

《智能移动设备软件开发》

期末项目报告

项目名称: SmartParty App (智慧党建app)

2022-2023学年 第 2学期

1. 项目概况

为学习贯彻习近平新时代中国特色社会主义思想，该项目通过学习和测试主题教育知识，扎实推进主题教育。

* + - * 用户在APP中可以观看、学习党的主题教育内容
      * 用户可以参加主题教育知识小测（3道选择题）。

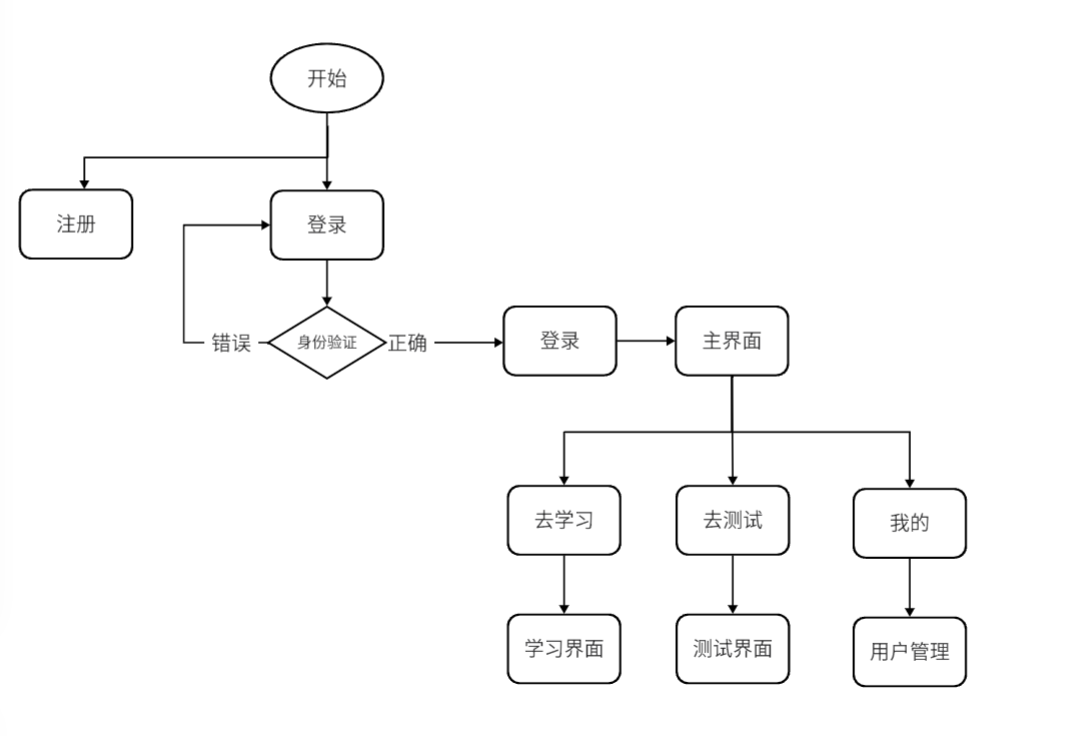


图1.1 业务流程图

1. UI设计

### 2.1 布局+视图控件

主要使用ConstraintLayout和LinearLayout布局，使用了Button、TextView、RadioButton、ImageView、EditText、ViewPager、RecyclerView等控件。

