E-Learning System 设计文档

Assignment 2

小组成员	
17301088	包天活
17301101	刘彬

一、 界面介绍

1) 登陆界面

实现了用户是否输入了账号、密码验证、以及邮箱格式、密码长度验证，以及使用 okhttp 连接 web server 服务器实现登录的账号密码验证，登录成功后跳转到课程列表界面。

```
private void login(){
    String user=et_username.getText().toString().trim();
    //判断是否输入了邮箱
    if(TextUtils.isEmpty(user)){
        Toast.makeText( context: this,"请输入邮箱",Toast.LENGTH_SHORT).show();
        return;
    }
    //判断邮箱格式是否正确
    if(!RegexUtil.isEmail(user)){
        Toast.makeText( context: this,"您输入的邮箱格式不正确",Toast.LENGTH_SHORT).show();
        return;
    }

    //判断是否输入了密码
    String pass=et_password.getText().toString().trim();
    if(TextUtils.isEmpty(pass)){
        Toast.makeText( context: this,"请输入密码",Toast.LENGTH_SHORT).show();
        return;
    }
    //判断密码长度
    if(pass.length()<6||pass.length()>15){
        Toast.makeText( context: this,"密码长度不正确",Toast.LENGTH_SHORT).show();
        return;
    }
```

```
private void login(){
    String user=et_username.getText().toString().trim();
    //判断是否输入了邮箱
    if(TextUtils.isEmpty(user)){
        Toast.makeText( context: this,"请输入邮箱",Toast.LENGTH_SHORT).show();
        return;
    }
    //判断邮箱格式是否正确
    if(!RegexUtil.isEmail(user)){
        Toast.makeText( context: this,"您输入的邮箱格式不正确",Toast.LENGTH_SHORT).show();
        return;
    }

    //判断是否输入了密码
    String pass=et_password.getText().toString().trim();
    if(TextUtils.isEmpty(pass)){
        Toast.makeText( context: this,"请输入密码",Toast.LENGTH_SHORT).show();
        return;
    }
    //判断密码长度
    if(pass.length()<6||pass.length()>15){
        Toast.makeText( context: this,"密码长度不正确",Toast.LENGTH_SHORT).show();
        return;
    }
```

2) 课程列表界面

使用 recyclerview 实现课程列表的展示，写了一个 item_view.xml 来定义 recyclerview 里每个 item 的版式，进入课程列表界面时通过 okhttp 实时获取课程列表信息并展示在界面上。

以下是课程列表界面所在 activity 里使用 recyclerview 适配器的代码：

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    spu = SharedPreferencesUtil.getInstance(getApplicationContext());

    recyclerView = (RecyclerView)findViewById(R.id.rv_course) ;
    //给recyclerview里的item设置布局，一行一个
    recyclerView.setLayoutManager(new LinearLayoutManager( context: MainActivity.this));

    et_search = (TextView)findViewById(R.id.et_search);

    getData();
}

private void getData(){
    Log.d(TAG,  msg: "getData: ");
    String address = "http://123.207.6.140:8080/getAllCourses";
    RequestBody requestBody = new FormBody.Builder().build();
    HttpUtil.sendOkHttpRequest(address, requestBody, new okhttp3.Callback() {
        @Override
        public void onFailure(Call call, IOException e) {

        }

        @Override
        public void onResponse(Call call, Response response) throws IOException {
            String responseData = response.body().string();
            showResponse(responseData);
        }
    });
}
```

以下为课程列表的适配器 recyclerviewAdapter，定义了一个 ViewHolder，来获取 item_view.xml 的组件，并给它赋值。

```java
public class RecyclerviewAdapter extends RecyclerView.Adapter<RecyclerviewAdapter.ViewHolder> {
    private static final String TAG = "RecyclerviewAdapter";
    private Context context;
    private List<Course> mCourseList;
    Course course;

    public RecyclerviewAdapter(Context context,List<Course> mCourseList){...}

    @Override
    public RecyclerviewAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                             int viewType) {
        // create a new view
        View v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_view, parent,  attachToRoot: false);
        // set the view's size, margins, paddings and layout parameters
        ViewHolder vh = new ViewHolder(v);
        return vh;
    }

