XML source, click the Design tab at the bottom of the window.

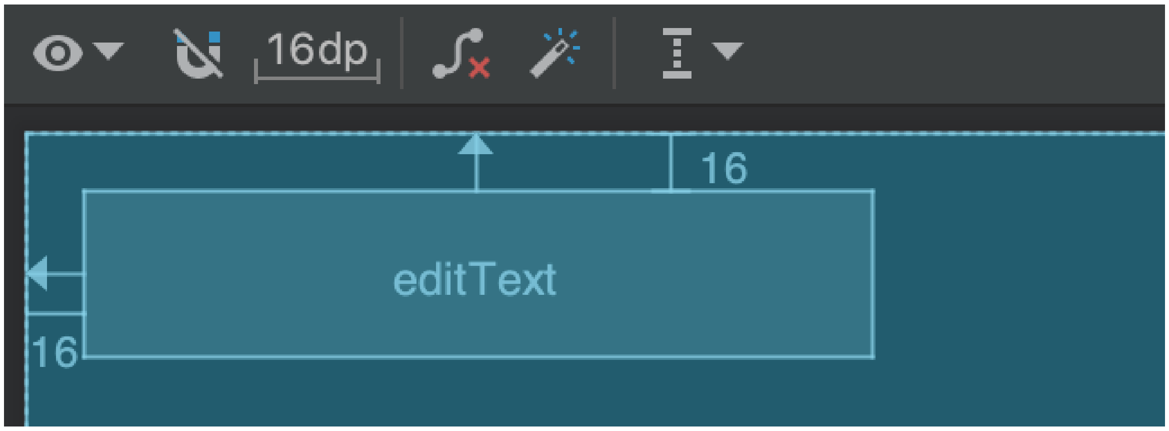
4. Click Select Design Surface and select Blueprint.

5. Click Show in the Layout Editor toolbar and make sure that Show All Constraints is checked.

6. Click Default Margins in the toolbar and select 16. If needed, you can adjust the margins for each view later.

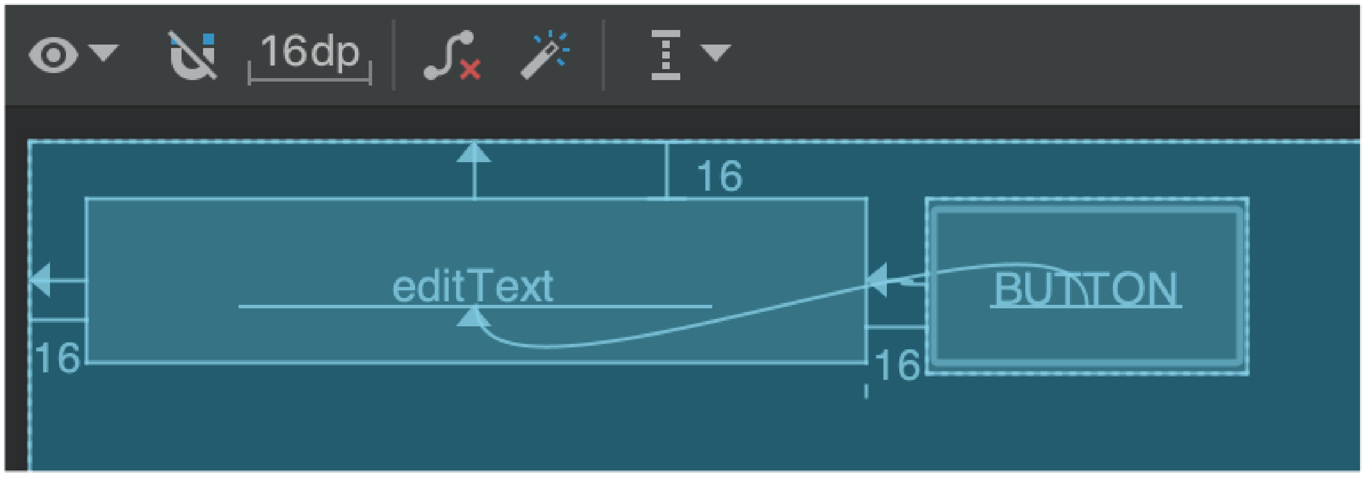
7. Click Device for Preview in the toolbar and select 5.5, 1440 × 2560, 560 dpi (Pixel XL).

Part 2:
Add a text box



1. First, you need to remove what's already in the layout. Click TextView in the Component Tree panel and then press the Delete key.
2. In the Palette panel, click Text to show the available text controls.
3. Drag the Plain Text into the design editor and drop it near the top of the layout. This is an EditText widget that accepts plain text input.
4. Click the view in the design editor. You can now see the square handles to resize the view on each corner, and the circular constraint anchors on each side. For better control, you might want to zoom in on the editor. To do so, use the Zoom buttons in the Layout Editor toolbar.
5. Click and hold the anchor on the top side, drag it up until it snaps to the top of the layout, and then release it. That's a constraint: it constrains the view within the default margin that was set. In this case, you set it to 16 dp from the top of the layout.
6. Use the same process to create a constraint from the left side of the view to the left side of the layout.

Part 3:
Add a button



1. In the Palette panel, click Buttons.
2. Drag the Button widget into the design editor and drop it near the right side.
3. Create a constraint from the left side of the button to the right side of the text box.
4. To constrain the views in a horizontal alignment, create a constraint between the text baselines. To do so, right-click the button and then select Show Baseline .
5. The baseline anchor appears inside the button. Click and hold this anchor, and then drag it to the baseline anchor that appears in the adjacent text box