### 2.2 布局设计效果



图2.1 主界面

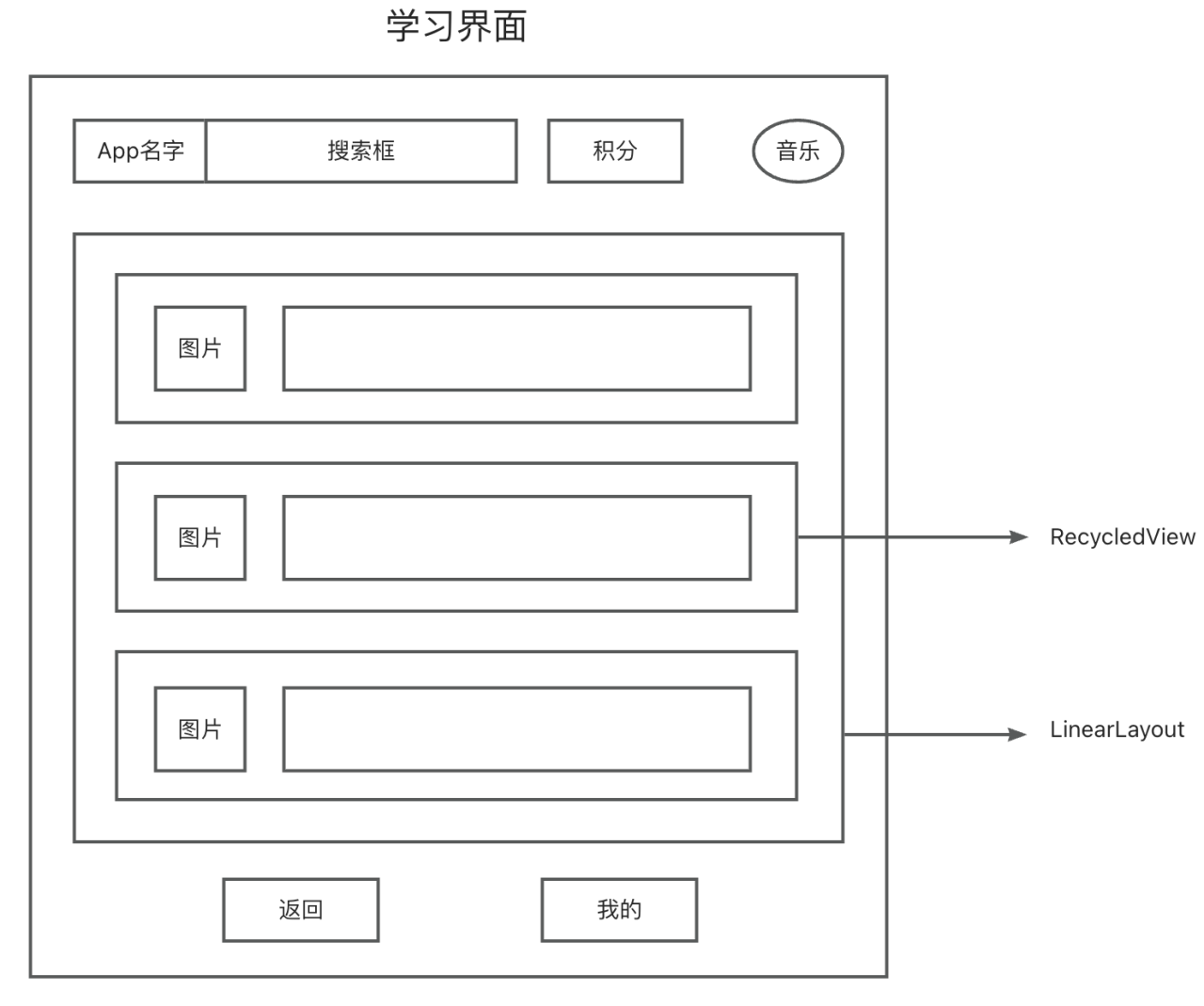


图2.2 学习界面

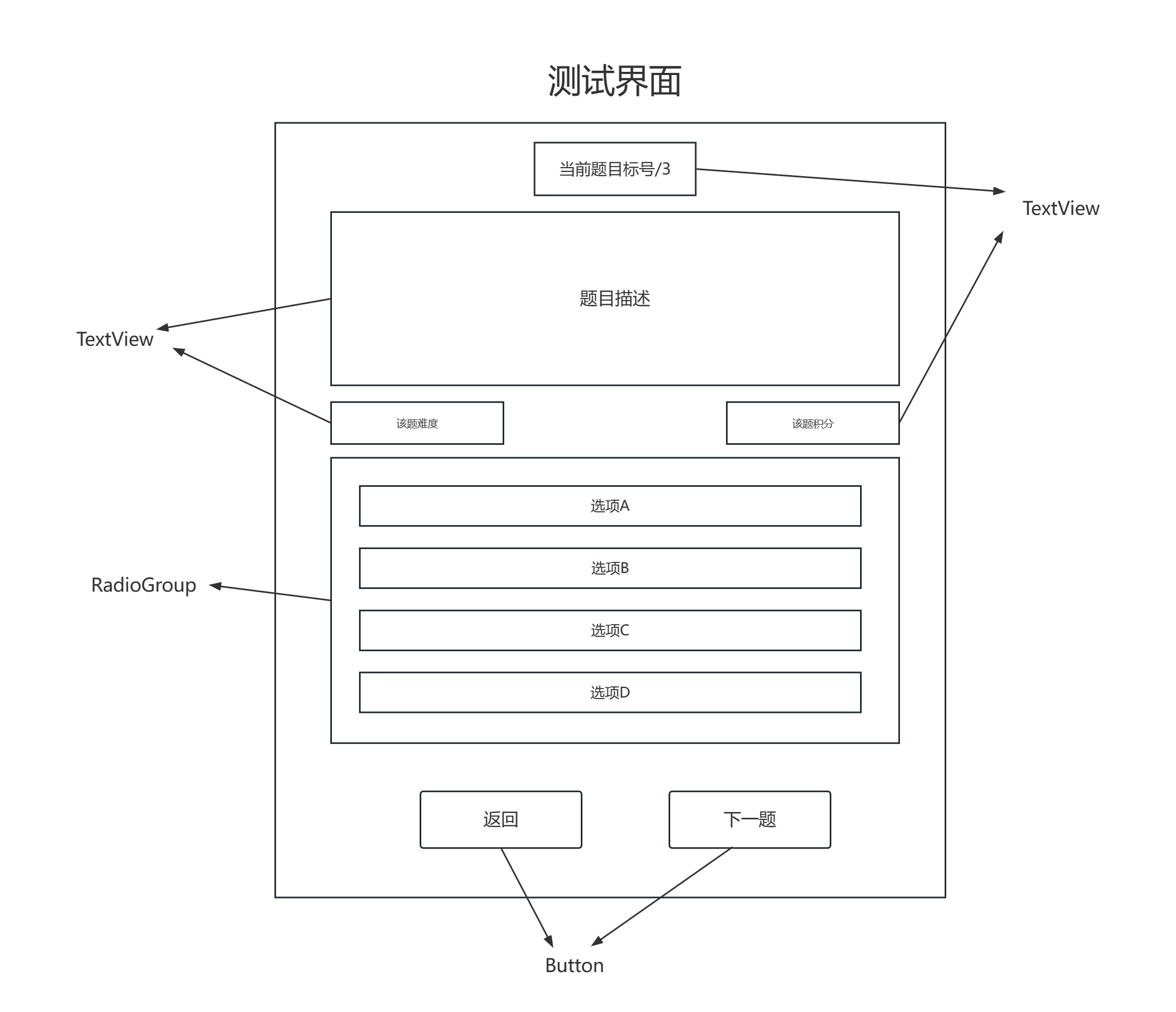


图2.3 测试界面

三、数据存储设计

1.主题教育知识点 knowledge

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 关键字段 | 说明 |
| id | int | √ | 知识点编号 |
| title | varchar(20) |  | 知识点标题 |
| content | varchar(150) |  | 知识点文本 |
| picture | varchar(20) |  | 知识点图片 |

2. 测试题库 test

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 关键字段 | 说明 |
| id | int | √ | 题目编号 |
| describe | varchar(50) |  | 题目描述 |
| level | varchar(5) |  | 题目难度 |
| score | int |  | 正确积分 |
| optionA | varchar(30) |  | 备选答案A |
| optionB | varchar(30) |  | 备选答案B |
| optionC | varchar(30) |  | 备选答案C |
| optionD | varchar(30) |  | 备选答案D |
| answer | varchar(5) |  | 正确答案 |

3. 用户信息 user

|  |  |  |  |
| --- | --- | --- | --- |
| 字段 | 类型 | 关键字段 | 说明 |
| id | varchar(20) | √ | 用户ID |
| username | varchar(20) |  | 用户名 |
| password | varchar(20) |  | 密码 |
| school | varchar(20) |  | 学校 |
| phone | varchar(11) |  | 电话 |
| gained\_score | int |  | 当前获得积分 |

