E-Learning System 设计文档

Assignment 4

小组成员	
17301088	包天活
17301101	刘彬

一、 MVVM

1、 MVVM 简介

MVVM 是 Model-View-ViewModel 的简写。它本质上就是 MVC 的改进版。MVVM 就是将其中的 View 的状态和行为抽象化，让我们将视图 UI 和业务逻辑分开。

经过讨论，本小组选择将项目课程详情页面通过使用 MVVM 开发模式进行重构。

2、 添加依赖及相关配置

```
implementation 'me.tatarka.bindingcollectionadapter2:bindingcollectionadapter:2.2.0'
```

```
dataBinding{
    enabled true
}
```

3、 编写 ViewHolder

ViewModel 不涉及任何的视图操作，只进行业务逻辑的处理。通过官方提供的 Data Binding 库，当 ViewModel 中的数据发生变化时，UI 将自动更新。

```
public class CourseViewHolder {
    private static final String TAG = "CourseViewHolder";
    private String courseId;

    public final ObservableField<String> id = new ObservableField<>();
    public final ObservableField<String> name = new ObservableField<>();
    public final ObservableField<String> teacher = new ObservableField<>();
    public final ObservableField<String> school = new ObservableField<>();
    public final ObservableField<String> time = new ObservableField<>();
    public final ObservableField<String> info = new ObservableField<>();

    public CourseViewHolder(String courseId){...}

    public void queryCourse(){
        String address = "http://123.207.6.140:8080/getCourseById";
        RequestBody requestBody = new FormBody.Builder()
                .add( name: "id", courseId)
                .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 {...}
        });
    }
}
```

4、 编写 View(XML 布局文件)

View 不涉及任何的业务逻辑处理，只进行界面的显示。在 xml 布局文件中，通过官方提供的 Data Binding 库，将 UI 与 ViewModel 中的数据进行绑定，当 ViewModel 中的数据发生变化时，UI 将自动更新。xml 布局文件的代码如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <data>

        <import type="android.view.View" />

        <variable
            name="viewModel"
            type="com.example.studycourse.vm.CourseViewHolder" />
    </data>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        tools:context=".activity.CourseDetailsActivity">

        <LinearLayout...>

        <ScrollView...>

    </LinearLayout>
</layout>
```

5、 编写 Activity 类

```
public class CourseDetailsActivity extends AppCompatActivity {

    private CourseViewHolder viewHolder;
    private ActivityCourseDetailsBinding binding;

    private SensorManager sensorManager;
    private JCVideoPlayer.JCAutoFullscreenListener sensorEventListener;
    private JCVideoPlayerStandard jcVideoPlayerStandard;

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}


    @Override
    protected void onResume() {...}

    @Override
    protected void onPause() {...}
    @Override
    public void onBackPressed() {...}

    public void onReturnClick(View view) {...}
}
```

二、 Room API for Database access

1、 Room API 简介

Room 提供了 SQLite 之上的一层抽象，既允许流畅地访问数据库，也充分利用了 SQLite。处理大量结构化数据的应用，能从在本地持久化的数据中极大受益。

本小组在使用 MVVM 重构了的课程详情的基础上，使用 Room API 进行课程数据的持久化存储和提取。

2、 添加依赖

```
// Room components
implementation "android.arch.persistence.room:runtime:2.2.0"
annotationProcessor "android.arch.persistence.room:compiler:2.2.0"
androidTestImplementation "android.arch.persistence.room:testing:2.2.0"

// Lifecycle components
implementation "android.arch.lifecycle:extensions:2.2.0-beta01"
annotationProcessor "android.arch.lifecycle:compiler:2.2.0-beta01"
```

3、 创建 Entity

```java
@Entity(tableName = "course")
public class Course {
    @PrimaryKey
    @NonNull
    private String courseID;

    private int imageId;
    private String courseName;
    private String teacher;
    private String school;
    private String courseDate;
    private String courseDescription;
    private String imageURL;

    public Course(@NonNull String courseID, int imageId, String courseName, String teacher, String school,
        this.courseID = courseID;
        this.imageId = imageId;
        this.courseName = courseName;
        this.teacher = teacher;
        this.school = school;
        this.courseDate = courseDate;
        this.courseDescription = courseDescription;
        this.imageURL = imageURL;
    }
```

4、 创建 DAO

