移动互联网技术及应用大作业报告

姓名	班级	学号
程年智	2017211309	2017211452

1、系统需求

1.1 需求分析

- 视频流式列表显示
- 视频播放
- 从指定URL获取视频信息
- 使用RecyclerView显示视频列表
- 使用Glide加载封面图

1.2 扩展功能实现

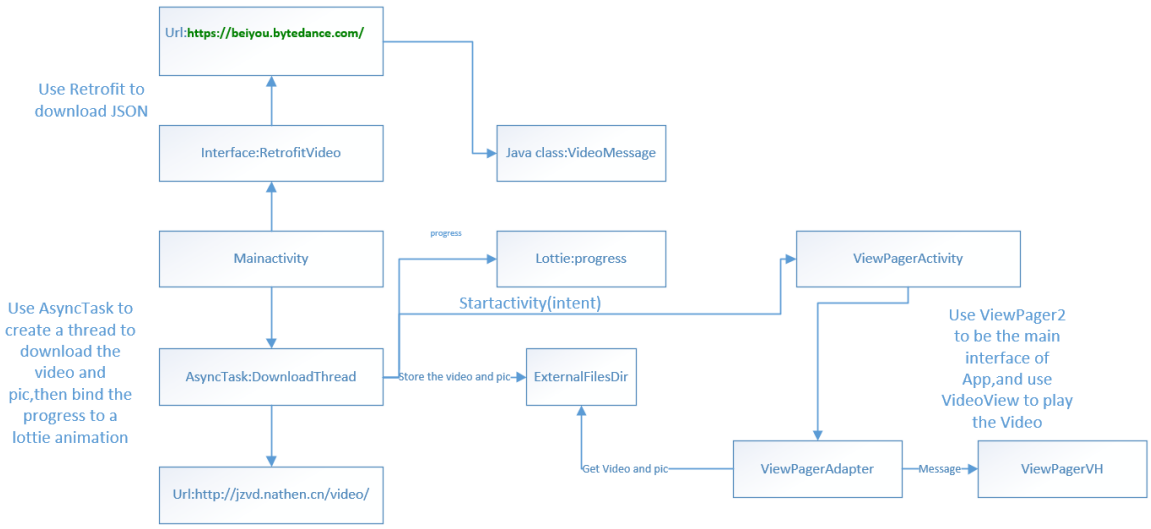
- 使用Viewpager2实现模仿抖音的界面。
- 每个视频可以显示作者信息，点赞数目等
- 双击点赞效果
- 加载页面和动画显示视频下载进度
- 无缝上下滑动切换视频
- 出色的按钮动画交互效果

2、App结构介绍

2.1 实现功能介绍和技术使用

- 使用retrofit从老师提供的URL的api中获取视频信息有关的JSON消息。
- 使用AsyncTask新建子线程，在子线程中下载视频和视频作者头像。
- 使用Lottie动画显示下载过程，在AsyncTask中调用方法在UI线程中设置进度。
- 使用ViewPager2进行列表显示，模仿抖音的上滑下滑切换视频效果。
- 使用VideoView播放视频。
- 使用MediaMetadataRetriever工具截取视频第一帧做视频封面，滑动的时候显示封面。
- 使用Glide图片库管理图片，Glide负责显示滑动时的下一视频的封面。
- 使用GestureDetector手势监听类，实现单击屏幕暂停，双击屏幕点赞的功能。
- 使用GitHubs上开源的库likebutton，点赞按钮实现点击爱心效果。
- 使用android studio自带的animator动画效果，实现分享评论等按钮的点击动态效果。
- 使用各种监听完成更多扩展需求。

