摘要

一个专为文案爱好者和创意工作者设计的应用程序。无论您是市场营销人员、社交媒体经理、内容创作者，还是只是喜欢收集和欣赏优美文字的人，文研 APP 都能为您提供无尽的灵感和创意。文研 APP 不仅是一个文案作品展示平台，更是一个文案创作交流的社区。用户可以在这里找到灵感、学习技巧、展示作品，与其他文案爱好者互动交流。无论是想要成为一名优秀的文案策划师，还是只是想在朋友圈里写出一段引人注目的文字，文研 APP 都能帮助实现，让文字更有魅力，让创意更有价值，让作品更受欢迎。

关键词：文案，语句，AI，创作

工作质量量化指标：

文档评价指标	是/否
文档有效页数	50
格式是否规范	是
内容是否详实	是
是否存在大量贴无效代码情况	否
是否包含背景分析	是
是否包含需求分析	是
是否包含界面设计	是
是否包含 APP 功能结构图	是
是否包含流程分析（非必须）	是
是否包含数据库表设计（非必须）	是
是否包含关键技术分析	是

图表 1

代码评价指标	个数	作用说明
类的个数	59	14 个 adapter、12 个 Bean（数据类）22 个 fragment、8 个辅助工具类、3 个与音乐相关的辅助类
Activity 个数	26	
Service 个数	1	管理音乐播放
是否用到 Broadcast Receiver	是	音乐播放器的控制
是否用到 Content Provider	是	头像拍照或图片选择功能
是否用到系统服务	是	通知栏的实现，用于显示通知栏音乐播放器
是否用到第三方控件	是	Okhttp
是否有后台	是	音乐播放的后台播放
是否用到文件存储	是	拍摄招聘与读取图片
是否用到 SQLite 存储	是	存储用户信息
是否用到多线程	是	更新 UI 界面

图表 2

目 录

第一章 APP概述

1.1 背景和意义

1.1.1 内容创作需求增加

随着互联网和社交媒体的发展，各类平台对内容的需求不断增加。无论是企业品牌营销、个人自媒体运营，还是日常社交分享，都离不开优质的文案。优质的文案不仅能提升传播效果，还能增强用户互动。

1.1.2 多样化文案需求

不同场景、不同用途的文案需求各异。广告推广需要引人注目的宣传文案，社交媒体需要引发共鸣的互动文案，励志语录需要激励人心的正能量文字。市场上急需一个能全面覆盖这些需求的工具。

1.1.3 现有文案工具的局限

市面上已有一些文案类应用，但大多数内容单一，更新不及时，无法满足用户多样化的需求。此外，许多应用缺乏智能化的创意生成功能，用户体验不佳。

1.2 主要工作内容

1.2.1 界面 UI 设计

在设计和开发 App 界面时，考虑到现代人对视觉美感的要求变得越来越高，采用多种不同的嵌套布局可以有效地满足他们的审美需求。

首先，多种不同的嵌套布局使得 App 界面呈现出更加有层次感和动态感的效果。通过将不同的组件嵌套在一起，可以创建出更加复杂和丰富的布局结构，使得界面看起来更具立体感和视觉吸引力。例如，可以在主视图上嵌套卡片式布局，在卡片内部再使用线性布局或相对布局来组织内容，这样可以有效地利用空间，并给用户一种更加整洁和有序的感觉。

其次，多种不同的嵌套布局可以为App界面增加更多个性化和创意的元素。通过将不同的布局组合在一起，可以让界面呈现出更加独特和个性化的样式。例如，可以在主界面上使用网格布局来展示不同的模块，然后在某些模块中使用约束布局或表格布局来组织具体的内容，这样可以在保持整体风格的基础上增加更多细节和创意。

此外，多种不同的嵌套布局也可以提升用户体验，使得 App 界面更加易用和直观。通过合理的布局嵌套，可以使得用户在使用 App 时更加舒适和便捷，减少繁琐的操作和信息查找。

1.2.2 数据对接实现

通过使用Apifox来模拟后端服务，可以更灵活地进行接口测试和数据交互，而不必依赖于实际的后端服务。这种方式可以帮助在前端开发过程中更快地进行数据调试和接口调用，从而提高开发效率。总的来说，与前后端分离的项目类似，使用 Apifox 来模拟后端服务可以更好地实现数据的动态变化，提升项目的灵活性和可维护性。

1.2.3 知识内容融合

在开发的 APP 中，将本学期学习到的知识全面应用，例如利用数据库知识设计和管理用户数据，确保数据安全和稳定性；利用网络编程知识实现与服务器的数据交互，实现实时更新和同步功能；利用用户界面设计知识打造简洁直观的界面，提升用户体验。通过将学习到的知识融入到 APP 的开发中，使 APP 更加完善和专业。

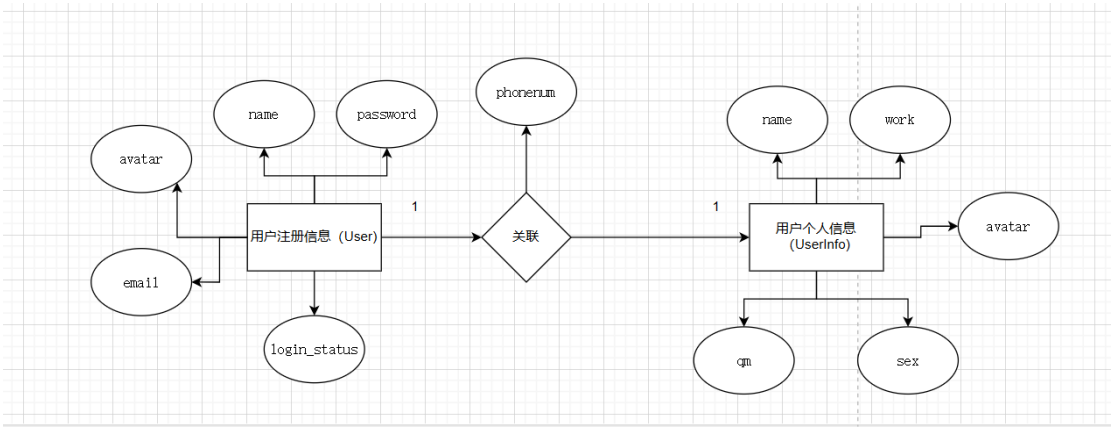
第二章 需求分析

2.1 APP 简介

文研 APP 是一款专为文案爱好者和创意工作者设计的应用，旨在帮助用户提升文案写作技巧，激发创意灵感，提高工作效率。用户可以在应用中找到各种文案范例、创意灵感、写作技巧等资源，帮助他们更好地表达自己的想法和情感。此外，文研 APP 还提供了编辑工具和分享功能，方便用户在写作过程中进行修改和分享，与他人交流和合作。无论是想要提升自己的文案写作水平，还是需要灵感创意的时候，文研 APP 都能成为用户的得力助手。

