华南师范大学
SOUTH CHINA NORMAL UNIVERSITY

系统开发报告

报告题目：“得闲饮茶”——
一款专注于广府养生文化的APP

课　　程：移动智能开发

指导老师：曹　阳

小组成员：王璐瑶、黄芷晴
王小玥、陈宇权

学　　院：计算机学院

目录

一、产品方案设计

1、项目实施可行性报告

1.1行业市场分析

广东特殊的气候，导致了广东人特别注重养生以对身体进行调理，其中汤文化和凉茶文化是岭南养生文化的重要内容。而当今很多人，尤其是年轻一代，由于各种各样的原因，出现了总是晚睡早起和精神压力过大的现象，最终导致了个人总是无精打采的结果。喝汤和饮凉茶是调理身体、保持精力充沛的有效方法，所以介绍广东汤文化和凉茶文化等养生知识的软件在市场上仍是有需求的。

现在市场上的饮食养生软件一般具有以下功能：编辑个人特色食谱、按气候季节变化推荐膳食搭配、对养生知识进行科普等。根据以上分析，我们计划开发一款专注于广府养生文化，名为“得闲饮茶”的饮食养生软件。

1.2竞争对手分析

- **薄荷营养师**

  可让用户根据需求选择减肥、血糖管理等标签进行功能的个性化定制。

  可记录用户日常饮食。

  让用户自定特色食谱。

- **饭橘**

  可根据用户健康情况提供饮食建议。

  对10万种左右的食材进行热量、饮食功效等方面的科普。

  让用户自定特色食谱。

- **雅卓**

  可记录用户日常饮食。

  让用户自定特色食谱。

  食材库全面，包括中餐和西餐，能够找到全面的日常饮食数据。

1.3 自身条件分析

我们计划开发的软件名称是“得闲饮茶”，与市场上的其他饮食养生软件相比，专注于岭南养生文化是我们的特色，同时我们也会设计简洁的UI界面便于用户使用。

“得闲饮茶”采用Kotlin+SQLite+jetpack进行实现，具有技术可行性和经济可行性。

2、产品定位及目标

2.1产品定位

“得闲饮茶”是一款介绍岭南养生文化的饮食养生app，具备以下功能：

- 可以让用户自由查看个人特色的汤谱和凉茶配方
- 根据季节进行汤和凉茶等饮食方面的推荐，并可让用户进行收藏
- 可以让用户点击汤或凉茶，以查看相关的配方以及对应的养生科普知识

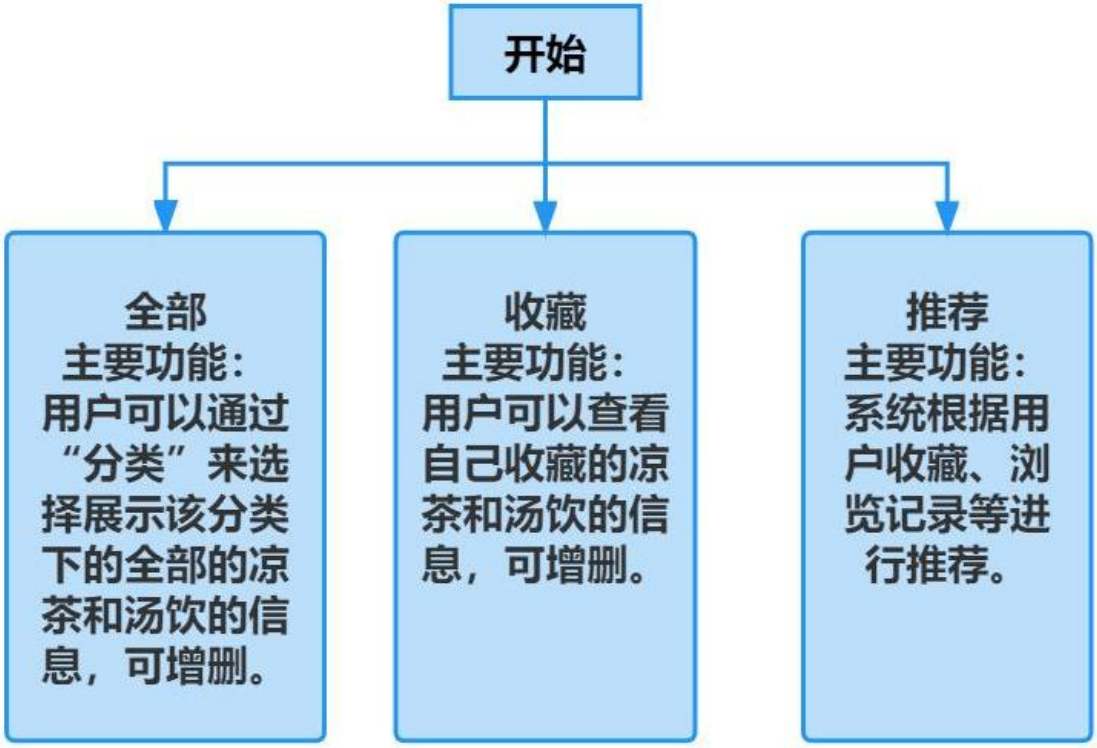
2.2 目标群体

“得闲饮茶”app致力于成为Android手机上具有特色的饮食养生app，专注于对岭南养生文化进行介绍，用户群体为对岭南养生文化感兴趣、希望以此对身体进行调理的用户。

3、产品内容策划

3.1应用流程规划

应用的主要流程规划如图1所示：



应用流程规划图

3.2设计与测试规范

主界面设计概要图，如图所示：

主界面设计概要图

测试规范：通过安卓开发环境模拟器使用多台虚拟设备进行测试，使用多台实体设备进行测试。

3.3开发日程表

开发日程表（初步）如表1所示：

表1 开发日程表（初步）

时间	项目

2023年4月份至2023年5月中旬	使用假数据完成各界面设计
2023年5月下旬至2023年6月中旬	完成后台开发
2023年6月下旬	测试

4、技术解决方案

本应用是智能手机上的养生软件，鉴于Android系统在国内的使用率最高，该产品基于Android开发，主要的开发环境是：JDK+Android Studio+Android SDK ，采用Kotlin+SQLite+jetpack进行实现。

5、推广方案

为充分吸引用户，本产品采用以下推广方案：

5.1前期宣传

前期在没有扎实用户的状况下，首先需要在内测阶段从团队成员身边发掘潜在用户，并发展内测用户，通过这部分用户的意见反馈来改善功能；尽量维持和这部分用户的联系，给予一定的福利和特权，使其发展成基础固定用户。

5.2产品推出阶段

5.2.1线上推广

- 基础上线：各大下载市场、应用商店、大平台、下载站的覆盖Android版本的发布渠道
- 应用商店：Google商店、小米商城、魅族商店、oppo应用商店、华为应用商店等
- 客户端：91手机助手、360软件管家

5.2.2线下推广

- 向身边的同学朋友推荐
- 采用免费用户通过邀请新用户注册可解锁高级功能的模式

5.2.3投放广告

- 信息流广告：分为社交信息流和新闻信息流，即：在腾讯的QQ空间和微信里展现一些广告；在浏览器+新闻头条的产品里展现一些广告
- 软文宣传：寻找影响力较大且风格符合的自媒体人，让专业写手为APP一到三篇专业软文，同时发布到APP上
- 行业APP广告直投：找出一些有流量，有影响力的相关行业网或是相关网站，在网站上直接投放广告

6、运营规划书

6.1产品功能

- 用户可以通过”分类”来选择展示该分类下的全部的凉茶和汤饮的信息；并且可以自由编辑个人特色的汤谱和凉茶配方
- 根据季节进行汤和凉茶等饮食方面的推荐，并可让用户进行收藏
- 可以让用户点击汤或凉茶，以查看相关的配方以及对应的养生科普知识

6.2产品发展

表2 运营规划表（初步）

时期	目标	关注数据
初期	稳定初期用户群体，产生基础流量并收集用户行为数据；保证应用的正常运行，与产品设计时的用户模型对比，有目的性调优	页面路径转化，按钮点击，启动次数，启动时间段，停留时长
中期	发展群体用户，产生持续的话题度和流量；定期进行应用的更新，具体从内容运营、活动策划、	新增，活跃，留存以及渠道数据

	用户运营、数据分析这几个方面着手实施	
后期	通过各种活动运营、增值服务创造营收；可实现广告投放盈利，开启会员制	付费用户数、付费金额、付费路径转化

二、产品实现方案

1、UI界面及系统功能描述

更加详细的 UI 设计，可以参考演示视频和截图和Github中的项目

1.1登录界面

有两个编辑框可供用户填写账号和密码，有两个按钮“登录”“注册”:若账号和密码正确，点击“登录”可进入APP中我的界面，若点击“注册”可跳转到注册界面进行注册。

1.2注册界面

有三个编辑框可供用户填写要进行注册的账号、密码和昵称，有两个按钮“注册”、“返回”：若账号密码不为空，点击“注册，将账号、密码和昵称写入数据库；若点击“返回”则返回登录界面。

