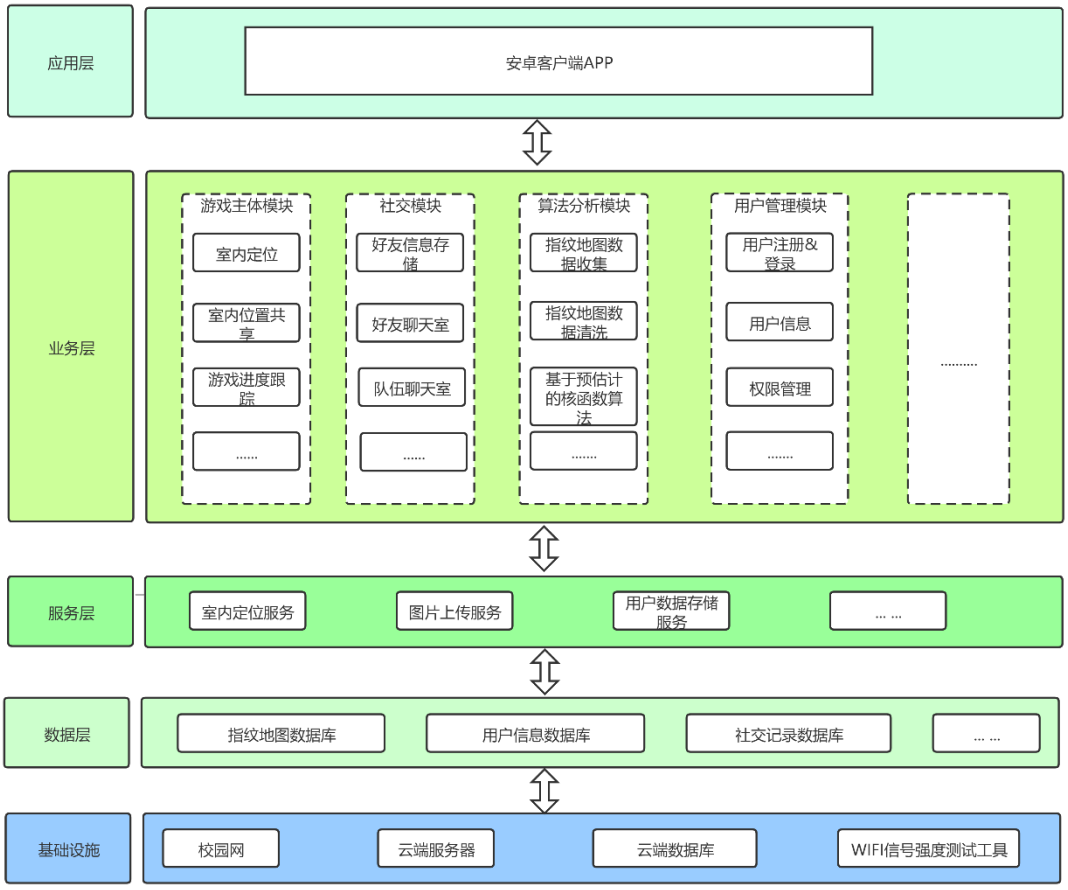
项目设计说明书

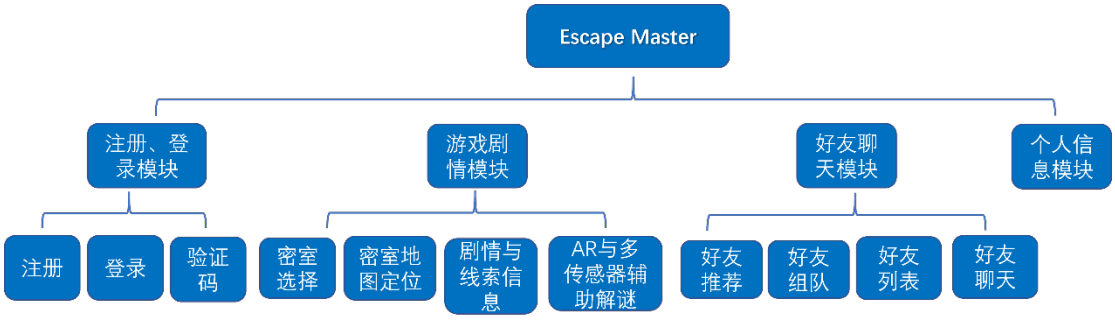
目录

1 概要设计

1.1 系统架构设计



1.2 系统总体功能结构设计



1.3 关键技术设计

(一) 定位算法设计

本项目将主要采用指纹地图定位法作为对移动设备进行室内定位的主要算法。相对于三边定位法，指纹地图很好地降低了复杂的环境因素对信号传播产生的影响，同时也不再使用距离参数进行位置估计，从根本上避免将众多影响因子引入到计算当中。

指纹地图法将定位分为两个阶段：训练阶段和定位阶段。

1. 训练阶段

最基础的工作是在工作区域完成指纹点的数据采集，包括每个指纹点的位置信息、信号强度平均值，方差以及分布图等。根据采集的数据分类存储，绘制完成整个区域的指纹地图。

2. 定位阶段

使用移动设备在工作区域中采集目标点的信号强度值数据，根据获得的数据在地图中进行匹配，利用精度低，但复杂性低的算法预估计设备的大致区域或若干个候选样本点，再在确定区域的基础上，用复杂性高但精度高的算法确定设备的精确位置。由此高速、精准地实现整个定位阶段。

下面将详细介绍本项目会应用到的关于指纹地图的具体算法。

1. 最邻近算法（Nearest Neighbor, NN）

顾名思义，NN 的基本思路就是再地图中寻找到一个与待确定点特征信号强度向量的欧氏距离最短的位置向量。

在训练阶段中，指纹地图中的每个点数据都表示为(x,y),(m1,m2, ,mn)，前者为标签点的坐标值，后者为对应点的 n 个 AP 节点的信号强度平均值。

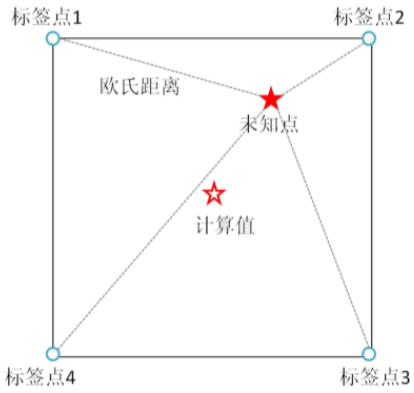
定位阶段，首先测得待确定点的信号强度向量(m1,m2, ,mn)，再计算各个标签点的信号强度向量到该点的欧氏距离，其中欧氏距离最短的标签点确定为结果点。