2.2 需求分析

2.2.1 数据库设计



图表 3 数据库设计图

在用户信息存储方面使用 sqllit 数据库，通过创建一个包含用户登录信息的表，可以轻松地存储用户的用户名、头像、密码、手机号、登陆状态、电子邮件等信息。再创建一个包含用户账号信息的表，包含用户名、头像、工作、签名、性别，并使用手机号将两个表进行关联。利用数据库存储登陆状态，实现用户登陆状态的持久化，保证用户无需多次重复登录。

2.2.2 APP 功能结构

文研 APP 主页有 4 个 Fragment、分别是首页、工具、社区、我的。

1、首页主要展示内容，最上面是一个轮播图，用于展示 APP 的消息信息等，下面是 10 个服务的分类，每个服务对应一种文案的类型，包括（励志奋斗、温柔治愈、影视台词、书籍摘抄、歌词摘录、广告文案、营销文案、美食推广、毒汤语录、全部分类）。

1.2、灵感创意模块，包含了文案合辑、美食推广、音乐鉴赏、节日感文案、今日好文、广告模板、品牌案例，每个页面展示不同的内容。并使用界面与标题关联，实现标题栏的可滑动性以及点击标题栏切换界面或者滑动界面标题栏对应切换。

1.3、该广告文案模块采用滑动展示效果，可以展示多样广告信息。在页面上，图片与文字结合在一起，图片位于上方，文字位于下方。当展示图片时，会自动放大，以便更好地展示广告内容。这种设计可以吸引用户的注意力，同时提供更多的信息展示空间，使广告更加生动和吸引人。

1.4、营销文案，用于展现可用于不同领域的营销文案。

2、AI 创作工具是指利用人工智能技术进行创作的工具。这些工具主要依靠 AI 对话功能，通过与用户进行对话交互，生成创作内容。AI 对话功能可以理解用户的需求和意图，并根据用户的输入生成相应的创作内容。这种工具可以用于创作各种形式的内容，如文章、诗歌、音乐、绘画等。

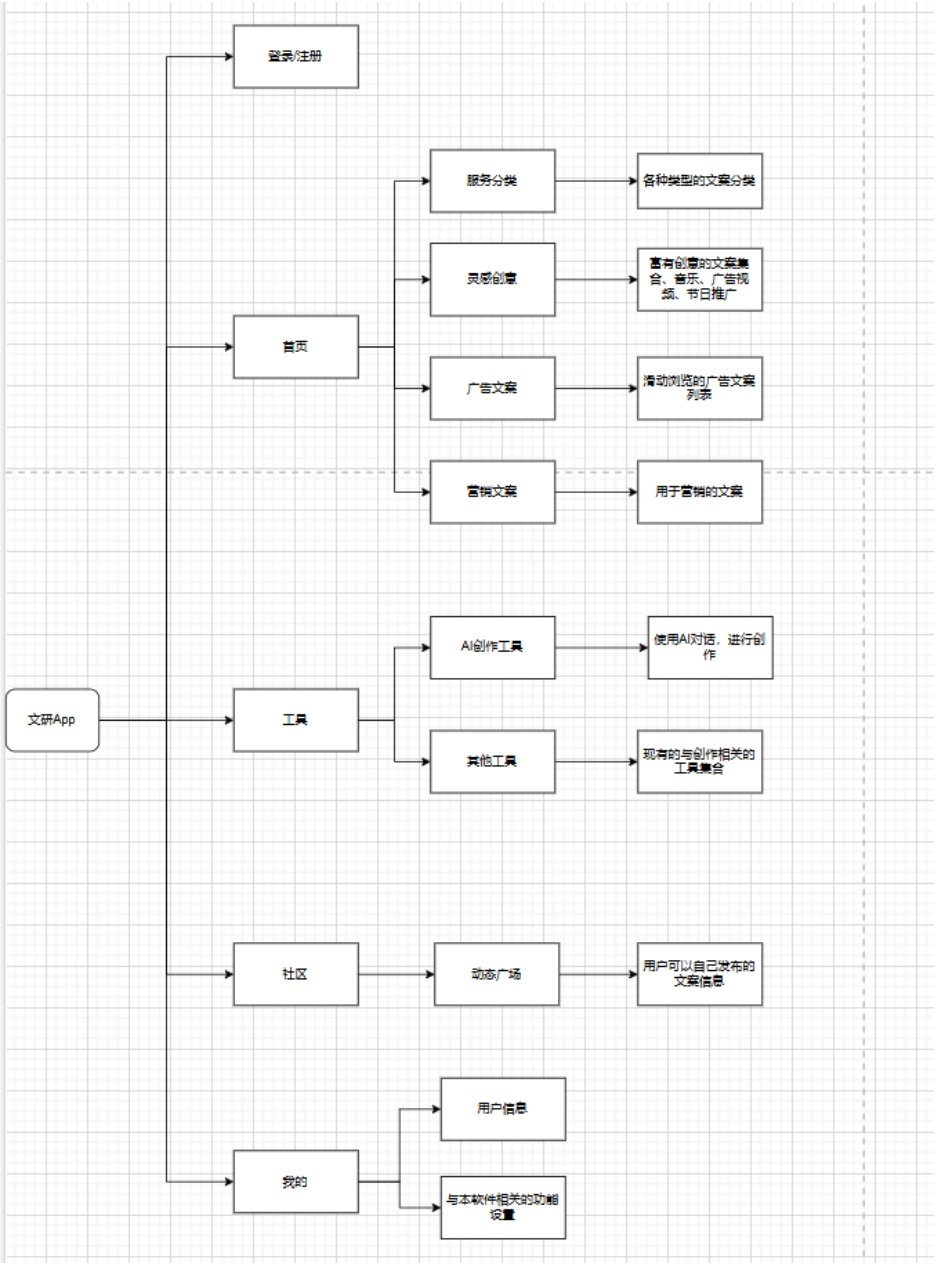
2.1、除了 AI 创作工具，工具界面还包括其他种类的创作工具。这些工具展示了 12 种不同种类的创作工具，每种工具都有其独特的功能和用途。这些工具可以帮助用户进行创作，提供各种创作所需的功能和工具。例如，有些工具可以帮助用户进行图像编辑和设计，有些工具可以帮助用户进行音频和视频编辑。

2.2、这些工具的种类多样，涵盖了各个创作领域的需求。用户可以根据自己的创作需求选择适合的工具进行使用。这些工具的功能和界面设计都经过精心设计，以提供用户友好的使用体验。无论是专业创作者还是普通用户，都可以通过这些工具实现自己的创作目标。工具界面的分为 AI 创作工具和其他工具，旨在帮助用户更好地利用人工智能技术和其他工具进行创作，提高创作效率和质量。

3、社区界面展示用户发布的文案信息，是一个集中了所有用户的发布消息的平台。在这个社区界面上，用户可以发布自己的文案信息，包括广告文案、宣传文案、推广文案等等。这些文案信息可以是文字、图片、视频等形式，用户可以根据自己的需求和目标来选择合适的方式进行发布。社区界面的集中发布功能使得用户可以在一个平台上找到各种类型和领域的文案信息，无论是商业广告、