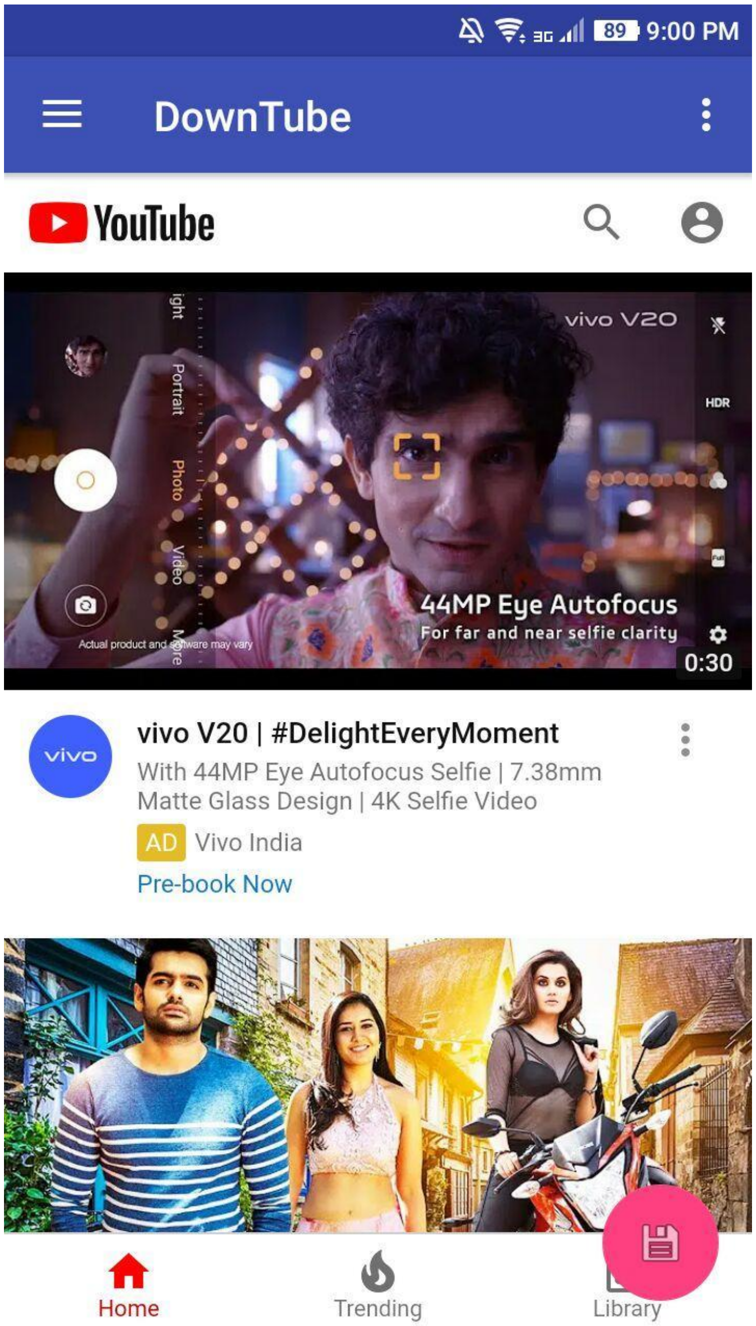
Progress

Currently the feature available are

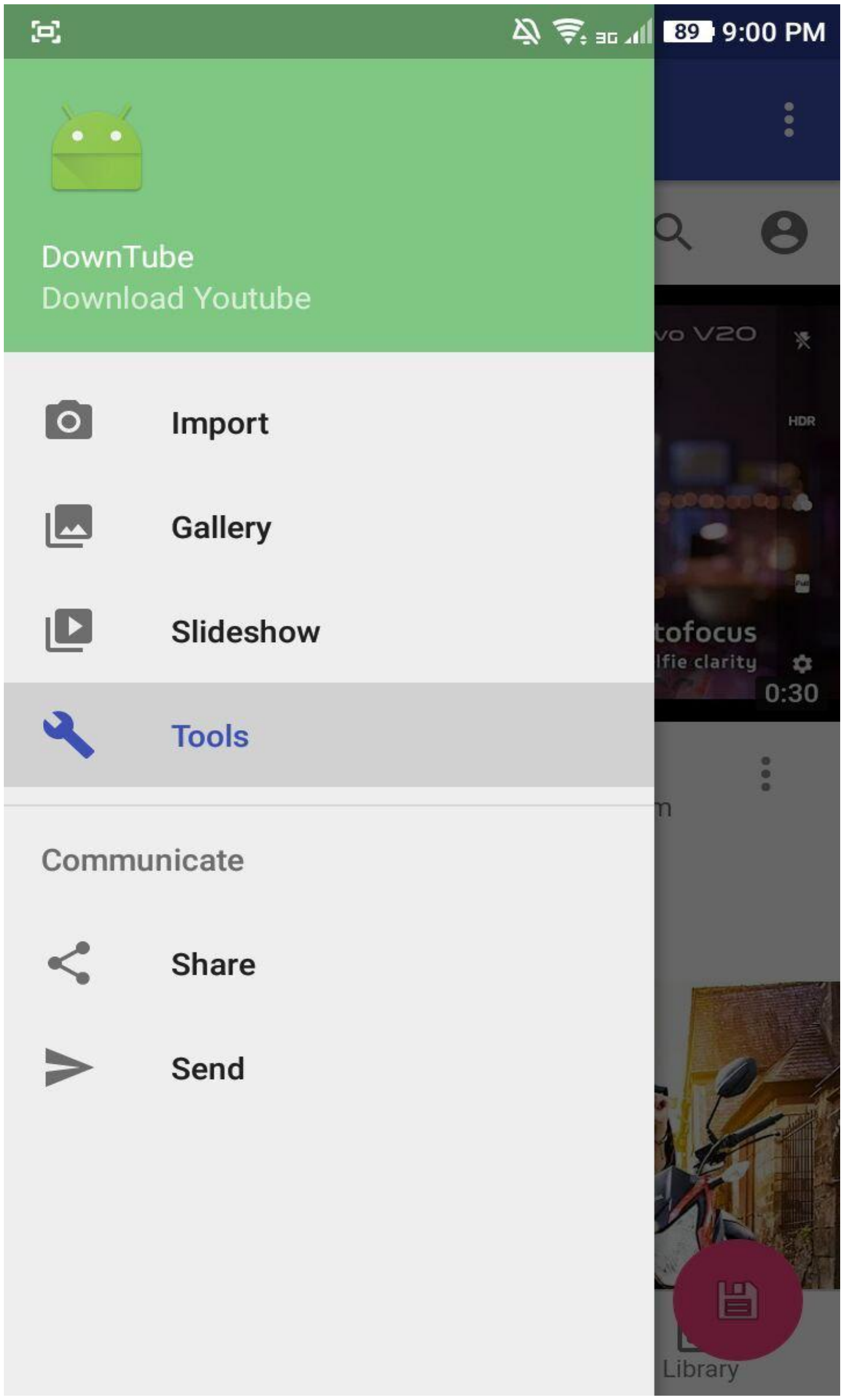
- Front end is dveloped
- UI and colors pannels are completed
- Now u can donwload youtube videos
- You can watch youtube videos in the app
- You can change the quality of the videos