登录、注册界面

1.3“得闲饮茶”首页

顶部是“得闲|饮茶”的logo，右侧为一个输入框。再往下是轮播图，用于展示茶饮、汤饮图片。在轮播图下方，APP为用户提供养生“汤饮推荐”和“茶饮推荐”。点击任意汤饮或茶饮后，用户可以进入详情页面。

“得闲饮茶”首页

1.4汤饮、茶饮详情页面

顶部是饮品图片，下面展示茶饮、汤饮的描述、口味、功效等详情内容。点击“<”按钮可以返回至首页。

汤饮、茶饮详情页

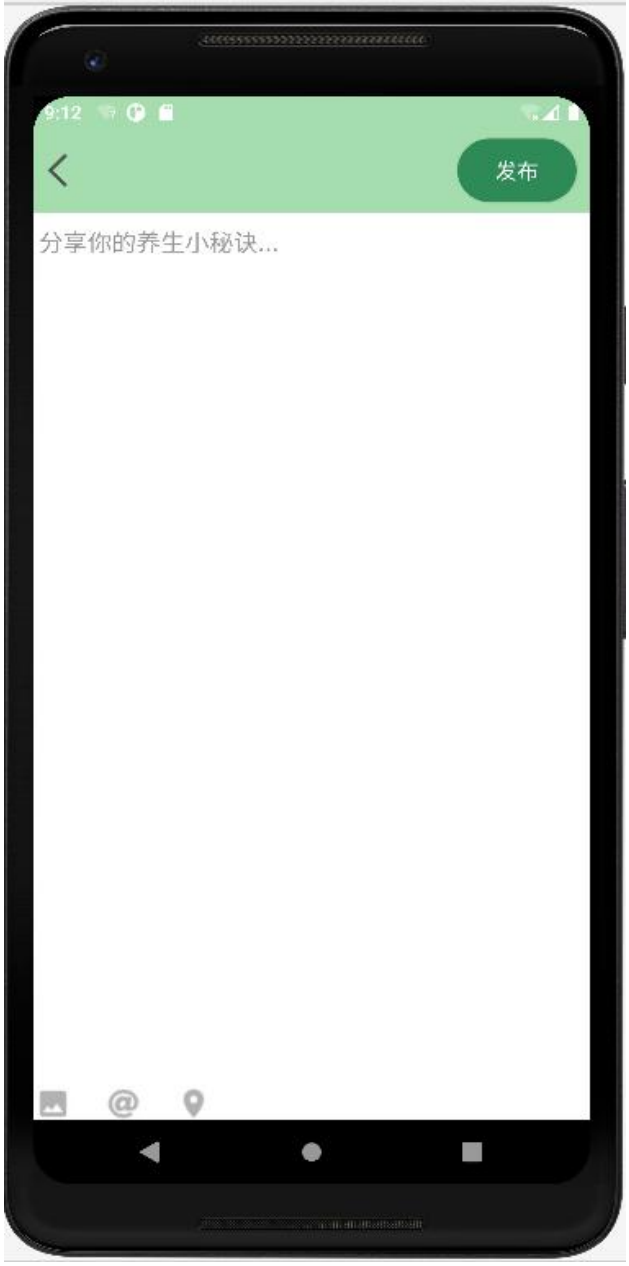
1.5社区页面

顶部有一个轮播图，用于展示与养生相关的图片。在轮播图下方，用户可以查看其他用户发布的动态，可以点击每一条动态下方的对应的按钮对动态进行点赞、收藏和评论。点击右下方的“+”号按钮后，用户可以进入发布动态页面。

社区页面UI设计

1.6发布动态页面

用户可以在输入框内输入想要分享的内容。动态编辑完成后，点击右上方的“发布”按钮发布动态。动态发布后自动回到社区页面，这时用户可以看到自己发布的动态已经显示了出来。如果想直接返回社区页面，可以点击左上方的“<”按钮返回社区页面。



发布动态页面UI界面设计

1.7社区页面的动态

点击评论按钮可以进入评论页面。在评论页面中，用户可以看到其他用户对该动态的评论，同时可以对这些评论进行点赞或者评论。用户可以在下方的输入框内输入自己的评论，点击右上方的“发布”按钮就可以发布评论。评论发布后，用户可以看到自己的评论已经显示了出来。如果想直接返回社区页面，可以点击左上方的“<”按钮返回社区页面。

评论页面UI设计

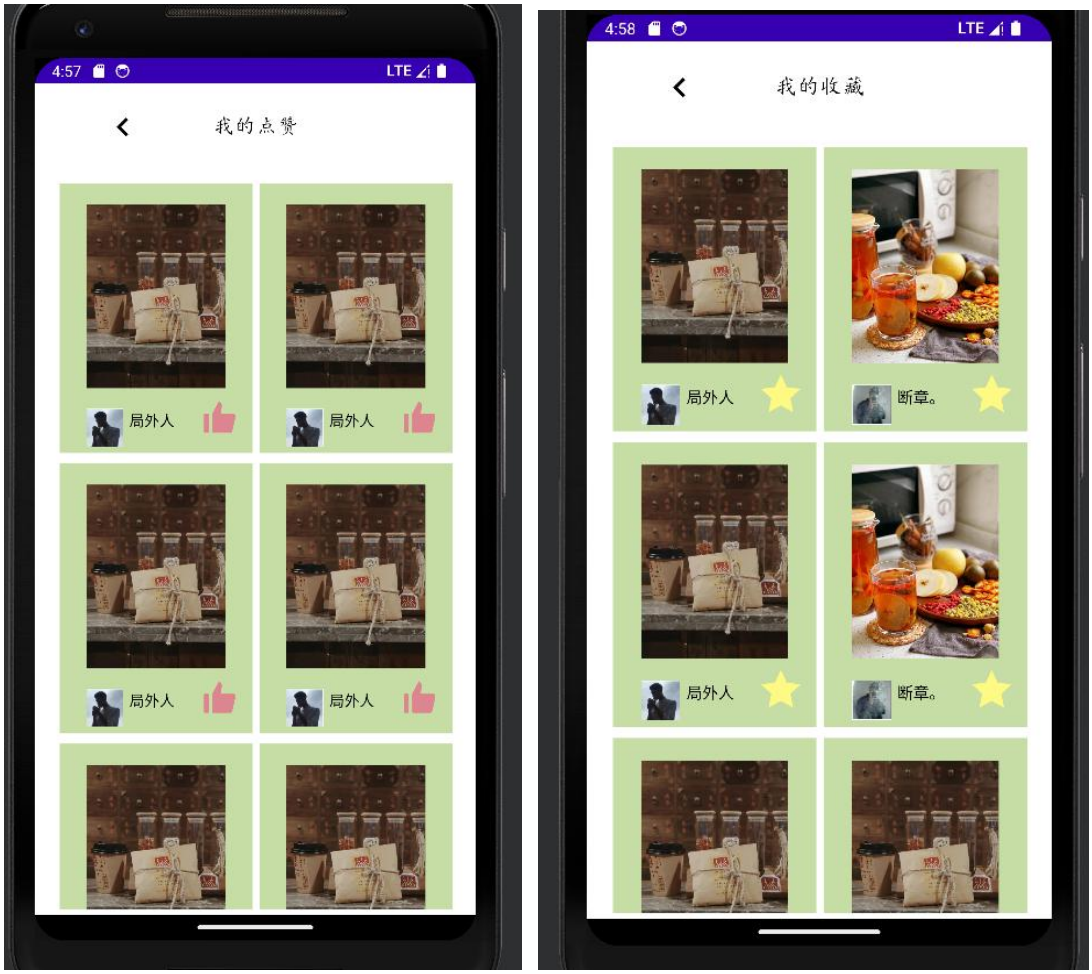
1.8我的界面

该界面主要是对用户个人信息进行展示和管理。展示了当前用户的头像、昵称、账号等信息。有四个按钮“我的收藏”“我的点赞”“我的口味”“退出登录”，点击按钮“我的收藏”，可跳转到我收藏动态的界面，对其进行管理。点击按钮“我的点赞”，可跳转到我点赞动态的界面，对其进行管理。点击按钮“我的口味”，可跳转到我的口味界面，对其进行管理；点击按钮“退出登录”可退出当前帐号。