个人创作还是社会公益等，都可以在这里找到相关的文案信息。这种集中发布的方式不仅方便了用户的浏览和搜索，也提高了文案信息的曝光率和传播效果。

4、用户可以在我的界面中查看自己的个人信息，包括用户名、头像、个性签名等。同时，用户还可以在该界面进行个人信息的编辑，如修改用户名、更换头像等操作。除此之外，用户还可以在我的界面中对 APP 进行各种操作设置，如修改密码、设置通知提醒、管理账号安全等功能。通过我的界面，用户可以方便地管理自己的账号信息，并对 APP 进行个性化的设置，提升用户体验。



图表 4 APP 功能结构图

第三章 APP设计与实现

3.1 界面设计

3.1.1 欢迎界面

进入程序，首先显示时长为 2 秒的欢迎界面。通过展示一个持续 2 秒的欢迎界面，可以给用户一个良好的第一印象，同时也可以为应用程序的加载和初始化过程提供一些缓冲时间，确保应用程序在展示主界面之前已经完成了必要的准备工作。

图表 5 欢迎界面展示

3.1.2 登录/注册界面

登录界面使用两个 EditText，用于输入用户名和密码，一个 CheckBox 用来

确认是否需要记住密码，以及一个用户注册的 Textview，点击后跳转到注册界面。注册界面头像处，使用 Imageview 显示默认头像，用户可以点击后选择或者拍摄自己的头像。

图表 6 登录界面

图表 7 注册界面

3.1.3 主界面

主界面为一个 Activity，该 Activity 下创建一个 Tablayout 和一个 viewpager，使用 List<Fragment>对 fragment 进行管理，每个 fragment 下实现一个界面，分别是{"首页","工具","社区","我的"}。

图表 8 主界面导航栏

3.1.4 首页

使用 LinearLayout 布局，右上到下，分别是搜索框、轮播图、RecyclerView 用于展示接口获取的服务、一个嵌套的 HorizontalScrollView+自定义的 CustomViewpager，用于实现滑动展示不同的界面。每个界面都是一个 fragment、又一个 RecyclerView 和 CustomViewpager，这里实现上面展示图片，下面展示图片对应的内容、LinearLayout 实现的静态的布局。为实现界面的美观，每个项目都使用 CardView 包围，实现圆角效果。

图表 9 首页展示

3.1.5 工具界面

使用 TabLayout+ViewPager，实现两个部分的切换展示。AI 创作工具界面，上面部分使用简单的 LinearLayout 布局，实现图片的展示，中间部分使用 RecyclerView 实现类似轮播图的展示效果。最下面使用 HorizontalScrollView+CustomViewpager，实现不同工具的展示。其他工具界面，上面使用 GridLayout，对每个按钮进行合理布局。

图表 10 工具界面展示

3.1.6 社区界面

使用 HorizontalScrollView+CustomViewpager 进行嵌套布局，实现对贴子的分类，分为最新和最热两个模块，每个对应一个页面。CustomViewpager 里使用 ListView 进行动态数据的展示。

图表 11 社区界面展示

3.1.7 我的页面

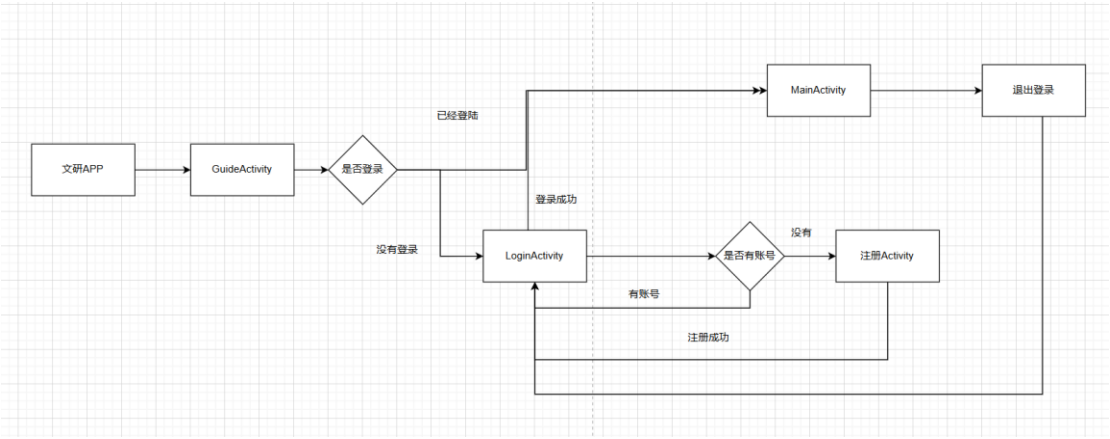
使用 ConstraintLayout+LinearLayout+CardView 等多种布局嵌套，实现界面的显示。

图表 12 我的界面展示

图表 13 编辑资料界面展示

3.2 APP 实现.

3.2.1 登录注册



图表 14 程序运行流程图

GuideActivity 读取数据库中的信息，获取当前登录的用户信息，如果存在已经登录的用户，则直接进入 MainActivity，否则进入 LoginActivity。

主要代码实现：

```
adapter= new GuideAdapter(imageViews);
vp.setAdapter(adapter);
mDBOpenHelper=new DBOpenHelper(this);
loginuser=mDBOpenHelper.getAllLoggedInUsers();
new Handler().postDelayed(new Runnable() {
   public void run() {
      if(loginuser.size()==1){
         Intent intent = new Intent(GuideActivity.this, MainActivity.class); //第二个参数即为执行完跳转后的
Activity
         startActivity(intent);
      }else {
         Intent intent = new Intent(GuideActivity.this, LoginActivity.class);   //第二个参数即为执行完跳转后的
Activity
         startActivity(intent);
      }
      GuideActivity.this.finish();  //关闭 splashActivity，将其回收，否则按返回键会返回此界面
   }
}, SPLASH_DISPLAY_LENGHT);
```

如果没有登录，在登录页面中，使用 SharedPreferences 实现记住密码。当用

户点击记住密码后，SharedPreferences 会保存当前 Editor 的数据，在下一次打开后，直接填充到输入框中。用户登录成功后，会跳转到 MainActivity 中，并且同时更新数据库中当前用户的登陆状态。

主要代码实现：

```
String user_phone = userPhone;
Intent intent1 = new Intent(LoginActivity.this, MainActivity.class);//设置自己跳转到成功的界面
mDBOpenHelper.updateLoginStatus(user_phone,1);
//intent1.putExtra("user_name",user_name);
startActivity(intent1);
finish();
```

当用户点击注册时，跳转到注册界面，在注册界面输入对应的用户信息。同时用户可以上传自己的头像，在头像编辑处，使用 fileprovider，以及图片的读写。主要代码实现：