SCREENSHOTS

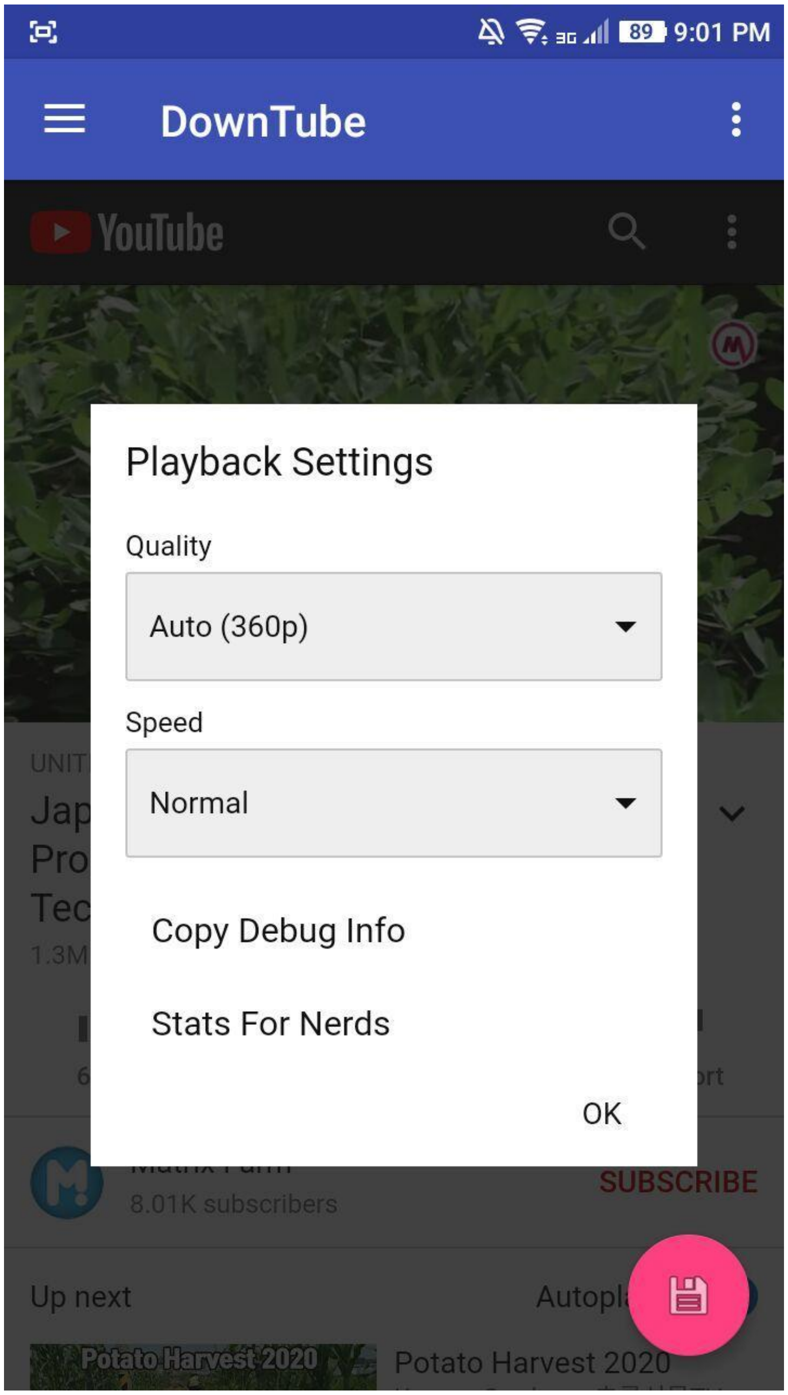
Main Ui of the app


9:00 PM
DownTube
YouTube
vivo V20
44MP Eye Autofocus
For far and near selfie clarity
0:30
vivo V20 | #DelightEveryMoment
With 44MP Eye Autofocus Selfie | 7.38mm Matte Glass Design | 4K Selfie Video
AD Vivo India
Pre-book Now
Home
Trending
Library

Menu button of the app



Playback setting of video



Playback Settings

Quality: Auto (360p)

