移动互联网技术及应用

大作业报告

题目： 仿 TikTok 的短视频程序的架构设计

姓名	班级	学号
肖璐	2020211311	2020211641

2023.6

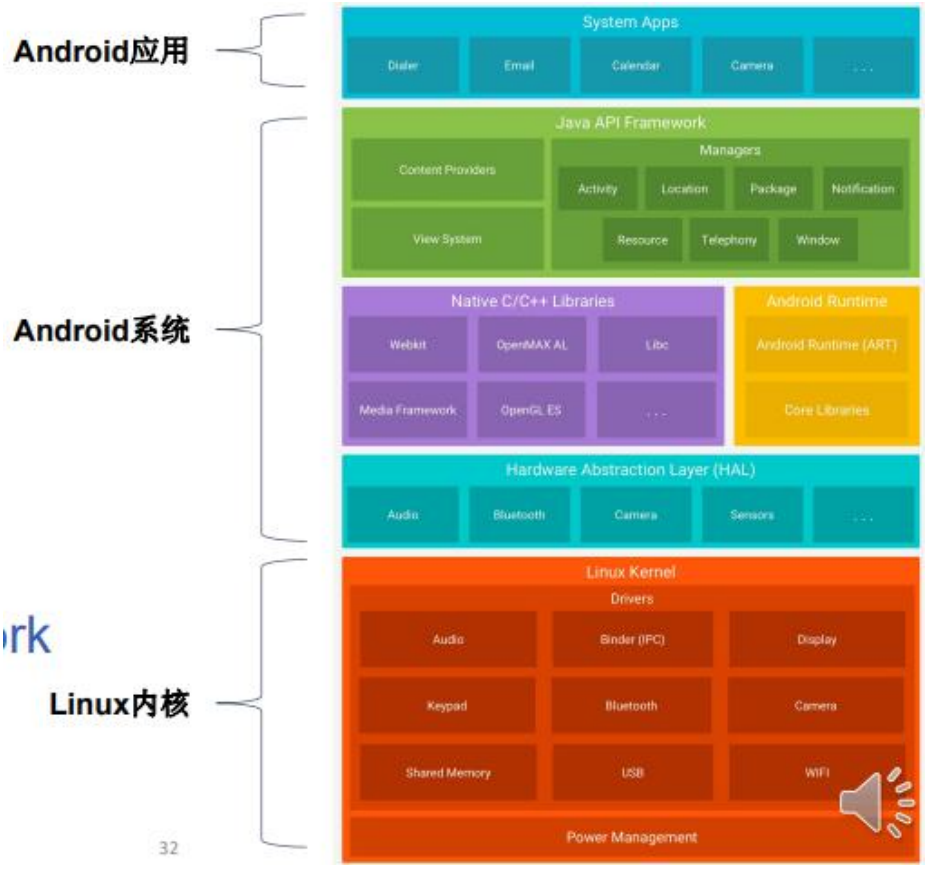
目录

1. 相关技术 2
2. 系统功能需求 5
3. 系统设计与实现 6
4. 系统可能的扩展 14
5. 总结体会 15

1. 相关技术

1.1. FrameWork、Native C/C++ Libraries、HAL、drivers 各层对 Android 提供功能分析

Android 系统结构如下图：



1.1.1. drivers

在 Android 系统的最底层是 Drivers 层，这里包含了具体的硬件驱动程序。Android 系统中的驱动程序通常是由硬件设备厂商来实现的，这些驱动程序会被编译成.so 文件，然后通过 HAL 层提供的接口来访问。

1.1.2. HAL 硬件抽象层

在 HAL 层，Android 系统会提供一系列的抽象接口，这些接口可以让不同的硬件设备厂商在不同的硬件平台上实现自己的 HAL 驱动程序。以传感器框架为例，Android 系统中的传感器功能是通过 HAL 层提供的接口来实现的，不同的硬件设备厂商可以通过实现这些接口来实现自己的传感器驱动程序。

1.1.3. Native C/C++ Libraries

在 Native C/C++ Libraries 层，Android 系统会提供不同的库文件，

这些库文件包含了对不同硬件设备的驱动程序、对不同功能的实现等。这些库文件在 Android 系统启动时会被加载到内存中，以便应用程序能够调用其中的函数。以音频框架为例，Android 系统中的音频功能是通过 libaudioflinger.so、libaudiopolicymanager.so 等库文件来实现的，这些库文件包含了音频数据的处理、音频输出的管理等功能。