四、功能设计与实现

### 3.1 功能点1

#### 3.1.1 流程分析与设计

（一）流程分析与设计   
主界面设计图：



当用户点击主界面的“去学习”按钮，进入学习界面；点击“去测试”，进入测试界面。

主界面每隔5秒轮播一条主题教育知识点，用户通过手势滑动实现“上一条”“下一条”的切换；如果用户点击“返回”，则回到主界面。

#### 3.1.2 实现方法+关键代码

**（一）轮播图及手动切换**

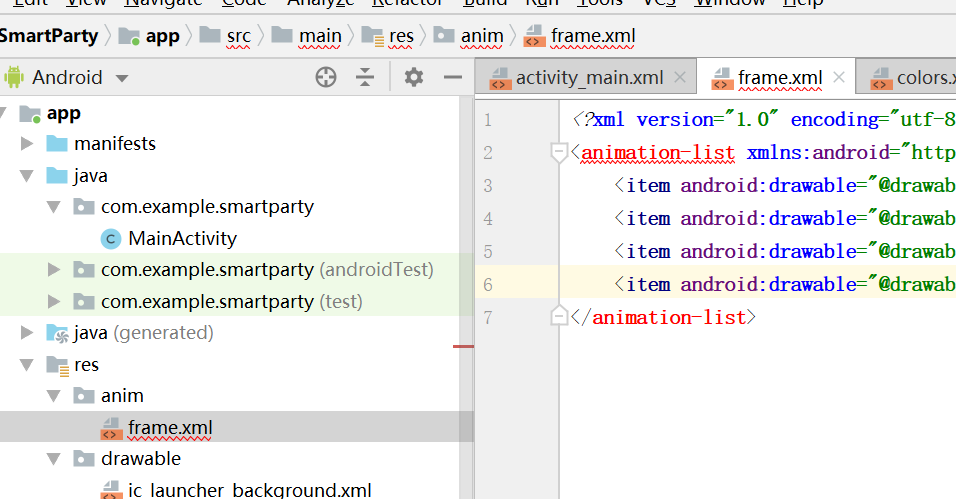
1、设计思想

利用AnimationDrawable实现帧动画每隔5秒循环播放，再利用其selectDrawable方法实现点击“上一张”、“下一张”的切换。

关键：用last current next分别记录上一张、当前的、预测下一张imageId。

遇到的问题：

（1）frame.xml文件位置放错了，应该放到drawable文件夹下。



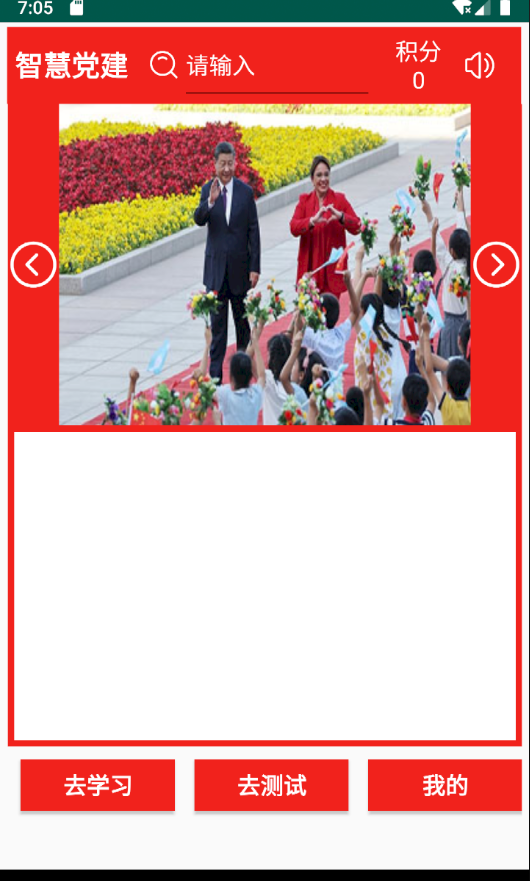
（2）用setImageResources实现切换就帧动画就会停止。

虽然这样能够获取当前的帧，但是并不能用setImageResources方法来切换上一张，因为它播放的是帧，不仅仅是图片。后来查询发现AnimationDrawable有个selectDrawable(int index)方法可以查看用于根据当前状态选择合适的Drawable资源。它可以根据不同的状态（例如按下、选中等）选择不同的Drawable资源，从而实现动态的UI效果。在开发中，可以使用selectDrawable方法来实现按钮按下、选中等状态下的不同背景、边框等效果。

2、关键代码：（右滑动同理）

|  |
| --- |
| *//左滑动* **img\_left**.setOnClickListener(**new** View.OnClickListener(){  @Override  **public void** onClick(View v) {  *//获取当前帧* currentFrame = **anim**.getCurrent();  **for** (**int** i = 0; i < **anim**.getNumberOfFrames(); i++) {  checkFrame = **anim**.getFrame(i);  **if** (checkFrame == currentFrame) {  **pic\_current** = i;  **break**;  }  }  Log.*i*(**"anim"**,**"pic\_current"**+**pic\_current**);  **if** (**pic\_current** == 0 ){  **pic\_last** = 3;  }**else**{  **pic\_last** = **pic\_current** - 1;  }  Log.*i*(**"anim"**,**"pic\_last"**+**pic\_last**); *// img\_center.setImageResource();* **anim**.selectDrawable(**pic\_last**);  }  }); |

3、运行效果：



**改进：**

ViewPager控件实现图片轮播和手势滑动切换。

上面那种方法能够成功实现手动切换图片功能，但是这种方法并不能实现手势滑动，不太符合手机app的特性。于是经过查询，发现利用ViewPager能够更好地实现功能。

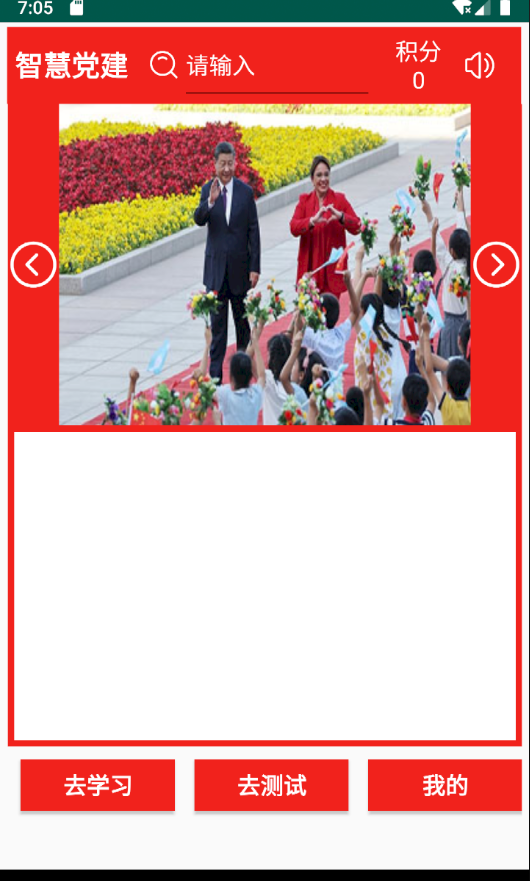
Android的ViewPager是一种可滑动的视图容器，它允许用户在同一个屏幕上浏览多个页面，并且可以通过左右滑动来切换这些页面。ViewPager通常用于实现轮播图、图片浏览、导航页等功能。

ViewPager可以与Fragment一起使用，每个页面就是一个Fragment。ViewPager中的Fragment可以通过FragmentPagerAdapter或FragmentStatePagerAdapter进行管理。其中，FragmentPagerAdapter适用于少量固定的Fragment，而FragmentStatePagerAdapter适用于大量动态生成的Fragment。ViewPager还提供了一些常用的方法和回调函数，如setCurrentItem()、addOnPageChangeListener()等，以便我们实现更加灵活的页面切换效果。

关键代码：

|  |
| --- |
| **class** ImagePagerAdapter **extends** PagerAdapter {  **private** Context **mContext**;  **private** List<Integer> **mImageIds**;  **public** ImagePagerAdapter(Context context, List<Integer> imageIds) {  **mContext** = context;  **mImageIds** = imageIds;  }  @Override  **public int** getCount() {  **return mImageIds**.size();  }  @Override  **public boolean** isViewFromObject(View view, Object object) {  **return** view == object;  }  @Override  **public** Object instantiateItem(ViewGroup container, **int** position) {  ImageView imageView = **new** ImageView(**mContext**);  imageView.setScaleType(ImageView.ScaleType.***CENTER\_CROP***);  imageView.setImageResource(**mImageIds**.get(position));  container.addView(imageView);  **return** imageView;  }  @Override  **public void** destroyItem(ViewGroup container, **int** position, Object object) {  container.removeView((View) object);  } } |

