雾霾探测系统设计实验报告

1、任务分析

本次任务要求设计一款手机端雾霾 app 探测系统，需要满足以下三个功能：

1.定位功能：获取当前定位城市信息，并显示在客户端。

2.界面设计：包含显示天气和空气质量指数的动态显示，湿度温度折线图。

3.天气详情和空气质量指数：通过和风天气、墨迹天气、我的天气等均可获取，百度地图、高德地图等可以辅助位置信息。

2、系统方案

2.1 app 实现方案

app 实现方案大致有三种：

1. 使用 html 进行网页开发，手机通过浏览器的方式进行访问。虽然这能够解决不同机型和操作系统的适配问题，但不是真正的 app。
2. 使用 uniapp 将网页开发成果在手机端进行部署，和网页开发方式类似，但可能存在一定兼容性问题。
3. 使用 Android 原生开发，选择 Android Studio 作为编辑器。这样能够在 Android 操作系统上获得最佳的性能表现，并且能够调用手机端的 Gps 定位信息和网络信息，用户体验较好。不足之处在于无法适用于 Apple 的操作系统。

综合上述考虑，我们考虑到绝大多数人是 Android 用户，因此，我们选择了 Android 原生开发。

2.2 app 定位方案

Android app 定位实现方案大致有两种：

1. GPS 定位。该方案需要用户打开 GPS 定位模块，该方案准确度较高。
2. 网络定位。该方案通过用户的网络 ip 进行定位，但准确率不高。

综合上述考虑，我们选择使用百度地图提供的高精度 GPS 定位。

2.3 天气查询实现方案

天气数据可以通过各种天气网站进行获取。我们选择了和风天气的数据接口。

3、系统实现

3.1 权限授予

为了获取用户的定位信息，我们需要用户在首次启动时，对 App 的网络和 GPS 调用权限进行授权。为了防止用户在未经授权的情况下进入时，直接看到空数据。我们设置了一个引导页。在 App 启动时，会首先看到引导页，并提示权限授予弹窗，如果用户授予权限则能在 2 秒之后跳转到主页；如果用户未授权，则不进行跳转，跳出提示引导用户在系统设置内进行授权。

相关核心代码：

```
private void checkPermissions(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        int q1 = ContextCompat.checkSelfPermission(getContext(), permissions[0]);
        int q2 = ContextCompat.checkSelfPermission(getContext(), permissions[1]);
        int q3 = ContextCompat.checkSelfPermission(getContext(), permissions[2]);
        int q4 = ContextCompat.checkSelfPermission(getContext(), permissions[3]);
        int q5 = ContextCompat.checkSelfPermission(getContext(), permissions[4]);
        int q6 = ContextCompat.checkSelfPermission(getContext(), permissions[5]);
        int q7 = ContextCompat.checkSelfPermission(getContext(), permissions[6]);
        // 权限是否已经授权 GRANTED---授权  DINIED---拒绝
       if (q1 != PackageManager.PERMISSION_GRANTED ||
                q2 != PackageManager.PERMISSION_GRANTED ||
                q3 != PackageManager.PERMISSION_GRANTED ||
                q4 != PackageManager.PERMISSION_GRANTED ||
                q5 != PackageManager.PERMISSION_GRANTED ||
                q6 != PackageManager.PERMISSION_GRANTED ||
                q7 != PackageManager.PERMISSION_GRANTED) {
            // 如果没有授予该权限，就去提示用户请求
          startRequestPermission();
        }
        else{
            //获取权限成功,跳转
          new android.os.Handler().postDelayed(new Runnable() {
                @Override
                public void run() {
                    Intent mainIntent = new Intent(Splash.this,MainActivity.class);
                    Splash.this.startActivity(mainIntent);
                    Splash.this.finish();
                }
            },2000);
        }
    }
}
```

3.2 定位功能实现

我们采用百度地图的 API 来实现定位功能，主要步骤如下：

1. 在百度地图开放平台申请账号并新建应用。
2. 填写应用包名信息以及开发版和发布版的 SHA1
3. 下载并配置百度地图 SDK
4. 通过百度地图提供的 BDAbstractLocationListener()读取定位数据