我的界面

1.9点赞和收藏页面

使用网格布局呈现点赞和收藏的内容，总体布局为两列。网格中每一个卡片包含一张内容图片、一个用户头像图片、用户名以及用于取消点赞或收藏的按钮。



点赞和收藏页面UI设计

1.10弹窗呈现点赞或收藏的具体信息

在点赞和收藏页面中，单击展现简略信息的卡片就会出现一个弹窗用于呈现点赞或收藏内容的具体信息，包括用户头像、用户名、评论信息以及评论配图。

点赞收藏详细信息UI界面设计

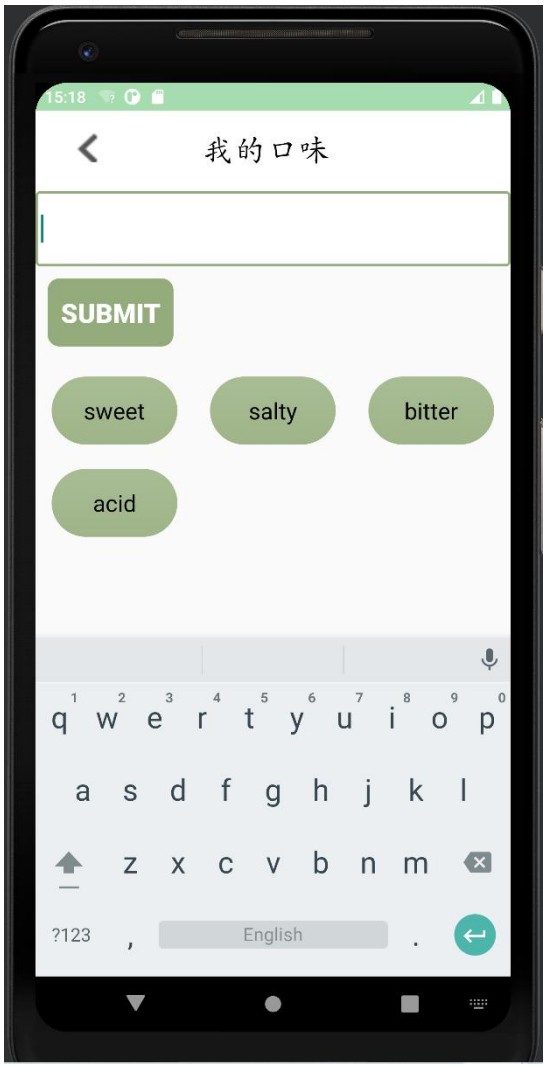
1.11点赞和收藏按钮

点击动态的按钮，在进行数据库操作的同时会让相应按钮改变颜色。

点赞和收藏动态的UI设计

1.12我的口味界面

该界面主要展示了当前用户已添加的口味，用户可在输入框中输入自己想要添加的口味，点击按钮“SUBMIT”进行提交，新口味将展示到界面中。用户也可长按某一项口味对其进行删除。



我的口味

2、关键技术与难点以及解决方案

2.1登录注册主界面：

2.1.1在UI设计方面：

（1）实现难点：自定义控件的Style：由于Android Studio中自带的控件样式很少，所以为了使UI界面看起来更加美观自然，我们需要自定义控件的样式。

（2）解决方案：对编辑框和按钮都新建Drawble Resource文件重新设置了它们的框内填充颜色、边框宽度和颜色、圆角半径、各边倒角大小。

（3）核心代码：如下代码是对登录界面编辑框样式的新定义：

```
<?xml version="1.0" encoding="utf-8"?>
<!-- shape定义形状，shape="rectangle"表示形状为长方形 -->
<shape
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:shape="rectangle" >
   <!-- 设置框内填充颜色 -->
   <solid android:color="@color/gray" />
   <!-- 设置边框宽度和颜色 -->
   <stroke
       android:width="1dip"
       android:color="@color/transparent" />
   <!-- 设置圆角半径 -->
   <corners android:radius="3dp" />
   <!-- 设置边距 -->
   <padding
       android:bottom="5dp"
       android:left="5dp"
       android:right="5dp"
       android:top="5dp" />
   <!-- 设置渐变角度angle和渐变颜色 -->
   <gradient
       android:angle="270"
       android:endColor="#00000000"
       android:startColor="#DDECE4" />
   <!-- 设置各边倒角大小 -->
   <corners
```

```
        android:bottomLeftRadius="200dp"
        android:bottomRightRadius="200dp"
        android:topLeftRadius="200dp"
        android:topRightRadius="200dp" />
</shape>
```

（1）实现难点：变换文字字体

（2）解决方案：对于文字的字体，使用了楷体，这个需要自己去字体官网下载相应的字体文件，将文件放入到这个路径中：.\app\src\main\assets\fonts，并在对应的.java文件中调用setTypeface方法设置字体样式。

（3）核心代码如下：

```
Typeface typeFace =Typeface.createFromAsset(getAssets(),"fonts/SanJiKaiShu-2.ttf");
taste_label=findViewById(R.id.taste_label);
taste_label.setTypeface(typeFace);
```

2.1.2在功能实现方面：

（1）实现难点： SQLite数据库的使用，主要是在注册时将用户的账号、密码、昵称存储到数据库中；在登录时将用户输入的账号和密码在数据库中进行比对，若有相同项则登录成功，若没有则登录失败。

（2）解决方案：使用Android Studio中的SQLite数据库，需要定义一个数据库帮助类，在这个类中创建数据库和表，并定义一系列对数据库进行增删查改的函数。

2.2首页搜索框

（1）实现难点：点击搜索框时出现下拉列表，并可以在搜索框中输入文字，实现查询，在数据库中使用模糊查询。

（2）解决方案：整个布局使用的是线性布局，搜索框又是一个线性布局（里面包含一个相对布局和一个TextView,相对布局里面有一个EditText，下面是一个ListView;搜索框其实就是一个EditText，背景是用shape自己画出来的。在Activity中给EditText设置一个监听，当输入文字的时候获取输入的内容然后查询数据库，将查询到的数据展示到ListView中；CharSequence s是EditText中的文本内容，判断如

果s长度为0隐藏ListView内容，否则显示；查询数据库获得Cursor获得CursorAdapter将内容展示到ListView中。

```
private void initView() {
       mTextView = (TextView) findViewById(R.id.textview);
       mEditText = (EditText) findViewById(R.id.edittext);
       mImageView = (ImageView) findViewById(R.id.imageview);
       mListView = (ListView) findViewById(R.id.listview);

       //设置点击事件
      View.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View v) {
              //把EditText内容设置为空
              mEditText.setText("");
              //把ListView隐藏
              mListView.setVisibility(View.GONE);
          }
       });
       //EditText添加监听
       mEditText.addTextChangedListener(new TextWatcher() {
          public void beforeTextChanged(CharSequence s, int start, int
count, int after) {}//文本改变之前执行
          @Override
          //文本改变的时候执行
          public void onTextChanged(CharSequence s, int start, int before,
int count) {
              //如果长度为0
              if (s.length() == 0) {
                  //隐藏
```

```
                View.setVisibility(View.GONE);
            } else {//长度不为0
                //显示ListView
                showListView();}
        }

        public void afterTextChanged(Editable s) { }//文本改变之后执行
    });
    mTextView.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            //如果输入框内容为空，提示请输入搜索内容
   if(TextUtils.isEmpty(mEditText.getText().toString().trim())){
       ToastUtils.showToast(context,"请输入您要搜索的内容");}
else {
                //判断cursor是否为空
                if (cursor != null) {
                    int columnCount = cursor.getCount();
                    if (columnCount == 0) {
                        ToastUtils.showToast(context, "没有搜索的内容");
                    } }
            }   }
    });
  }

  private void showListView() {
    mListView.setVisibility(View.VISIBLE);
    //获得输入的内容
    String str = mEditText.getText().toString().trim();
    //获取数据库对象
```

```
        MyOpenHelper myOpenHelper = new
MyOpenHelper(getApplicationContext());

        SQLiteDatabase db = myOpenHelper.getReadableDatabase();

        //得到cursor

        cursor = db.rawQuery("select * from lol where name like '%" +

str + "%'", null);

        MyListViewCursorAdapter adapter = new
MyListViewCursorAdapter(context, cursor);

        mListView.setAdapter(adapter);

        mListView.setOnItemClickListener(new
AdapterView.OnItemClickListener() {

            @Override


public void onItemClick(AdapterView<?> parent, View view, int position,
long id) {

                //把cursor移动到指定行

                cursor.moveToPosition(position);

                String name = cursor.getString(cursor.getColumnIndex("name"));

                ToastUtils.showToast(context, name); }

        });

    }

}
```

2.3首页轮播图

（1）实现难点：要实现轮播图样式为圆角；图片可以无限循环左右滑动切换，页可以自动切换；手触摸上去时停止切换。

（2）解决方案：banner_radius: 设置轮播图的圆角；banner_loop_time: 设置轮播间隔时间，默认3000；Banner框架的适配器BannerImageAdapter，可以直接使用；设置自动循环播放Banner.isAutoLoop(true)，true为自动播放；设置指示器Banner.setIndicator(new CircleIndicator(this))；设置指示器选中时的颜色(即选中时小点的颜色)Banner.setIndicatorSelectedColor(Color.