图片左右的左滑右滑不太美观，考虑还是去掉，实现手势左右滑动更符合常规。

**（二） 将MySql数据库的数据导进SQLite**

1、实现步骤：

（1）将MySql数据库表导出为sql文件；

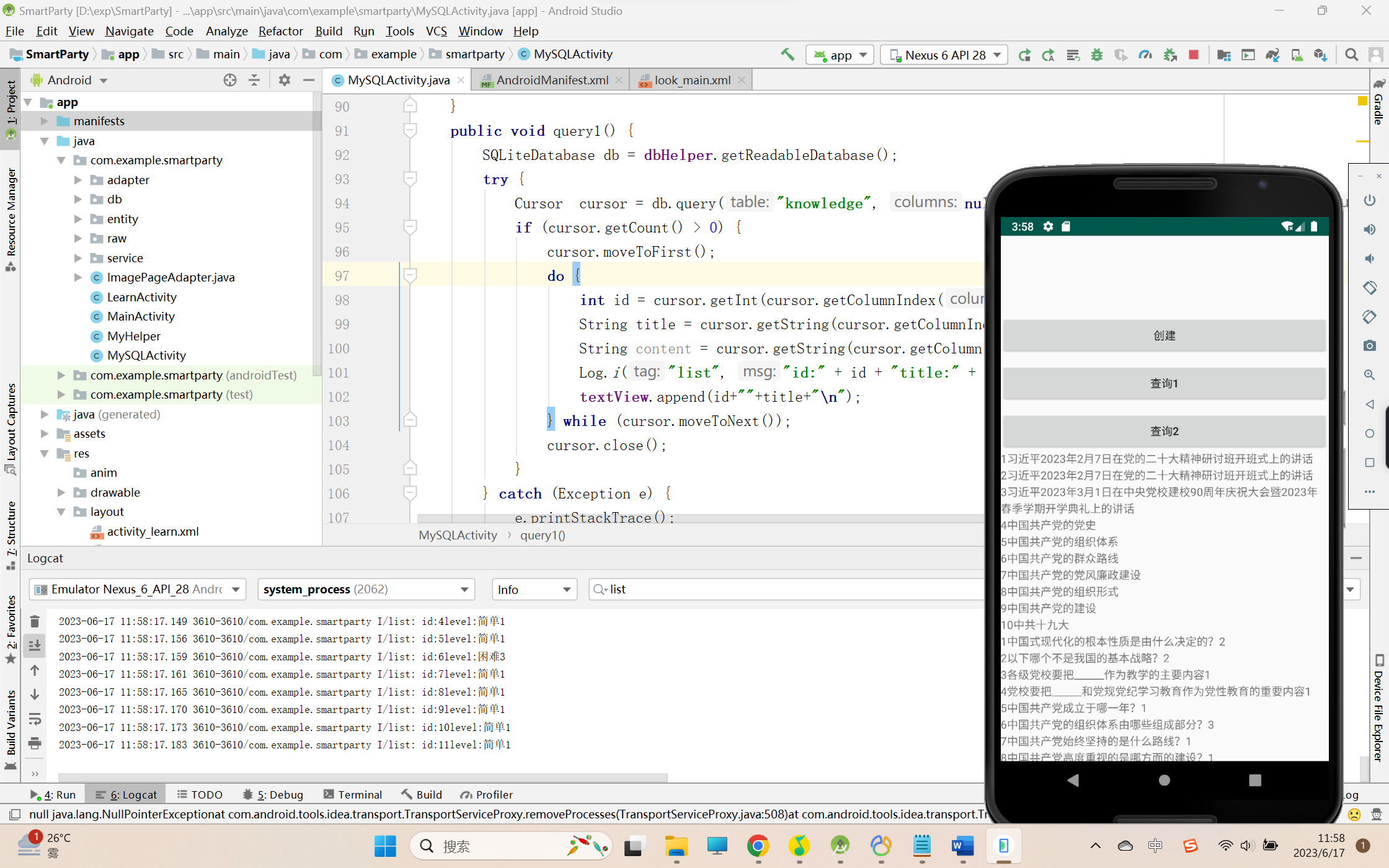
（2）SQLiteOpenHelper创建SQLite数据库和相应的表和字段；

（3）在Android项目中创建一个名为SqlParser的java类，用于遍历插入执行文件中的sql语句进行数据插入；

|  |
| --- |
| **class** SqlParser {  **public void** parseSqlFile(Context context) {  **db** = **dbHelper**.getWritableDatabase();  String[] table = {**"knowledge.sql"**, **"test.sql"**};  **for** (**int** i = 0; i < table.**length**; i++) {  **try** {  InputStream is = context.getAssets().open(table[i]);  BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(is));  String line = **null**;  **while** ((line = reader.readLine()) != **null**) {  **if** (line.trim().length() > 0) {  **db**.execSQL(line);  }  }  Log.*i*(**"list"**, **"成功"**);  reader.close();  is.close();  } **catch** (IOException e) {  e.printStackTrace();  Log.*i*(**"list"**, **"失败"**);  }  }  **db**.close();  **dbHelper**.close();  } } |

（4）利用Cursor进行查询，检查是否插入成功。

最终查询结果如下：



注：MyHelper的构造器中只有第一次调用会创建数据库，之后再创建对象时也只会调用onCreate方法。

**（三）RecyclerView展示知识点**

1、RecyclerView相比ListView的优点

（1）更好的性能：RecyclerView使用ViewHolder模式，可以**避免重复创建视图**，减少内存占用和布局的时间。

（2）更灵活的布局：RecyclerView支持多种布局方式，包括线性、网格、瀑布流等，而ListView**只支持线性布局**。

（3）更好的动画效果：RecyclerView提供了默认的动画效果，还可以自定义动画，使得列表的操作更加流畅。

（4）更好的扩展性：RecyclerView提供了更多的回调和接口，方便扩展和自定义，可以满足更多的需求。

（5）更好的分割线：RecyclerView支持添加分割线，而ListView需要手动添加。

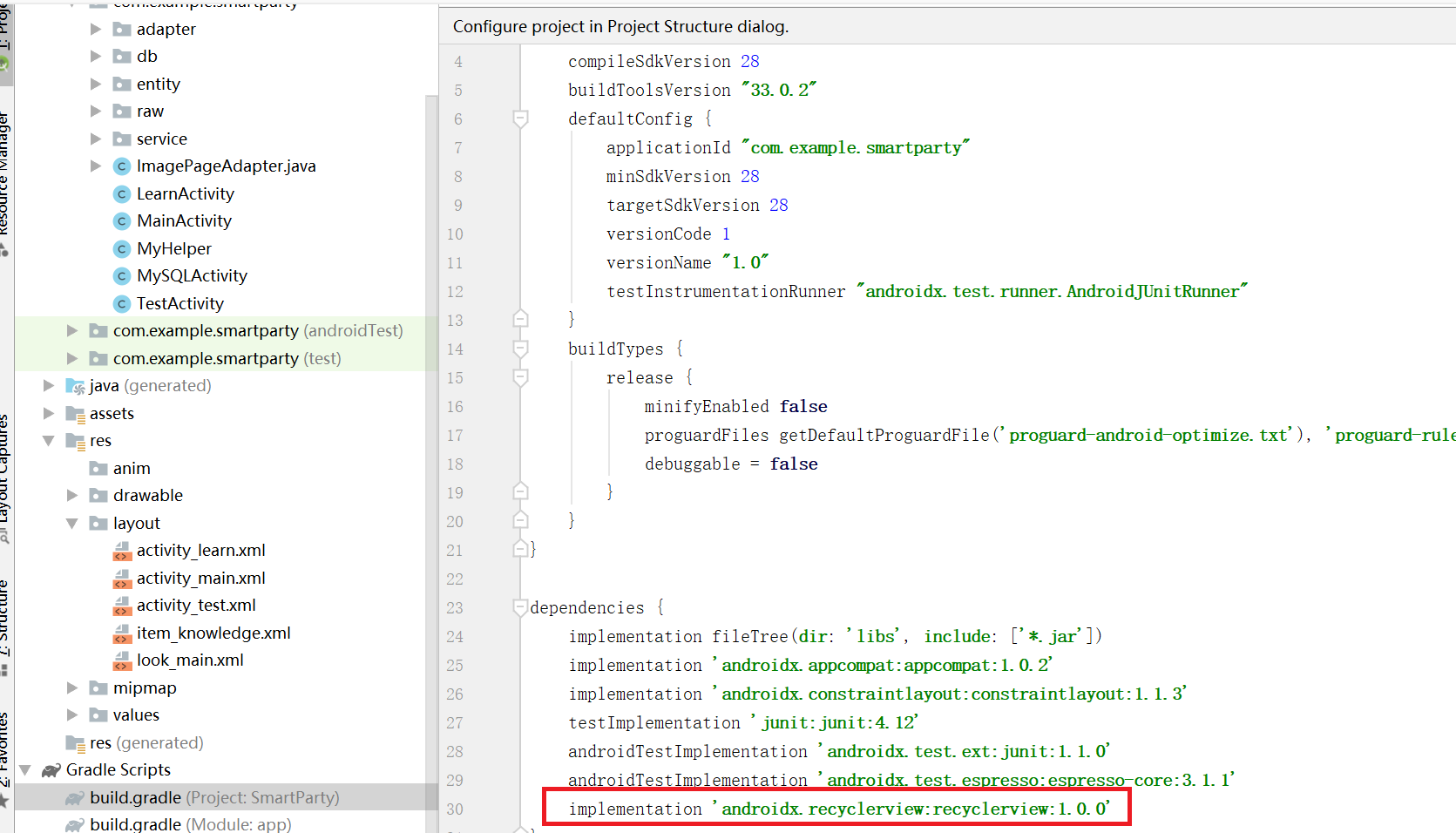
（6） 更好的缓存机制：RecyclerView提供了更好的缓存机制，可以**提高列表的加载速度**。