2.2 模块图



2.3 流程说明

1. mainactivity首先通过Retrofit定义的接口RetrofitVideo中的GET方法访问URL，通过网站的API获取我们播放视频所需要的的JSON信息。
2. 通过Retrofit自带的功能将JSON信息转换成JAVA Bean对象，再生成列表，将该列表存入Intent中，以备切换activity时传送信息。
3. 运行自己写的AsyncTask线程，根据JSON中的下载地址分别下载各个视频的视频文件、作者头像图片到外部存储中。
4. 在AsyncTask线程中，使用独立的UI线程设置Lottie的动画进度。
5. 下载完成后，跳转到播放视频的ViewPagerActivity。
6. 将intent中Mainactivity传给Viewpageractivity的JSON信息通过构造函数的方式传给ViewPagerAdapter，Adapter解析这些信息创建视频播放画面，获取本地的视频地址并且播放。
7. 使用其它很多工具完善视频界面的功能。
8. OVER。

2.4 GitHub

https://github.com/DoChEnGzZ/TikTok

3、代码实现设计

3.1 Retrofit设计

3.1.1 Java Bean对象：VideoMessage

序列化方便后面放入到intent中，完全和JSON中的信息对应。

```
public class VideoMessage implements Serializable{

    @SerializedName("_id")
    private String userid;
    @SerializedName("feedurl")
    private String VideoUrl;
    @SerializedName("nickname")
    private String NickName;
    @SerializedName("description")
    private String DesCripTion;
    @SerializedName("likecount")
    private int LikeCount;
    @SerializedName("avatar")
    private String avatorUrl;
```

3.1.2 Retrofit 接口

定义Retrofit下载接口，指定get方法和URL地址，方法getvideo返回类型是List<VideoMessage>。

```java
public interface RetrofitVideo {
    @GET("api/invoke/video/invoke/video")
    Call<List<VideoMessage>> getvideo();
}
```

2.2.3 使用Retrofit获取JSON，并且转换成Bean对象

```java
        Retrofit retrofit=new Retrofit.Builder()
                .baseUrl("https://beiyou.bytedance.com/")
                .addConverterFactory(GsonConverterFactory.create())
                .build();
        RetrofitVideo retrofitVideo=retrofit.create(RetrofitVideo.class);
        retrofitVideo.getvideo().enqueue(new Callback<List<VideoMessage>>()
{
            @Override
            public void onResponse(Call<List<VideoMessage>> call,
Response<List<VideoMessage>> response) {
                //获取转换的Bean列表
                messageList=response.body();
            }

            @Override
            public void onFailure(Call<List<VideoMessage>> call, Throwable
t) {
            }
        });
```

3.2 AsyncTask 设计

新建AsyncTask类，重写其中的方法。

在新建的线程中根据下载地址下载内容。

```java
    private class DownLoadThread extends AsyncTask<String, Integer, String>
{

        @Override
        protected String doInBackground(String... strings) {
            /*
            doInBackground相当于thread的run(),在这里下载视频
            */
            return null;
        }
        @Override
        protected void onPreExecute() {
            /*
            preExecute是在执行run前执行的函数
            */
            super.onPreExecute();
        }
```

```
        @Override
        protected void onPostExecute(String s) {
            /*
            onPostExecute是在执行run后执行的函数
            */
            super.onPostExecute(s);
        }

        @Override
        protected void onProgressUpdate(Integer... values) {
            /*
            onProgressUpdate是在执行run时可以调用的UI线程，用来更新动画
            */
            super.onProgressUpdate(values);
        }
    }
```

3.3 Lottie设计

使用Lottie动画显示下载进度，动画资源下载自网络。

```
    <com.airbnb.lottie.LottieAnimationView
        android:id="@+id/progress"
        app:lottie_fileName="progress.json"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        android:layout_marginTop="330dp"
        android:layout_width="60pt"
        android:layout_height="60pt"
        />
```