拍照实现：

```
private void takePhoto() {
    tempFile = new File(this.getExternalFilesDir(DIRECTORY_DCIM), System.currentTimeMillis() + ".png");
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
        intent.setFlags(Intent.FLAG_GRANT_WRITE_URI_PERMISSION);
        Uri contentUri = FileProvider.getUriForFile(RegisteredActivity.this, "com.ai.ai_gen.fileprovider", tempFile);
        intent.putExtra(MediaStore.EXTRA_OUTPUT, contentUri);
    } else {
        intent.putExtra(MediaStore.EXTRA_OUTPUT, Uri.fromFile(tempFile));
    }
    startActivityForResult(intent, TAKE_PHOTO);
}
```

打开相册，读取图片实现：

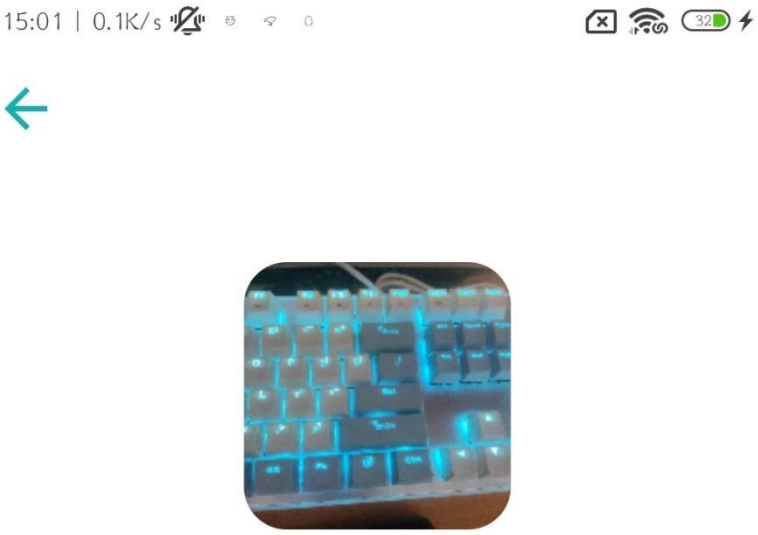
```
private void openAlbum() {
        Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
        intent.setType("image/*");
        startActivityForResult(intent, SELECT_PHOTO);
}
```

图表 15 更换图片

在拍照后，或者读取图片后，会返回一个 ActivityResult，这里根据返回的状态码来判断具体是什么操作。并且对返回结果进行处理，保存图片到数据库中，这里将图片转为 Base64 编码进行保存，以便后续的读取使用。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    switch (requestCode) {
        case TAKE_PHOTO:
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N) {
                Uri contentUri = FileProvider.getUriForFile(RegisteredActivity.this,
"com.ai.ai_gen.fileprovider", tempFile);
                cropPhoto(contentUri);
            } else { cropPhoto(Uri.fromFile(tempFile));    }
            break;
        case SELECT_PHOTO:
            if (resultCode == RESULT_OK) {
                Uri uri = intent.getData();
                if (uri != null)
                    cropPhoto(uri); }
            break;
        case CROP_REQUEST_CODE:
            if (resultCode == RESULT_OK) {
                Uri croppedImageUri = intent.getData();
                if (croppedImageUri != null) {
                    try {
                        Bitmap bitmap = CameraUtils.getBitmapFromUri(this, croppedImageUri);
                        avatar = BitmapUtils.bitmapToBase64(bitmap);
                        iv_avatar.setImageBitmap(bitmap);
                    } catch (IOException e) {
                        showMsg("无法加载图片");    } }    }
            break;
        default:
            break;}}
```

实现效果：



图表 16 头像选择实现

3.2.2 主页实现

在登录成功后，MainActivity 中，tablayout+viewPager 对页面进行处理，实现对应的四个 Fragment。这里需要使用 Adapter 对页面进行分配。

对于底部 Tab 的设置：

```
public View getView(int position) {
        View view = View.inflate(MainActivity.this, R.layout.main_tab_item, null);
        ImageView img = view.findViewById(R.id.img);
        TextView tv = view.findViewById(R.id.tv);
        if (tablayout.getTabAt(position).isSelected()) {
            img.setImageResource(onSele[position]);
        } else {
            img.setImageResource(unSele[position]);
        }
        tv.setText(titles[position]);
        tv.setTextColor(tablayout.getTabTextColors());
        return view;
    }
}
```

对 Tab 对应的页面的展示：这里使用 getSupportFragmentManager 对多个

Fragment 进行管理

```
MainTabAdapter mainTabAdapter = new MainTabAdapter(getSupportFragmentManager());
viewPager.setAdapter(mainTabAdapter);
tablayout.setupWithViewPager(viewPager);

for (int i = 0; i < tablayout.getTabCount(); i++) {
    TabLayout.Tab tab = tablayout.getTabAt(i);
    if (tab != null) {
        tab.setCustomView(mainTabAdapter.getView(i));
    }
}
```

图表 17 底部导航栏

首页 HomeFragment 中，首先初始化服务页面，服务界面请求 API 获取数据来实现填充，API 返回的数据如：

```
{
    "id": 1,
    "serviceName": "励志奋斗",
    "imgUrl": "/profile/lizhi.png",
    "link": "service/lizhi"
},
```

通过访问 API，将获得服务的 list，使用 Adapter 进行分配。

```
private void getServiceData() {
    Request request = new Request.Builder()
            .url(APIConfig.BASE_URL + "/service/service/list")
            .build();
    try {
        Call call = client.newCall(request);
        call.enqueue(new Callback() {
            @Override
            public void onFailure(Call call, IOException e) {
                Log.i("onFailure", e.getMessage());
            }
            @Override
            public void onResponse(Call call, Response response) throws IOException {
                if (response.isSuccessful()) {
                    final String result = response.body().string();
                    getActivity().runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            Gson gson = new Gson();
                            ServiceBean serviceBean = gson.fromJson(result, ServiceBean.class);
                            Message msg = new Message();
                            msg.what = 0;
                            msg.obj = serviceBean;
                            handler.sendMessage(msg); } });} } });
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

使用 RecyclerView.Adapter，对数据的每一项单独分配，对图片资源使用 Glide 进行加载

```
holder.service_name.setText(rowsBeans.get(position).getServiceName());
String url = APIConfig.IMAGE_URL + rowsBeans.get(position).getImgUrl();
Glide.with(context).load(url).into(holder.service_img);
```