```java
@Dao
public interface CourseDao {
    @Insert
    void insert(Course course);

    @Query("SELECT * FROM course")
    LiveData<List<Course>> getCourseAll();

    @Query("SELECT * FROM course")
    List<Course> findAllCourses();

    @Query("SELECT * FROM course WHERE courseID = :courseId")
    Course findCourseById(String courseId);
}
```

5、 添加 Room 数据库

```
@Database(entities = {Course.class}, version = 1, exportSchema = false)
public abstract class CourseRoomDatabase extends RoomDatabase {

    private static volatile CourseRoomDatabase INSTANCE;

    public static CourseRoomDatabase getDatabase(Context context){
        if(INSTANCE == null){
            synchronized (CourseRoomDatabase.class){
                if(INSTANCE == null){
                    INSTANCE = Room.databaseBuilder(context,
                            CourseRoomDatabase.class,  name: "elearning").build();
                }
            }
        }
        return INSTANCE;
    }

    @NonNull
    @Override
    protected SupportSQLiteOpenHelper createOpenHelper(DatabaseConfiguration config) {
        return null;
    }

    @Override
    public void clearAllTables() {

    }

    public abstract CourseDao courseDao();
}
```

6、 加载课程列表时将数据存入数据库中

```
JSONArray jsonArray = JSONArray.parseArray(response);
if(jsonArray.size()>0){
    for(int i=0;i<jsonArray.size();i++){
        JSONObject json = jsonArray.getJSONObject(i);
        String courseID = json.getString( key: "id");
        String name = json.getString( key: "name");
        String teacher = json.getString( key: "teacher");
        String school = json.getString( key: "school");
        String imageURL = json.getString( key: "imageUrl");
        Course course = new Course(courseID,name,teacher,school,imageURL);
        courseList.add(course);
    }
}


new Thread(new Runnable() {
    @Override
    public void run() {
        List<Course> courses = courseDao.findAllCourses();
        if(courses.isEmpty()){
            for(Course course:courseList){
                courseDao.insert(course);
            }
        }
    }
}).start();
```

7、 进入课程详情页，通过访问数据库获取相关资料

```
public void queryCourse(){
    new Thread(new Runnable() {
        @Override
        public void run() {
            CourseDao courseDao = database.courseDao();
            Course course = courseDao.findCourseById(courseId);

            id.set(courseId);
            name.set(course.getcourseName());
            teacher.set(course.getTeacher());
            school.set(course.getSchool());
            time.set(course.getCourseDate());
            info.set(course.getCourseDescription());
        }
    }).start();
}
```

三、 实现第三方登录与分享

由于微信开发者账号审核太慢，只做了 QQ 登录

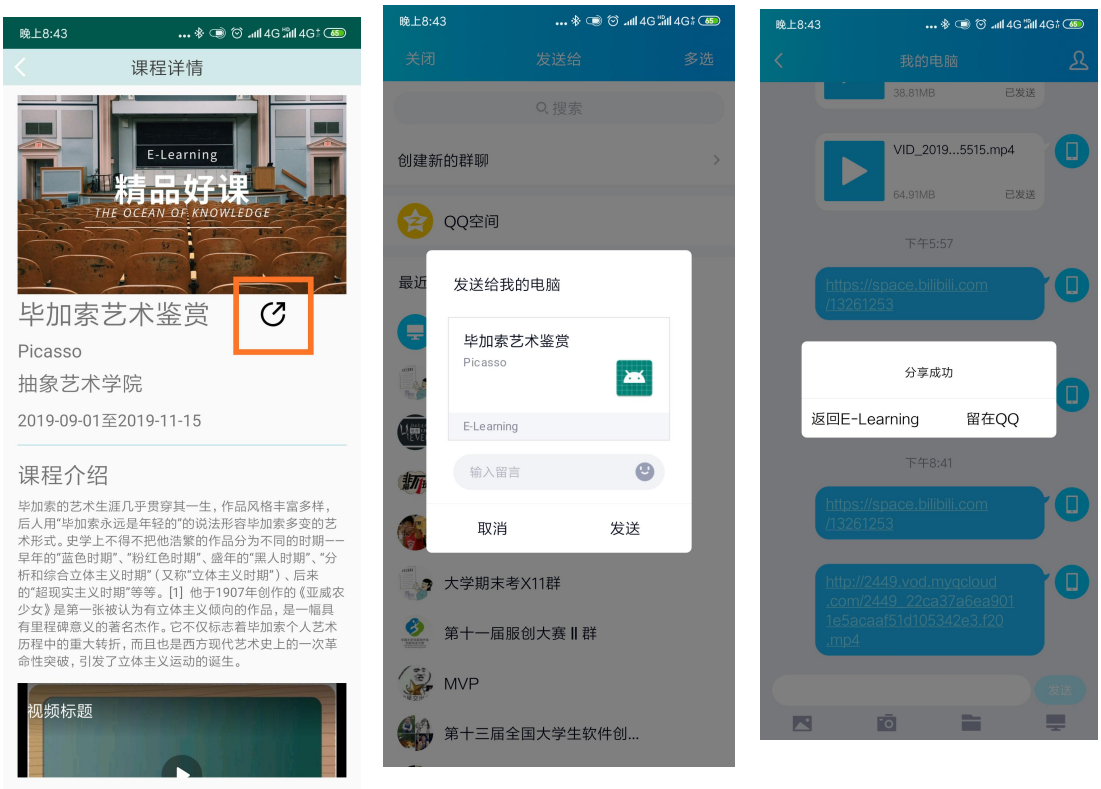
1. 登录的实现，利用腾讯提供的 API，新建一个 Tencent 获取参数，然后在新建 IUilistener 示例，点击跳转，获取到 QQ 用户昵称、QQ 号、头像等信息实现登录，登录后跳转到课程列表。

登录的具体代码如下：