核心代码：

```
public BDAbstractLocationListener oneLocationListener = new
BDAbstractLocationListener() {
    /**
     * 定位请求回调函数
    * @param location 定位结果
    */
    @Override
    public void onReceiveLocation(BDLocation location) {
        if (null == location) {
            return;
        }
        LatLng latLng = new LatLng(location.getLatitude(), location.getLongitude());
        LatLngBounds.Builder builder = new LatLngBounds.Builder();
        builder.include(latLng);
        StringBuffer sb = new StringBuffer(256);
        // 更新地图状态
       if (location.getLocType() == BDLocation.TypeGpsLocation) {// GPS 定位结果
          sb.append("gps 定位成功");
        } else if (location.getLocType() == BDLocation.TypeNetWorkLocation) {// 网络定
位结果
```

```
            sb.append("网络定位成功");
        } else if (location.getLocType() == BDLocation.TypeOffLineLocation) {// 离线定
位结果
            sb.append("离线定位成功");
        } else if (location.getLocType() == BDLocation.TypeServerError) {
            sb.append("服务端网络定位失败");
        } else if (location.getLocType() == BDLocation.TypeNetWorkException) {
            sb.append("网络不同导致定位失败，请检查网络是否通畅");
        } else if (location.getLocType() == BDLocation.TypeCriteriaException) {
            sb.append("无法获取有效定位依据导致定位失败，请开启 GPS 和网络后重启 App");
        }
}
```

3.3 天气查询功能实现

我们采用和风天气的接口来查询天气数据，空气质量数据以及一周内的温湿度数据，我们首先在和风天气 App 注册账号，然后获取网站提供的免费 key 值，最后就能调用接口。

接口地址如下：

天气数据接口：

https://free-api.heweather.net/s6/weather/now?key=3086e91d66c04ce588a7f538f917c7f4&location=城市名

空气质量数据接口：

https://free-api.heweather.net/s6/air/now?key=3086e91d66c04ce588a7f538f917c7f4&location= 城市名

一周温湿度数据接口：

https://free-api.heweather.net/s6/weather/now?key=3086e91d66c04ce588a7f538f917c7f4&location=城市名

有了接口之后，我们需要在 App 内发起网络请求。我们使用了 okhttp 框架，采用 get 异步请求的方式向接口发送请求。由于 Android 不能在主线程内发送网络请求，我们采用了多线程的方式，开辟子线程进行数据获取，然后将数据传递到主线程，进行 UI 的更新。

从接口返回获取的数据是 Json 形式，因此我们引入了谷歌开发的 Gson 库进行 json 数据的解析。

核心代码如下：

```
//获取今天的天气和温度
private void get_weather(String area) {
    //使用 Get 异步请求
    OkHttpClient client = new OkHttpClient();
    Request request = new Request.Builder().url("https://free-
api.heweather.net/s6/weather/now?key=3086e91d66c04ce588a7f538f917c7f4&location=" +
area).build();
    client.newCall(request).enqueue(new Callback() {
```

```
        @Override
        public void onFailure(Call call, IOException e) {
            System.out.println("调用失败");
        }
        @Override
        public void onResponse(Call call, Response response) throws IOException {
            if(response.isSuccessful()){//回调的方法执行在子线程。
             JSONObject result_json2= null;
                try {
                    result_json2 = JSON.parseObject(response.body().string());
                    JSONArray jsonArray2 = result_json2.getJSONArray("HeWeather6");
                    JSONObject object2 = jsonArray2.getJSONObject(0);
                    // 天气和温度
                 String now_weather = object2.getString("now");
                    JSONObject json3 = JSON.parseObject(now_weather);
                    weather = json3.getString("cond_txt");
                    temp = json3.getString("tmp");
                    // 更新时间
                 String up_time = object2.getString("update");
                    JSONObject json4 = JSON.parseObject(up_time);
                    time = json4.getString("loc");
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
            MainActivity.this.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    mweather.setText(weather);
                    mtemp.setText(temp);
                    mtime.setText("最近更新时间: " + time);
                }
            });
        }
    });
}
```

3.4 日期的显示

为了在主界面直观展示当前是星期几，我们采用了 Android 自带的 Calendar 类来获取系统当前日期，并将其转换成星期，核心代码如下：

```
/**
 * 根据系统时间返回今天是星期几
 */
public class Weektime {
    private static String mWay;
    public static String StringData(){
        final Calendar c = Calendar.getInstance();
        c.setTimeZone(TimeZone.getTimeZone("GMT+8:00"));
        mWay = String.valueOf(c.get(Calendar.DAY_OF_WEEK));
        if("1".equals(mWay)){
            mWay ="日";
        }else if("2".equals(mWay)){
            mWay ="一";
        }else if("3".equals(mWay)){
            mWay ="二";
        }else if("4".equals(mWay)){
            mWay ="三";
        }else if("5".equals(mWay)){
            mWay ="四";
        }else if("6".equals(mWay)){
            mWay ="五";
        }else if("7".equals(mWay)){
            mWay ="六";
        }
        return mWay;
    }
}
```