GREEN)；设置指示器之间的距离Banner.setIndicatorSpace(int)。

（3）核心代码：

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


       initData();
       banner = findViewById(R.id.main_banner);
       banner.setAdapter(new BannerImageAdapter<Integer>(banner_data) {


           @Override
           public void onBindView(BannerImageHolder holder, Integer data,
int position, int size) {
               holder.imageView.setImageResource(data); }
       });


       // 开启循环轮播
       banner.isAutoLoop(true);
       banner.setIndicator(new CircleIndicator(this));
       banner.setScrollBarFadeDuration(1000);
       // 设置指示器颜色(TODO 即选中时那个小点的颜色)
       banner.setIndicatorSelectedColor(Color.GREEN);
       // 开始轮播
       banner.start();
   }
```

2.4首页图片列表

（1）实现难点：汤饮、茶饮的图片以横向列表的形式展现，并且可以左右滑动。点击图片可以跳转到详情页。

（2）解决方案：建立.xml文件，通过HorizontalScrollView类嵌套一个GridView来实现并将其封装，再定义一个经典的adapter并绑定来调控水平滑动方法。之后只要修改横向item的大小尺寸以及item的布局即可在汤饮展示和茶饮展示中重复使用。

（3）核心代码：

```
//绑定GridView布局
Private GridView weekDateGridView=(GridView)findViewById
(R.id.gridViewHorizontal);
//构建数据源
private List<String> weekDateData=new List<String>();
for (int i = 6; i < 14; i++) {
          String S=i+2+":00";
          weekDateData.add(S);
      }
//定义适配器
private WeekDateHorizontalAdatper horizontalAdatper=
new WeekDateHorizontalAdatper (context, weekDateData);
//调用控制水平滚动的方法
setHorizontalGridView(weekDateData.size(), weekDateGridView);
weekDateGridView.setAdapter(horizontalAdatper);
//绑定适配器
weekDateGridView.setAdapter(horizontalAdatper);


   // 水平滚动的GridView的控制
   private void setHorizontalGridView(int siz, GridView gridView) {
       int size = siz;
//     int length = (int) getActivity().getResources().getDimension(
//          R.dimen.coreCourseWidth);
       int length=60;
       DisplayMetrics dm = new DisplayMetrics();
```

```
        this.getActivity().getWindowManager().getDefaultDisplay()
                .getMetrics(dm);
        float density = dm.density;
        int gridviewWidth = (int) (size * (length) * density);
        int itemWidth = (int) ((length) * density);

        @SuppressWarnings("deprecation")
        LinearLayout.LayoutParams params = new LinearLayout.LayoutParams
(gridviewWidth, LinearLayout.LayoutParams.FILL_PARENT);
        gridView.setLayoutParams(params);
// 设置GirdView布局参数,横向布局的关键
        gridView.setColumnWidth(itemWidth); // 设置列表项宽
        gridView.setHorizontalSpacing(0); // 设置列表项水平间距
        gridView.setStretchMode(GridView.NO_STRETCH);
        gridView.setNumColumns(size); // 设置列数量=列表集合数
    }
```

2.5详情页数据

（1）实现难点：存储各个汤饮、茶饮图片对应的详情内容；并将点击的图片与其详情内容对应并且可以实现搜索查找。

（2）解决方法：首先建立一个与数据库存取操作相关的DetailDB类，主要实现在SQLite中建表、插入数据、查询所有数据等功能，可以将详情页的相关内容存储到数据库的搜索表当中。然后在首页中点击图片时获取图片地址的第一个字符，按照字符搜索相关内容。

2.6社区页面轮播图

（1）实现难点：要实现轮播图的无限循环左右滑动切换，图片可以自动切换，手触摸上去时停止切换。

（2）解决方案：使用了ViewPager实现，将ImgeView放进ViewPger这个控件里，并通过给ViewPager写适配器来设置图片，同时使用句柄的postDelayed方法设置切换的间隔时间为2000ms（即2s），通过设置currentItem来实现自动切换。通过onAttachedToWindow和onDetachedFromWindow两个方法来实现手触摸轮播图时停止切换。

（3）核心代码：

轮播图的适配器PagerAdapter内部类，实现将轮播图的数据绑定到控件中，并且实现无限左右切换：

```
//适配器写成内部类
private PagerAdapter mpagerAdapter=new PagerAdapter() {
    @Override
    public int getCount() {
        //实现无限左右滑动
        return Integer.MAX_VALUE;}

    @Override
    public boolean isViewFromObject(@NonNull View view, @NonNull
Object object) {
        return view==object;}

    @NonNull
    @Override
    public Object instantiateItem(@NonNull ViewGroup container, int
position) {
        //初始化:把ImageView放进父控件container里
        //载入布局
        Viewitem=LayoutInflater.from(container.getContext()).inflate
(R.layout.item_pager,container,false);
        ImageView iv=item.findViewById(R.id.cover);
```

```
        //设置数据
        int realPosition=position % mData.size();
        iv.setImageResource(mData.get(realPosition));
        if(iv.getParent() instanceof ViewGroup){
            ((ViewGroup) iv.getParent()).removeView(iv);
        }
        container.addView(iv);
        return iv;
    }

    @Override
    public void destroyItem(@NonNull ViewGroup container, int
position, @NonNull Object object) {
        //销毁:从container里删除ImageView
        container.removeView((View) object);
    }
};
```

实现自动切换和触摸时切换停止功能的SobViewPager类：

```
public class SobViewPager extends ViewPager {

    private static final String TAG = "SobViewPager";
    private Handler handler;

    public SobViewPager(@NonNull Context context) {
        //统一构造方法的入口，跳转到第二个构造方法
        this(context,null);
    }

    public SobViewPager(@NonNull Context context, @Nullable AttributeSet
```

```
attrs) {
        super(context, attrs);
        //使用句柄设置切换时间
        handler = new Handler(Looper.getMainLooper()); }

    @Override
    protected void onAttachedToWindow() {
        super.onAttachedToWindow();
        Log.d(TAG,"onAttachedToWindow...");
        startLooper(); }

    private void startLooper() {
        //设置切换时间
        handler.post(mTask);}
    private Runnable mTask=new Runnable() {
        @Override
        public void run() {
            //自动切换
            int currentItem=getCurrentItem();
            currentItem++;
            setCurrentItem(currentItem);
            postDelayed(this,2000);}
    };

    @Override
    protected void onDetachedFromWindow() {
        super.onDetachedFromWindow();
        Log.d(TAG,"onDetachedFromWindow...");
        stopLooper();}
```

```
    private void stopLooper() {
        //停止自动切换
        handler.removeCallbacks(mTask);
    }
}
```

2.7对社区动态列表项的设计

（1）实现难点：如图所示，社区的每一条动态都由用户头像、昵称、发布动态的日期、动态的内容、两张图片以及收藏、点赞和评论三个按钮组成。组件的布局十分复杂，且逻辑功能上还要分别实现点击收藏按钮切换按钮颜色并加入到我的收藏、点击点赞按钮切换按钮颜色并加入到我的点赞和点击评论按钮切换到评论页面的功能。

每一条动态的效果

如下图所示，评论列表的每一项跟动态列表类似，只是少了两张图片和一个收藏按钮，总体功能和布局变化不大。

每一条评论的效果

（2）解决方案：使用RecyclerView实现列表，建立community_list_item.xml文件，对列表的每一项进行布局设计，总体采用线性布局，按钮采用ImageButton。为了方便对头像进行裁剪，专门设计了CricleImage类，用于实现将图片裁剪成圆形并

实现缩放效果。同时，通过给ImageButton设置selector来实现点击收藏/点赞按钮切换按钮颜色。

（3）核心代码：

community_list_item.xml布局文件，实现对每一个列表项的布局：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    ...
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <!--头像图片，这个CricleImageView是另写的一个类继承自ImageView，用来实现圆形的图片效果和缩放效果的-->
        <com.example.community.CircleImage
            android:id="@+id/comment_item_head"
            android:layout_width="40dp"
            android:layout_height="40dp"
            android:layout_gravity="center"
            android:layout_margin="8dp"
            android:scaleType="centerCrop"
            android:src="@drawable/cm_tx1" />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">
            <!--昵称区域-->
            <TextView
                android:id="@+id/comment_item_username"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:layout_marginRight="17dp"
                android:text="柒柒"
                android:textColor="#333333"
                android:textSize="15sp" />
            <!--创建时间-->
            <TextView
                android:id="@+id/comment_item_createdtime"
                android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:layout_marginRight="17dp"
            android:text="2023年4月15日"
            android:textColor="#999999"
            android:textSize="12sp" />
        </LinearLayout>
    </LinearLayout>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">
        <!--内容区域-->
        <TextView
            android:id="@+id/comment_item_context"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginLeft="53dp"
            android:layout_marginRight="50dp"
            android:text="看起来很好喝哦！"
            android:textColor="#333"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="right"
        android:orientation="horizontal"
        android:layout_marginLeft="50dp"
        android:layout_marginBottom="5dp">
        <!--评论区域-->
        ...
        <!--点赞区域-->
       ...

    </LinearLayout>
</LinearLayout>
```

评论列表的每一项跟动态列表类似，只是少了两张图片和一个收藏按钮，总体布局变化不大，因此这里不展示相关代码。实现图片裁剪成圆形并缩放的工具类CircleImage，继承自ImageView类。

实现点击点赞按钮切换颜色的dz_btn_pressed.xml资源文件：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_selected="true" android:drawable="@drawable/dz2"/>
    <item android:state_selected="false" android:drawable="@drawable/dz"/>
</selector>
```

同时，要在适配器HomeAdapter的onBindViewHolder方法（这个方法时为了获取数据并显示到社区页面对应的控件中）中为点赞按钮设置对应的点击事件：