2. K 阶最邻近算法（K- Nearest Neighbor, KNN）

NN 算法的精度与标签点的密度呈正相关，但密度过大会导致训练阶段的投入开销呈几何式增加。

K 阶最邻近算法在不增加开销的基础上提升了指纹地图定位的精度，即在 t 个样本点的欧式距离中找到 K 个最小值，计算对应 K 个坐标值的平均值作为结果值(x, y)。其中

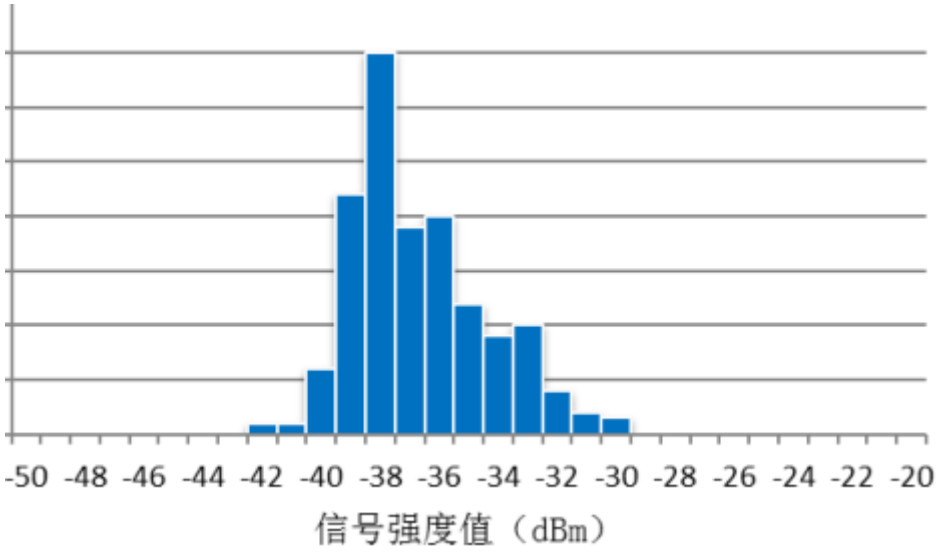
$$x = \frac{\sum x_i}{k}, y = \frac{\sum y_i}{k}$$



3. 直方图算法（Histogram Method）

由于未知点到不同样本点的距离大多数情况下并不相同，以取平均值为主要思想的 KNN 算法仍存在很大的误差。相比 K 阶临近算法这种确定性算法，直方图算法引入了概率论与数理统计方面的知识，即根据某个位置的某个 AP 节点的信号强度在多次测量中的分布情况，为未知点的信号强度分配权重，再根据未知点的信号强度向量以及贝叶斯公式，计算出待选样本点中最可能的结果。

信号强度值在测量次数上的分布最终是以直方图的形式呈现出来的。直方图中柱形越高表明在测量点处出现该信号强度值的出现次数越多，概率越大。



在测量出未知点的特征向量值 R 后，它的坐标为 L 的概率可表示为：

$$P(L|R) = \frac{P(R|L) * P(L)}{P(R)}$$

其中，P(R)为在整个地图中信号强度为 R 的概率，因此，在一定范围内 P(R)**可以视为一个常数**。P(L)表示在不考虑信号强度值的情况下位置为 L 的概率，显然在无任何条件约束的情况下未知点出现在任一位置服从平均分布，**依然可以视为常数**。

因此该公式值的大小取决于条件概率 P(R|L)，其值可计算成为：

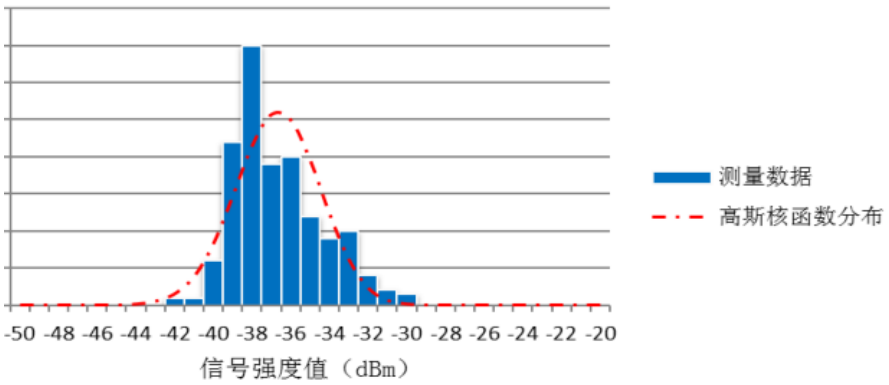
$$P(R|L) = P(rss_1, rss_2, \quad , rss_n|L) = \frac{1}{n} * \sum_{i=1}^{n} P(rss_i|L)$$

（这也将作为下文计算合成点时的点权重因子。）

这种算法很好地适应了各个样本点的变化性，根据样本点测量数据可以很好地绘制出对应 AP 节点的直方图。

4. 核函数算法 （Kernel Method）

由于直方图算法的测量数据都是离散值，呈阶梯分布，精度依然会受到影响，此外，记录每个 AP 节点在所有位置的分布需要巨大的存储空间。为了避免以上问题，核函数算法采用高斯曲线来拟合信号规律，通过使用参数量少的核函数计算定位结果的概率，极大地节约了存储空间并提升了测量精度。



核函数算法只需要在测量的基础上存储计算获得的信号强度均值以及方差来模拟正态分布曲线。定位时，条件概率$P(rss_i|L)$可以计算为：

$$P(rss_i|L) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(rss_i - \overline{rss_i})^2}{2\sigma^2}\right]$$