（7）更好的滑动体验：RecyclerView支持滑动到指定位置，滑动到顶部或底部等操作，更加方便用户使用。

综合以上优点考虑，本项目选取RecyclerView控件进行知识点的展示。

2、RecyclerView的实现步骤

（1）添加RecyclerView依赖库：在build.gradle文件中添加依赖库



（2）在布局文件中添加RecyclerView：在布局文件中添加RecyclerView控件

|  |
| --- |
| <**androidx.recyclerview.widget.RecyclerView  android:id="@+id/recyclerView"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"**  **android:background="@color/colorWhite"**  /> |

（3）创建ViewHolder：创建一个继承RecyclerView.ViewHolder的类，用来保存RecyclerView中每个item的视图。

|  |
| --- |
| **public class** MyViewHolder **extends** RecyclerView.ViewHolder {  **public** TextView **textView**;  **public** MyViewHolder(View itemView) {  **super**(itemView);  **textView** = (TextView) itemView.findViewById(R.id.textView);  }} |

（4）创建Adapter：创建一个继承RecyclerView.Adapter的类，用来管理RecyclerView中的数据和视图。

|  |
| --- |
| **public class** MyAdapter **extends** RecyclerView.Adapter<MyViewHolder> {  **private** List<String> **mData**;  **public** MyAdapter(List<String> data) {  **mData** = data;  }  @Override  **public** MyViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType) {  View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item\_layout, parent, **false**);  **return new** MyViewHolder(view);  }  @Override  **public void** onBindViewHolder(MyViewHolder holder, **int** position) {  holder.textView.setText(**mData**.get(position));  }  @Override  **public int** getItemCount() {//返回数据条数  **return mData**.size();  } } |

注：Recyclerview的onBindViewHoler主要负责将数据与holder绑定，它在列表滑动时会不停的被调用。

（5）设置LayoutManager：设置RecyclerView的LayoutManager，用于控制RecyclerView中item的布局方式。

|  |
| --- |
| recyclerView.setLayoutManager(new LinearLayoutManager(this)); |

（6）设置RecyclerView的Adapter，用于管理RecyclerView中的数据和视图。

|  |
| --- |
| recyclerView.setAdapter(adapter); |

（7）补充：

RecyclerView.ItemDecoration可以给RecyclerView添加分割线或其他装饰;

RecyclerView.ItemAnimator可以给RecyclerView添加动画效果。

3、效果展示：



**（四）积分统计**

1、实现方法：

利用startActivityForResult方法启动另一个Activity会在跳转到的活动页面销毁时，返回一个数据给上一个活动。

2、关键代码：

（1）启动测试界面时调用startActivityForResult：

|  |
| --- |
| **btn\_goTest**.setOnClickListener(**new** View.OnClickListener(){  @Override  **public void** onClick(View v) {  Intent intent = **new** Intent(MainActivity.**this**, TestActivity.**class**);  *//启动跳转，这里使用startActivityForResult（）方法，这个方法会在跳转到的活动页面销毁时，返回一个数据给上一个活动  /\*\*  \* param1:intent  \* param2：请求码，用于回调时判断数据的来源  \*/* startActivityForResult(intent, 1);}  }); |

（2）在主界面中重写onActivityResult方法：

|  |
| --- |
| @Override **protected void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {  **super**.onActivityResult(requestCode, resultCode, data);  Log.*e*(***TAG***, **"onActivityResult1: "**+data );  **switch** (resultCode) {  *//下面的1为startActivityForResult(intent, 1);中的1* **case** 1:  *//这里的1为setResult(1, intent);中的1* **if** (resultCode==1){  **int** score = data.getIntExtra(**"addScore"**,0);  Log.*i*(**"score"**,**"score"**+score);  *totalScore* += score;  **totalScoreView**.setText(*totalScore*+**""**);  Log.*i*(**"score"**,**"totalScore"**+*totalScore*);  }  **break**;  **default**:  **break**;  } } |

3、运行示例：



#### 3.1.3 效果展示



### 3.2 功能点2

#### 3.2.1 流程分析与设计

当用户选择知识小测选项时，他们将进入知识小测活动。知识小测活动从数据库中使用QuestionDatabaseHelper类检索一组问题，然后逐个显示给用户。用户选择答案并提交，应用程序检查答案是否正确，并向用户显示反馈。一次给出一个问题，一共有三个问题，直到回答完所有问题。回答完所有问题后，应用程序将计算用户的分数并将其显示给用户。为了实现此功能，创建一个TestActivity类，该类将显示每个问题和用户的得分。还需要创建一个Question类来表示每个问题，并创建一个QuestionDatabaseHelper类来检索问题列表。

TestActivity类将使用QuestionDatabaseHelper类从数据库中检索问题列表，并将其存储在ArrayList中。TestActivity类将使用currentQuestionIndex变量跟踪当前问题的索引，并使用score变量跟踪用户的得分。对于每个问题，TestActivity类将显示问题和选项，并等待用户选择答案。一旦用户选择答案，TestActivity类将检查答案是否正确，并显示反馈。TestActivity类将重复这个过程，直到所有问题都已回答。最后，TestActivity类将计算用户的得分并将其显示给用户。

#### 3.2.2 实现方法+关键代码

**（一）创建实体类Question**

表示一个问题对象，包含问题的描述、难度、分数、选项和答案等信息。在类中，通过定义私有属性，包括问题的ID、描述、难度、分数、选项A、B、C、D和答案等信息。并在构造方法中初始化这些属性。通过公有的getter方法，获取问题对象的描述信息、选项信息、正确答案、难度和分数等信息。在获取选项信息的方法中，将选项A、B、C、D的值存入List集合中，并返回该集合。通过封装问题对象的信息，方便其他程序调用和操作。

|  |
| --- |
| public Question(Integer id,String describe, String level, int score, String optionA, String optionB, String optionC, String optionD, String answer) {  this.id = id;  this.describe = describe;  this.level = level;  this.score = score;  this.optionA = optionA;  this.optionB = optionB;  this.optionC = optionC;  this.optionD = optionD;  this.answer = answer; }  public String getDescribe() {  return describe; }  public List<String> getOptions() {  List<String> options = new ArrayList<>();  options.add(optionA);  options.add(optionB);  options.add(optionC);  options.add(optionD);  return options; } |