1.1.4. Frameworks

在 Android 的 Framework 层中，NFC 框架、音频框架、传感器框架等功能是通过 Java API 的方式提供给应用程序开发者使用的。这些功能在 Framework 层中是通过调用 Native C/C++ Libraries、HAL、drivers 等底层组件来实现的。以 NFC 框架为例，Android 系统中的 NFC 功能是通过 NFCManagerService 提供的服务来实现的，同时 NFCManagerService 也会调用 Native C/C++ Libraries、HAL、drivers 等底层组件来实现 NFC 的数据交互、数据解析等功能。

1.2. 编程实例相关技术说明

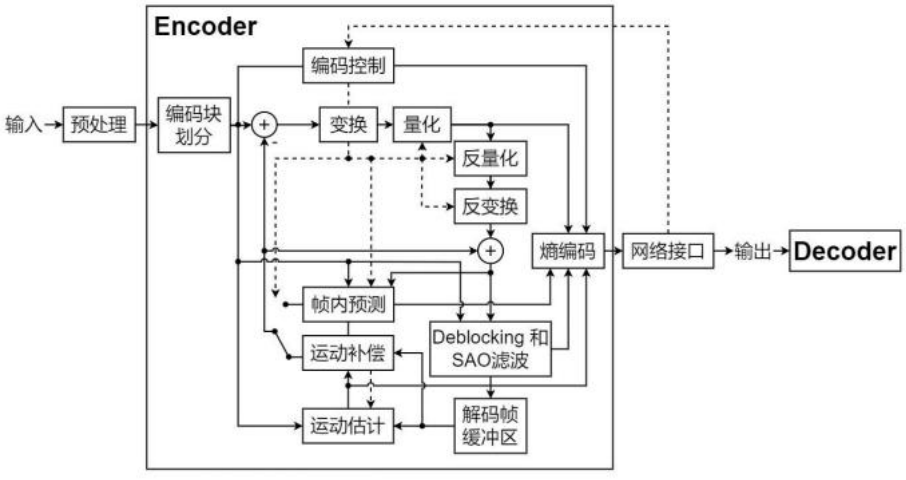
我对于同学开发的仿 TikTok 短视频 APP 进行架构分析，短视频播放器 APP 开发需要以下几项 Android 技术：

1.2.1. 短视频和图片的编码和解码技术

为了能够流畅播放各种格式的视频文件，需要掌握视频编解码技术。Android 平台支持的视频编码格式包括 H.264、VP8、VP9 等，需要了解这些编码格式的特点和优缺点。

在我们设计的短视频类中有七个属性：id、feedurl、nickname、description、likecount、avatar、thumbnails。其中 avatar 和 thumbnails 为 png 图片类型，feedurl 是 mp4 视频类型。我们采用的是本地访问视频，因此，为了减少空间的使用和其他资源的占用，需要将图片和视频进行编码压缩。对于图片我们常用离散余弦变换 DCT 技术根据图像信号在频率域的统计特性进行压缩图像数据。对于图片解码，我们采用图片加载库中的解码器和 Glide 代码进行图片解码。

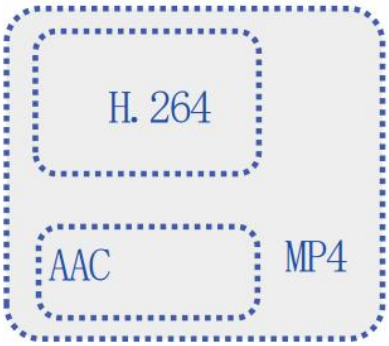
对于视频编码，下图是视频编码框架：



PC 和移动设备上都离不开视频播放，涉及的视频格式也比较多，不同扩展名实际上使用了不同的视频和音频编码，而我们的程序不一定会对所有的视频播放都提供支持，因此需要视频编码，通过特定的压缩技术，将某个视频内容数据转换成另一种特定格式文件（我们将视频都默认转换为 mp4 格式）。

1.2.2. 媒体处理技术

短视频播放器需要处理视频的播放、暂停、快进、后退、截图等操作，需要掌握 Android 提供的媒体处理技术，包括 MediaPlayer、VideoView、SurfaceView、TextureView 等。视频在程序中的封装格式如下：