5.合成目标点

由上文分析，最后合成目标点时，j点的权重因子：

$w_j = \frac{1}{n} * \sum_{i=1}^{n} P(rss_i|L)$

计算目标点时：

$$x = \frac{\sum w_j x_i}{\sum w_i},\ y = \frac{\sum w_j y_i}{\sum w_j}$$

1.4 数据库设计

表1 指纹地图表
Table 1 fingerprint map Table

字段名	类型	描述
room_id	int	房间ID
pos_id	int	位置ID
pos_x	double	位置横坐标
pos_y	double	位置纵坐标
avg_1	double	点1平均值
avg_2	double	点2平均值
avg_3	double	点3平均值
avg_4	double	点4平均值
avg_5	double	点5平均值
avg_6	double	点6平均值
sd_1	double	点1标准差
sd_2	double	点2标准差
sd_3	double	点3标准差
sd_4	double	点4标准差
sd_5	double	点5标准差
sd_6	double	点6标准差

表2 密室评论表
Table 2 Room comments Table

字段名	类型	描述
id	int	ID
userID	int	用户ID
dateTime	datetime	评论时间
content	varchar(255)	评论内容
rate	double	评分

表3 密室信息表
Table 3 Room info Table

字段名	类型	描述
id	int	ID
room_name	varchar(255)	密室名
info_text	varchar(255)	密室信息
rate	varchar(255)	评分

表4 用户信息表
Table 4 User info Table

字段名	类型	描述
id	int	ID
phone_number	int	手机号
password	varchar(255)	密码
nickname	varchar(255)	用户名
avatar	mediumblob	头像
gender	int	性别
signature	varchar(255)	个性签名
gameTime	time	游戏时长
isLogin	int	登录状态

1.5 错误/异常处理设计

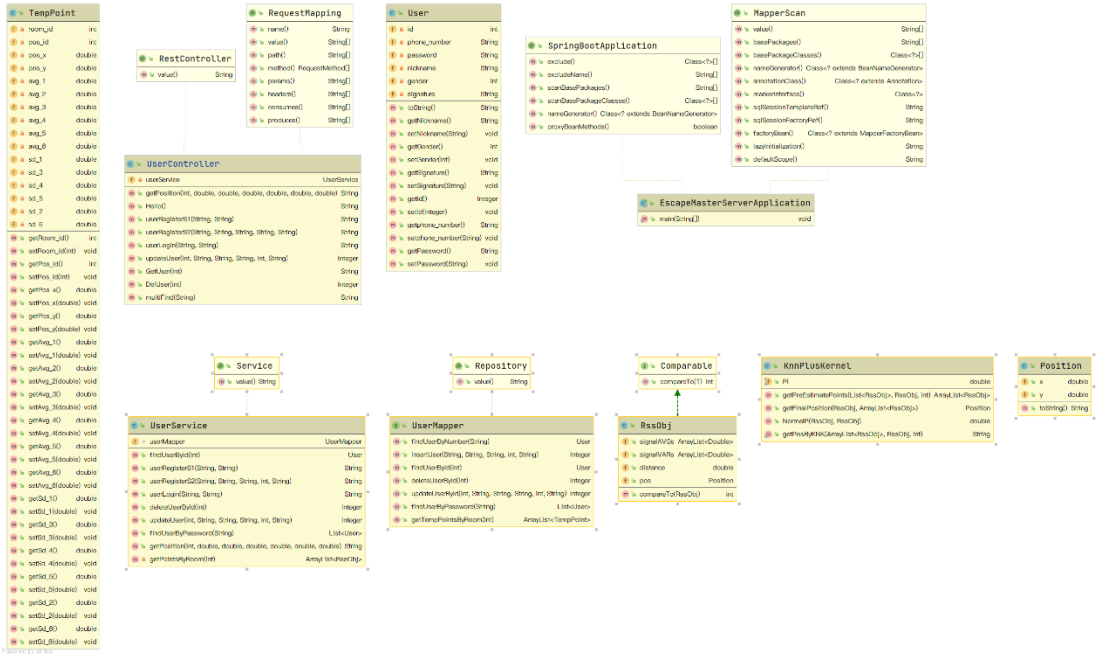
一、用户体验方面
1.信息输入不完全不能进行下一步操作。
2.已有账户不能再次进行注册
3.注册需通过手机号注册（体现账号唯一性，如第二点）
4.短信验证码具有时效性（3 分钟）
5.密码错误会给予密码错误反馈
6.设置密码及修改密码前后得输入两次，两次需保持一致
二、数据库方面
1. 插入用户：通过对单条插入语句 SQLException 进行异常判断
2. 增加、删除、修改用户：返回值为 0 则为异常进行单独处理