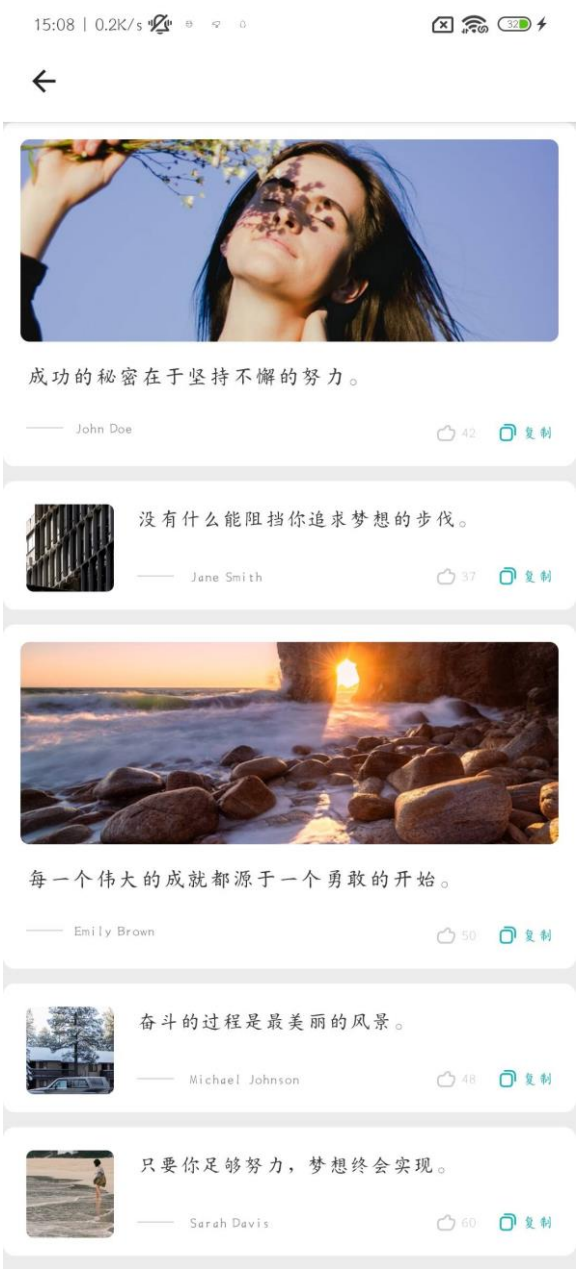
图表 18 服务列表

同时设置点击事件，在点击后，可以跳转到相应的界面中

```
public void onItemClick(View view, int position) {
    String url = APIConfig.BASE_URL + "/" + rowsBeanList.get(position).getLink();
    Intent intent = null;
    if (position != 10) {
        intent = new Intent(getActivity(), ServiceViewActivity.class);
    } else {
        intent = new Intent(getActivity(), ServiceAllActivity.class);
    }
    Bundle bundle = new Bundle();
    bundle.putString("title", rowsBeanList.get(position).getServiceName());
    bundle.putString("url", url);
    intent.putExtras(bundle);
    getActivity().startActivity(intent);
}
```

多次使用 recyclerview+ viewPager 进行页面的美化。以灵感创意界面为例：将每个标题对应的界面加入到 list 中，以便后续调用，同时初始化界面

```
tj_titleList.add(tj_List[6]);
pinpaiFragment pinpaifragment = new pinpaiFragment(getContext(), tj_idList[6]);
tj_fragmentsList.add(pinpaifragment);
add_tj_TitleLayout(tj_titleList.get(6), tj_idList[6]);
```



图表 19 文案列表

初始化 Tablayout 的标题

```
final TextView textView = (TextView) getLayoutInflater().inflate(R.layout.tj_horizontal_title, null);
textView.setText(title);
textView.setTag(position);
textView.setOnClickListener(new posOnClickListener());
LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
params.leftMargin = 25;
params.rightMargin = dip2px(getActivity(), mTitleMargin) * 4;
tj_titleLayout.addView(textView, params);
tj_textViewList.add(textView);
```

图表 20 灵感创意板块

3.2.3 工具界面实现

工具界面分为 AI 创作工具和其他工具

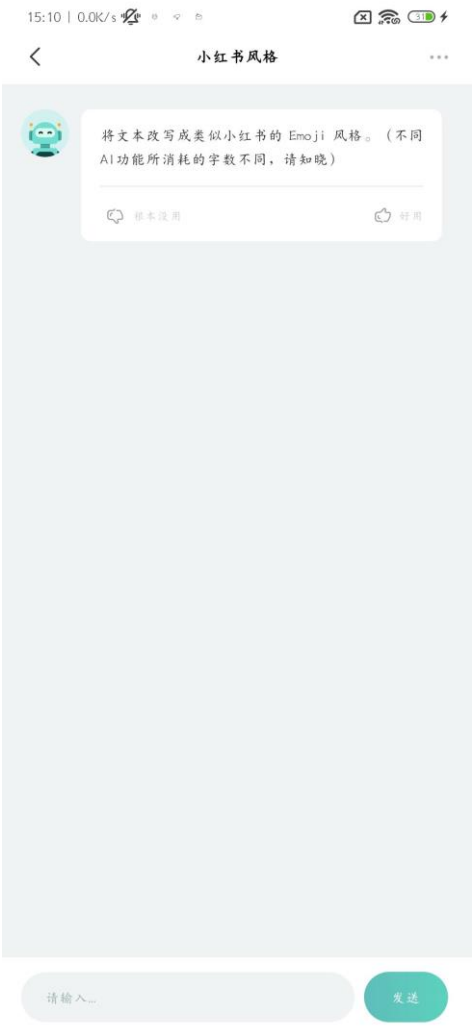
AI 创作工具界面使用与首页相同的布局方式，设置界面的代码：

```
gj_Adapter = new HomeViewPagerAdapter(getChildFragmentManager());
gj_titleList.add(gj_List[0]);
AimycollectionFragment colllectionfragment = new AimycollectionFragment(getActivity(), gj_idList[0]);
gj_fragmentsList.add(colllectionfragment);
add_gj_TitleLayout(gj_titleList.get(0), gj_idList[0]);
for (int i = 1; i < gj_List.length; i++) {
    gj_titleList.add(gj_List[i]);
    AiutilsFragment fragment = new AiutilsFragment(getActivity(), gj_idList[i]);
    gj_fragmentsList.add(fragment);
    add_gj_TitleLayout(gj_titleList.get(i), gj_idList[i]);
}
// 如果不设置，可能第三个页面以后就显示不出来了，因为offset 就是默认值1 了
gj_Adapter.setData(gj_fragmentsList);
gj_viewPager.setAdapter(gj_Adapter);
```

图表 21 AI 工具板块

在点击每一个 Item 后，会跳转到对应的 AI 聊天界面，在该界面使用 RecyclerView 来显示与 Ai 的对话。通过控制 Adapter 中的 type 来控制 AI 的消息框或者自己的消息框的展示或隐藏，依次实现相互聊天的展示效果。

```
if(conlist.getRows().get(positon).gettype()==0){
    holder.cl_user.setVisibility(View.GONE);
    holder.tv_gpt_answer.setText(conlist.getRows().get(positon).getContent());
}else {
    holder.cl_gpt.setVisibility(View.GONE);
    holder.tv_user_ask.setText(conlist.getRows().get(positon).getContent());
}
```