我们对于视频播放器的设计采用 VideoView 技术，使用 start()函数进行视频播放，以及启动手势监听技术来进行调用 stopPlayBack()、pause()、resume()、seekTo()等函数，以及采用 isPlaying()、getDuration()、getCurrentPosition()等函数进行视频播放监听。

1.2.3. 网络通信技术

短视频播放器需要从网络中获取视频数据，需要掌握 Android 提供的网络通信技术，包括 HttpURLConnection、OkHttp、Retrofit 等。

短视频的载入采用的是 HttpURLConnection 技术，建立 URL 与 HTTP 进行连接，其中 HTTP 的 JSON 文件中存储着每个视频的属性和信息，我们通过读取 JSON 文件中的信息来获取视频信息，然后循环播放视频。

```
URL url;
HttpURLConnection httpsURLConnection = null;
try {
    //建立连接并发送请求
    //url = new URL("https://beiyou.bytedance.com/api/invoke/video/invoke/video");
    url = new URL("http://10.128.241.110:8080/myVideo.ini");
```

1.2.4. 异步处理技术

为了避免在主线程中进行耗时操作导致界面卡顿，需要掌握 Android 提供的异步处理技术，包括 AsyncTask、Handler、Thread、ThreadPoolExecutor 等。

1.2.5. UI 设计技术

短视频播放器的用户界面需要美观、易用，需要掌握 Android 提供的 UI 设计技术，包括 Layout、View、Animation、Fragment 等。

我们使用 ImageView 设置了爱心图标和双击“666”丰富了我们的 UI 设计。

2. 系统功能需求

2.1. 整体界面设计和控件布局设计

在 Android 中，可以使用 XML 文件来进行界面设计和控件布局设计。可以使用 Android Studio 等开发工具,通过可视化界面设计工具进行布局，也可以手动编写 XML 文件进行布局。在布局过程中，需要注意不同屏幕尺寸和分辨率的适配问题，保证用户在不同设备上的使用体验。

在我们的设想中，初始界面设计应该尽量简单，没有多余的 TextView 和 Button，只有一个 ViewGroup 栏，这个界面的 label 我们初始化设置为 douyin。

通过对移动互联网课程的学习，我们使用 Android Studio 中的 AndroidManifest.xml 进行界面设计。在 AndroidManifest.xml 中，我们注册编写布局 xml，创建 Activity 类。在 MainActivity.java 文件中，我们采用 setContentView()、setRequestedOrientation()、ViewPager2 等控件对界面进行处理。

2.2. 网络信息获取

在 Android 中，可以使用 Volley、OkHttp、Retrofit 等网络框架来进行网络信息获取。这些框架可以简化网络请求的代码编写，提高网络请求的效率和稳定性。

进行短视频 APP 的开发需要有一个稳定的视频获取库，我们读取这个库里的信息，将它存储在本地，然后采用 RecycleView 来进行信息处理和视频播放。

我们打算采用 URL 连接和 HttpURLConnection 来进行视频提取，我们先将所有视频文件的信息封装在一个 JSON 文件中，这个文件包含所有视频的所有属性信息，然后采用一个 GetJson.java 文件来读取信息。这个文件先与 Json 文件建立 URL 连接发送请求，然后使用 HttpURLConnection 对信息进行处理，然后使用数据流读取器 BufferedReader 来读取信息，最后将数据信息都存储在 JSONArray 中，这样成功获取网络信息。

但是这样存在一个问题就是，它是一次性读取所有文件，如果 Json 文件中的信息很多，那么这个 JsonArray 数组将开的特别大超出内存空间，因此我们的这个 APP 只能简单模拟有限视频的播放。

2.3. 视频信息流展示

可以使用 RecyclerView 来展示视频信息流，通过自定义 Adapter 和 ViewHolder 实现不同布局和交互效果。同时，可以使用 Glide 或者 Fresco 等图片加载框架来加载视频封面图和用户头像，提高用户体验。

视频展示需要循环播放，并且每个视频的左下方应该显示这个视频的相关信息。我们需要完成的是初始打开 APP 就自动播放视频，然后向下向上划动会播放别的视频，完成视频 videoView 的跳转。