```
holder.btn_dz.setOnClickListener(new View.OnClickListener() {
    boolean b=true;
    @Override
    public void onClick(View v) {
        //为点赞按钮设置点击切换颜色的事件
        if(!b){
            v.setSelected(false);
            b=true;
        }
        else{
            v.setSelected(true);
        }
    }
});
```

收藏按钮的实现同上。

2.8列表的动态增加操作：

（1）实现难点：发布动态功能的实现，要将发布页面记录下来的数据转换成对应的列表中的一项，并在社区页面展示出来。同时要将创建时间地记录下来，并显示在列表项中。由于Activity的生命周期，页面跳转时数据不会被记录下来，因此需要设计存储相关数据的表，并存储到数据库中。发布评论的功能同理。

（2）解决方案：使用RecyclerView的LinearLayoutManager 来实现线性布局，即将各个项排列在一维列表中来实现。同时，为“发布”按钮设置点击事件，只要用户一点击按钮，就将发布页面中的数据存储到Android Studio的SDK包自带的SQLite数据库中。同时，为RecyclerView新建一个item，并将数据库中的数据绑定到对应的

控件上，再利用Activity的onResume方法，对列表进行刷新，这样就能实现列表的动态增加功能。

（3）核心代码：社区页面的列表项的实体类，主要展示类的成员（定义了列表项的数据类型），get和set方法不再展示：

```
public class CommunityItem implements Serializable {
    //社区页面列表item的定义
    private String createdTime;//创建时间
    private int imgs1;//图片1路径
    private int imgs2;//图片2路径
    private String mContext;//动态内容
    private int headIcons;//头像
    private String usernames;//昵称

    private String id;
    ……}
```

社区页面的RecyclerView的适配器类HomeAdapter：

```
public class HomeAdapter extends
RecyclerView.Adapter<HomeAdapter.MyViewHolder>{

    private List<CommunityItem> itemList;
    private LayoutInflater layoutInflater;
    private Context context;
    private CommuItemDB cmdb;
    public HomeAdapter(Context context, List<CommunityItem> itemList){
        this.itemList=itemList;
        this.context=context;
        layoutInflater=LayoutInflater.from(context);
    }
    @NonNull
    @Override
    //加载布局文件并返回MyViewHolder对象
    public MyViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
int viewType) {
        //创建view对象
```

```
        View view =
layoutInflater.inflate(R.layout.coummunity_list_item,parent,false);
        //创建MyViewHolder对象
        MyViewHolder myViewHolder=new MyViewHolder(view);
        return myViewHolder;
    }
    @Override
    //获取数据并显示到对应控件
    public void onBindViewHolder(@NonNull MyViewHolder holder,
@SuppressLint("RecyclerView") int position) {
        //给控件获取一下数据，注意不同类型调用不同的方法，设置图片用
setImageResource（），设置文字用setText（）
        CommunityItem item=itemList.get(position);
        holder.createdtime.setText(item.getCreatedTime());
        holder.img1.setImageResource(item.getImgs1());
        holder.img2.setImageResource(item.getImgs2());
        holder.content.setText(item.getmContext());
        holder.head.setImageResource(item.getHeadIcons());
        holder.username.setText(item.getUsernames());
        holder.btn_dz.setOnClickListener(new View.OnClickListener() {
            boolean b=true;
            @Override
            public void onClick(View v) {
                //为点赞按钮设置点击切换颜色的事件
                if(!b){
                    v.setSelected(false);
                    b=true;
                }
                else{
                    v.setSelected(true);
                }
            }
        });
        holder.btn_sc.setOnClickListener(new View.OnClickListener() {
            boolean b=true;
            @Override
            public void onClick(View v) {
                //为收藏按钮设置点击切换颜色的事件
                if(!b){
                    v.setSelected(false);
                    b=true;
                }
```

```
                else{
                    v.setSelected(true);
                }
            }
        });
        holder.btn_pl.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //点击评论按钮跳转到评论页面
                Intent intent = new Intent(context,
CommentActivity.class);
                context.startActivity(intent);
            }
        });
    }

    @Override
    public int getItemCount() {
        //获取列表条目总数
        return itemList.size();
    }
    public void refreshData(List<CommunityItem> itemlist){
        //刷新列表数据
        this.itemList=itemlist;
        //通知ViewHolder数据刷新了
        notifyDataSetChanged();
    }

    class MyViewHolder extends RecyclerView.ViewHolder{
        //初始化控件
        ImageView img1,img2,head;
        TextView createdtime,content,username;

        ImageButton btn_dz,btn_sc,btn_pl;


        public MyViewHolder(@NonNull View itemView) {
            super(itemView);
            //绑定控件，使适配器能够找到对应的控件

createdtime=itemView.findViewById(R.id.community_item_createdtime);
```

```
            img1=itemView.findViewById(R.id.community_item_img1);
            img2=itemView.findViewById(R.id.community_item_img2);
            content=itemView.findViewById(R.id.community_item_context);
            head=itemView.findViewById(R.id.community_item_head);
            username=itemView.findViewById(R.id.community_item_username);
            btn_dz=itemView.findViewById(R.id.btn_dz);
            btn_sc=itemView.findViewById(R.id.btn_sc);
            btn_pl=itemView.findViewById(R.id.btn_pl);
        }
    }

}
```

与数据库存取操作相关的CommuItemDB类，主要实现在SQLite中建表、插入数据、查询所有数据等功能：

```
public class CommuItemDB extends SQLiteOpenHelper {
    //存时间和内容
    private static final String DB_NAME="noteSQLite.db";
    public static final String TABLE_NAME = "community";
    public static final String ID = "id";//id
    public static final String TIME = "time";//时间
    public static final String CONTENT = "content";//内容
    public CommuItemDB(Context context){
        super(context,DB_NAME,null,1);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        //建表
        db.execSQL("CREATE TABLE " + TABLE_NAME + " (" + ID
                + " INTEGER PRIMARY KEY AUTOINCREMENT,"
                + " TEXT ," + TIME
                + " TEXT ," + CONTENT
                + " TEXT )");
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int
newVersion) {}
```

```
public long insertData(String time,String content) {
    //将数据插入到数据库中(增)
    SQLiteDatabase db = getWritableDatabase();

    ContentValues values = new ContentValues();
    values.put("time",time);
    values.put("content",content);
    return db.insert(TABLE_NAME, null, values);
}
public List<CommunityItem> queryAllFromDb() {
    //从数据库中查找全部内容
    SQLiteDatabase db = getWritableDatabase();
    List<CommunityItem> itemList = new ArrayList<>();
    //要展示的对应item的数据
    List<String> createdTime=new ArrayList<>();
    createdTime.add("2023年4月13日");
    createdTime.add("2023年4月15日");
    List<Integer> imgs1=new ArrayList<>();
    imgs1.add(R.drawable.dt1);
    imgs1.add(R.drawable.dt2);
    List<Integer> imgs2=new ArrayList<>();
    imgs2.add(R.drawable.dt4);
    imgs2.add(R.drawable.dt3);
    List<String> contents=new ArrayList<>();
    contents.add("打卡一家街边凉茶铺。");
    contents.add("自制下火凉茶——\n润喉罗汉果胎菊水!");
    List<Integer> headsIcon=new ArrayList<>();
    headsIcon.add(R.drawable.tx1);
    headsIcon.add(R.drawable.tx2);
    List<String> usernames=new ArrayList<>();
    usernames.add("局外人");
    usernames.add("断章。");
    CommunityItem item1=new CommunityItem();
    item1.setCreatedTime(createdTime.get(0));
    ...
    CommunityItem item2=new CommunityItem();
    item2.setCreatedTime(createdTime.get(1));
    ...
    itemList.add(item2);
    Cursor cursor = db.query(TABLE_NAME, null, null, null, null, null,
```

```
null);
       if (cursor != null) {
           while (cursor.moveToNext()) {
               @SuppressLint("Range") String time =
cursor.getString(cursor.getColumnIndex("time"));
               @SuppressLint("Range") String content =
cursor.getString(cursor.getColumnIndex("content"));
               createdTime.add(time);
               contents.add(content);
               CommunityItem item=new CommunityItem();
               item.setCreatedTime(time);
               ...
           }
           cursor.close();
       }
       return itemList;
   }
}
```

“发布”按钮的点击事件，一点击就将数据写进数据库中：

```
//发布动态
public void msubmit(View view) {
   //发布动态
   String time=getCurrentTimeFormat();
   String content=edtext.getText().toString();
   //将数据保存到数据库中
   long row=cmdb.insertData(time,content);
   if(row!=-1){
       Toast.makeText(AddCoummnityActivity.this,"发布成功
",Toast.LENGTH_SHORT).show();
       //自动回到上一个页面
       this.finish();
   }else{
       Toast.makeText(AddCoummnityActivity.this,"发布失败
",Toast.LENGTH_SHORT).show();
   }
}
```

同时，在社区页面中还要添加初始化控件、绑定数据、渲染列表、刷新数据等功能的方法，这里列出部分方法，具体代码可见CommunityActivity.java文件：