```
mTencent = Tencent.createInstance( s: "101840549", context: LoginActivity.this);
View.OnClickListener qqClickListener = (view)->{
    qq_login_listener = new IUiListener() {
        @Override
        public void onComplete(Object o) {
            org.json.JSONObject jsonObject = (org.json.JSONObject) o;
            String openid = jsonObject.optString( name: "openid");
            String token,expires_in;
            try{
                token = jsonObject.getString( name: "access_token");
                expires_in = jsonObject.getString( name: "expires_in");
            } catch (JSONException e) {
                e.printStackTrace();
                Toast.makeText( context: LoginActivity.this, text: "网络错误",Toast.LENGTH_SHORT).show();
                return;
            }

            QQToken qqToken = mTencent.getQQToken();
            mTencent.setOpenId(openid);
            mTencent.setAccessToken(token,expires_in);
            UserInfo userinfo = new UserInfo( context: LoginActivity.this,qqToken);
            userinfo.getUserInfo(new IUiListener() {
                @Override
                public void onComplete(Object o) {
                    org.json.JSONObject json = (org.json.JSONObject) o;
                    String nickname = json.optString( name: "nickname");
                    String figureurl = json.optString( name: "figureurl_qq_2");
                    spu.setLogin(true);
```

2. 分享到第三方，分享按钮设计在了课程详情界面，点击分享，会将这节课的课程名称。讲课教师以及课程附带的视频链接分享给好友



分享功能实现代码如下

```
mTencent = Tencent.createInstance( s: "101840549", context: LoginActivity.this);
View.OnClickListener qqClickListener = (view)->{
    qq_login_listener = new IUiListener() {
        @Override
        public void onComplete(Object o) {
            org.json.JSONObject jsonObject = (org.json.JSONObject) o;
            String openid = jsonObject.optString( name: "openid");
            String token,expires_in;
            try{
                token = jsonObject.getString( name: "access_token");
                expires_in = jsonObject.getString( name: "expires_in");
            } catch (JSONException e) {
                e.printStackTrace();
                Toast.makeText( context: LoginActivity.this, text: "网络错误",Toast.LENGTH_SHORT).show();
                return;
            }

            QQToken qqToken = mTencent.getQQToken();
            mTencent.setOpenId(openid);
            mTencent.setAccessToken(token,expires_in);
            UserInfo userinfo = new UserInfo( context: LoginActivity.this,qqToken);
            userinfo.getUserInfo(new IUiListener() {
                @Override
                public void onComplete(Object o) {
                    org.json.JSONObject json = (org.json.JSONObject) o;
                    String nickname = json.optString( name: "nickname");
                    String figureurl = json.optString( name: "figureurl_qq_2");
                    spu.setLogin(true);
```