2 详细设计与实现

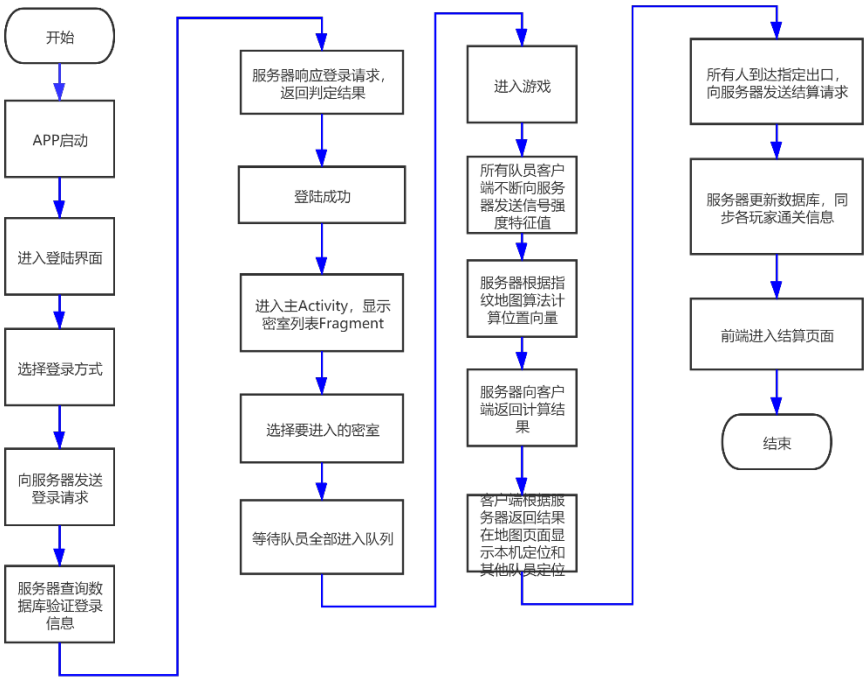
2.1 子系统的设计与实现

2.1.1 类图、流程图、时序图等

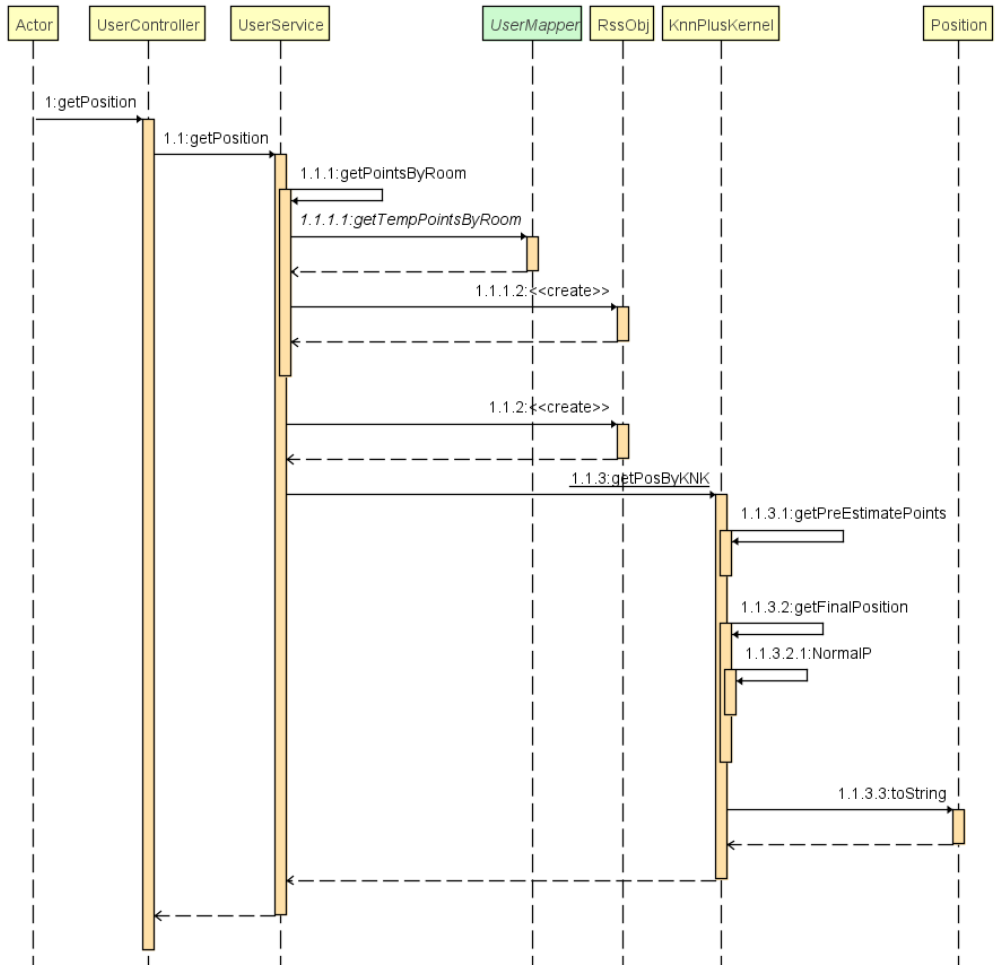
1. 类图

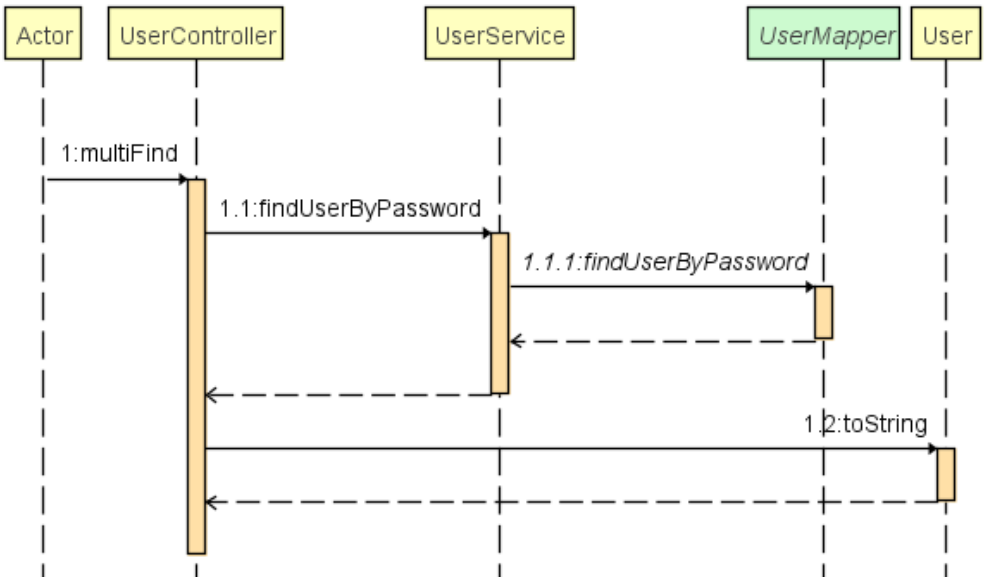


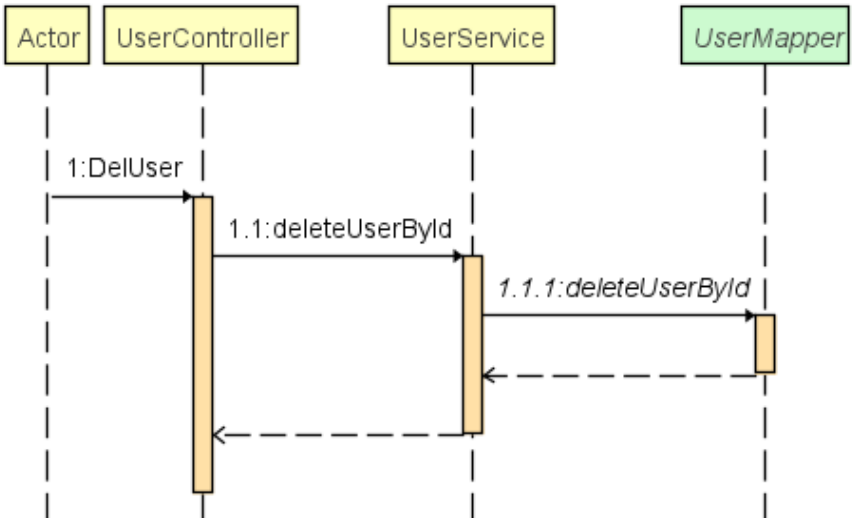
2. 流程图

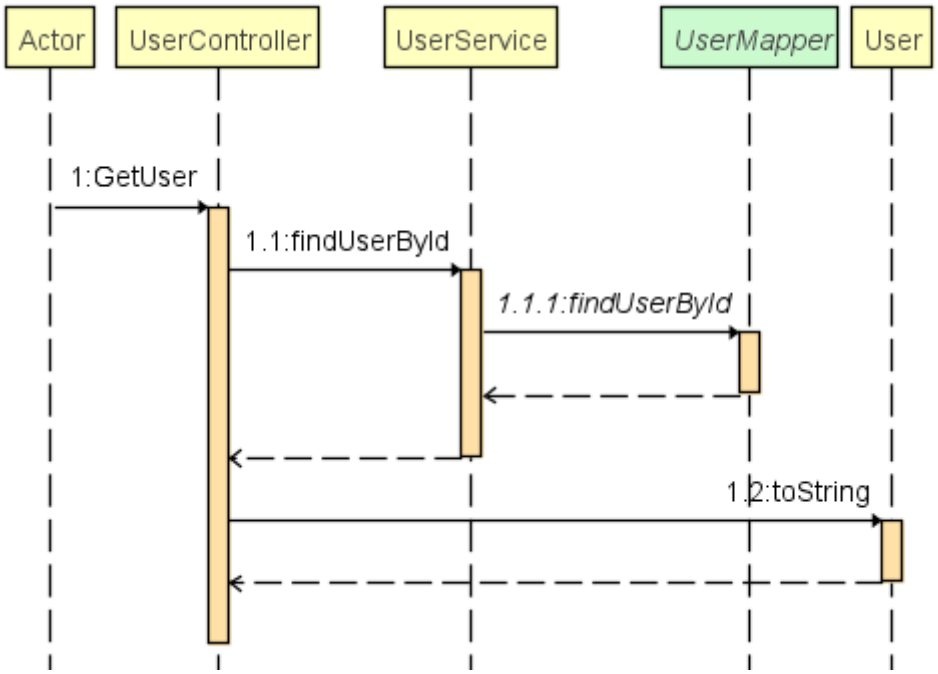


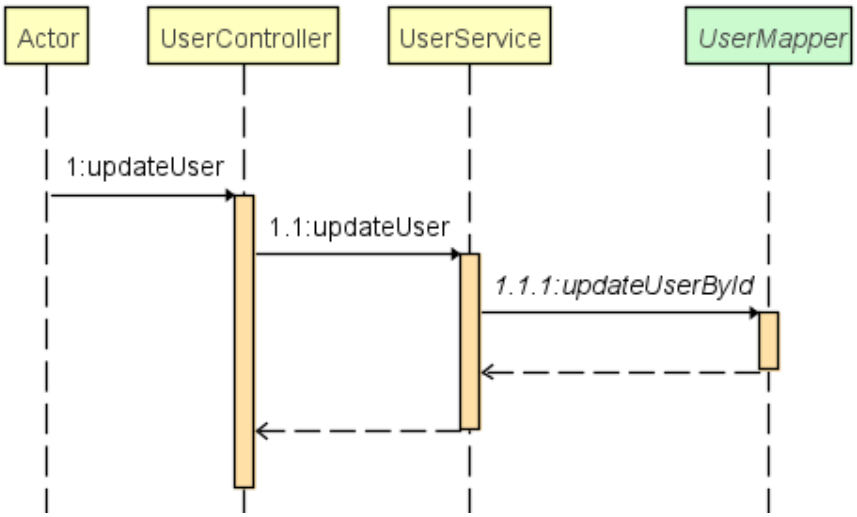
3. 时序图

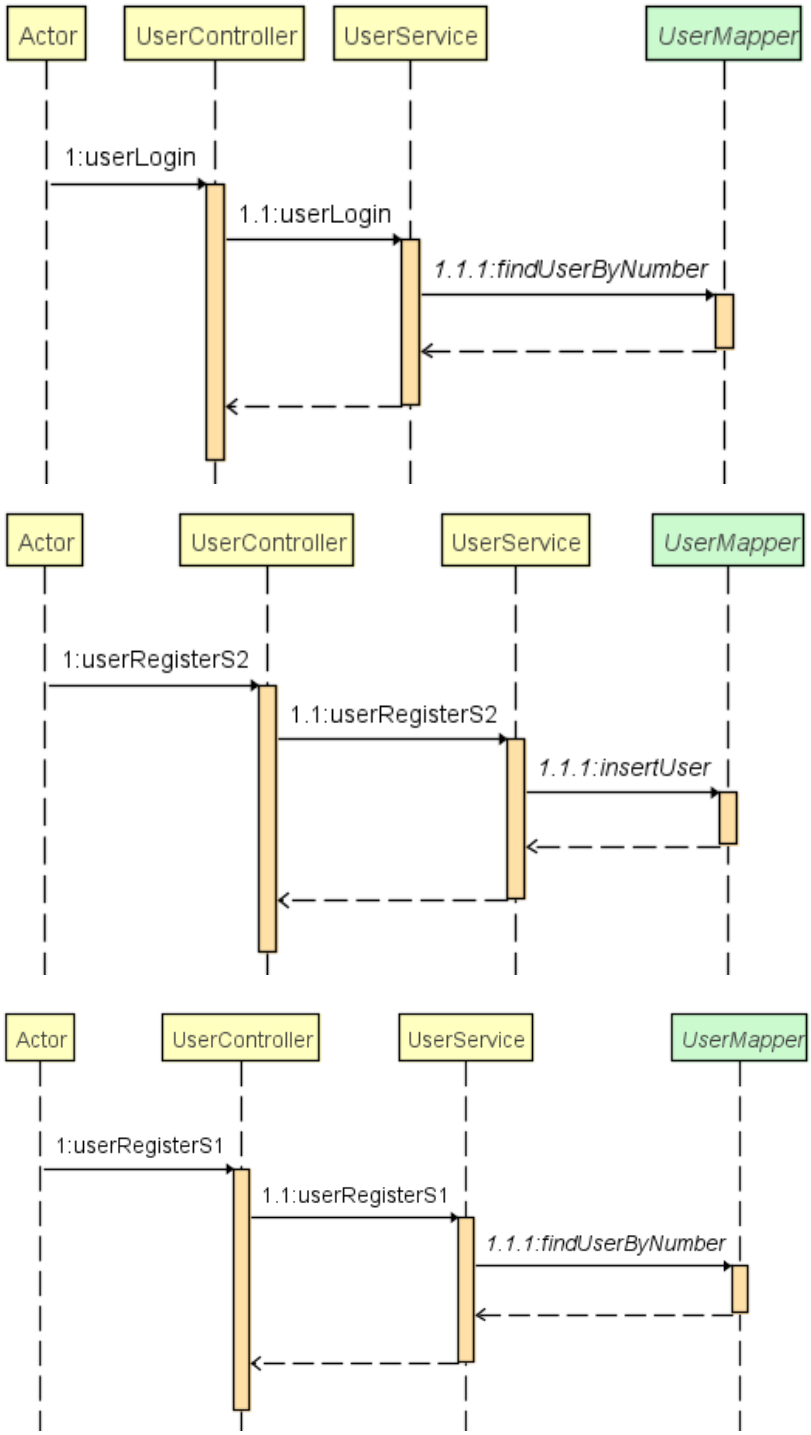

Actor
UserController
UserService
UserMapper
RssObj
KnnPlusKernel
Position
1:getPosition
1.1:getPosition
1.1.1:getPointsByRoom
1.1.1.1:getTempPointsByRoom
1.1.1.2:<<create>>
1.1.2:<<create>>
1.1.3:getPosByKNK
1.1.3.1:getPreEstimatePoints
1.1.3.2:getFinalPosition
1.1.3.2.1:NormalP
1.1.3.3:toString


Actor
UserController
UserService
UserMapper
User
1:multiFind
1.1:findUserByPassword
1.1.1:findUserByPassword
1.2:toString


Actor
UserController
UserService
UserMapper
1:DelUser
1.1:deleteUserById
1.1.1:deleteUserById


Actor
UserController
UserService
UserMapper
User
1:GetUser
1.1:findUserById
1.1.1:findUserById
1.2:toString


Actor
UserController
UserService
UserMapper
1:updateUser
1.1:updateUser
1.1.1:updateUserById


Actor
UserController
UserService
UserMapper
1:userLogin
1.1:userLogin
1.1.1:findUserByNumber
Actor
UserController
UserService
UserMapper
1:userRegisterS2
1.1:userRegisterS2
1.1.1:insertUser
Actor
UserController
UserService
UserMapper
1:userRegisterS1
1.1:userRegisterS1
1.1.1:findUserByNumber