    @Override
    public void onBindViewHolder(RecyclerviewAdapter.ViewHolder holder,final int position) {...}

    @Override
    public int getItemCount() { return mCourseList.size(); }

    public static class ViewHolder extends RecyclerView.ViewHolder {
        // each data item is just a string in this case
        public ImageView imageView;
        public TextView tvTitle,tvTime,tvContext;

        public ViewHolder(View v) {...}
    }
```

3) 点击课程列表的某一行可以查看该课程的详细信息同时课程详情里还包括了该课程的视频、图片，点击视频可以进行播放，

视频播放器选择的是 JCVideoPlayer

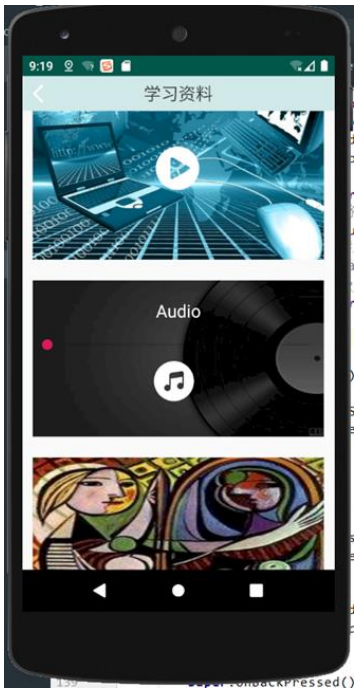
此处为加载视频和图片的关键代码，加载图片使用 Picasso，加载视频使用 JCVideoPlayer 的复写方法，同时还使用 Glide 来加载视频缩略图（因为 picasso 不能加载视频缩略图，而 Glide 可以实现）

```
            Picasso.get()
                    .load(image)
                    .into(iv_course);

            jcVideoPlayerStandard= (JCVideoPlayerStandard) findViewById(R.id.jiecao_Player);
            //添加视频缩略图
            ImageView thumbImageView = jcVideoPlayerStandard.thumbImageView;
            //使用Glide添加
            Glide.with( activity: CourseDetailsActivity.this).load(imageUrl).into(thumbImageView );
            //配置jiecaovideoplayer
            jcVideoPlayerStandard.setUp(s1,jcVideoPlayerStandard.SCREEN_LAYOUT_NORMAL, ...objects: "视频标题");
        });
    }
    @Override
    protected void onResume() {
        super.onResume();
        Sensor accelerometerSensor = sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sensorManager.registerListener(sensorEventListener, accelerometerSensor, SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(sensorEventListener);
        JCVideoPlayer.releaseAllVideos();
    }
    @Override
    public void onBackPressed() {
        if (JCVideoPlayer.backPress()){
            return;
        }
        super.onBackPressed();
    }
```

4) 查询功能及界面

用户输入自己想要查询的关键词，然后点击按钮跳转到课程材料界面，课程材料采用 recyclerview 展示，这里可以展示查询到的视频、

音频、图片，通过适配器里不同的 ViewHolder 来实现，点击视频和音频都可以实现跳转播放。

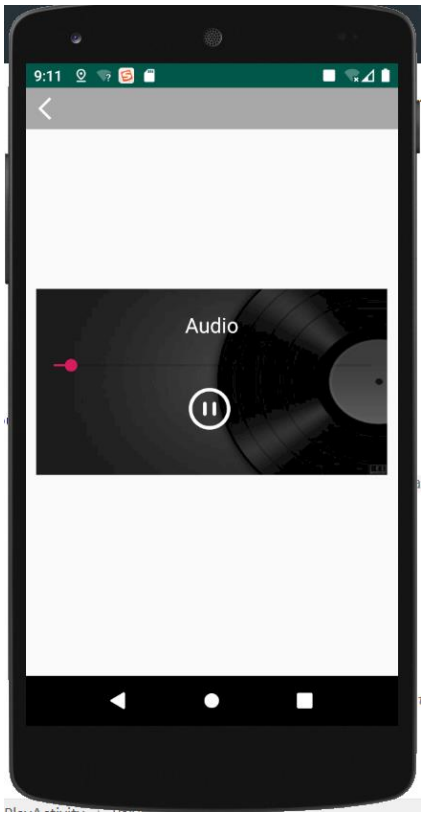


以下为适配器的关键代码，分别给视频、音频、图片写了三个 xml，在适配器里写了 3 个 ViewHolder，并定义了 type 来判断使用哪种类型的 ViewHolder。