在上面的AsyncTask线程中的UI线程中根据下载进度对动画进行更新，基本逻辑如下：

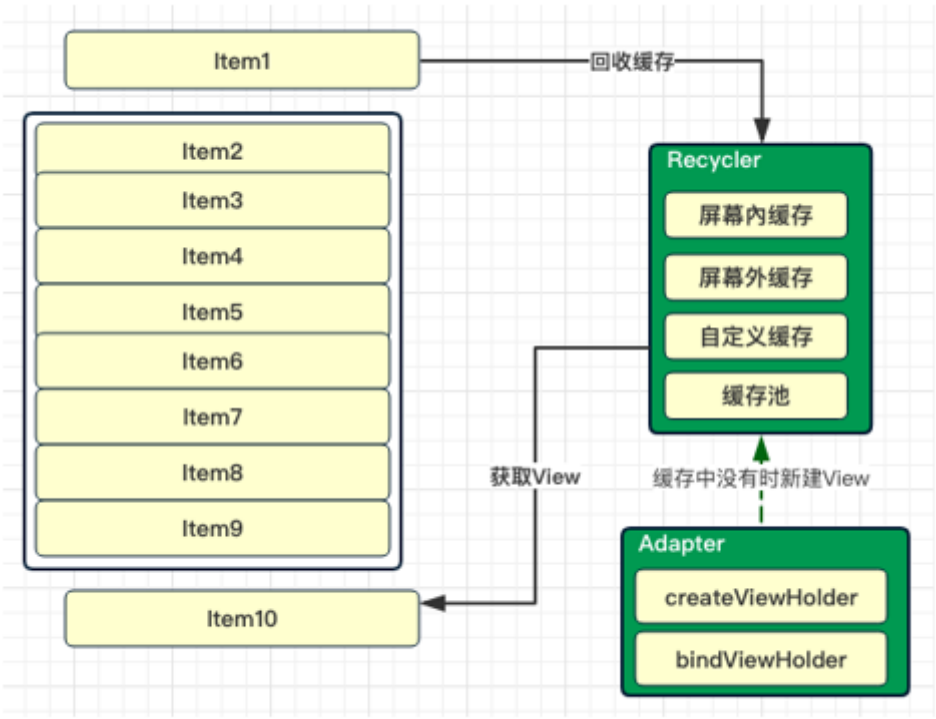
```
        @Override
        protected String doInBackground(String... strings) {
            int count_download=0;
            /*
            根据下载进度设置进度，使用publishprogress设置进度，然后会自动调用
onProgressUpdate
            */
            publishProgress(count_download*TimePiece);
            return null;
        }
        @Override
        protected void onProgressUpdate(Integer... values) {
            /*
            使用lottie的setProgress方法设置进度
            */
```

```
            lottie.setProgress(values[0]/100f);
            super.onProgressUpdate(values);
        }
```

3.4 ViewPager2的使用

ViewPager2类似RecycleView,需要设置一个Adapter和Viewholder，同时必须重写对应的方法。

在Adapter中重写OnCreateVh,OnbindVH,getItemCount,VH中需要建立和itemvie的对立关系。



> recyclerViewd的复用机制

3.4.1 ViewPagerAdapter

构造函数中使用参数List<VideoMessage>和context传入activity中的视频参数和上下文环境context。根据List.size()获取item的数量。

```
public class ViewPagerAdapter extends RecyclerView.Adapter<PagerViewHolder>
{

    public int PagerNums;
    List<VideoMessage> messageList;
    private Context context;

    public ViewPagerAdapter(Context context, List<VideoMessage>
messageList) {
        PagerNums=messageList.size();
        this.messageList=messageList;
        this.context=context;
    }

    @NonNull
    @Override
```

```
    public PagerViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        View
view=LayoutInflater.from(parent.getContext()).inflate(R.layout.pager_view,p
arent,false);
        return new PagerViewHolder(view);
    }

    @Override
    public void onBindViewHolder(@NonNull PagerViewHolder holder, int
position) {
            holder.Bind(messageList.get(position),context);
    }

    }

    @Override
    public int getItemCount() {
        return PagerNums;
    }
```