我们打算使用 VideoInfo 类来存储需要展示的视频信息，将所有视频信息类都存储在 videoInfos 数组中，然后使用 ViewPagerAdapter 来完成 videoInfos 的数据处理和打印展示。但这会存在一个问题，由于是一次性的视频信息存储，因此当视频过多时，会出现 videoInfos 数组非常答，导致内存过载，因此这个 APP 只能模拟有限个视频的播放，不能完成真正的短视频播放器。

2.4. 视频播放

可以使用 ExoPlayer 或者 VideoView 等系统提供的视频播放组件进行视频播放。在使用 ExoPlayer 时，需要注意视频格式的兼容性问题，同时可以通过自定义 Renderers 实现更多的播放能力。

在上面的设计中，我们将所有视频信息都存放在了 JsonArray 中。我们打算采用提取视频 mp4 的连接然后使用 videoView.start()的方式进行初始视频的播放，循环播放视频我们使用 RecycleView。

2.5. 基本触控操作和手势监控

Android 提供了丰富的事件处理机制，可以通过重载 View 的 onTouchEvent 方法来实现基本触控操作，比如单击、长按、滑动等。同时，也可以通过 GestureDetector 或者 MotionEvent 等系统提供的类来实现更复杂的手势操作。

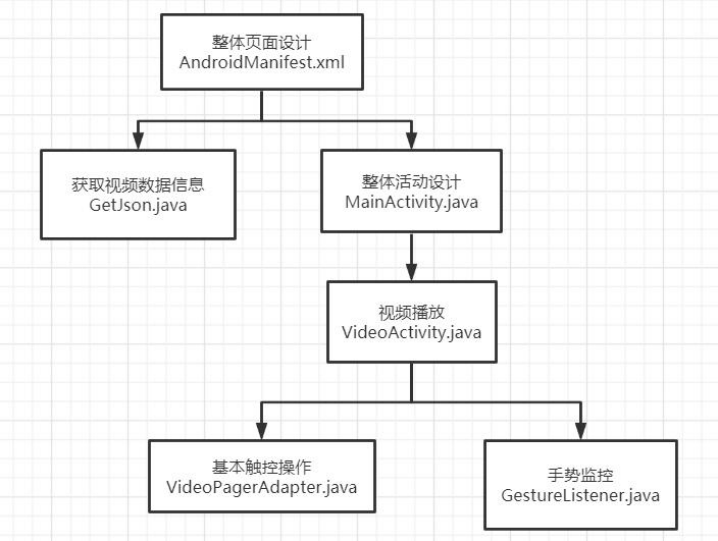
由于短视频 APP 需要识别手势，判断视频是否需要跳转以及是否出现点赞等情况，因此需要进行基本触控操作的定义和实现以及手势的监听。

2.6. 整体润色

Android 提供了丰富的 ImageView 可以对图像进行加工处理润色，我们也可以使用其他动画处理插件来对 APP 操作进行润色处理。

3. 系统设计与实现

3.1. 总体设计和系统组成



3.1.1. 文件说明

文件名	功能
AndroidManifest.xml	整体界面设计和 Activity 类型的定义
GetJson.java	创建 JSONArray 类，实现视频数据信息的读取与载入
MainActivity.java	对视频播放界面进行定于与控件布局；将 JSONArray 中的数据封装在 VideoInfo 结构中，并形成 VideoInfos 数组；采用 ViewPager2 对视频数据进行处理打印。
VideoActivity.java	对 APP 中的控件进行进一步细化，对 VideoView 进行布局；完成第一条视频的播放；完成基础触控操作的监控和调度，启动手势监听。
VideoPagerAdapter.java	定义 VideoInfo 结构类，采用 RecyclerView 对视频进行循环播放；完成视频信息的打印；
GestureListener.java	实现手势监控和相关的调度功能。

3.1.2. 结构类说明

- JSONArray
  存储视频所有信息。相关函数有：getJson()、getData()等。
- 自定义结构类——VideoInfo

```
class VideoInfo {//视频信息
    public String feedurl;
    public String nickname;
    public String description;
    public String likecount;
```

```
    public String avatar;

    public VideoInfo(String feedurl, String nickname, String
description, String likecount, String avatar){
        this.feedurl = feedurl;
        this.nickname = nickname;
        this.description = description;
        this.likecount = likecount;
        this.avatar = avatar;
    }
}
```