**（二）随机挑选三个问题**

通过Random对象生成一个随机数，然后在while循环中，判断questionIndexList集合的大小是否小于3。如果小于3，则再次生成一个随机数，并判断该随机数是否已经存在于questionIndexList集合中。如果不存在，则将该随机数添加到questionIndexList集合中。这个过程会一直循环，直到questionIndexList集合的大小等于3，即随机选择了3个不同的索引值，用于在questionList集合中获取对应的问题。这样可以实现随机选择问题的功能。

|  |
| --- |
| Random random = new Random();  while (questionIndexList.size() < 3) {  int index = random.nextInt(questionList.size());  if (!questionIndexList.contains(index)) {  questionIndexList.add(index);  }  } |

**（三）显示第一个问题**

showQuestion(currentQuestionIndex)是一个方法调用，用于显示当前问题的信息，currentQuestionIndex是当前问题的索引值。

submitButton.setText("下一题")是设置按钮的文本内容为“下一题”。这段代码通常用于在用户回答完当前问题后，将按钮的文本内容修改为“下一题”，以便用户点击后跳转到下一个问题。

|  |
| --- |
| showQuestion(currentQuestionIndex); submitButton.setText("下一题"); |

**（四）设置提交按钮的单击事件**

当用户点击提交按钮时，会执行onClick()方法中的代码。

在onClick()方法中，首先调用checkAnswer()方法，用于检查用户的答案并更新得分。然后，将当前问题的索引值加1，用于显示下一个问题或者显示得分。如果当前问题的索引值小于questionIndexList集合的大小，则调用showQuestion()方法显示下一个问题，并判断当前问题是否为最后一个问题，如果是，则将按钮的文本内容修改为“提交”。

如果当前问题的索引值等于questionIndexList集合的大小，则调用showScore()方法显示得分。

通过监听提交按钮的点击事件，实现了在用户回答完当前问题后，自动跳转到下一个问题或者显示得分的功能。

|  |
| --- |
| submitButton.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View v) {  *// 检查用户的答案，更新得分* checkAnswer();  *// 显示下一个问题或显示得分* currentQuestionIndex++;  if (currentQuestionIndex < questionIndexList.size()) {  showQuestion(currentQuestionIndex);  if (currentQuestionIndex == questionIndexList.size()-1)  submitButton.setText("提交");  } else {  showScore();  }  } }); |

**（五）在页面上显示问题**

在方法中，首先根据传入的索引值获取对应的问题对象。然后，通过获取问题对象的描述信息和选项信息，分别设置问题文本框和答案选项。接下来，通过循环遍历问题对象的选项信息，创建RadioButton对象，并将其添加到答案选项的RadioGroup中。然后，通过计算当前问题的索引值和总问题数，设置进度文本框的文本内容。最后，获取问题对象的难度和积分信息，并设置到对应的文本框中。通过显示问题的描述信息和选项信息，以及相关的难度、积分、进度等信息，可以帮助用户更好地理解和回答问题。

|  |
| --- |
| private void showQuestion(int index) {  Question question = questionList.get(questionIndexList.get(index));  questionTextView.setText(question.getDescribe());  answerRadioGroup.removeAllViews();  for (String option : question.getOptions()) {  RadioButton radioButton = new RadioButton(this);  radioButton.setText(option);  answerRadioGroup.addView(radioButton);  }  String p = index+1 + "/3";  processTextView.setText(p);  String levelInfo = "难度："+question.getLevel();  String scoreInfo = "积分：" +question.getScore();  levelTextView.setText(levelInfo);  scoreTextView.setText(scoreInfo); } |

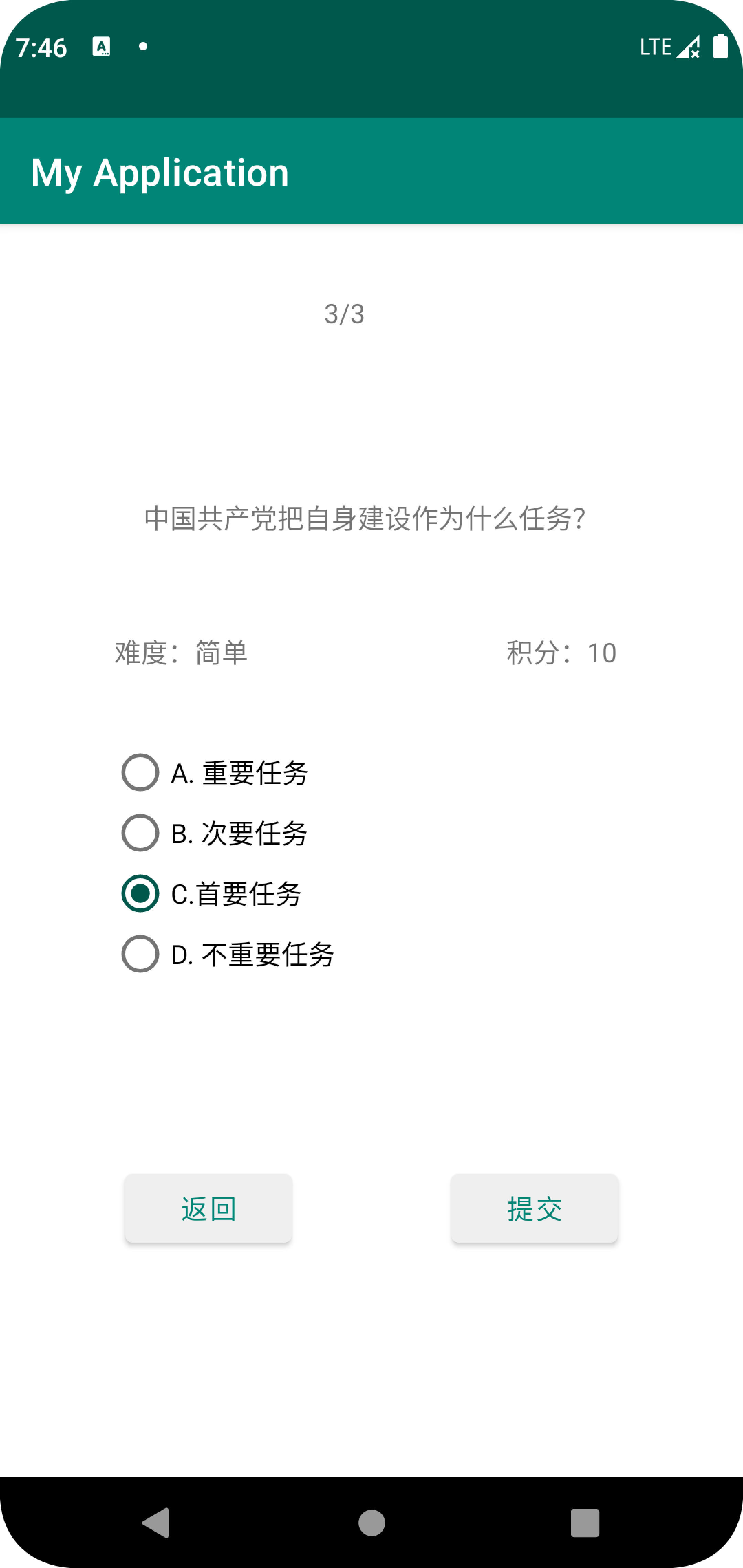
**（六）检查答案正确**

在方法中，首先获取当前问题的问题对象。然后，通过获取答案选项RadioGroup的选中的RadioButton的ID，判断用户是否有选择答案。

如果用户选择了答案，则获取选中的RadioButton对象，并将其文本内容转换为字符串类型的变量checkedAnswer。接着，通过比较checkedAnswer和问题对象的正确答案是否相等，判断用户的答案是否正确。如果用户的答案正确，则将问题对象的得分加入到用户的总分数score中。通过判断用户选择的答案是否正确，更新用户的总分数。

|  |
| --- |
| private void checkAnswer() {  Question question = questionList.get(questionIndexList.get(currentQuestionIndex));  int checkedRadioButtonId = answerRadioGroup.getCheckedRadioButtonId();  if (checkedRadioButtonId != -1) {  RadioButton checkedRadioButton = findViewById(checkedRadioButtonId);  String checkedAnswer = checkedRadioButton.getText().toString();  if (checkedAnswer.equals(question.getAnswer())) {  score += question.getScore();  }  } } |