2.1.2 用户界面展示

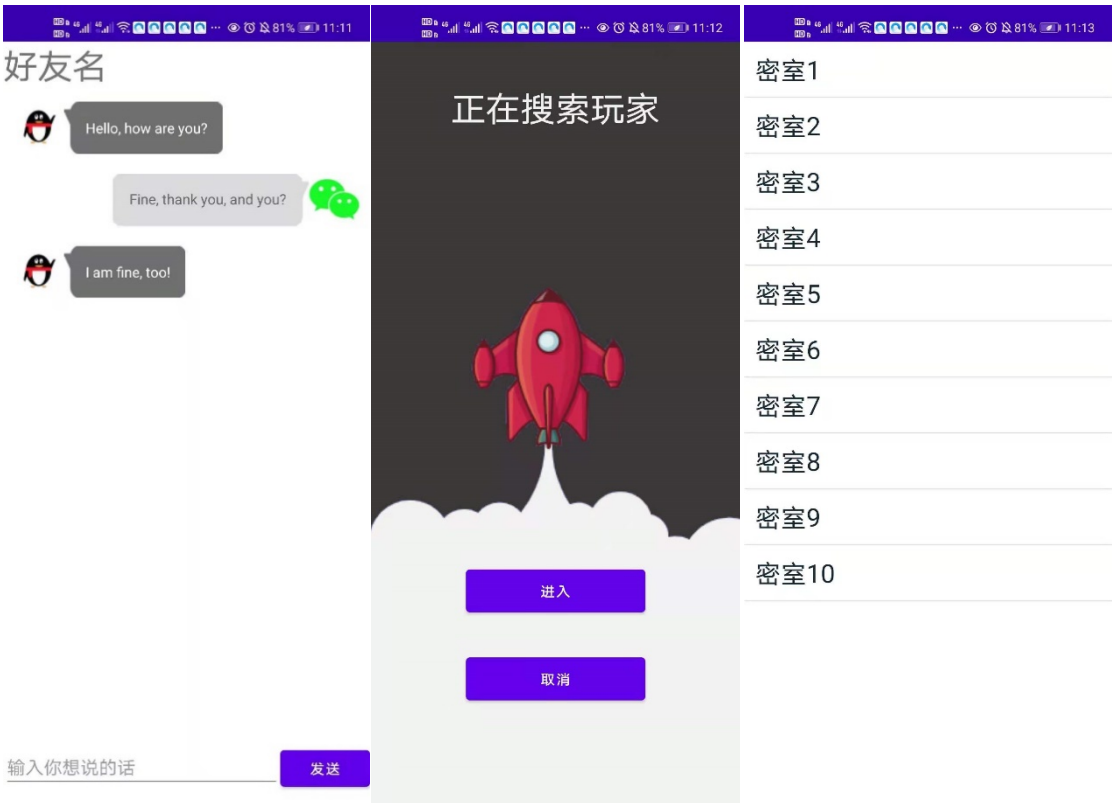
登录界面

注册界面

好友列表

密室列表

个人信息



聊天界面　　等待界面　　密室任务列表

2.2 核心算法（伪代码）

```
//预估计伪代码
function preEstimate(List<RssObj> fingerPrintingMap, RssObj targetPoint,
int estimateNum) {
    List<RssObj> tempList = new List<RssObj>(estimateNum);
    for(RssObj rssObj : fingerPrintingMap) {
        RssObj preEstimateObj = new RssObj();
        preEstimateObj.id = rssObj.id;
        //计算曼哈顿距离
        preEstimateObj.distance = Math.abs(targetPoint.x-
rssObj.x)+Math.abs(targetPoint.y-rssObj.y);
        tempList.add(preEstimateObj);
    }
    //对 tempList 的 distance 进行排序
    ...
    List<RssObj> preEstimateList = new List<RssObj>(estimateNum);
    //将 tempList 中距离最小的 estimateNum 个数据存入 preEstimateList
    ...
    //返回 estimateNum 个曼哈顿距离最小的指纹点
```

```
    return preEstimateList;
}

//高斯函数计算权重
function getWeight(targetPoint, lablePoint){
    for(i < signalDimentionSize){
        //根据标签点信号强度的均值和标准差
        //计算 lablePoint 确定为 targetPoint 的概率 p
        weight = weight + p
    }
    return weight / signalDimentionSize
}

//Kernel 算法拟合
function Kernel(fingerPrintingMap, targetPoint, estimateNum){
    preEstimateList   =   preEstimate((fingerPrintingMap,   targetPoint,
estimateNum);
    for(rssObj : preEstimateList){
        weight = getWeight(rssObj,  targetPoint);
        //加权平均分母
        sumWeight = sumWeight + weight;
        //加权平均分子
        estimateX = estimateX + rssObj.pos.x * weight;
        estimateY = estimateY + rssObj.pos.y * weight;
    }
    resultPosition = (estimateX, estimateY) / sumWeight;
    return resultPosition;
}
```

2.3 接口设计

在 Spring'Boot+Mybatis 框架的基础上，在数据库增删改查的接口设计

```
@Repository
public interface UserMapper {
    User findUserByNumber(String phone_number);
    Integer insertUser(String phone_number, String password, String nickna
    Integer insertUser(User user);
    Integer updatePassword(String phone_number,String password);

    User findUserById(int id);
    Integer deleteUserById(int id);
    Integer updateUserById(int id,String phone_number,String password,
                           String nickname,int gender,
                           String signature);

    List<User> findUserByPassword(String password);

    ArrayList<TempPoint> getTempPointsByRoom(int room_id);
}
```

处理用户各项操作的接口，接受 json 格式的请求信息

```
//短信验证登录
@RequestMapping(value = "/LoginBySMS",method = RequestMe
public String getSMS(@RequestBody JSONObject json){...}

//短信验证
@RequestMapping(value = "/LoginVerifySMS",method = Reque
public String checkSMS(@RequestBody JSONObject json){..

//密码登录
@RequestMapping(value="/LoginByPw",method = RequestMeth
public String userLogin(@RequestBody JSONObject json){.

//注册-首先验证号码
@RequestMapping(value="RegisterVerifyNumber",method = Re
public String userRegisterVerifyNumber(@RequestBody JSO

//注册-验证短信
@RequestMapping(value="RegisterVerifyCode",method = Requ
public String userRegisterVerifyCode(@RequestBody JSONO

//注册-设置密码-完善信息
@RequestMapping(value="RegisterSetPassword",method = Re
public String userRegisterSetPassword(@RequestBody JSON

//修改密码
@RequestMapping(value = "ModifyPassword",method = Reques
public String modifyPassword(@RequestBody JSONObject js

//请求位置
@RequestMapping(value = "getPosition",method = RequestMe
public String getPosition(@RequestBody JSONObject json)
```

网络接口，基于 okhttp 设计的网络工具类，提供 get 和 post 两种方式的异步请求静态方法，用于完成登陆，注册，修改密码等需要与服务器进行交互的功能。

```
public class OkHttpUtils {
    private static int TimeOut = 120;
    //单例获取ohttp3对象
    private static OkHttpClient client = null;
    private static synchronized OkHttpClient getInstance() {
        if (client == null) {
            client = new OkHttpClient.Builder()
                    .readTimeout(TimeOut, TimeUnit.SECONDS)
                    .connectTimeout(TimeOut, TimeUnit.SECONDS)
                    .writeTimeout(TimeOut, TimeUnit.SECONDS)
                    .build();
        }
        return client;
    }
    //使用get方式向服务器提交数据并获取返回提示数据
    public static void sendOkHttpRequest(final String address, final Callback callback){
        client=getInstance();
        Request request=new Request.Builder()
                .url(address)
                .build();
        client.newCall(request).enqueue(callback);
    }
    //使用POST方式向服务器提交数据并获取返回提示数据
    public static void sendOkHttpResponse(final String address,
                                          final RequestBody requestBody, final Callback callback){
        client=getInstance();
        //JSONObject这里是要提交的数据部分
        Request request=new Request.Builder()
                .url(address)
                .post(requestBody)
                .build();
        client.newCall(request).enqueue(callback);
    }

}
```

（1）登录

```
public void Login (String username, String password){
    if (username.equals("") || password.equals(""))
        Toast.makeText( context: MainActivity.this,  text: "用户名或密码不能为空", Toast.LENGTH_SHORT).show();
    else {
        JSONObject jsonParam = new JSONObject();
        jsonParam.put("ph", username);
        jsonParam.put("pw", password);
        String json = jsonParam.toJSONString();
        MediaType mediaType = MediaType.Companion.parse( $this$toMediaTypeOrNull: "application/json;charset=utf-8");
        RequestBody requestBody = RequestBody.Companion.create(json, mediaType);
        OkHttpUtils.sendOkHttpResponse( address: "http://o414e98134.wicp.vip/user/LoginByPw", requestBody, new Callback() {
            @Override
            public void onFailure(Call call, IOException e) { System.out.println(e); }

            @Override
            public void onResponse(Call call, Response response) throws IOException {
                final String data = response.body().string();
                Looper.prepare();
                if (data.equals("number not registered"))
                    Toast.makeText( context: MainActivity.this,  text: "账号未注册", Toast.LENGTH_SHORT).show();
                else {
                    if (data.equals("incorrect password"))
                        Toast.makeText( context: MainActivity.this,  text: "密码错误", Toast.LENGTH_SHORT).show();
                    else {
                        User.user = JSON.parseObject(data, User.class);
                        Toast.makeText( context: MainActivity.this,  text: "登陆成功", Toast.LENGTH_SHORT).show();
                        SharedPreferences settings = getSharedPreferences( name: "setting",  mode: 0);
                        SharedPreferences.Editor editor = settings.edit();
                        editor.putString("ph", username);
                        editor.putString("password", password);
                        editor.commit();
                        Intent i = new Intent( packageContext: MainActivity.this, MainpageActivity.class);
                        startActivity(i);
                    }
                }
                Looper.loop();

            }
        });
    }
}
```