```
@Override
protected void onPostResume() {
    //由于app的生命周期的缘故，要在onResume中再执行一遍查找所有数据的操作来进行刷新
    super.onPostResume();
    refreshDataFromDb();
}
private void refreshDataFromDb() {
    //拿到新的数据
    itemList=getDataFromDB();
    //通知适配器列表已刷新
    homeAdapter.refreshData(itemList);
}
private List<CommunityItem> getDataFromDB() {
    return cmdb.queryAllFromDb();
}

private void initData() {
    //准备数据
    //轮播图
    mData.add(R.drawable.banner5);
    mData.add(R.drawable.banner2);
    mData.add(R.drawable.banner3);
    mData.add(R.drawable.banner4);
    mData.add(R.drawable.banner1);
    //数据准备完后进行更新
    mpagerAdapter.notifyDataSetChanged();
    //设置中间位置
    mviewPager.setCurrentItem(Integer.MAX_VALUE / 2 + 1);
    //列表
    itemList=new ArrayList<>();
    cmdb=new CommuItemDB(this);
}
private void initView(){
    //初始化视图
    //绑定控件
    mviewPager=this.findViewById(R.id.view_pager);
    //设置适配器
    mviewPager.setAdapter(mpagerAdapter);
```

```
    //绑定RecycleView
    recyclerView = findViewById(R.id.community_item);
    //设置为表格布局
    recyclerView.setLayoutManager(new LinearLayoutManager(this));
    recyclerView.addItemDecoration(new space_item(space));//给recycleView
添加item的间距
}

private void initEvent() {
    //渲染列表
    homeAdapter=new HomeAdapter(this,itemList);
    recyclerView.setAdapter(homeAdapter);
}
```

2.9悬浮按钮的设计及功能实现

（1）实现难点：如下图所示，要实现圆形的悬浮按钮，按钮的颜色是墨绿色，中心的“+”号图案是白色的。同时按钮的位置是不变的，即便用户滑动列表，按钮也是固定在界面的右下方。

（2）解决方案：悬浮按钮使用FloatingActionButton来实现，按钮的填充色通过back groundTint属性来设置，中心的“+”号通过src来引用系统自带的“+”号图片，并通过tint属性来设置颜色。由于FloatingActionButton只能被放在相对布局Relati veLayout中，因此在布局时要充分考虑其他控件的位置以及可以滚动的Scroll View的区域。

（3）核心代码：社区页面的布局文件activity_community.xml：

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

    <com.example.community.views.SobViewPager
        android:id="@+id/view_pager"
        android:layout_width="match_parent"
        android:layout_height="120dp" />

    <ScrollView
        android:layout_width="match_parent"
```

```
        android:layout_height="wrap_content">
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="120dp"
            android:layout_marginBottom="5dp"
            android:background="#f2f2f2">
            <androidx.recyclerview.widget.RecyclerView
                android:id="@+id/community_item"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginBottom="5dp"/>
        </RelativeLayout>
    </ScrollView>

    <com.google.android.material.floatingactionbutton.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="73dp"
        android:onClick="add"
        android:src="@android:drawable/ic_input_add"
        app:backgroundTint="#387633"
        app:elevation="0dp"
        app:tint="@color/white"
        tools:ignore="SpeakableTextPresentCheck" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="15dp"
        android:layout_alignParentBottom="true"
        android:layout_centerHorizontal="true"
        android:background="@color/white">

        <ImageButton
            android:id="@+id/first"
            android:layout_width="57dp"
            android:layout_height="58dp"
```

```
            android:layout_marginLeft="60dp"
            android:background="@drawable/ic_first"
            android:onClick="first" />

        <ImageButton
            android:id="@+id/community"
            android:layout_width="51dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="70dp"
            android:background="@drawable/commu"
            android:onClick="community" />

        <ImageButton
            android:id="@+id/my"
            android:layout_width="41dp"
            android:layout_height="match_parent"
            android:layout_marginLeft="70dp"
            android:background="@drawable/commu_my"
            android:onClick="my" />

    </LinearLayout>

</RelativeLayout>
```

2.10我的界面

2.10.1在UI设计方面：

（1）实现难点：我的界面在UI设计方面实现较为繁琐，该页面所含元素过多，布局较为复杂，色彩的使用较为丰富，颜色的选取需要不断地对比设计图，在各控件布局方面需要进行多次的参数调整，才能使得界面被充分利用且美观协调。

（2）解决方案：在布局方式上我仍然选择了较为熟悉的嵌套线性布局，其中需要不断地调整各控件之间的大小比例和间隔距离，在颜色选取方面我借助了网上的RGB取色工具，直接获得设计图中每个颜色的编码，将编码应用到页面设计中。其UI设计图和控件布局图如下：

如上图所示，主要采用了ImageView、TextView、Button、ImageButton这四种控件，其中ImageView和ImageButton这两种空间的背景均填充了相应的图片，这些图片应事先保存到.\app\src\main\res\drawable文件夹中。

2.10.2在功能实现方面：

（1）实现难点：如何将用户的账号和昵称从登录界面传递到本界面并显示。

（2）解决方案：使用到了SharedPreference轻量级的存储辅助类，在用户登录成功后，将用户的账号和昵称以键值对的形式进行保存在以xml形式的文件中，当跳转到我的界面后，将数据从文件中读取出来，并显示到界面中相应的文本框中。由于其需要操作的代码重复性比较多，所以在项目中把它封装成一个工具类PrefercesService，使用SharedPreferences存储少量数据时比使用数据库更加方便高效且页面间数据传递不是临时的。

2.11在点赞和收藏页面单击预览卡片后显示动态的具体信息：

（1）实现难点：如果使用activity之间的跳转做为实现呈现动态具体信息的方案，会增加页面之间的跳转，不仅影响用户的使用体验，用于呈现动态预览卡片的网格视图也会反复加载数据，进而影响性能。

（2）解决方案：使用了PopupWindow控件，将要呈现的动态的详细信息用一个弹窗进行实现，在方便用户查看点赞收藏的详细动态信息，提升用户体验的同时提升系统性能。

（3）核心代码：

创建PopupWindow对应的布局文件popup_window.xml：

```
<?xml version="1.0" encoding="utf-8"?>

<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"

  ...

    <ImageView

        android:id="@+id/winLogo"

        android:layout_width="79dp"

        android:layout_height="79dp"

        app:srcCompat="@mipmap/headscu"

        android:layout_marginTop="20dp"

        android:layout_marginLeft="20dp"/>

    <TextView

        android:id="@+id/winUsername"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:text="断章"

        android:textColor="@color/black"

        android:layout_marginTop="45dp"

        android:layout_marginLeft="130dp"

        android:textSize="20dp"/>

    ...

    <TextView
```

```
        android:id="@+id/winContent"

        android:textSize="16dp"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:textColor="@color/black"

        android:text="自制下火凉茶——\n润喉罗汉果胎菊水+竹蔗茅根水！"

        android:layout_marginTop="110dp"

        android:layout_marginLeft="30dp"/>

</RelativeLayout>
```

创建PopupWindow对象同时获取要呈现的数据：

```
public void onItemClick(AdapterView<?> adapterView, View view, int
i, long l) {

            //获取数据

            Collection collection = mCollectionList.get(i);

            View inflate =
LayoutInflater.from(context).inflate(R.layout.popup_window,null);

            //创建弹窗对像

            PopupWindow popupWindow = new PopupWindow(view,

                    ViewGroup.LayoutParams.WRAP_CONTENT,

                    ViewGroup.LayoutParams.WRAP_CONTENT,true);

            //对应弹窗和布局

            popupWindow.setContentView(inflate);

            //设置弹窗位置

            popupWindow.setTouchable(true);

            popupWindow.setFocusable(false);

           popupWindow.showAtLocation(inflate,Gravity.CENTER,0,0);
```

```
            popupWindow.showAsDropDown(inflate,0,0);

            popupWindow.setOutsideTouchable(false);

            //设置展现数据

            ImageView logo = (ImageView)
                inflate.findViewById(R.id.winLogo);

            ImageView image =
                (ImageView)inflate.findViewById(R.id.winImage);

            TextView username = (TextView)
                    inflate.findViewById(R.id.winUsername);

            TextView content =
                  (TextView)inflate.findViewById(R.id.winContent);

            logo.setImageBitmap(collection.getLogo());

            image.setImageBitmap(collection.getImage());

            username.setText(collection.getUsername());

            content.setText(collection.getContent());

            //设置布局内点击事件

            ImageView button =
inflate.findViewById(R.id.winCloseBtn);

            button.setOnClickListener(new View.OnClickListener() {

                @Override

                public void onClick(View view) {

                    popupWindow.dismiss();}

            });

        }
```

2.12从数据库动态获取点赞和收藏的内容

（1）实现难点：由于每次运行app时点赞和收藏的内容只是存储在内存中，退出app后再次进行app就会丢失掉点赞的收藏的内容。

（2）解决方案：使用数据库做为数据持久化的解决方案，同时数据库也为呈现点赞和收藏内容的网格视图提供数据源。关键是写好数据库相关的工具类，为点赞收藏相关的一系列功能提供相应的接口。