存放需要打印在屏幕上的视频信息。

3.2. 整体界面设计和控件布局设计

整体界面设计在 AndroidManifest.xml 文件中，规定了开始界面的 label 和 icon 等属性，并且规定了 Activity 类所在的文件名，实现了整个 APP 的框架设计，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.shortvideo">
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:usesCleartextTraffic="true"
        android:screenOrientation="portrait"
        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>
        <activity android:name=".VideoActivity" />
    </application>
```

```
</manifest>
```

在我们的设想中，初始界面设计应该尽量简单，没有多余的 TextView 和 Button，只有一个 ViewGroup 栏，这个界面的 label 我们初始化设置为 douyin。

通过对移动互联网课程的学习，我们使用 Android Studio 中的 AndroidManifest.xml 进行界面设计。在 AndroidManifest.xml 中，我们注册编写布局 xml，创建 Activity 类。在 MainActivity.java 文件中，我们采用 setContentView()、setRequestedOrientation()、ViewPager2 等控件对界面进行处理。具体分析如下：

我们会采用手势监控来完成用户对 APP 操作的控件设计：

```
gestureDetector = new GestureDetector(this, new GestureListener());
```

我们对短视频 APP 的设计中，要求视频以固定竖屏的方式播放，因此用 setRequestedOrientation()控件来设置：

```
setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
```

对于基本触控操作，我们也需要使用 ViewPager2 控件来进行监视和处理：

```
ViewPager2 vp = findViewById(R.id.vp);
vp.setOrientation(ViewPager2.ORIENTATION_VERTICAL);
ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(videoInfos,
this);
vp.setAdapter(viewPagerAdapter);
```

3.3. 网络信息获取

一开始，我们打算采用 URL 连接和 HttpURLConnection 来进行视频提取，我们先将所有视频文件的信息封装在一个 JSON 文件中，这个文件包含所有视频的所有属性信息，然后采用一个 GetJson.java 文件来读取信息。这个文件先与 Json 文件建立 URL 连接发送请求，然后使用 HttpURLConnection 对信息进行处理，然后使用数据流读取器 BufferedReader 来读取信息，最后将数据信息都存储在 JSONArray 中，这样成功获取网络信息。

但是这样存在一个问题就是，它是一次性读取所有文件，如果 Json 文件中的信息很多，那么这个 JsonArray 数组将开的特别大超出内存空间，因此我们的这个 APP 只能简单模拟有限视频的播放。

我们使用了以下代码来实现我们的功能：

```
URL url;
HttpURLConnection httpsURLConnection = null;
try {
//建立连接并发送请求
url = new URL("http://10.128.241.110:8080/myVideo.ini");
File file = new File("C:\\Users\\lxqqqqqqq\\Desktop\\video.json");
 url = file.toURI().toURL();
```

后来在实际操作过程中，由于本地的 json 文件始终无法正常读

取，读取的数据总是存在乱码现象，以及视频的编码解码存在一定问题，因此我们放弃使用 URL 连接和 HttpURLConnection 来进行视频提取。打算在本地代码上直接将视频信息手动输入，这样保证了视频读取的准确性以及避免了出现数组过大的情况。举例如下，我们将一条视频的七个属性全部初始化赋值到 JSONArray 数组中：

```
JSONArray jsonArray = new JSONArray();

JSONObject jsonObject1 = new JSONObject();
jsonObject1.put("_id", "111");
jsonObject1.put("feedurl", "http://10.128.241.110:8080/1.mp4");
jsonObject1.put("nickname", "南鸢");
jsonObject1.put("description", "这是第一条 Feed 数据");
jsonObject1.put("likecount", 10000);
jsonObject1.put("avatar", "http://10.128.241.110:8080/1.png");
jsonObject1.put("thumbnails", "http://10.128.241.110:8080/1.png");
jsonArray.put(jsonObject1);
```

3.4. 视频信息流展示

视频展示需要循环播放，并且每个视频的左下方应该显示这个视频的相关信息。我们需要完成的是初始打开 APP 就自动播放视频，然后向下向上划动会播放别的视频，完成视频 videoView 的跳转。

我们打算使用 VideoInfo 类来存储需要展示的视频信息，将所有视频信息类都存储在 videoInfos 数组中，然后使用 ViewPagerAdapter 来完成 videoInfos 的数据处理和打印展示。但这会存在一个问题，由于是一次性的视频信息存储，因此当视频过多时，会出现 videoInfos 数组非常答，导致内存过载，因此这个 APP 只能模拟有限个视频的播放，不能完成真正的短视频播放器。

在 ViewPagerAdapter.java 文件中，我们规定了需要展示的视频信息类，代码如下：