#### 3.2.3 效果展示

****

### 3.3 功能点3

#### 3.3.1 流程分析与设计

创建一个服务类，用于封装MediaPlayer对象，实现播放背景音乐的功能。在服务类的onCreate()方法中，创建一个MediaPlayer对象，设置音乐资源并开始播放。在服务类中提供两个方法：playMusic()和pauseMusic()，用于控制音乐的播放和停止。在服务类的onDestroy()方法中，停止音乐的播放并释放MediaPlayer对象。

在APP启动主活动界面时，通过startService()方法启动服务类，并在服务类的onCreate()方法中初始化MediaPlayer对象，设置音乐资源并开始播放。在测试活动界面中，通过stopService()方法停止服务类，从而停止背景音乐的播放。在活动界面的onDestroy()方法中调用stopService()方法，以确保每次离开测试活动界面时都能停止背景音乐的播放。

Android Studio 中使用虚拟设备可能会遇到没有声音的情况，需要关闭虚拟设备的Snapshot功能后才能听到声音，如果仍然没有解决，则可以使用物理设备。

#### 3.3.2 实现方法+关键代码

1.创建一个服务类，实现播放音乐的功能，包括初始化MediaPlayer，加载音乐文件，播放、暂停、停止等操作。

|  |
| --- |
| public class MusicService extends Service implements MediaPlayer.OnPreparedListener, MediaPlayer.OnCompletionListener {       public static final String ACTION\_PLAY = "PLAY";      public static final String ACTION\_PAUSE = "PAUSE";  ​      private MediaPlayer mediaPlayer;  ​      @Override      public void onCreate() {          super.onCreate();          mediaPlayer = new MediaPlayer();          mediaPlayer.setOnPreparedListener(this);          mediaPlayer.setOnCompletionListener(this);     }        @Override      public int onStartCommand(Intent intent, int flags, int startId) {          String action = intent.getAction();          if (action != null) {              switch (action) {                  case ACTION\_PLAY:                      playMusic();                      break;                  case ACTION\_PAUSE:                      pauseMusic();                      break;                  default:                      break;             }         }          return super.onStartCommand(intent, flags, startId);     }  ​      @Override      public void onDestroy() {          if (mediaPlayer != null) {              mediaPlayer.stop();              mediaPlayer.release();         }          super.onDestroy();     }  ​      @Override      public void onPrepared(MediaPlayer mp) {          mediaPlayer.start();     }  ​      @Override      public void onCompletion(MediaPlayer mp) {          stopSelf();     }  ​      private void playMusic() {          if (!mediaPlayer.isPlaying()) {              mediaPlayer.start();         }     }  ​      private void pauseMusic() {          if (mediaPlayer.isPlaying()) {              mediaPlayer.pause();         }     }  ​      @Nullable      @Override      public IBinder onBind(Intent intent) {          return null;     }  } |

2. 在AndroidManifest.xml文件中注册服务。

|  |
| --- |
| <service android:name=".MusicService"/> |

3. 在MainActivity中启动服务，并在onStop方法中停止服务。

当退出主页面时，MainActivity的onStop方法被调用，停止了服务，但是当重新回到主页面时，MainActivity的onCreate方法不会被重新调用。因此，需要在MainActivity的onRestart方法中重新启动服务，以便重新播放音乐。

|  |
| --- |
| public class MainActivity extends AppCompatActivity {  ​      private Intent musicIntent;      private Button pauseButton;      private boolean isPlaying = true;        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);  ​          musicIntent = new Intent(this, MusicService.class);          startService(musicIntent);  ​          pauseButton = findViewById(R.id.pause\_button);          pauseButton.setOnClickListener(new View.OnClickListener() {              @Override              public void onClick(View v) {                  if (isPlaying) {                      pauseButton.setText("Play");                      pauseMusic();                 } else {                      pauseButton.setText("Pause");                      playMusic();                 }                  isPlaying = !isPlaying;             }         });     }  ​      @Override      protected void onStop() {          super.onStop();          stopService(musicIntent);     }       @Override      protected void onRestart() {          super.onRestart();          startService(musicIntent);     }  ​      private void pauseMusic() {          Intent intent = new Intent(this, MusicService.class);          intent.setAction(MusicService.ACTION\_PAUSE);          startService(intent);     }  ​      private void playMusic() {          Intent intent = new Intent(this, MusicService.class);          intent.setAction(MusicService.ACTION\_PLAY);          startService(intent);     }  } |

成功实现了在MainActivity启动时会自动播放背景音乐，当打开其他页面或退出应用时会停止播放。

（二） RecyclerView展示知识点

1、设计思想

知识点页面布局、视图控件与主界面保持相同风格，主体使用 RecyclerView 展示每一条知识点。创建item\_knowledge.xml文件，作为 RecyclerView 中每个 item 的布局文件，在 KnowledgeAdapter 的 onCreateViewHolder 方法中，使用 LayoutInflater 从 item\_knowledge.xml 中获取视图并返回 ViewHolder 对象。