（3）核心代码：数据库工具类CollectionDBOpenHelper.java：

```
/**
 * 收藏和点赞增删改查工具类
 */
public class CollectionDBOpenHelper extends SQLiteOpenHelper {
    //数据库名称
    private static final String DB_NAME = "collection.db";
    //表名
    //收藏的表
    public static final String TABLE_NAME_COLLECTION =
"collection";
    //点赞的表
    public static final String TABLE_NAME_LIKE = "myLike";
    //建表sql语句
    private static final String create_table_sql_collection =
        "create table if not exists " + TABLE_NAME_COLLECTION + "(id
integer primary key autoincrement,username text,content text,logo
blob,image blob)";
    private static final String create_table_sql_like =
            "create table if not exists " + TABLE_NAME_LIKE + "(id
integer primary key autoincrement,username text,content text,logo
blob,image blob)";

    public CollectionDBOpenHelper(Context context){
        super(context,DB_NAME,null,1);
    }


    @Override
    public void onCreate(SQLiteDatabase sqLiteDatabase) {
        sqLiteDatabase.execSQL(create_table_sql_collection);
        sqLiteDatabase.execSQL(create_table_sql_like);
    }

    @Override
    public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int
i1) {}

    /**
     * 将动态添加收藏或点赞
     * @param collection 收藏对象
     * @param tableName 表名
     * @return 返回-1表示插入失败
     */
    public long insert(Collection collection,String tableName){
        SQLiteDatabase db = getWritableDatabase();
        ContentValues values = new ContentValues();
        //装数据
        values.put("username",collection.getUsername());
        values.put("content",collection.getContent());
        ByteArrayOutputStream os1 = new ByteArrayOutputStream();
        collection.getLogo().compress(Bitmap.CompressFormat.PNG,
100, os1);
        values.put("logo",os1.toByteArray());

        ByteArrayOutputStream os2 = new ByteArrayOutputStream();
```

```
        collection.getImage().compress(Bitmap.CompressFormat.PNG,
100, os2);
        values.put("image",os2.toByteArray());

        return db.insert(tableName,null,values);
    }

    /**
     * 取消收藏
     * @param id 收藏或点赞编号
     * @param tableName 表名
     * @return 返回0表示删除该列失败
     */
    public int deleteById(Integer id,String tableName){
        SQLiteDatabase db = getWritableDatabase();
        String[] strArr = {id.toString()};
        return db.delete(tableName,"id = ?",strArr);
    }

    /**
     * 查询所有收藏或点赞记录
     * @param tableName 表名
     * @return 收藏或点赞数据
     */
    @SuppressLint("Range")
    public List<Collection> queryAll(String tableName){
        SQLiteDatabase db = getWritableDatabase();
        List<Collection> list = new ArrayList<>();
        Cursor cursor =
db.query(tableName,null,null,null,null,null,null);
        if(cursor!=null){
            while (cursor.moveToNext()){
                //(id integer primary key autoincrement,username
text,content text,logo blob,image blob)
                int id = cursor.getInt(cursor.getColumnIndex("id"));
                String username =
cursor.getString(cursor.getColumnIndex("username"));
                String content =
cursor.getString(cursor.getColumnIndex("content"));
                //图片处理
                byte[] logoBlob =
cursor.getBlob(cursor.getColumnIndex("logo"));
                Bitmap logoBmp =
BitmapFactory.decodeByteArray(logoBlob, 0, logoBlob.length);
                byte[] imageBlob =
cursor.getBlob(cursor.getColumnIndex("image"));
                Bitmap imageBmp =
BitmapFactory.decodeByteArray(imageBlob, 0, imageBlob.length);
                //封装数据
                Collection collection = new
Collection(id,username,content,logoBmp,imageBmp);
                list.add(collection);
            }
            cursor.close();
        }
        return list;
    }

    /**
     * 清空表中数据
     * @param tableName 表名
     */
```

```
    public void deleteAll(String tableName){
        SQLiteDatabase db = getWritableDatabase();
        db.delete(tableName,null,null);
    }
}
```

2.13呈现点赞和收藏内容的预览图

（1）实现难点：由于在UI设计阶段，点赞和收藏预览图的设计要出现动态配图、用户头像等信息，整体呈现为一个高比宽长的矩形，而且所有预览图排成两列进行展示，使用listview显然不合适。

（2）解决方案：使用控件gridview就可以实现网格的分布，并将网格设置成两列就可以达到UI设计的效果，其中使用查询数据库的结果做为数据源，就可以实现整体的效果。

（3）核心代码：

预览图布局文件collection_card_layout.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="180dp"
    android:layout_height="251dp"
    android:background="@color/cardColor">
    <TextView
        android:id="@+id/userId"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="id"
        android:visibility="gone">

    </TextView>

    <ImageView
        android:id="@+id/image"
        android:layout_width="130dp"
        android:layout_height="180dp"
        app:srcCompat="@mipmap/examplepicture"
        android:layout_marginTop="15dp"
        android:layout_centerHorizontal="true"
        />

    <ImageView
        android:id="@+id/logo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:srcCompat="@mipmap/headscu"
        android:layout_marginTop="210dp"
        android:layout_marginLeft="25dp"/>

    <TextView
        android:id="@+id/username"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="65dp"
        android:layout_marginTop="210dp"
        android:textSize="14dp"
```

```
        android:textColor="@color/black"
        android:text="断章" />

    <ImageView
        android:id="@+id/collectBtn"
        android:layout_width="37dp"
        android:layout_height="37dp"
        android:layout_marginLeft="130dp"
        android:layout_marginTop="200dp"

        app:srcCompat="@drawable/star" />

</RelativeLayout>
```

适配器类CollectionAdapter：

```
public class CollectionAdapter extends BaseAdapter {
    private List<Collection> mCollectionList;
    private LayoutInflater mLayoutInflater;
    private Context mContext;
    private CollectionDBOpenHelper collectionDBOpenHelper;
    public CollectionAdapter(Context mContext,List<Collection>
mCollectionList) {
        this.mCollectionList = mCollectionList;
        this.mContext = mContext;
        mLayoutInflater = LayoutInflater.from(mContext);
        collectionDBOpenHelper = new CollectionDBOpenHelper(mContext);}

    @Override
    public int getCount() {
        return mCollectionList.size();}

    @Override
    public Object getItem(int i) {
        return mCollectionList.get(i);}

    @Override
    public long getItemId(int i) {
        return i;}

    @Override
    public View getView(int i, View view, ViewGroup viewGroup) {
        if(view==null){
            view =
mLayoutInflater.inflate(R.layout.collection_card_layout,viewGroup,false);}
        //获取控件
        TextView username = view.findViewById(R.id.username);
        TextView userId = view.findViewById(R.id.userId);
        ImageView image = view.findViewById(R.id.image);
        ImageView logo = view.findViewById(R.id.logo);
        //获得对象数据
        Collection collection = mCollectionList.get(i);
        //展示数据
        username.setText(collection.getUsername());
        userId.setText(collection.getId().toString());
        image.setImageBitmap(collection.getImage());
        logo.setImageBitmap(collection.getLogo());

        ImageView collectBtn = view.findViewById(R.id.collectBtn);
        collectBtn.setOnClickListener(new View.OnClickListener() {
            @Override
```

```
            public void onClick(View view) {
                collectionDBOpenHelper.deleteById(collection.getId(),
CollectionDBOpenHelper.TABLE_NAME_COLLECTION);
                collectBtn.setVisibility(View.GONE);
                notifyDataSetChanged();
            }
        });
        return view;
    }
}
```

数据库查询做为数据源，配合适配器将数据呈现在控件上：