```
class VideoInfo {//视频信息
    public String feedurl;
    public String nickname;
    public String description;
    public String likecount;
    public String avatar;

    public VideoInfo(String feedurl, String nickname, String description,
String likecount, String avatar){
        this.feedurl = feedurl;
        this.nickname = nickname;
        this.description = description;
        this.likecount = likecount;
        this.avatar = avatar;
```

```
    }
}
```

我们需要输出 feedurl、nickname、description、likecount、avater 等信息，将每条视频的信息都封装在 VideoInfo 中，然后将这个 VideoInfo 节点添加到 VideoInfos 中，使用以下代码实现该功能：

```
List<VideoInfo> videoInfos = new ArrayList<VideoInfo>();
for(int i = 0; i < jsonArray.length(); i++){
    try {
        JSONObject temp = jsonArray.getJSONObject(i);

        String feedurl = temp.getString("feedurl");
        String nickname = temp.getString("nickname");
        String description = temp.getString("description");
        String likecount = temp.getString("likecount");
        String avatar = temp.getString("avatar");

        VideoInfo video = new VideoInfo(feedurl, nickname, description,
likecount, avatar);

        videoInfos.add(video);
    } catch (JSONException e) {
        e.printStackTrace();
    }
}
```

对于视频信息的打印，我们使用 ViewPagerAdapter 来实现，将 videoInfos 中的全部信息依次打印内容，代码实现如下：

```
ViewPager2 vp = findViewById(R.id.vp);
vp.setOrientation(ViewPager2.ORIENTATION_VERTICAL);
ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(videoInfos,
this);
vp.setAdapter(viewPagerAdapter);
```

3.5. 视频播放

在上面的设计中，我们将所有视频信息都存放在了 JsonArray 中。我们打算采用提取视频 mp4 的连接然后使用 videoView.start()的方式进行初始视频的播放，循环播放视频我们使用 RecycleView。

通过对 saveInstanceState 的状态查询来进行第一条视频的播放，实现代码如下，当 saveInstanceState 存在时，表明是刚打开 APP，于是进行第一条视频的播放：

```
if (savedInstanceState != null){
    Log.d("saved", "success");
```

```
    int sec1 = (int) savedInstanceState.getLong("time");
    Log.d("saved", String.valueOf(sec1));
    videoView.seekTo(sec1);
    boolean isStart1 = savedInstanceState.getBoolean("play");
    if (isStart1 == true) {
        videoView.start();
    }
}
else//修改！视频点开后自动播放
{
    videoView.start();
```

视频的循环播放我们使用 RecyclerView 来实现，实现代码如下：

```
class ViewPagerViewHolder extends RecyclerView.ViewHolder {
    TextView mTvTitle;
    RelativeLayout mContainer;
    ImageView mImageView;
    public ViewPagerViewHolder(@NonNull View itemView) {
        super(itemView);
        mContainer = itemView.findViewById(R.id.container);
        mTvTitle = itemView.findViewById(R.id.tvTitle);
        mImageView = itemView.findViewById(R.id.imageView);
    }
}
```

3.6. 基本触控操作和手势监听

Android 提供了丰富的事件处理机制，可以通过重载 View 的 onTouchEvent 方法来实现基本触控操作，比如单击、长按、滑动等。同时，也可以通过 GestureDetector 或者 MotionEvent 等系统提供的类来实现更复杂的手势操作。

由于短视频 APP 需要识别手势，判断视频是否需要跳转以及是否出现点赞等情况，因此需要进行基本触控操作的定义和实现以及手势的监听。

通过对基本触控操作的调度以及手势的监控，当向下滑动时，完成视频的转换；当双击屏幕时完成视频的点赞；当单击屏幕时，实现视频暂停；实现代码如下：

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    outState.putLong("time", sec);
    outState.putBoolean("play", isStart);
    Log.d("save", String.valueOf(sec));
    Log.d("save", String.valueOf(isStart));
    super.onSaveInstanceState(outState);
}
```

```
@Override
protected void onPause() {
    super.onPause();
    videoView = findViewById(R.id.videoView);
    sec = videoView.getCurrentPosition();
    isStart = videoView.isPlaying();
}

@Override
public boolean onTouchEvent(MotionEvent event) {
    return gestureDetector.onTouchEvent(event); //启动手势监听
}

@Override
public boolean onSingleTapConfirmed(MotionEvent e) //单击暂停
{
    if (VideoActivity.vactivity.videoView.isPlaying()){
        VideoActivity.vactivity.videoView.pause();
    }else {
        VideoActivity.vactivity.videoView.start();
    }
    return false;
}
```