```
public static class VideoHolder extends RecyclerView.ViewHolder {
    private ImageView imageView;
    public VideoHolder(View v) {
        super(v);
        imageView = (ImageView)v.findViewById(R.id.iv_video);
    }
}

public static class AudioHolder extends RecyclerView.ViewHolder {
    // each data item is just a string in this case
    public AudioHolder(View v) { super(v); }
}

public static class ImageHolder extends RecyclerView.ViewHolder {
    // each data item is just a string in this case
    public ImageView imageView;

    public ImageHolder(View v) {
        super(v);
        imageView = (ImageView)v.findViewById(R.id.iv_image);
    }
}
```

```java
@Override
public RecyclerView.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View v;

    if(viewType == TYPE_VIDEO){
        v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_video, parent,  attachToRoot: false);
        VideoHolder vh = new VideoHolder(v);
        return vh;
    }else if(viewType == TYPE_AUDIO){
        v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_audio, parent,  attachToRoot: false);
        AudioHolder vh = new AudioHolder(v);
        return vh;
    }else {
        v = LayoutInflater.from(parent.getContext())
                .inflate(R.layout.item_image, parent,  attachToRoot: false);
        ImageHolder vh = new ImageHolder(v);
        return vh;
    }
}
@Override
public void onBindViewHolder(final RecyclerView.ViewHolder holder, final int position) {
    String resource = "";
    if(holder instanceof VideoHolder){
        resource = mData.get(position).material;
    }else if(holder instanceof AudioHolder){
        resource = mData.get(position).material;
    }else{
        resource = mData.get(position).material;
        Picasso.get()
                .load(resource)
                .into(((ImageHolder)holder).imageView);
    }
```

以下为视频、音频的播放界面



二、 通过 HTTP 请求网络资源

本次作业要求课程信息等资源需要从网络上进行请求，所以

需要给项目配置相应权限，并采用相关技术。android 中规定耗时较长的操作不能放在主线程中，应该放在子线程中进行操作。此外，更改界面也放在 UI 线程中进行。

本小组研究后决定采用 OkHttp + Picasso + Glide 进行网络资源请求。

1) 基础配置

a) 要发送 http 请求，需要连接网络，需要获取 INTERNET 权限。在 AndroidManifest.xml 中添加以下：

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

b) 本服务器是简单搭建，采用 HTTP 协议，没有采用 HTTPS 协议进行数据传输。在高版本 android 中，不允许明文传输数据，所以 http 协议是被禁止的，需要使用 https 协议进行通信。如果想要使用 http 协议，则需要在“AndroidManifest.xml”配置

```
android:usesCleartextTraffic="true"
```

c) 如果添加以上配置还不能向服务器发送 HTTP 请求，则有可能是项目在配置前已经安装到虚拟机（或手机）上，配置之后，在此运行项目，并不会自动覆盖原来的配置文件，需要将该项目从虚拟机（或手机）上卸载，再次安装则可以成功解决。

2) OkHttp

a) 添加相应依赖项

在 build.gradle（Module:app）的 dependencies 中添加

```
implementation 'com.squareup.okhttp3:okhttp:3.10.0'
```

b) 此处通过使用 okhttp3 的 callback 来开子线程，将 okHttp 封装成两个静态方法，代码为：