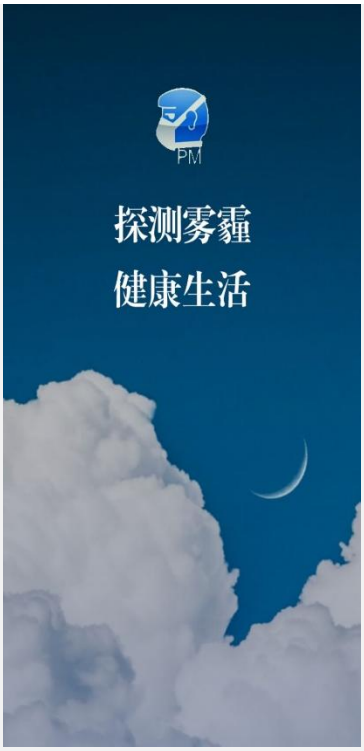
3.5 系统界面设计

系统主界面我们主要参考了手机自带的天气 App 的设计布局，并使用 RoundProgressBar 和 MPAndroidChart 两款开源的 Android 绘图框架实现图表的绘制，实现数据的可视化，同时增加了动画效果，进一步优化了用户体验。

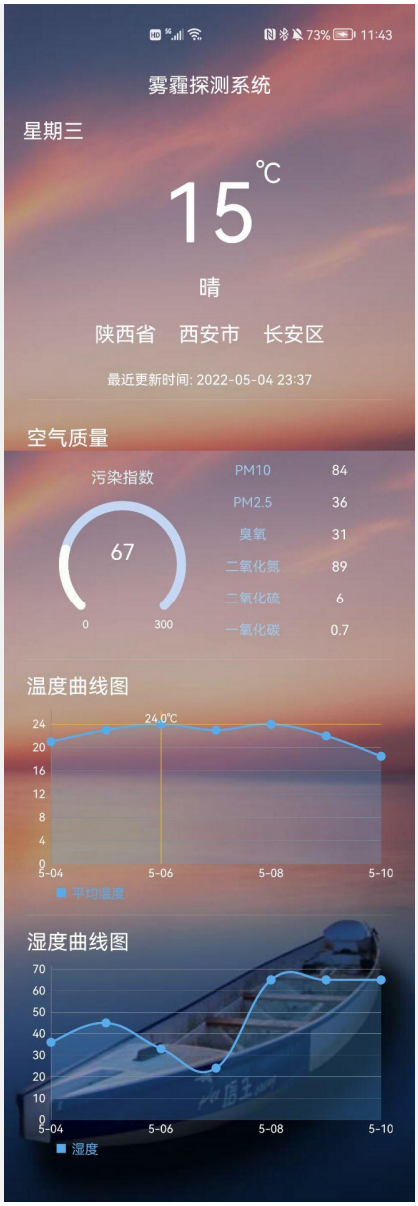
4、结果分析

在实机上，我们的界面展示如下：

引导页



主页



同时，我们设计了一个应用图标： 

我们的应用成果符合了本次测试要求，并实现了良好的用户体验。

为了方便应用的推广，我们将安装包上传到了蒲公英分发平台上，下载二维码如下：

5、小组分工

章星宇：App 定位功能开发，天气与温湿度查询功能开发

李宝富：App 界面布局开发，权限功能开发

高寓杨：UI 设计，图表绘制

6、心得体会

章星宇：通过本次实验，我对 Android 应用开发有了进一步的了解和掌握。首次尝试了定位功能开发，网络请求开发和动态数据解析。期间遇到了很多 Bug，但最终都一一克服。对我而言，本次实验给我提供了一个对未知领域的练手机会，我也再次加深了对网络的认识，收获颇丰。

李宝富：通过本次实验，我学习了 Android App 的界面布局开发以及各种权限功能的申请和调用。同时，和队友合作，对接口调用有了一些新的理解和体会。并且，对于 App 的设计和用户体验方面有了全新的认识。

高寓杨：通过本次实验，我对于接口的运用有了进一步的了解和掌握。在原来仅接触过前端的情况下，这次与队友一起尝试合作了定位功能以及接口运用，同样也让我对 app 的布局一些权限功能有了全新的了解。本次实验给我一个很好的机会让我能够尝试 Android 开发，加深了理解。