（2）注册

```
public void Register(String username,String cache)
{
    if(username.equals("") || cache.equals("")){
        Toast.makeText( context: RegisterActivity.this, text: "手机号码或验证码不能为空",Toast.LENGTH_SHORT).show();
    }
    else {
        SharedPreferences settings = getSharedPreferences( name: "setting",  mode: 0);
        SharedPreferences.Editor editor=settings.edit();
        String msgId = settings.getString( key: "msgId",  defValue: "");
        JSONObject jsonParam=new JSONObject();
        jsonParam.put("phone_number",username);
        jsonParam.put("cache",cache);
        jsonParam.put("msgId", msgId);
        String json = jsonParam.toJSONString();
        MediaType mediaType=MediaType.Companion.parse( $this$toMediaTypeOrNull: "application/json;charset=utf-8");
        RequestBody requestBody=RequestBody.Companion.create(json,mediaType);
        OkHttpUtils.sendOkHttpResponse( address: "http://o414e98134.wicp.vip/user/RegisterVerifyCode", requestBody,  new Callback() {
            @Override
            public void onFailure(Call call, IOException e) { System.out.println(e); }
            @Override
            public void onResponse(Call call, Response response) throws IOException {
                final String data = response.body().string();
                Looper.prepare();
                if(data.equals("wrong cache code"))
                    Toast.makeText( context: RegisterActivity.this,  text: "验证码错误", Toast.LENGTH_SHORT).show();
                else {
                    Toast.makeText( context: RegisterActivity.this,  text: "验证成功", Toast.LENGTH_SHORT).show();
                    Intent i = new Intent( packageContext: RegisterActivity.this,SetPasswordActivity.class);
                    startActivity(i);
                }
                Looper.loop();

            }
        });


    }
}
```

（3）修改密码

```
public void Modifypassword(String pw1,String pw2)
{
    if (!pw1.equals(pw2))
        Toast.makeText( context: SetPasswordActivity.this,  text: "两次密码不一致", Toast.LENGTH_SHORT).show();
    else {
        SharedPreferences settings = getSharedPreferences( name: "setting",  mode: 0);
        SharedPreferences.Editor editor = settings.edit();
        editor.putString("password", pw1);
        editor.commit();
        Intent i = new Intent( packageContext: SetPasswordActivity.this, InforActivity.class);
        startActivity(i);
    }
}
```

动态获取权限接口，提供动态申请权限的静态方法，用于获取危险权限时（如位置信息，安卓 8 以后获取一些 wifi 信息需要获取位置信息以及打开定位）动态获取权限。
从 Android6.0 以上开始，系统引入了一个新的应用权限模型，运行时权限模型（Runtime Permissions）。
运行时权限：用户在应用运行时，对应用授予危险权限。由应用决定何时去申请权限（例如，在应用打开摄像机时或访问通讯录时），但必须容许用户来授予或者拒绝应用对特定权限的访问。所以对于危险权限，应用运行时必须动态申请权限。

```
public class PermissUtil {
    public static final int PERMISSON_REQUESTCODE = 123;
    //app需要进行检测的权限数组
    public static String[] appNeedPermissions = {
            Manifest.permission.ACCESS_FINE_LOCATION,
            Manifest.permission.ACCESS_COARSE_LOCATION,
            Manifest.permission.ACCESS_WIFI_STATE,
            Manifest.permission.CHANGE_WIFI_STATE
    };
     //app权限检测
    public static boolean checkPermissions(Activity activity, String... permissions) {
        List<String> needRequestPermissonList = findDeniedPermissions(activity, permissions);
        if (null != needRequestPermissonList
                && needRequestPermissonList.size() > 0) {
            ActivityCompat.requestPermissions(activity,
                    needRequestPermissonList.toArray(
                            new String[needRequestPermissonList.size()]),
                    PERMISSON_REQUESTCODE);
            return false;
        } else return true;
    }
    //获取权限集中需要申请权限的列表
    public static List<String> findDeniedPermissions(Activity contexts, String[] permissions) {
        List<String> needRequestPermissonList = new ArrayList<~>();
        for (String perm : permissions) {
            if (ContextCompat.checkSelfPermission(contexts,
                    perm) != PackageManager.PERMISSION_GRANTED) {
                needRequestPermissonList.add(perm);
            } else {
                if (ActivityCompat.shouldShowRequestPermissionRationale(
                        contexts, perm)) {
                    needRequestPermissonList.add(perm);
                }
            }
        }
        return needRequestPermissonList;
    }
    //检测是否说有的权限都已经授权
    public static boolean verifyPermissions(int[] grantResults) {

        for (int result : grantResults) {
            if (result != PackageManager.PERMISSION_GRANTED) {
                return false;
            }
        }
        return true;
    }
}
```

wifi 信号强度获取接口，结合广播及定时器，使程序每 30 秒获取一次附近 WiFi 信号强度。安卓 9 以后 wifimanager.startscan()方法被 Google 限制在前台应用 2 分钟最多只能调用 4 次以节省资源。而这 4 次可以在 2 分钟内的任意时间，所以为了使 4 次调用权限更合理，需要合理分配调用方法的时机。

```
private void startTimeTask() {
    if (timer == null) {
        timer = new Timer();
    }
    if (timerTask == null) {
        timerTask = (TimerTask) () -> {
                ((WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE)).startScan();
        };
    }
    timer.schedule(timerTask, delay: 0, period: 10*1000);
}

private void obtainWifiInfo()
{
    WifiManager wifiManager = (WifiManager) getApplicationContext().getSystemService(WIFI_SERVICE);
    WifiInfo info = wifiManager.getConnectionInfo();
    List<ScanResult> results = wifiManager.getScanResults();

    for (ScanResult result : results) {
        if (rssiMap.containsKey(result.BSSID)) {
            rssiMap.put(result.BSSID,result.level);
        }
    }

}
public BroadcastReceiver rssiReceiver = (context, intent) -> {
        final String action = intent.getAction();
        if (action.equals(WifiManager.RSSI_CHANGED_ACTION) || action.equals(WifiManager.WIFI_STATE_CHANGED_ACTION)
                || action.equals(WifiManager.SCAN_RESULTS_AVAILABLE_ACTION)) {
            SharedPreferences settings=getSharedPreferences( name: "setting", mode: 0);
            String phone_number= settings.getString( key: "ph", defValue: "");
            obtainWifiInfo();
            sendWiFi(phone_number,rssiMap);
        }
};
```