3.7. 整体润色

Android 提供了丰富的 ImageView 可以对图像进行加工处理润色，我们也可以使用其他动画处理插件来对 APP 操作进行润色处理。

我们采用爱心图标来表示 likecount 点赞数目，对 UI 设计进行优化，实现代码如下：

public ImageView imageView; //此处作了修改，新增爱心图标

我们还设计了双击点赞“666”的动画，来优化整体设计，实现代码如下：

```
@Override
public boolean onDoubleTap(MotionEvent e) //双击 666
{
    Toast toast=Toast.makeText(VideoActivity.vactivity, "666",
Toast.LENGTH_SHORT);
    //显示 toast 信息
    toast.show();
    //设置弹出图片
```

```
VideoActivity.vactivity.imageView.setImageResource(R.drawable.love);
    VideoActivity.vactivity.imageView.setVisibility(View.VISIBLE);
    //设置弹出淡出效果
    VideoActivity.vactivity.imageView.setAlpha(1.0f);
    alphaAnimation = new AlphaAnimation(1.0f, 0.0f);
    alphaAnimation.setDuration(1000);
    VideoActivity.vactivity.imageView.setAnimation(alphaAnimation);
    alphaAnimation.start();
    VideoActivity.vactivity.imageView.setVisibility(View.INVISIBLE);

    //之前妄图采用 lottie 动画,但是 VideoView 和 Lottie 控件两个图层叠加会 crash,
参考其官方答疑区
    // https://github.com/airbnb/lottie-android/issues/1252
    //修改 json 文件后仍然没有能力让 lottie 控件淡入淡出与 VideoView 图层，因此放
弃 lottie，采用基本的 imageview
    return false;
}
```

4. 系统可能的扩展

4.1. 站在系统架构设计师的角度，阐述如何设计一个移动互连网终端的操作系统和实现自主可控进行技术

设计一个移动互连网终端的操作系统需要考虑到多个方面，包括系统的可靠性、安全性、易用性和可扩展性等。同时，要实现自主可控技术，需要采用一些特殊的技术手段，如加密算法、访问控制、审计等。设计一个移动互连网终端操作系统需要考虑到以下几个方面：

安全设计：操作系统的安全设计是非常重要的。需要采用先进的防护措施，如基于硬件的安全机制、安全启动和可信计算等。同时，需要加强用户身份认证、访问控制、数据加密等方面的保护，防止未经授权的访问和篡改。

可靠性设计：移动互连网终端的操作系统需要具备高可靠性，避免系统崩溃和数据丢失。可以采用多种技术手段来提高系统的可靠性，如备份、容错、故障转移等。

易用性设计：操作系统的易用性是用户体验的重要组成部分。需要设计简单直观的用户界面和操作流程，使用户能够轻松地完成各种任务，同时还要考虑到用户的个性化需求和习惯。

可扩展性设计：移动互连网终端的操作系统需要具备较高的可扩展性，以满足不断变化的业务需求和用户需求。可以采用模块化设计、插件机制等技术手段，以便于系统的扩展和升级。

自主可控技术：为了实现自主可控技术，需要采用一些特殊的技术

手段。例如，采用国产芯片和操作系统内核，采用自主研发的加密算法和安全协议，加强对系统的自主控制和管理等。

总之，设计一个移动互连网终端的操作系统需要综合考虑多个因素，以满足用户的需求和安全要求。在实现自主可控技术方面，需要采用一些特殊的技术手段，以确保系统的安全性和可控性。