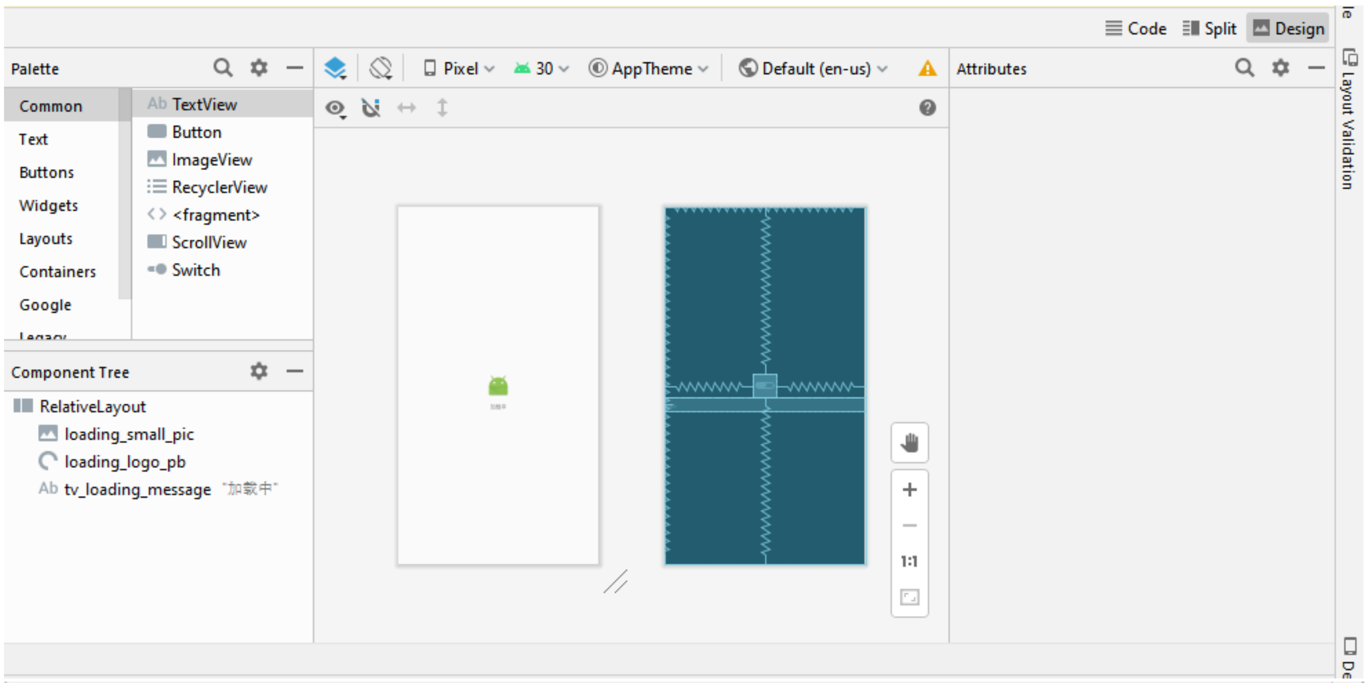
Speed: Normal

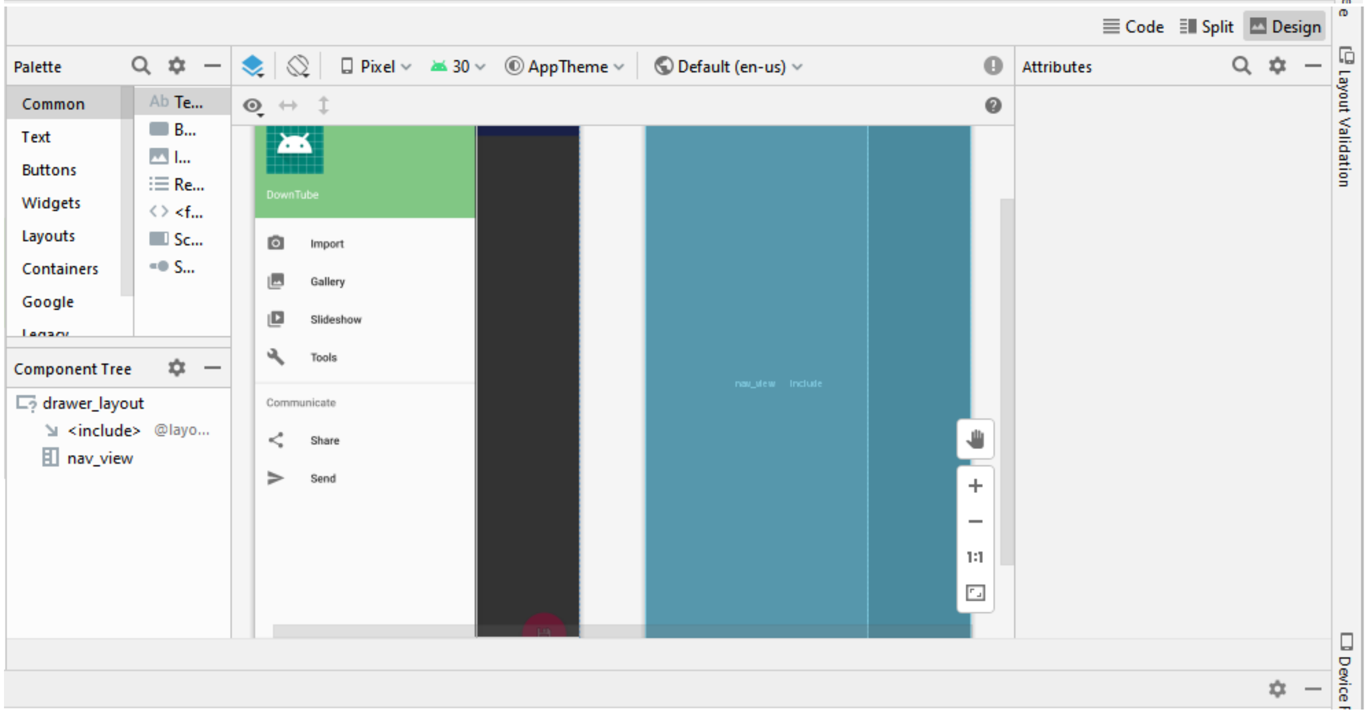
Copy Debug Info

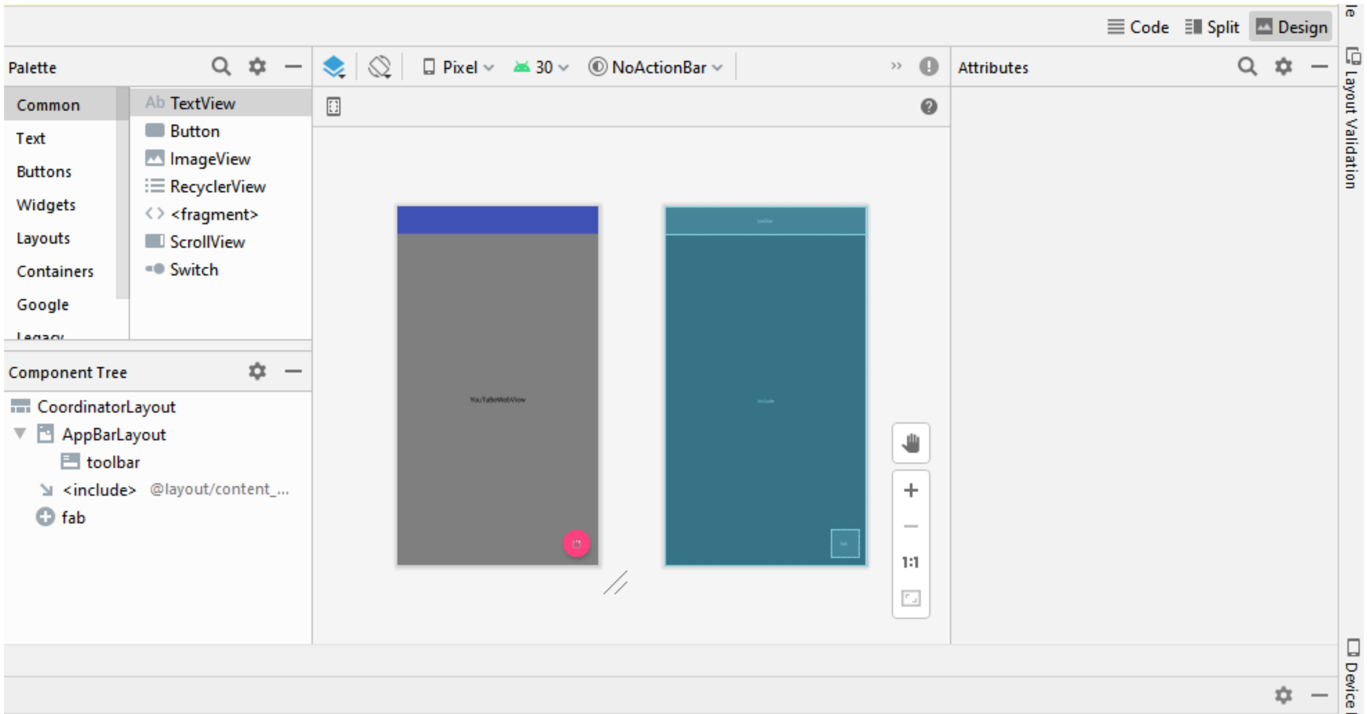
Stats For Nerds

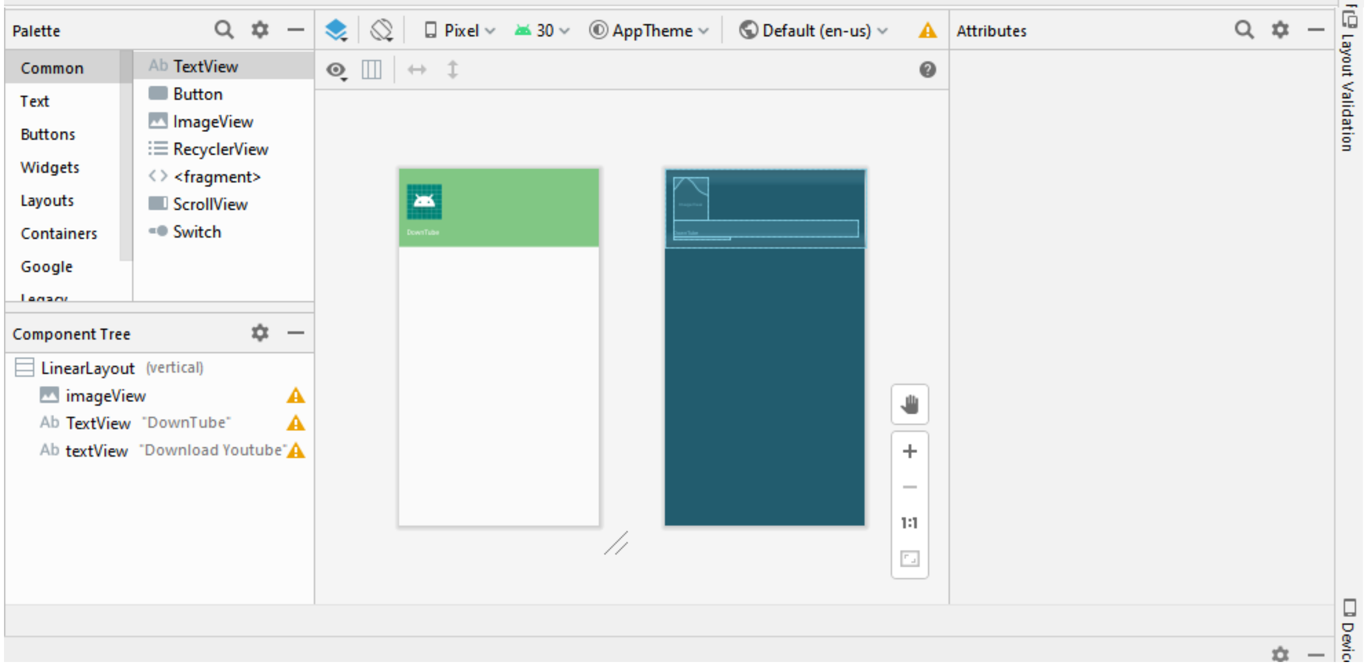
OK

Front end screen shots

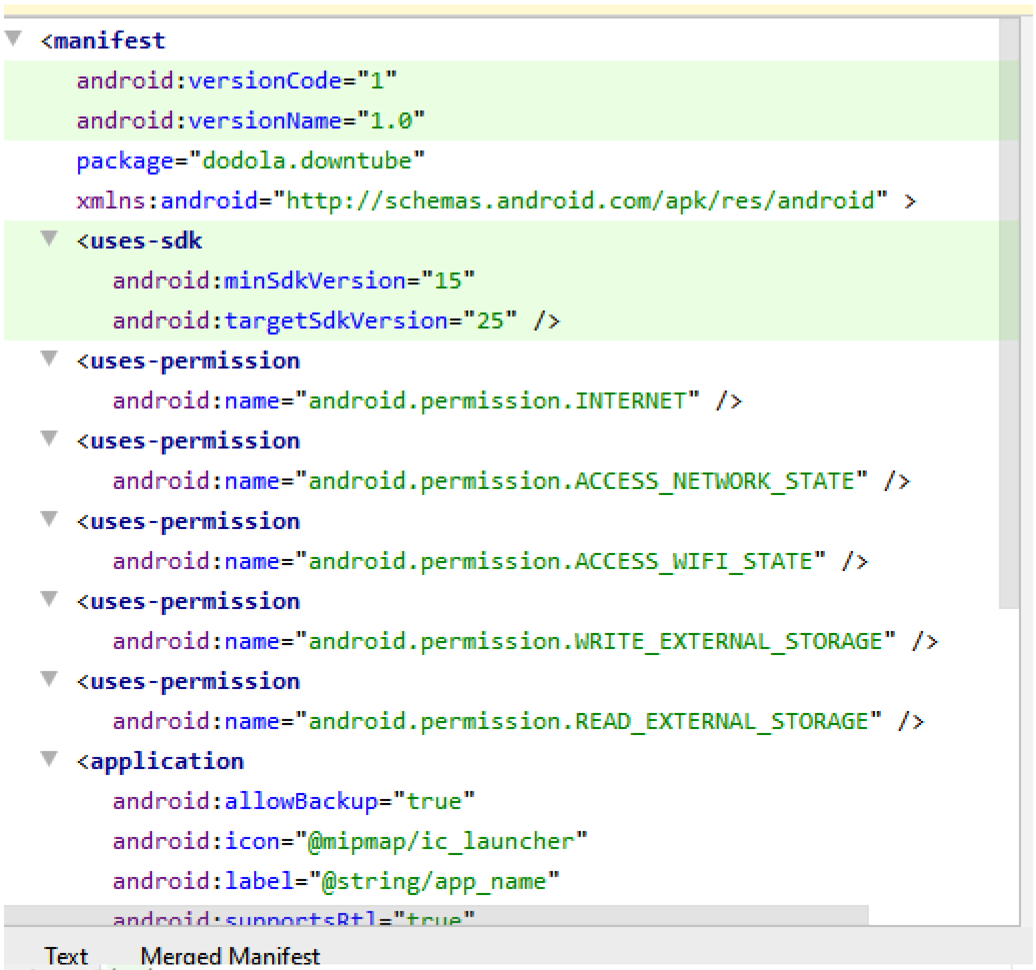




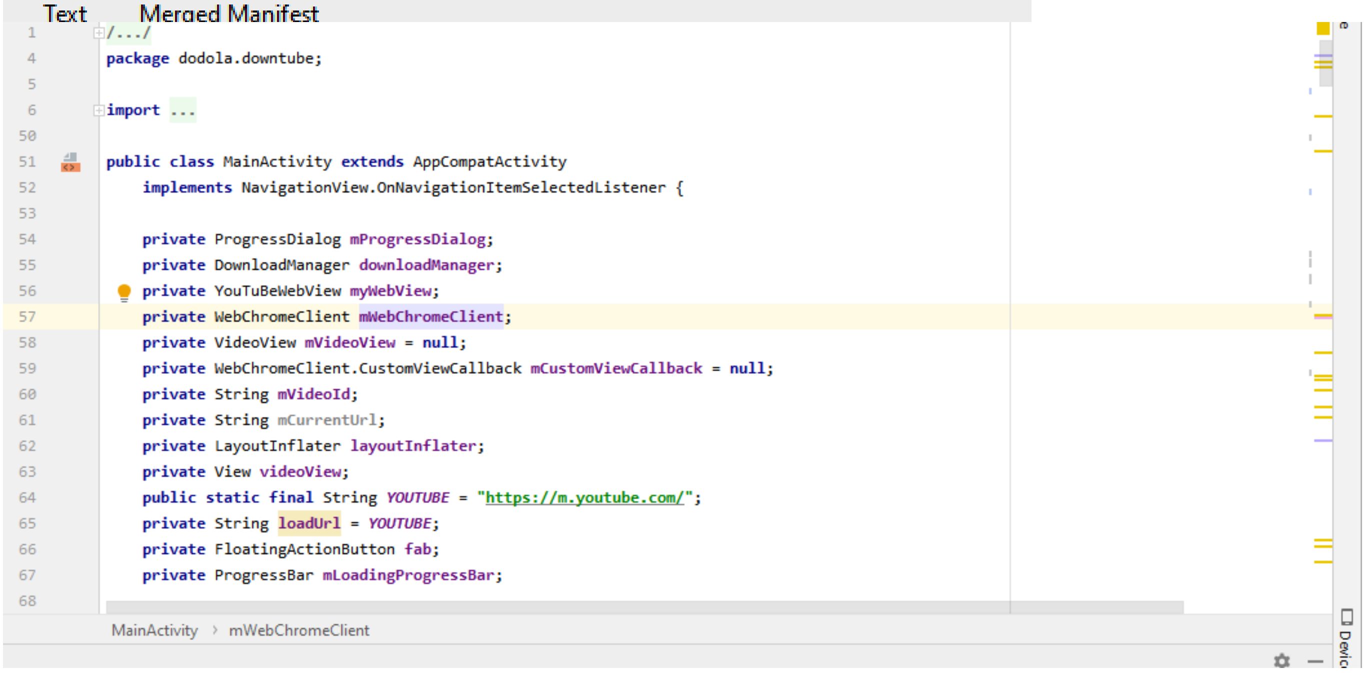

Code
Split
Design
Palette
Pixel
30
NoActionBar
Attributes
Common
Text
Buttons
Widgets
Layouts
Containers
Google
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Component Tree
CoordinatorLayout
AppBarLayout
toolbar
<include> @layout/content_...
fab
Layout Validation
Device F


Palette
Pixel
30
AppTheme
Default (en-us)
Attributes
Common
Text
Buttons
Widgets
Layouts
Containers
Google
TextView
Button
ImageView
RecyclerView
<fragment>
ScrollView
Switch
Component Tree
LinearLayout (vertical)
imageView
TextView "DownTube"
textView "Download Youtube"
Layout Validation
Devic