3.4.2 PagerViewHolder

PagerVH中的构造方法需要传入一个view，然后java代码中的每个view都需要通过findViewByid找到对应的view。

bind方法通过传入一个视频信息videomessage来初始化view中的信息。

```

public class PagerViewHolder extends RecyclerView.ViewHolder {
    public LikeButton likeButton;
    public ImageView share;
    public ImageView comment;
    public TextView sharenum;
    public TextView commentnum;
    public TextView likenum;
    public CircleImageView avator;
    public TextView nickname;
    public TextView description;
    public VideoView videoView;
    public ImageView pause;
    public ImageView f1pic;
    public Uri VideoPath;
    private Bitmap bitmap;

    private Uri AvatorPath;


    public PagerViewHolder(@NonNull View itemView) {
        super(itemView);
        likeButton = itemView.findViewById(R.id.likeBtn);
        share = itemView.findViewById(R.id.iv_share);
        comment = itemView.findViewById(R.id.iv_comment);
        sharenum = itemView.findViewById(R.id.tv_share);
        commentnum = itemView.findViewById(R.id.tv_comment);
        likenum = itemView.findViewById(R.id.tv_like);
        avator = itemView.findViewById(R.id.ci_avator);
```

```
        nickname = itemView.findViewById(R.id.tv_nickname);
        description = itemView.findViewById(R.id.tv_description);
        pause=itemView.findViewById(R.id.iv_pause);
        videoView=itemView.findViewById(R.id.vv);
        f1pic=itemView.findViewById(R.id.iv_1framepic);
    }

    public void Bind(final VideoMessage message, Context context) throws MalformedURLException {
        VideoPath=Uri.parse(message.getVideoUrl());
        likenum.setText(TransformLikenum(message.getLikeCount()));
        nickname.setText("@"+message.getNickName());
        description.setText(message.getDesCripTion());

VideoPath=Uri.parse("file:///"+context.getExternalFilesDir(null)+"/mp4/"+message.getVideoName());

 AvatorPath=Uri.parse("file:///"+context.getExternalFilesDir(null)+"/pic/"+message.getPicName());

}

```

3.4.3 MediaMetadataRetriever获取视频第一帧

```
  /*
  使用setDataSource设置视频地址
  */
retriever.setDataSource(context.getExternalFilesDir(null)+"/mp4/"+message.getVideoName());
  /*
  通过getFrameAtTime获取对应时间的第一帧，类型是bitmap
  */
bitmap=retriever.getFrameAtTime(0,MediaMetadataRetriever.OPTION_CLOSEST);
```

3.4.5 Glide图片库管理图片

Glide我使用的比较简单，第一项with是glide的生命周期，我选择和播放视频的videoview同周期，使用传进的activity的context一样的。第二项是载入的图片，这里是上面获取的视频第一帧做封面。第三项是载入图片的位置，实际上是和VideoView位置大小一致的一个imageview。

```
        Glide.with(videoView)
                .load(bitmap)
                .into(ImageView);
```

3.4.6 视频封面的工作原理：

视频在videoview播放，封面显示在同样位置大小的imageview中。在视频未加载好或者滑动没有完全到底时，设置videoview不可见，封面可见。等到视频加载完成，或者滑动到底，再设置封面不可见视频可见，这样就实现了简单的视频滑动的流畅性。但是有个问题，就是手机卡的时候，视频加载慢的时候会有明显的卡顿，不知道字节跳动官方是如何解决这个问题的。

下面是viewpager2自带的页面监听事件。

onPageSelected是页面切换完成，onPageScrollStateChanged是页面切换时的监听。

```
        viewPager2.registerOnPageChangeCallback(new
ViewPager2.OnPageChangeCallback() {
            @Override
            public void onPageSelected(int position) {
                /*
                页面切换完成，使视频可见
                */
                super.onPageSelected(position);
            }

            @Override
            public void onPageScrollStateChanged(int state) {
                super.onPageScrollStateChanged(state);
                switch (state) {
                    case ViewPager2.SCROLL_STATE_DRAGGING:
                        /*
                        页面开始切换，视频暂停。
                        */
                        break;
                    case ViewPager2.SCROLL_STATE_IDLE:
                        break;
                    case ViewPager2.SCROLL_STATE_SETTLING:
                        break;
                }
            }
        });
```

```
        videoView.setOnPreparedListener(new
MediaPlayer.OnPreparedListener() {
            @Override
            public void onPrepared(MediaPlayer mp) {
                mp.setOnInfoListener(new MediaPlayer.OnInfoListener() {
                    @Override
                    public boolean onInfo(MediaPlayer mp, int what, int
extra) {
                        /*
                        视频加载完成，使封面不可见
                        */
                        return true;
                    }
                });
            }
        });
```