图表 22 AI 工具详情

在用户点击发送按钮后，获取输入框的信息，将消息封装到一个 conversionBean 中，在将该数据发送到 Adapter 以展示。

```
public void onClick(View v) {
    // 获取输入内容
    String ask = et.getText().toString();
    ConversionBean.RowsBean con_info = new ConversionBean.RowsBean(ask, 1);
    conversionBean.rows.add(con_info);
    con_num += 1;
    conversionBean.setTotal(con_num);
    // 更新 RecyclerView 适配器
    aiUtilComAdapter = new AiUtilComAdapter(AiUtil_comActivity.this, conversionBean);
    recyclerView.setAdapter(aiUtilComAdapter);
    // 清空输入框
    et.setText("");
    // 显示加载动画
    ll_loading.setVisibility(View.VISIBLE);
    // 创建 Handler 实现 1 秒延迟
    new Handler().postDelayed(new Runnable() {
        @Override
        public void run() {
            // 隐藏加载动画
            ll_loading.setVisibility(View.GONE);
        }
    }, 1000); // 1 秒延迟
}
```

图表 23 AI 聊天

其他工具界面的实现：以其中一个CardView为例，设置不同工具的界面。

```
<GridLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:columnCount="2"
    android:paddingTop="5dp"
    android:rowCount="5">
    <androidx.cardview.widget.CardView
        android:id="@+id/util_spqsy"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_row="0"
        android:layout_column="0"
        android:layout_gravity="fill|center"
        android:layout_marginEnd="10dp"
        android:layout_marginBottom="15dp"
        app:cardCornerRadius="10dp"
        app:contentPaddingBottom="15dp"
        app:contentPaddingLeft="10dp"
        app:contentPaddingRight="10dp"
        app:contentPaddingTop="10dp">
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="vertical">
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:text="视频去水印"
                    android:textColor="@color/textblackcolor"
                    android:textSize="18sp"
                    android:textStyle="bold" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:paddingTop="8dp"
                    android:text="一键去除视频水印"
                    android:textColor="@color/textgreycolor"
                    android:textSize="13sp" />
            </LinearLayout>
```

实现效果：

图表 24 其他工具布局

3.2.4 社区界面实现

社区界面使用 MainActivity 的方法 CustomViewpager+HorizontalScrollView，实现动态数据的展示。

```
com_Adapter = new HomeViewPagerAdapter(getChildFragmentManager());
com_titleList.add(com_List[0]);
CominfoFragment comfragment = new CominfoFragment(getActivity(), com_idList[0]);
com_fragmentsList.add(comfragment);
add_com_TitleLayout(com_titleList.get(0), com_idList[0]);
    com_titleList.add(com_List[1]);
CominfoFragment fragment = new CominfoFragment(getActivity(), com_idList[1]);
    com_fragmentsList.add(fragment);
    add_com_TitleLayout(com_titleList.get(1), com_idList[1]);
```

对页面的变化监听，对不同的页面进行不同的显示效果。

```
@Override
public void onPageSelected(final int position) {
    // 切换到当前页面，重置高度
    com_viewPager.requestLayout();
    com_textViewList.get(com_currentPos).setTextColor(Color.rgb(94, 94, 94));
    com_textViewList.get(com_currentPos).setTypeface(null, Typeface.NORMAL);
    com_textViewList.get(com_currentPos).setTextSize(14);
    com_textViewList.get(position).setTextColor(Color.rgb(20, 178, 177));
    com_textViewList.get(position).setTextSize(18);
    com_textViewList.get(position).setTypeface(null, Typeface.BOLD);
    com_currentPos = position;
    com_scrollView.scrollTo(moveToList.get(position), 0);
}
```

同时，与 AI 创作界面一样，对每个展示的 item 进行点击事件的监听，当点击后，会跳转到对应的界面。

```
final CommunityBean e = mList;
if (mItemClickListener != null) {
    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            mItemClickListener.onItemClick(view,position);
            Intent intent = new Intent(context, CommunityActivity.class);
            Bundle bundle = new Bundle();
            bundle.putString("name", mList.getRows().get(position).getName());
            bundle.putString("content",    mList.getRows().get(position).getContent());
            bundle.putString("img",    mList.getRows().get(position).getImgUrl());
            bundle.putString("like",    mList.getRows().get(position).getlikenum());
            bundle.putString("collection",    mList.getRows().get(position).getColection());
            intent.putExtras(bundle);
            context.startActivity(intent);
        });}
```

图表 25 社区界面

图表 26 评论详情

3.2.5 我的界面实现

在我的页面使用多种布局方式，实现页面布局的美化，并对每个 Button 进行监听，在点击后，可以跳转到对应的界面中。

在个人编辑界面，再次与数据库进行交互，进入页面时，首先读取数据库中存在的 UserInfo 的用户信息，并且将该信息进行赋值到界面 UI 上，如果读取的信息不存在，则先使用 User 中存在的用户信息，进行初始化一个用户的信息。代码实现：

```
mDBOpenHelper=new DBOpenHelper(mContext);
user=mDBOpenHelper.getAllLoggedInUsers().get(0);
userinfo=mDBOpenHelper.getUserInfoByPhoneNum(user.getPhonenum());
if(userinfo==null){
   mDBOpenHelper.upsertUserInfo(user.getPhonenum(),user.getName(),user.getAvatar(),"","","");
}
userinfo=mDBOpenHelper.getUserInfoByPhoneNum(user.getPhonenum());
if(userinfo.getAvatar()!=null){
      Bitmap iv_photo= BitmapUtils.base64ToBitmap(userinfo.getAvatar());
      Glide.with(mContext).load(iv_photo).into(iv_user_icon);
}else {
      iv_user_icon.setImageResource(R.mipmap.ic_mine_avatar);
}
avatar=userinfo.getAvatar();
tv_username.setText(userinfo.getName());
tv_sex.setText(userinfo.getSex());
tv_work.setText(userinfo.getWork());
et_qm.setText(userinfo.getQm());
tv_save.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
            String name=tv_username.getText().toString();
            String sex=tv_sex.getText().toString();
            String work=tv_work.getText().toString();
            String qm=et_qm.getText().toString();
            mDBOpenHelper.upsertUserInfo(user.getPhonenum(),name,avatar,work,qm,sex);
            mDBOpenHelper.updateUserInfo(user.getPhonenum(),name,avatar);
            finish();
      }
});
```

在更新用户信息后，返回到 UserFragment 中，fragment 中的信息也需要随之更新，因此使用 onResume 方法，在视图显示后，更新视图 UI。

```
    @Override
    public void     onResume() {
        super.onResume();
        initInfo();
    }
public void initInfo(){
    mDBOpenHelper=new DBOpenHelper(context);
    user=mDBOpenHelper.getAllLoggedInUsers().get(0);
    iv_username.setText(user.getName());
    String advatar=user.getAvatar();
    if(advatar!=null){
        Bitmap bitmap=BitmapUtils.base64ToBitmap(advatar);
        if(bitmap!=null){
            iv_advatar.setImageBitmap(bitmap);
        }
    }
}
```