Some screen shots of code



```xml
<manifest
    android:versionCode="1"
    android:versionName="1.0"
    package="dodola.downtube"
    xmlns:android="http://schemas.android.com/apk/res/android" >
    <uses-sdk
        android:minSdkVersion="15"
        android:targetSdkVersion="25" />
    <uses-permission
        android:name="android.permission.INTERNET" />
    <uses-permission
        android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission
        android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
```

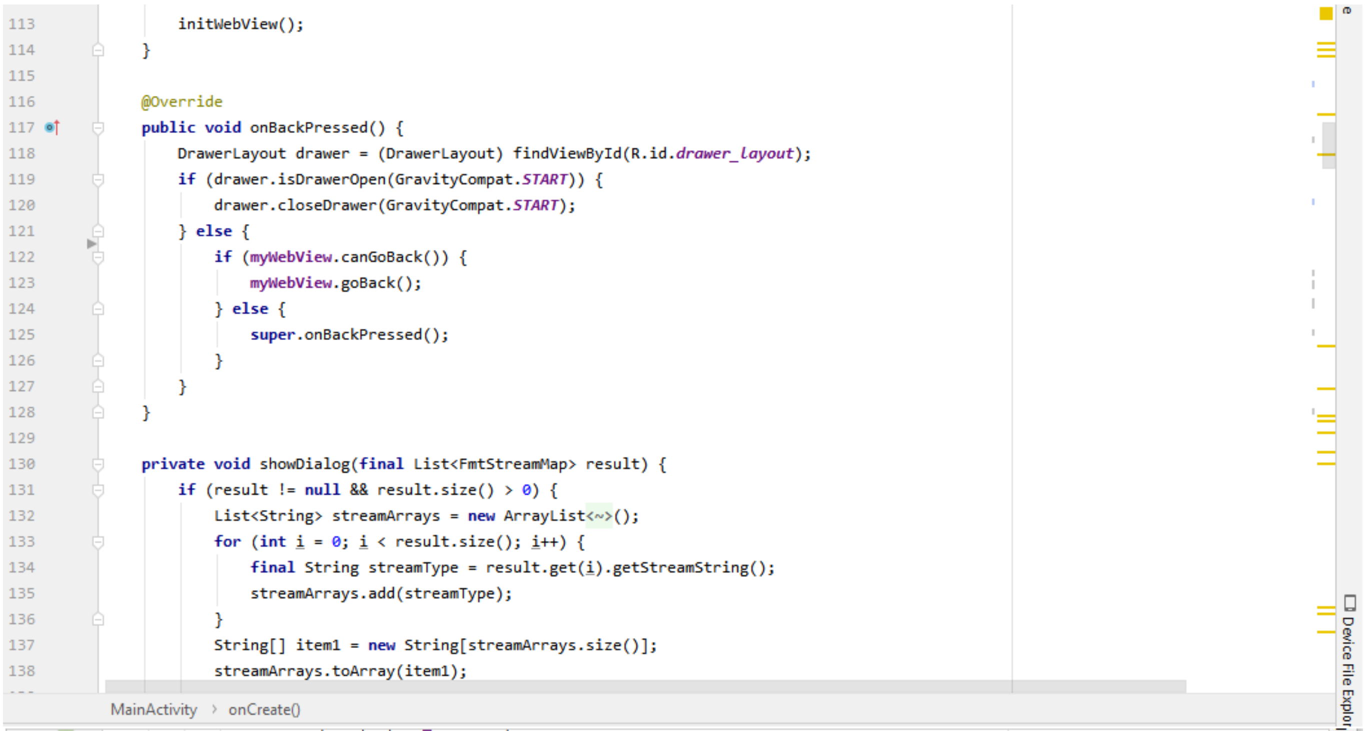


```java
/.../
package dodola.downtube;

import ...

public class MainActivity extends AppCompatActivity
    implements NavigationView.OnNavigationItemSelectedListener {

    private ProgressDialog mProgressDialog;
    private DownloadManager downloadManager;
    private YouTubeWebView myWebView;
    private WebChromeClient mWebChromeClient;
    private VideoView mVideoView = null;
    private WebChromeClient.CustomViewCallback mCustomViewCallback = null;
    private String mVideoId;
    private String mCurrentUrl;
    private LayoutInflater layoutInflater;
    private View videoView;
    public static final String YOUTUBE = "https://m.youtube.com/";
    private String loadUrl = YOUTUBE;
    private FloatingActionButton fab;
    private ProgressBar mLoadingProgressBar;
```

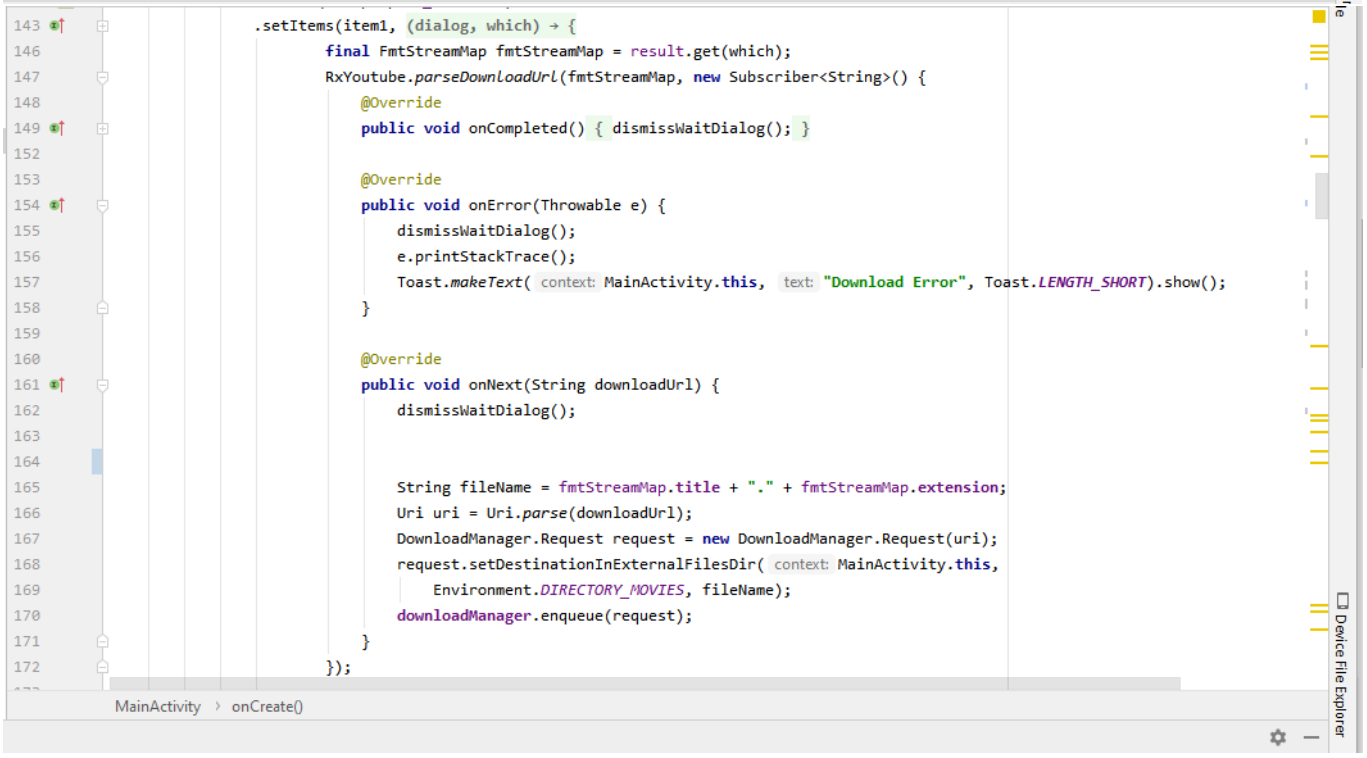


```
        initWebView();
    }

    @Override
    public void onBackPressed() {
        DrawerLayout drawer = (DrawerLayout) findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            if (myWebView.canGoBack()) {
                myWebView.goBack();
            } else {
                super.onBackPressed();
            }
        }
    }

    private void showDialog(final List<FmtStreamMap> result) {
        if (result != null && result.size() > 0) {
            List<String> streamArrays = new ArrayList<~>();
            for (int i = 0; i < result.size(); i++) {
                final String streamType = result.get(i).getStreamString();
                streamArrays.add(streamType);
            }
            String[] item1 = new String[streamArrays.size()];
            streamArrays.toArray(item1);
```