3.4.7 likebutton的使用

偶然在github上找到的一个开源库，可以实现简单的点赞动画。

地址：https://github.com/jd-alexander/LikeButton

3.4.8 GestureDetector 手势监听

通过设置Gesture来监听手势，监听单机屏幕，双击屏幕的动作，并进行暂停视频和点赞/取消点赞。

首先通过videoview的setOnTouchListener监听触摸事件，将触摸事件传给GestureDetector。

```
videoView.setOnTouchListener(new View.OnTouchListener() {
    @Override
    public boolean onTouch(View v, MotionEvent event) {
        mgesture.onTouchEvent(event);
        return false;
    }
});
```

然后设置GestureDetector的监听事件

```
class GestureListner extends GestureDetector.SimpleOnGestureListener{



        public GestureListner() {
            super();
        }
        @Override
        public boolean onDoubleTap(MotionEvent e) {
            /*
            双击事件，设置点赞
            */
            return super.onDoubleTap(e);
        }

        @Override
        public boolean onSingleTapConfirmed(MotionEvent e) {
            /*
            单机事件，暂停视频播放，设置暂停图标可见
            */
    }
```

3.4.9 按钮的点击动画效果实现

```
        ObjectAnimator
scalex=ObjectAnimator.ofFloat(imageView,"scaleX",0.8f,1.2f);
        ObjectAnimator
scaley=ObjectAnimator.ofFloat(imageView,"scaleY",0.8f,1.2f);
        ObjectAnimator
scalex2=ObjectAnimator.ofFloat(imageView,"scaleX",1.2f,1f);
        ObjectAnimator
scaley2=ObjectAnimator.ofFloat(imageView,"scaleY",1.2f,1f);
        AnimatorSet animatorSet=new AnimatorSet();
        AnimatorSet animatorSet2=new AnimatorSet();
        animatorSet.playTogether(scalex,scaley);
        animatorSet2.playTogether(scalex2,scaley2);
        animatorSet.setDuration(200);
        animatorSet2.setDuration(200);
        animatorSet.start();
        animatorSet2.start();
```

点击分享评论等图标后，图标会有一个短暂变大的动画效果，提升手感。

4 感悟

感谢段鹏瑞老师和字节跳动的各位老师给了我一次学习我一直想学习的东西的机会，感谢我自己选了这门课。

可能这门课是我选了这么多选修课最忙的一门课也是收获最多的一门课。

我一直是个科技发烧友，从手机、电脑到耳机都是我的涉猎对象，曾经我一直想要学习android开发，但是苦于不会java加上懒一直没有机会，这次可以说是抓住了机会，学到了很多东西。

一开始段鹏瑞老师的手机网络课程，我学习到了GPRS,3g，4g，5g的协议和网络结构，可以说是学到了很多。

后面的android开发部分，一开始可以说是噩梦般的难度，不会java，环境配置一直失败，甚至一度有过想法退了这门课，但是我还是坚持了下来。事实说明，虽然很困难，但是学到的明显是等于你的付出的。

首先我初步掌握了JAVA这门语言，虽然没有系统的学习，但是我觉得JAVA在语法上和C++是有很多相同点的，区别在于java封装的更好，同时我也对面向对象的编程方法有了更深刻的理解，因为android开发的过程就是在不断的调用类和重写方法。

其次，感谢字节跳动的各位老师，每位老师都详细的讲解了每段代码的功能，不仅让我明白了如何实现app的每种功能，而且我学到了很多编程技巧，包括IDEA的经典IDE使用方法，github等等。

最大的收获是成功入门了安卓开发，学到了UI，线程，网络，存储，多媒体等等功能或者技巧的实现，让我对安卓系统有了更多的兴趣，对app如何工作有了更深刻的理解。

期末的大作业可以说是对自己一学期辛苦学习的完美考验，不仅复习了每节课的内容，而且为了完成自己目标内的拓展功能，我花了很多时间去学习，虽然很累但是都是值得的。

最后，感谢各位老师，在这次疫情下我们共同克服了网上授课的难题，谢谢你们！！！