图表 27 编辑资料界面

3.3 关键技术和难点

3.3.1 我的页面

在 HomeFragment 中，要实现横向滑动标题栏，并且点击标题栏来显示不同的界面的效果，由于传统的 TabLayout 只能在界面中全部显示，当存在很多项时，就无法完整显示，因此滑动效果十分必要。为解决此问题，使用了 HorizontalScrollView+ CustomViewpager

HorizontalScrollView 负责实现一个横向的滑动的条目，自定义的 Viewpager 则实现界面的展示，通过将二者关联，以此实现了点击一个 Tabitem，则展示一个对应的页面，当 Tabitem 较多时，则会通过滑动，来实现展示更多的条目。

完整代码实现：

```
private void init_tj_page() {
    tj_fragmentsList = new ArrayList<>();
    tj_titleList = new ArrayList<>();
    tj_textViewList = new ArrayList<>();
    moveToList = new ArrayList<>();
    tj_Adapter = new HomeViewPagerAdapter(getChildFragmentManager());
    tj_titleList.add(tj_List[0]);
    wenanFragment fragment = new wenanFragment(getActivity(), tj_idList[0]);
    tj_fragmentsList.add(fragment);
    add_tj_TitleLayout(tj_titleList.get(0), tj_idList[0]);
                                                    |
                                                    |
    tj_titleList.add(tj_List[5]);
    guanggaoFragment guanggaofragment = new guanggaoFragment(getContext(), tj_idList[5]);
    tj_fragmentsList.add(guanggaofragment);
    add_tj_TitleLayout(tj_titleList.get(5), tj_idList[5]);
    tj_titleList.add(tj_List[6]);
    pinpaiFragment pinpaifragment = new pinpaiFragment(getContext(), tj_idList[6]);
    tj_fragmentsList.add(pinpaifragment);
    add_tj_TitleLayout(tj_titleList.get(6), tj_idList[6]);
    // 如果不设置，可能第三个页面以后就显示不出来了，因为 offset 就是默认值 1 了
    tj_Adapter.setData(tj_fragmentsList);
    tj_viewPager.setAdapter(tj_Adapter);
    tj_viewPager.setOffscreenPageLimit(9);
    tj_textViewList.get(0).setTextColor(Color.rgb(255, 255, 255));
    tj_textViewList.get(0).setBackgroundResource(R.drawable.tj_text_back);
    tj_currentPos = 0;
```

视图界面添加监听事件，当界面发生变化，这改变导航栏样式

```
tj_viewPager.addOnPageChangeListener(new ViewPager.OnPageChangeListener() {
        @Override
        public void onPageScrolled(int position, float positionOffset, int positionOffsetPixels) {
        }
        @Override
        public void onPageSelected(final int position) {
            // 切换到当前页面，重置高度
            tj_viewPager.requestLayout();
            tj_textViewList.get(tj_currentPos).setTextColor(Color.rgb(94, 94, 94));
            tj_textViewList.get(tj_currentPos).setBackgroundResource(0);
            tj_textViewList.get(position).setTextColor(Color.rgb(255, 255, 255));
            tj_textViewList.get(position).setBackgroundResource(R.drawable.tj_text_back);
            tj_currentPos = position;
            tj_scrollView.scrollTo((int) moveToList.get(position), 0);
        }
        @Override
        public void onPageScrollStateChanged(int state) {
        }
    });
}
```

导航栏初始化，设置导航栏数据到 Adaptar，实现数据与视图的绑定。

```
private void add_tj_TitleLayout(String title, int position) {
   final TextView textView = (TextView) getLayoutInflater().inflate(R.layout.tj_horizontal_title, null);
   textView.setText(title);
   textView.setTag(position);
   textView.setOnClickListener(new posOnClickListener());
   LinearLayout.LayoutParams params = new
LinearLayout.LayoutParams(LinearLayout.LayoutParams.WRAP_CONTENT,
LinearLayout.LayoutParams.WRAP_CONTENT);
   params.leftMargin = 25;
   params.rightMargin = dip2px(getActivity(), mTitleMargin) * 4;
   tj_titleLayout.addView(textView, params);
   tj_textViewList.add(textView);
   int width;
   if (position == 0) {
      width = 0;
      moveToList.add(width);
   } else {
      int w = View.MeasureSpec.makeMeasureSpec(0, View.MeasureSpec.UNSPECIFIED);
      int h = View.MeasureSpec.makeMeasureSpec(0, View.MeasureSpec.UNSPECIFIED);
      tj_textViewList.get(position - 1).measure(w, h);
      width = tj_textViewList.get(position - 1).getMeasuredWidth() + mTitleMargin * 8;
      moveToList.add(width + moveToList.get(moveToList.size() - 1));
   }
}
```

为导航栏添加对应的事件监听，当导航栏 iteem 被点击时，则改变导航栏样式

```
class posOnClickListener implements View.OnClickListener {
    @Override
    public void onClick(View view) {
        if ((int) view.getTag() == tj_currentPos) {
            return;
        }
        tj_textViewList.get(tj_currentPos).setTextColor(Color.rgb(94, 94, 94));
        tj_textViewList.get(tj_currentPos).setBackgroundResource(0);
        tj_currentPos = (int) view.getTag();
        tj_viewPager.setCurrentItem(tj_currentPos);
        tj_viewPager.requestLayout();
    }
}
```

在这里也需要使用 adapter 进行页面的设置，由于代码量过大问题，这里不在展示。

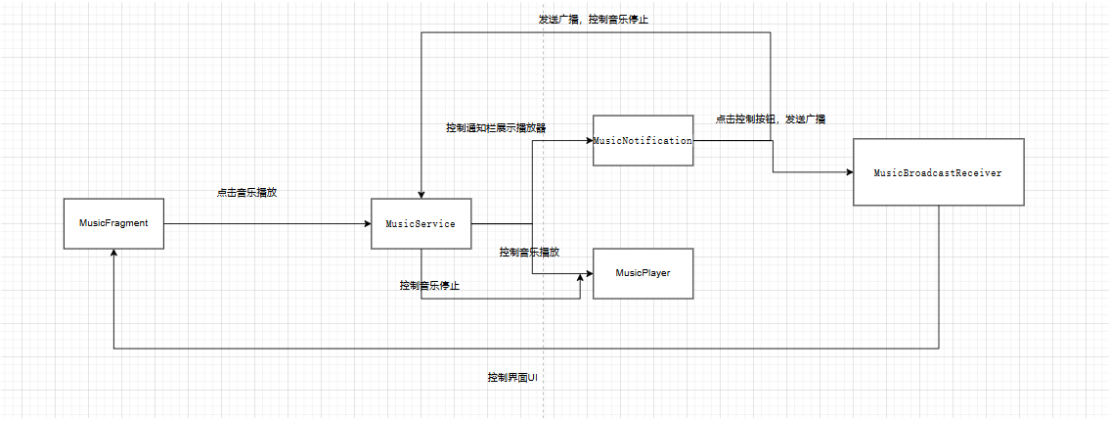
图表 28 实现效果

图表 29 广告视图

3.3.2 音乐播放器实现

音乐播放器实现过程中，需要实现通知栏的同步展示，因此这里使用了自定义的 MusicService，以及 Boardcast 的广播，还有 Notiaication 的通知栏。由于音乐的点击需要传递点击位置和数据，因此将原本定义在 Fragment 中的点击实现逻辑，下移到了 Adapter 中，在点击后，会将点击的数据使用 Parcelable