MainActivity › onCreate()



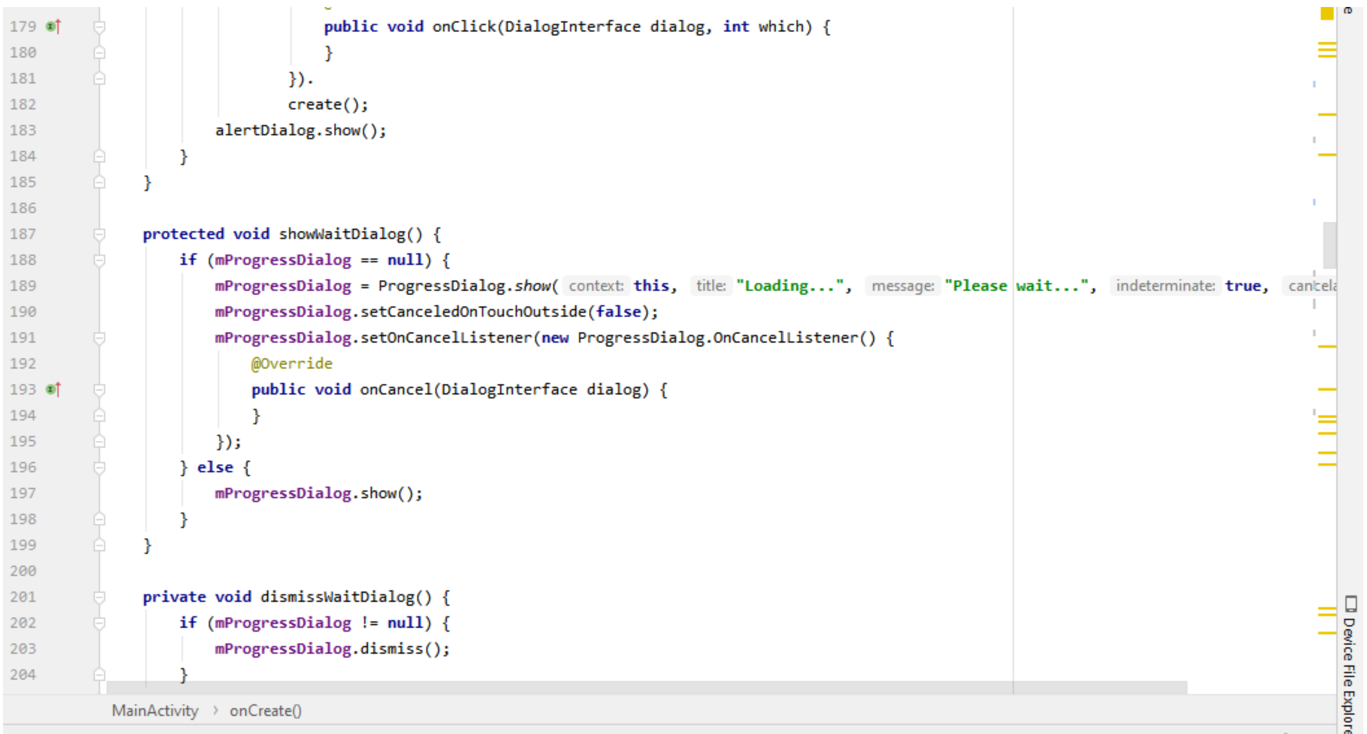
```
                .setItems(item1, (dialog, which) → {
                        final FmtStreamMap fmtStreamMap = result.get(which);
                        RxYoutube.parseDownloadUrl(fmtStreamMap, new Subscriber<String>() {
                            @Override
                            public void onCompleted() { dismissWaitDialog(); }

                            @Override
                            public void onError(Throwable e) {
                                dismissWaitDialog();
                                e.printStackTrace();
                                Toast.makeText( context: MainActivity.this,  text: "Download Error", Toast.LENGTH_SHORT).show();
                            }

                            @Override
                            public void onNext(String downloadUrl) {
                                dismissWaitDialog();


                                String fileName = fmtStreamMap.title + "." + fmtStreamMap.extension;
                                Uri uri = Uri.parse(downloadUrl);
                                DownloadManager.Request request = new DownloadManager.Request(uri);
                                request.setDestinationInExternalFilesDir( context: MainActivity.this,
                                    Environment.DIRECTORY_MOVIES, fileName);
                                downloadManager.enqueue(request);
                            }
                        });
```

MainActivity › onCreate()