4.2. 该短视频 APP 的系统拓展

4.2.1. 拍摄、编辑、发布视频功能

短视频 APP 的核心功能是视频录制和编辑，需要使用 Android 提供的 MediaRecorder 类和 Camera 类来实现视频录制功能。同时还需要掌握视频编辑技术，如视频剪辑、添加音乐、添加滤镜等，可以使用第三方库或自行开发。

短视频 APP 需要将录制好的视频进行压缩和上传，以便用户可以分享和观看。需要掌握视频压缩技术，如 H.264、H.265 等，以及视频上传技术，如 HTTP 协议、FTP 协议等。

4.2.2. 采用大数据技术进行视频推送

短视频 APP 需要具有社交分享和推荐功能，需要掌握社交分享 API，如微信、QQ 等，以及推荐算法技术，如基于内容的推荐、基于用户行为的推荐等。

4.2.3. 用户登录和数据管理技术

短视频 APP 需要用户登录、注册和数据管理功能，需要掌握 Android 提供的账号系统和数据库管理技术，如 SQLite 数据库、SharedPreference 等。

5. 总结体会

本次移动互联网大作业，我先自己进行了第一类应用系统设计实现的内容，从应用开发者的角度去尝试开发一个应用系统，首先尝试的就是备忘录 APP 的开发，在这次开发过程中，我实现了全部编程和演示，重新回顾了 ViewPaper 和多线程的知识点，但是我认为备忘录 APP 的开发过于简单，对于移动互联网课堂上的 NFC 框架开发、音频框架开发、传感器框架开发、SOCKET 传输和网络编程等知识点都没有涉及，因此我放弃了自己开发的备忘录 APP。我在学长学姐的代码上进行研究，发现我更擅长于从系统架构设计者的角度去分析一个软件，因此，我选择去分析短视频 APP 的架构设计，并亲手去修改了学长学姐的代码完成了我的设计。

移动互联网的课程，从纵向来说，从硬件的 drivers 层的设备驱动和硬件抽象，直接控制硬件设备的访问和操作；到 HAL 层，比起 drivers 更接近于软件，但同样也是一种抽象的硬件访问层，实现 Android 系统与不同的硬件进行交互，完成一些硬件抽象层的接口和实现；到 Native C/C++ Libraries 层，作为 Android

系统的底层库，提供 Android 底层的系统服务和 API，到用于开发系统级别应用程序的编程库。再到 FrameWork 层，成熟且最高层次的应用程序框架，提供各种 API 和服务来用于应用程序的开发。最后完成程序开发生成 APK 文件实现 AndroidAPP 的运行。从最底层到最高层，我们都进行了深入的学习，而移动互联网课程的最深入学习在于引导我们从开发者的角度去真实使用 Java 语言以及 Android 的 API 开发软件 APP。让我们作为学生去体会一个架构师和一个工程师的任务与责任。

移动互联网课程，从横向来说，使用 Android 四大组件：Activity、Service、BroadcastReceiver、ContentProvider 来实现 Android 程序的底层逻辑和架构，然后使用 UI 组件完成 Android 软件的整体实现。更深入的，ABI 接口实现软件在不同 CPU 之间的运行，实现软件兼容性，人工智能的深度学习与软件 APP 的结合能加深软件 APP 的数据挖掘深度，增加软件大数据的可用性等。

最后，结合移动互联网课程的三次实验，从第一次实验的 ViewPaper 时钟作业让我们对基础的 Android 知识有了初步认知并采用编程成功实现了基础概念；到第二次 musicplayer 的编程，让我们在最基础的页面设计上加入了音频设备来实现软件 APP；到第三次作业，从整个移动互联网课程的总体知识架构出发，设计分析一个成熟的 Android 软件；从浅至深，一步一步引导我们深入 Android 编程和移动互联网知识，我认为我们到了第六章之后才是真正的学习移动互联网，前五章我们只在本地以及页面上设计实现程序，第六章之后，引入线程和 SOCKET 网络通信，真正意义上的实现了互联网知识的学习和 APP 的开发。整个课程前五章是计算机类基础课程与 Android 程序开发的对接，第六章是学校基础知识与现实互联网开发的对接，实现了计算机学生的实操，个人认为这门课程对我的帮助教育意义远大于我学会了 Android 程序开发的意义。