```
public class HttpUtil {

    public static void sendOkHttpRequest(String address, okhttp3.Callback callback){
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url(address).build();
        client.newCall(request).enqueue(callback);
    }

    public static void sendOkHttpRequest(String address, RequestBody requestBody, okhttp3.Callback callback){
        OkHttpClient client = new OkHttpClient();
        Request request = new Request.Builder().url(address).post(requestBody).build();
        client.newCall(request).enqueue(callback);
    }
}
```

第一个方法用于发送不携带任何参数 HTTP 请求，若需要携带参数则使用第二个方法。

c) 调用实例

```
String address = "http://123.207.6.140:8080/getCourseById";
RequestBody requestBody = new FormBody
        .Builder()
        .add("id",course)
        .build();
HttpUtil.sendOkHttpRequest(address, requestBody, new okhttp3.Callback() {
    @Override
    public void onFailure(Call call, IOException e) {

    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
        String responseData = response.body().string();
        System.out.println(responseData);
        showResponse(responseData);
    }
});
```

```
private void showResponse(final String response){
    runOnUiThread(new Runnable() {
        @Override
        public void run() {
            JSONObject jsonObject = JSONObject.parseObject(response);

            String course = jsonObject.getString("name");
            String teacher = jsonObject.getString("teacher");
            String school = jsonObject.getString("school");
            String time = jsonObject.getString("time");
            String info = jsonObject.getString("info");
            String image = jsonObject.getString("imageUrl");

            tv_course_id.setText(course);
            tv_teacher.setText(teacher);
            tv_school.setText(school);
            tv_date.setText(time);
            tv_description.setText(info);
            Picasso.get()
                    .load(image)
                    .into(iv_course);

            jcVideoPlayerStandard= (JCVideoPlayerStandard) findViewById(R.id.jiecao_Player);
            //添加视频缩略图
            ImageView thumbImageView = jcVideoPlayerStandard.thumbImageView;
            //使用Glide添加
            Glide.with(CourseDetailsActivity.this).load(imageUrl).into(thumbImageView );
            //配置jiecaovideoplayer
            jcVideoPlayerStandard.setUp(s1,jcVideoPlayerStandard.SCREEN_LAYOUT_NORMAL,"视频标题");
        }
```

3) Picasso

picasso 是专门用来加载图片所用的框架。

a) 添加依赖项

```
implementation 'com.squareup.picasso:picasso:2.71828'
```

b) 调用实例：

```
resource = mData.get(position).material;
Picasso.get()
        .load(resource)
        .into(((ImageHolder)holder).imageView);
```

c) 遇到的有趣的问题

在写服务器时，使用 IDEA 直接在本机运行 spring boot，

测试时使用的 ip 是本机“127.0.0.1”，直接通过浏览器访问获取图片是完全可行的，但是当在 android 项目里面直接访问本机的服务时，没有反应，当我将 server 项目部署到服务器上时，ip 为“123.207.6.***”，就可以成功获取了。

4) Glide

a) 添加依赖项

```
implementation 'fm.jiecao:jiecaovideoplayer:5.5'
implementation 'com.github.bumptech.glide:glide:3.7.0'
```

b) 调用实例

```
//使用Glide添加
Glide.with( activity: this).load(imageUrl).into(thumbImageView );
```

三、 Server 简介

由于本次作业要求从网络获取课程信息、图片、视频和音频等数据，所以本小组成员根据项目需求简单搭建了一个服务器，现在就服务器进行简单介绍：

- 编程语言：JAVA、SQL
- 服务器框架：spring boot
- 工具：IDEA Ultimate 2019.2.3
- 数据库：MYSQL 8.0.16
- 接口文档:（此处简单介绍，详情见附件“服务器接口文档.txt”）

四、 Broadcast 技术

本项目中，采用了 Broadcast 技术和 Notifications 技术进行课

程信息广告。逻辑为：用户在进入登录界面时，APP 后台会新建一个线程，线程中向服务器请求一条课程信息作为 Notifications 中的课程广告。请求完成后，通过本地广播管理器发送 action 为："com.example.broadcast.LESSON_ADVERTISEMENT"的广播，并将请求得到的课程的数据一并发送。

当本地广播接收器接收到了相应 action 的广播后，开始构建一条 Notifications，并推送。

1) 定义广播接收器（继承 BroadcastReceiver 类， 实现 onReceive 方法）。接收到广播后具体的操作在 onReceive 方法中进行。