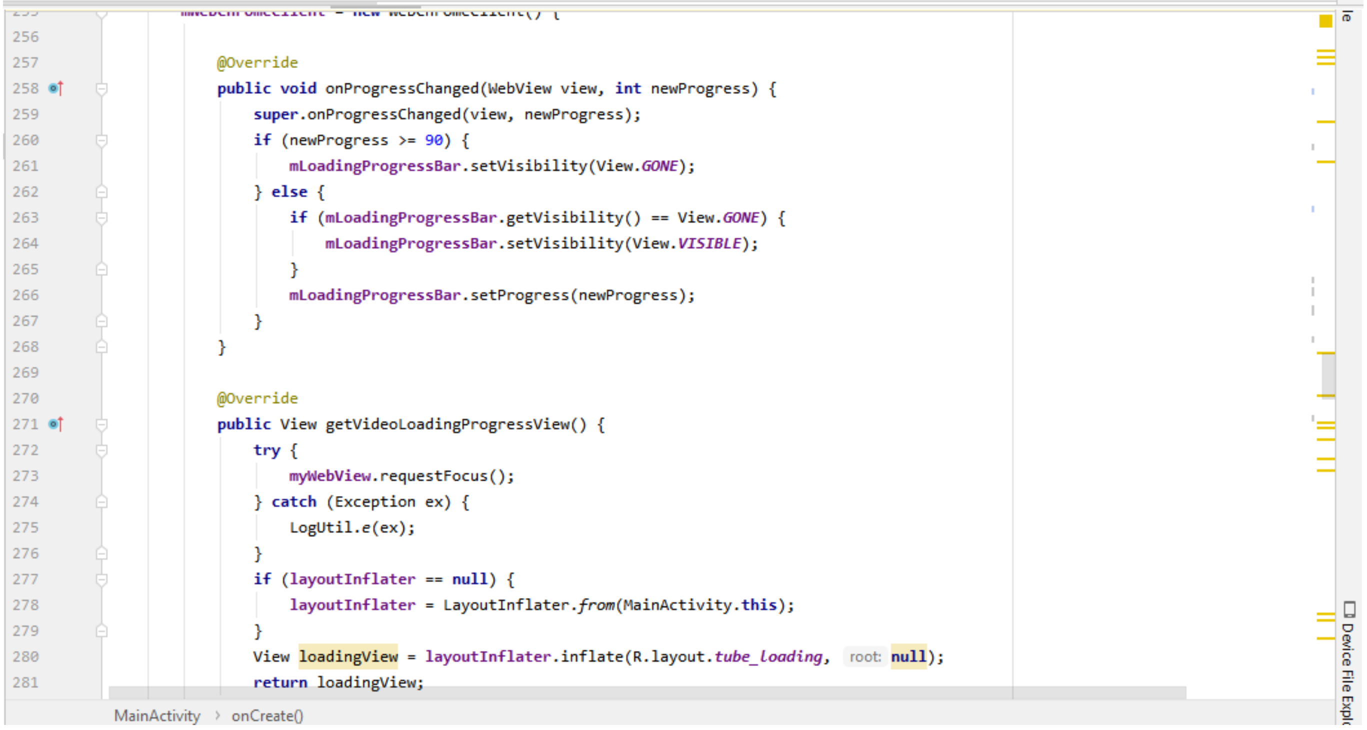


```
                    public void onClick(DialogInterface dialog, int which) {
                    }
                }).
                create();
            alertDialog.show();
        }
    }

    protected void showWaitDialog() {
        if (mProgressDialog == null) {
            mProgressDialog = ProgressDialog.show( context: this,  title: "Loading...",  message: "Please wait...",  indeterminate: true,  cancela
            mProgressDialog.setCanceledOnTouchOutside(false);
            mProgressDialog.setOnCancelListener(new ProgressDialog.OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog) {
                }
            });
        } else {
            mProgressDialog.show();
        }
    }

    private void dismissWaitDialog() {
        if (mProgressDialog != null) {
            mProgressDialog.dismiss();
        }
```

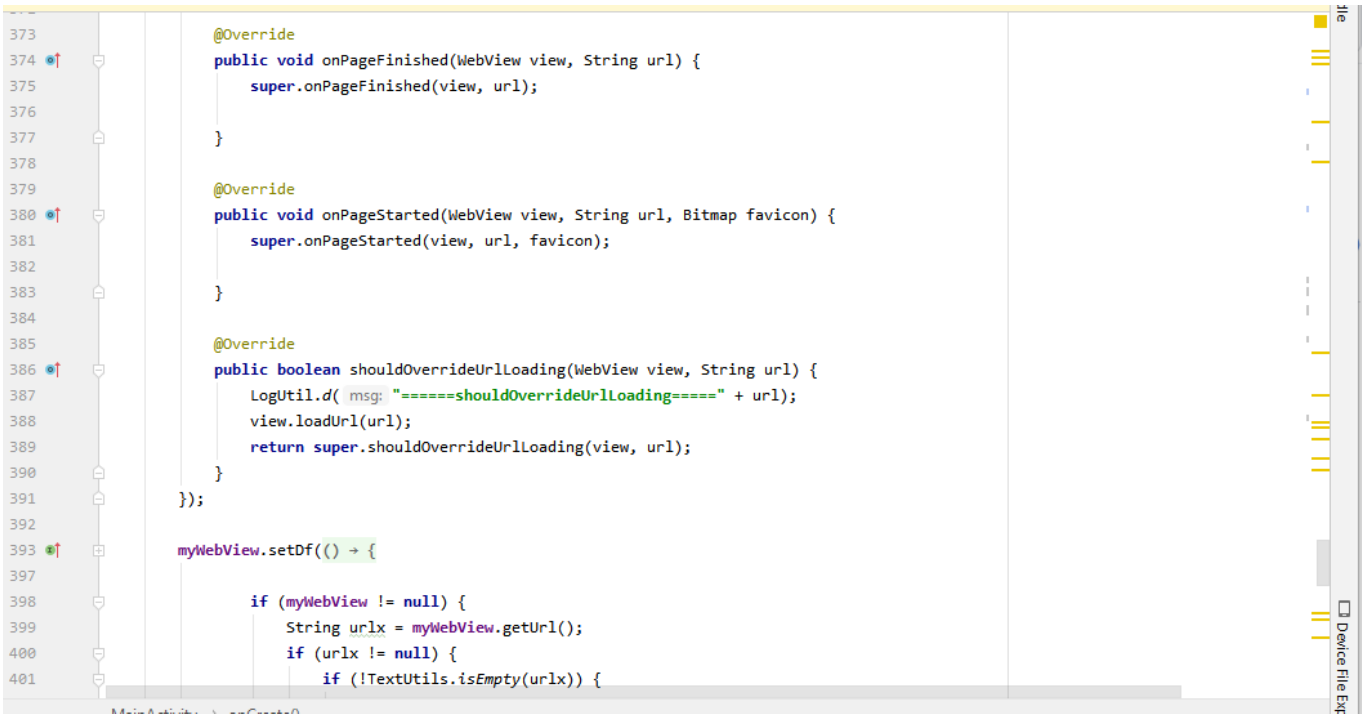
MainActivity › onCreate()



```
            @Override
            public void onProgressChanged(WebView view, int newProgress) {
                super.onProgressChanged(view, newProgress);
                if (newProgress >= 90) {
                    mLoadingProgressBar.setVisibility(View.GONE);
                } else {
                    if (mLoadingProgressBar.getVisibility() == View.GONE) {
                        mLoadingProgressBar.setVisibility(View.VISIBLE);
                    }
                    mLoadingProgressBar.setProgress(newProgress);
                }
            }

            @Override
            public View getVideoLoadingProgressView() {
                try {
                    myWebView.requestFocus();
                } catch (Exception ex) {
                    LogUtil.e(ex);
                }
                if (layoutInflater == null) {
                    layoutInflater = LayoutInflater.from(MainActivity.this);
                }
                View loadingView = layoutInflater.inflate(R.layout.tube_loading,  root: null);
                return loadingView;
```

MainActivity › onCreate()



```
            @Override
            public void onPageFinished(WebView view, String url) {
                super.onPageFinished(view, url);

            }

            @Override
            public void onPageStarted(WebView view, String url, Bitmap favicon) {
                super.onPageStarted(view, url, favicon);

            }

            @Override
            public boolean shouldOverrideUrlLoading(WebView view, String url) {
                LogUtil.d( msg: "======shouldOverrideUrlLoading=====" + url);
                view.loadUrl(url);
                return super.shouldOverrideUrlLoading(view, url);
            }
        });

        myWebView.setDf(() → {

                if (myWebView != null) {
                    String urlx = myWebView.getUrl();
                    if (urlx != null) {
                        if (!TextUtils.isEmpty(urlx)) {
```

References

- https://wwwyoutube.com/ code with harry - code with harry youtube page helped me alot in learning how to use android studio
- www.w3school.com
- www.tutorialspoint.com
- **Mentor . Mr. Piyush Vashisth Sir**