```
gridView = (GridView) findViewById(R.id.gridview_collection);
mCollectionDBOpenHelper = new CollectionDBOpenHelper(this);
mCollectionList =
mCollectionDBOpenHelper.queryAll(CollectionDBOpenHelper.TABLE_NAME_COLLECT
ION);
mCollectionAdapter = new CollectionAdapter(this,mCollectionList);
gridView.setAdapter(mCollectionAdapter);
```

2.14我的口味界面

2.14.1在UI设计方面

（1）实现难点：GridView网格视图：网格视图主要是按照行列来展示数据和图片，与ListView的使用方法类似，难点在于如何对布局参数进行设置和如何构建GridView每一项中的布局。

（2）解决方案：在本次项目中我使用到两个参数来进行界面布局：numColumns设置网格列数，verticalSpacing设置每一项之间的列间隔。列数设为3列，每项垂直间距设置为20dp。 GridView每一项中的布局仅包含一个TextView显示口味。所以需要新建一个layout文件，这个文件中仅布局了一个TextView。

2.14.2在功能实现方面

（1）实现难点：GridView网格视图的使用

（2）解决方案：其作用就是将适配器Adapter中的数据以行列形式展示出来，所以在实现功能时需要定义一个适配器，除此以外，我们还需要定义一个列表用来存储数据，当我们需要将数据展示到界面中时，才把列表中的数据传递给适配器。一般来

说，如果GridView中的项里面包含了太多数据，且需要实现较繁琐的操作，这时候需要我们自定义适配器类，将其封装起来。在本项目中使用到了GridView，且GridView每一项的内容仅有一个TextView，所以还需声明一个Android Studio中自带的适配器ArrayAdapter<String>用来向GridView提供数据，声明一个List<String>用来存储String类型的口味信息。初始化适配器和列表，并将适配器与列表和GridView每一项的内容进行绑定。

3、用户体验记录

我们也在线下寻找十名同学进行软件使用体验，并将反馈的问题进行记录和分析。

序号	用户	身份	应用评价	改进意见	开发者分析
1	郭婷婷	2020级计算机学院学生	很好。	增加改头像的功能	图片选择器的功能目前还没实现，后续会改进。
2	潘玮莹	2020级计算机学院学生	挺好玩的。	发布动态那里增加选择图片的功能	图片选择器的功能目前还没实现，后续会改进。
3	李虹	2020级计算机学院学生	界面挺不错	可以增加点击轮播图查看详情的功能	轮播图的点击事件还没有实现，后续会改进。
4	肖洽桐	2020级计算机学院学生	功能很完善	可以增加点击的音效	后续会考虑增加点击音效
5	黄金昀	2020级计算机学院学生	界面很好看	汤饮详情可以增加分享功能	后续会考虑增加分享汤饮的功能
6	陈林萍	2020级计算机学院学生	很有创意	希望可以增加推荐汤饮的功能	推荐算法还没有实现，后续会改进
7	姚思婷	2020级教育与科学学院学生	挺新颖的	可以增加关注用户的功能	后续会考虑增加关注用户的功能
8	卢艳萍	2020级教育与科学学院学生	设计得很好	可以增加个性签名的功能	后续会考虑增加个性签名的功能
9	林樱婷	2020级教育与科学学院学生	点子很不错	希望个人主页可以自定义背	图片选择器的功能目前还没实现，后续会改进

				景	
10	区富强	校外人士	很好用	希望能够在应用商店下载到这个app	完善功能后会考虑上线

4、已完成的改进和存在的问题

4.1存在问题：

4.1.1UI设计部分：

（1）某些控件在模拟机上运行时显示大小正常，但是在真机上运行时大小显示会不正常，甚至会不显示。

（2）选取的颜色不能和第二阶段设计图中颜色完全一致，有些界面的颜色选取过于鲜艳和暗淡。

（3）字体设置，有些显示英文的部分没有将它们设置为楷体。

（4）轮播图显示大小与原设计图相比有些过小，导致界面有些不是很协调。

4.1.2功能实现部分：

（1）在用户发布动态页面和我的页面，仅能发布文字，用户不能选择图片，也不能自定义头像，目前还没有添加图片选择器。

（2）没有开发后台管理系统，用户的数据仅能存储在手机中，不能进行多用户在线和资源共享。

（3）目前仅设计了用户可以添加口味，首页展示汤茶饮，还未实现app可以根据用户口味自动化推荐汤茶饮的功能。

（4）用户之间目前仅能通过评论动态来进行够通，还未实现用户间私聊功能。

（5）根据标准兼容性测试，目前app的兼容性比较差。

4.2产品改进

（1）已调整了某些界面中颜色过于鲜艳或黯淡的地方。

（2）后续可以继续改善轮播图，使界面协调。

（3）后续可以继续实现根据用户口味自动化推荐汤茶饮的功能。

（4）后续可以实现用户之间的私聊功能。

（5）除了收藏用户动态以外，接下来还可继续扩展收藏汤茶饮的功能。

接下来将会投入更多时间，希望能不止做到预先那样，还希望做到更多。

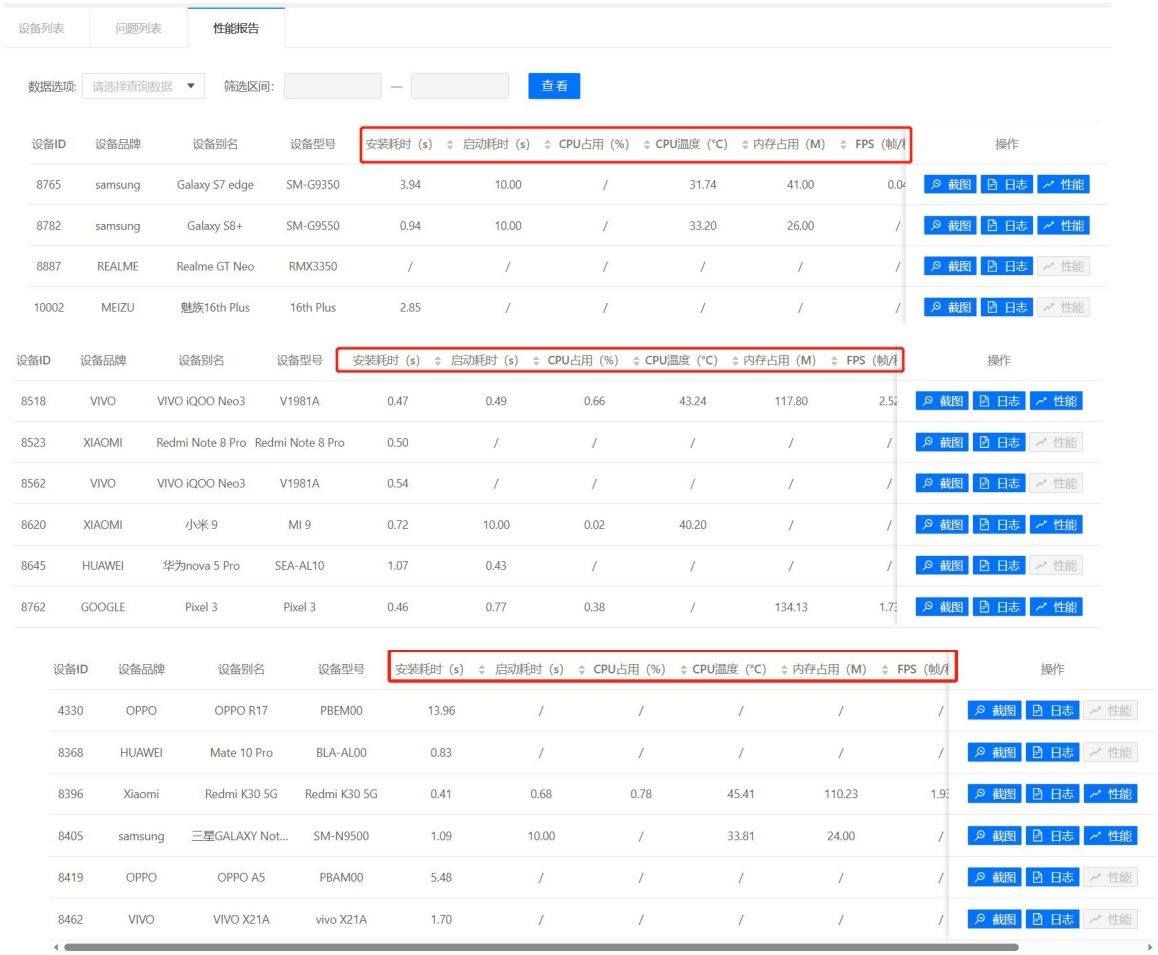
三、测试大纲和测试报告

1、软件测试与软件完成情况基本展示

我们使用的测试方案是百度 MTC 移动 App 测试方案，进行脚本兼容性测试以及标准兼容性测试。

1.1脚本兼容性测试

本测试覆盖市场主流机型，测试安装、启动、主要功能、卸载；可定制测试脚本，覆盖所需页面和核心功能；测试报告包含用例细节、截图、日志、性能数据和 BUG 详情。



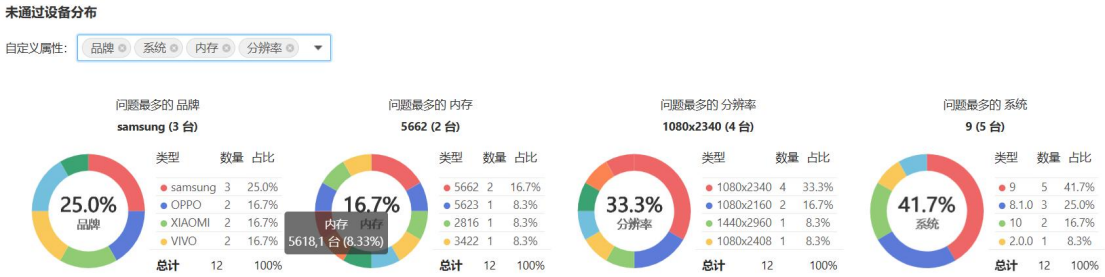
性能分析

1.2标准兼容测试

在选定机型上安装、启动、monkey、卸载等；记录测试过程中完整日志、截图、录像捕 获 CPU、内存、流量、电量等性能数据助您定位闪退、crash、ANR 等问题

兼容结论

问题列表



问题终端分布

2、测试结果分析

我们对前面的测试结果一一作分析。

2.1脚本兼容性分析

安装时间最多高达13.96秒的。同时可以看到内存占用量最高为134MB，启动耗时最高为10s。

2.2标准兼容性分析

从结论总览中可以看到我们App的兼容性比较差，只有18%的通过率，问题最多的品牌是samsung，问题最多的内存是5662类型，问题最多的分辨率似乎1080×2340，问题最多的系统是为9号。

2.3人工测试分析

功能还是太少了，可以继续增加APP的功能；没有兼容全面屏，若使用有不完整屏幕的手机，会出现缺陷区域部分黑暗的情况；使用过程中会出现轻微卡顿；用户界面可以设计的更加美观。

四、产品安装和使用说明

本 App 要求 Android 手机版本最低为6.0。安装时在应用商店搜索“Yingcha”并下载则可完成安装。

具体使用说明可以参考本文档中的以下部分：〈二〉产品实现方案　二、 UI界面及功能描述