```
class LocalReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent){
        String data = intent.getStringExtra( name: "data");
        Log.d(TAG, msg: "onReceive: 接收广播");
        try {
            JSONObject jsonObject = new JSONObject(data);

            String courseName = jsonObject.getString( name: "name");
            String courseTeacher = jsonObject.getString( name: "teacher");
            String courseSchool = jsonObject.getString( name: "school");
            String text = courseSchool + "新开课程: " + courseName + ", 专业名师 " + courseTeacher
                    + " 领衔主讲。";
            Log.d(TAG, msg: "onReceive: text:" + text);
            Notification notification = new NotificationCompat.Builder(context, channelId: "subscribe")
                    .setContentTitle("E-Learning 课程提醒")
                    .setContentText(text)
                    .setWhen(System.currentTimeMillis())
                    .setSmallIcon(R.drawable.audio)
                    .setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.audio))
                    .build();
            NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
            manager.notify( id: 1, notification);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

2) 注册广播接收器

本处拦截自定义 action 为：

"com.example.broadcast.LESSON_ADVERTISEMENT"的广播

```
localBroadcastManager = LocalBroadcastManager.getInstance(this);

intentFilter = new IntentFilter();
intentFilter.addAction("com.example.broadcast.LESSON_ADVERTISEMENT");
localReceiver = new LocalReceiver();
localBroadcastManager.registerReceiver(localReceiver, intentFilter);
```

3) 发送广播

```
Intent intent = new Intent( action: "com.example.broadcast.LESSON_ADVERTISEMENT");
intent.putExtra( name: "data", responseData);
localBroadcastManager.sendBroadcast(intent);
```

4) 注销广播接收器

本项目中为动态注册广播接收器，需要手动注销。

```
@Override
protected void onDestroy(){
    super.onDestroy();
    localBroadcastManager.unregisterReceiver(localReceiver);
}
```

五、 Notification 技术

1) 创建通知通道

封装方法 createNotificationChannel()

- channelId 通道的 id，在通知在发送时需要通过 id 指定通道;
- channelName 通道的名字;
- importance 重要等级。

```
@RequiresApi(api = Build.VERSION_CODES.O)
private void createNotificationChannel(String channelId, String channelName, int importance) {
    NotificationChannel channel = new NotificationChannel(channelId, channelName, importance);
    NotificationManager notificationManager = (NotificationManager) getSystemService(
            NOTIFICATION_SERVICE);
    notificationManager.createNotificationChannel(channel);
}
```

2) 发送通知

```
JSONObject jsonObject = new JSONObject(data);

String courseName = jsonObject.getString( name: "name");
String courseTeacher = jsonObject.getString( name: "teacher");
String courseSchool = jsonObject.getString( name: "school");
String text = courseSchool + "新开课程：" + courseName + ", 专业名师 " + courseTeacher
        + " 领衔主讲。";
Log.d(TAG, msg: "onReceive: text:" + text);
Notification notification = new NotificationCompat.Builder(context, channelId: "subscribe")
        .setContentTitle("E-Learning 课程提醒")
        .setContentText(text)
        .setWhen(System.currentTimeMillis())
        .setSmallIcon(R.drawable.audio)
        .setLargeIcon(BitmapFactory.decodeResource(getResources(), R.drawable.audio))
        .build();
NotificationManager manager = (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
manager.notify( id: 1, notification);
```