2、实现方法：

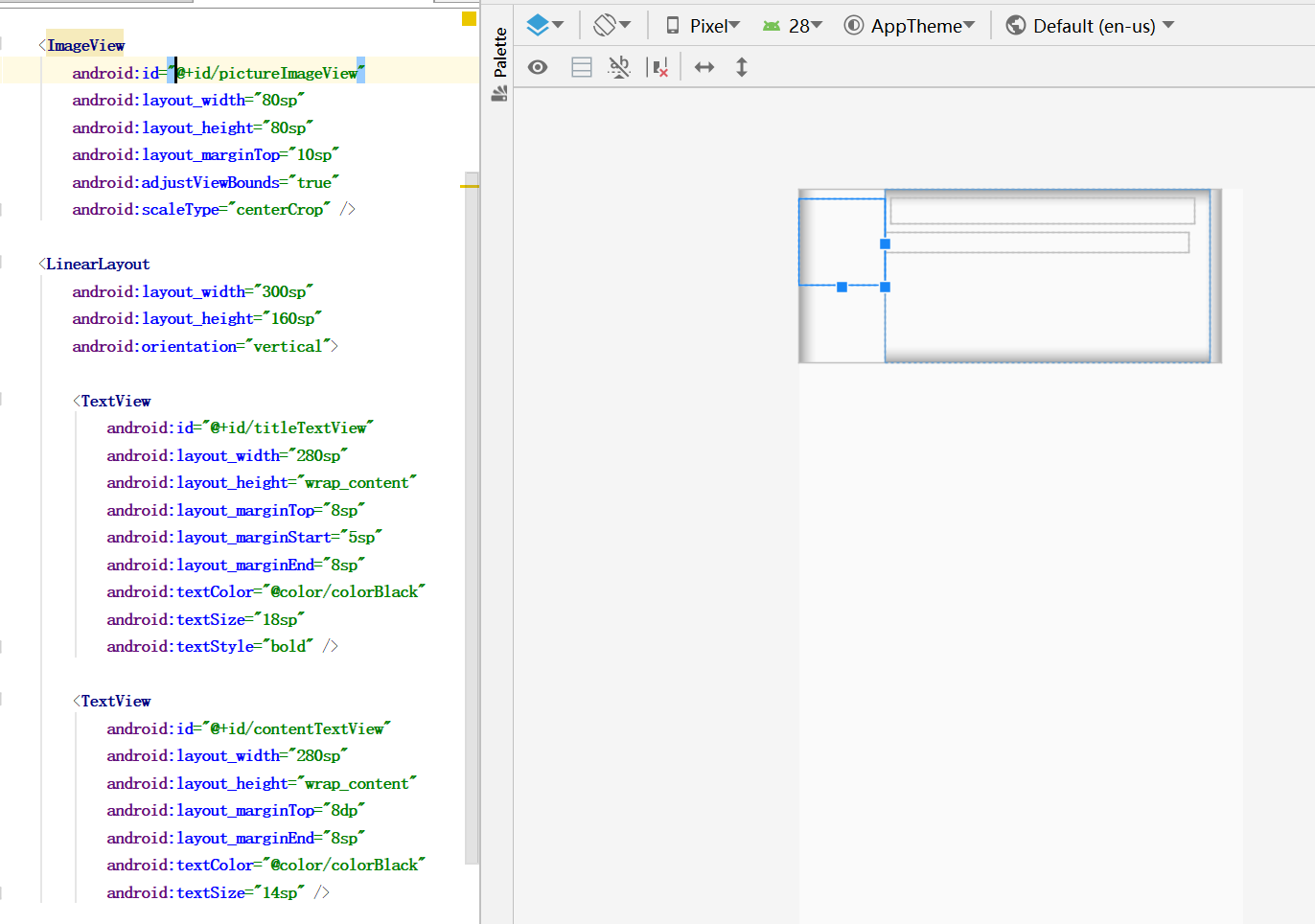
（1） 知识点页面 activity\_learn.xml 设计

主要使用 RecyclerView 展示每一条知识点



（2）布局文件 item\_knowledge.xml 设计

item\_knowledge 是 RecyclerView 中每个 item 的布局文件



（3）编写适配器 KnowledgeAdapter

KnowledgeAdapter 继承了 RecyclerView.Adapter，实现了 onCreateViewHolder、onBindViewHolder 和 getItemCount 方法。

在 onCreateViewHolder 方法中，通过 LayoutInflater 从 item\_knowledge.xml 中获取视图并返回 ViewHolder 对象; 在 onBindViewHolder 方法中，获取当前位置的 Knowledge 对象，并将其标题、内容和图片分别设置到 ViewHolder 的对应控件中; 在 getItemCount 方法中，返回数据源中 Knowledge 对象的数量。

（4）通过 setAdapter 方法将 Adapter 设置给控件RecyclerView

3、关键代码：

|  |
| --- |
| **public class** KnowledgeAdapter **extends** RecyclerView.Adapter<KnowledgeAdapter.ViewHolder> {  **private** List<Knowledge> **knowledgeList**;  **public** KnowledgeAdapter(List<Knowledge> knowledgeList) {  **this**.**knowledgeList** = knowledgeList;  }  @NonNull  @Override  **public** ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, **int** viewType) {  View view = LayoutInflater.*from*(parent.getContext()).inflate(R.layout.***item\_knowledge***, parent, **false**);  **return new** ViewHolder(view);  }  @Override  **public void** onBindViewHolder(@NonNull ViewHolder holder, **int** position) {  Knowledge knowledge = **knowledgeList**.get(position);  holder.**titleTextView**.setText(knowledge.getTitle());  holder.**contentTextView**.setText(knowledge.getContent());  Bitmap bitmap = BitmapFactory.*decodeFile*(knowledge.getPicture());  holder.**pictureImageView**.setImageBitmap(bitmap);  }  @Override  **public int** getItemCount() {  **return knowledgeList**.size();  }  **public static class** ViewHolder **extends** RecyclerView.ViewHolder {  TextView **titleTextView**;  TextView **contentTextView**;  ImageView **pictureImageView**;  **public** ViewHolder(@NonNull View itemView) {  **super**(itemView);  **titleTextView** = itemView.findViewById(R.id.***titleTextView***);  **contentTextView** = itemView.findViewById(R.id.***contentTextView***);  **pictureImageView** = itemView.findViewById(R.id.***pictureImageView***);  }  } } |

#### 3.3.4 效果展示



五、总结与反思

*(列举项目开发中的2个（或更多个）难点，简单介绍你们组是如何解决的？本项目在哪些方面可以提升？)*

**难点1：**数据库设计和实现：如何设计数据库以存储问题和知识点，并在应用程序中使用它们？

**解决1：**

为了解决这个问题，小组使用了SQLite数据库，并创建了一个QuestionDatabaseHelper类和KnowledgeDatabaseHelper来处理数据库操作; 创建一个测试表（test）来存储问题和答案，并使用了适当的数据类型来存储不同的字段，例如文本和整数; 创建一个Question类来表示每个问题和Knowledge类来表示每个知识点，并使用QuestionDatabaseHelper类从数据库中检索问题列表。

**难点2：**小组初始时在Navicat里面创建了Mysql数据库以及对应的表及字段，但是如何把Mysql的数据导入SQLite使用呢？

**解决2：**

（1）将MySql数据库表导出为sql文件；

（2）SQLiteOpenHelper创建SQLite数据库和相应的表和字段；

（3）在Android项目中创建一个名为SqlParser的java类，用于遍历插入执行文件中的sql语句进行数据插入；

**难点3：**在设计功能点1的时候，利用AnimationDrawable的selectDrawable方法虽然能实现点击切换图片，但是帧动画还是会从点击前的那一帧动画开始播放。

**解决3：**

通过查询，了解到通常使用ViewPager实现轮播图、图片浏览、导航页等功能。Android的ViewPager是一种可滑动的视图容器，它允许用户在同一个屏幕上浏览多个页面，并且可以通过左右滑动来切换这些页面。

**界面改进：**

**改进1：**主界面的轮播只是图片的轮播，加上知识点的文字轮播会更好。  
**改进2：**在背景音乐功能中增加用户自主选择音乐的功能，增加用户的个性化体验

**改进3：**还可以增加“白天”“黑夜”自主切换模式，增加个性化效果。

**功能改进：**

**改进4：**还需要再增加搜索界面和用户信息界面更符合app的规范。

**改进5：**增加用户登录功能，记录用户的学习和测试成绩，以便用户进行更好的学习规划

**改进6：**在测试界面中增加解析功能，让用户了解正确答案和错误的原因

**改进7：**可以在测试页面添加一个计时器，来限制用户回答每个问题的时间。当时间到了之后，可以自动提交答案并显示下一个问题；还可以将时间限制与得分相关联，在规定时间内回答问题的用户可以获得更高的得分。

**改进8：**允许用户选择测试的难度级别（例如，初级、中级、高级），并相应地选择问题进行测试。

**心得体会：**

本次团队项目完成的较为顺利，在这个过程中大家先协商计划，先完成合作部分，再分阶段完成个人部分，包括设计、代码实现和文档撰写，最后一起总结改进。个人部分不会的大家也都会互相给出建议，相互促进。但是也遇到一些困难，每个组员完成自己的部分后，将各个部分整合起来是个难题，我们小组的方法是由一个同学加上小组的沟通交流结合完成了代码的整合。