进行封装后，利用 Intent 进行传输到 MusicService 中。与此同时，应该调用通知栏的展示，但是由于后续需要对通知栏进行点击的监听与控制，因此将原定与 Adapter 的 notification 控制，下移到了 Service 中，在 Service 中，对通知栏进行控制。并且在通知栏中点击后，对应的 Fragment 的显示也需要变化，因此这里另定义了一个 BoardcastReceiver，在这里接受通知栏发送的广播，根据广播的内容，来控制 Fragment 中的显示。



图表 30 实现逻辑流程图

主要实现代码：

初始化 Service：开启 MUSIC 服务

```
@SuppressLint("WrongConstant")
public void getMusicModelList(MusicViewBean models) {
    Intent intent = new Intent(this.getActivity(), MusicService.class);
    intent.putExtra(MUSIC_INTENT_KEY, models);
    intent.addFlags(MUSIC_INTENT_FLAG);
    mContext.startService(intent);
}
```

Adapter 发送广播到 MusicService

```
public void onClick(View view) {
    mItemClickListener.onItemClick(view, position);
    musicIntent.putExtra(MAIN_MUSIC_INTENT_KEY, position);
    context.sendBroadcast(musicIntent);
    holder.start_button.setImageResource(R.drawable.stop_music);
    notifyItemChanged(lastSelectedPosition); // 新增这一行
    lastSelectedPosition = position; // 新增这一行
}
```

MusicService 接收，并控制播放

```
private void mainToService(Intent intent) {
    if (MAIN_MUSIC_INTENT_FLAG == flag) {
        position = intent.getIntExtra(MAIN_MUSIC_INTENT_KEY, -1);
        if (position > -1) {
            musicNotifi.onCreateMusicNotifi();
            // 播放
            if (musics != null) {
                mm = musics.getRows().get(position);
            } else {
                showToast("MUSICS IS NULL");
            }
            if (mm != null) {
                play(mm.getSong_url());
            }
```

播放，并设置通知栏

```
public void play(String path) {
    if (mp.isPlaying()) {
        //如果再次获得的 url 与播放的相同则暂停
        if (TextUtils.equals(mp.getMusicpath(), path)) {
            mp.pause();
            musicNotifi.updateMusicNotification(mm, true);
        } else {//否则播放新音乐
            mp.stop();
            mp.playOnOne(path);
            musicNotifi.updateMusicNotification(mm, false);
        }
    } else {
        mp.playOnOne(path);
        musicNotifi.updateMusicNotification(mm, false);
    }
    sendModelToMusicActivity();
}
* musicNotification 来的控制
 */
private void musicNotificationService(int k) {
    switch (k) {
        case 30001:
            // 播放
            play(mm.getSong_url());
            break;
        case 30003:
            break;}}
```

MusicBoardcastReceiver 更新界面 UI

```
private void handleMusicServiceIntent(Intent intent) {
    isPlaying = intent.getBooleanExtra(MUSIC_SERVICE_TO_ACTIVITY_ISPLAY, true);
    currentMusic = intent.getParcelableExtra(MUSIC_SERVICE_TO_ACTIVITY_MODEL);
    if (currentMusic != null && TextUtils.equals(currentMusic.getMusicname(),
musicViewHolder.music_title.getText())) {
        updatePlayButton(isPlaying);
    } else {
        resetPlayButton();
    }
}
```

实现效果：

图表 31 音乐播放器与通知联动

第四章 实现效果

4.1 用户登录注册界面

用户登录界面实现了注册以及登录，如果没有账号，则需要注册。在登录成功后跳转到主界面

图表 32 登录注册界面

4.2 首页展示

实现了多个视图的嵌套展示，以及各种滑动展示效果，采用多种复杂视图结合实现。

图表 33 首页界面效果

4.3 音乐播放界面展示

使用音乐服务以及广播，同时实现状态栏通知。并与主界面 UI 绑定，实现通知与 UI 的同步。

图表 34 音乐播放效果

4.4 内容展示界面

采用 RecyclerView 与 adaptar，将后端数据动态绑定，增强可维护性以及可拓展性。

图表 35 文案列表及内容

4.5 视频播放界面

结合媒体播放，实现网络视频的播放。

图表 36 视频播放界面效果

4.6 AI 创作界面

结合多种复杂布局以及 Listview，实现界面的滑动、滚动效果

图表 37 工具界面效果图

4.7 社区与消息展示

数据动态绑定，可以轻松更换，并在评论界面也实现数据的绑定，使评论动态加载。

图表 38 社区界面效果图

4.8 个人展示与编辑资料

采用多种界面结合，并实现多个跳转，同时实现个人资料的编辑，结合数据库的操作，保存用户信息。

图表 39 我的界面效果图

第五章 总结

5.1 工作总结

通过本次课程设计，将学习的全部知识进行了应用，在产品设计过程中，对于不同的界面的设计会产生很多问题，尤其是在复杂布局的嵌套中，很容易出现位置偏移以及不符合预期的情况，在整个产品界面设计中，我应用最多的还是LinearLayout，对于动态数据展示的界面，Recycleview 的应用更加适合，虽然实现较为复杂，但是展示的效果也更加符合正常生产环境中的要求，只需要改变数据源的内容，即可更新界面的内容，而静态界面在修改中则较为痛苦。通过本次的设计，也遇到了很多的 bug，比如单独创建的 Activity，由于没有在 Manifeat 中声明，而导致的报错，有时候报错却显示的是其他界面为正常初始化，这一个问题第一次遇到时发费了不少时间去定位。本次产品的开发，基本实现了大部分功能，有些功能实在需要后端支撑，则只做了静态的界面。但是产品的界面展示效果已经符合预期。

5.2 遇到的问题与不足

本次开发中，遇到许多问题，面临许多 bug，比如音乐播放器的实现中，由于该过程涉及诸多组件的联动使用，因此在实现中，产生了许多无法定位的 bug，且由于代码量的增加，很多地方已经不敢再随意改动，切身体验到代码维护与规范的重要性。目前产品的不足，存在许多静态界面，没有做动态数据的适配，且由于许多功能依赖后端的实现，因此简单使用 API 无法满足需求，则只做了静态页面进行展示，而未实现具体的返回。