《2023秋季 Android平台开发技术》

大作业项目总结报告

1 + 北航运动助手

组号 27（小组成员按姓名字典序排列）

组员1 黄泓亮 21371227

组员2 李嘉鹏 21373216

组员3 宁 然 21371455

组员4 潘卓然 21371249

组员5 张博豪 21371173

2023年12月

一、功能要求

本作业要求完成的功能有：

- 支持北航校内常见运动的运动记录。
- 支持活动的团体组局。
- 攻略分享。
- 活动推荐。
- City Walk。
- 卡路里检测。
- 健身小计划。
- 其他自定义选做计划。

二、已完成的任务

必做任务完成情况（已完成数量/必做任务总数 = 3 / 3）

- 支持北航校内常见运动的运动记录。
- 支持活动的团体组局。
- 攻略分享。

选做任务完成情况（已完成数量/选做任务总数 = 5 / 5）

- 活动推荐
- City Walk
- 卡路里检测
- 健身小计划
- 其他自定义选做任务
    - 较为完善的个人信息管理功能，用户可自行设置有关个人信息并对其进行修改，个人信息具体有：头像，生日，身高，体重，个性签名等。
    - 用户账号信息管理功能，具体有：登陆，注册，修改密码，注销账户等功能。
    - 运动收藏功能。用户可以对自己喜欢的运动进行收藏，推荐时会优先考虑用户的偏好和兴趣。
    - 添加了运动风险须知，模拟了app隐私政策等栏目，增强app的真实性。
    - 实现了攻略评论功能，用户可在其他人的帖子下跟帖评论
    - 实现了组局内交流功能，在组局详情页面放置了聊天框，方便用户间交流协商
    - 实现了攻略搜索功能，用户键入关键词可搜索到相关的帖子
    - 实现了组局检索功能，用户键入关键词可检索到相关的活动
    - 添加了天气以及气温的真实的实时播报（调用外部接口）功能，有利于用户更好地选择合适的运动，也能以此为参考给用户提供更加人性化的活动推荐
    - 添加了运动日历，也就是历史运动记录的可视化功能，参考 keep ，完成了日历与历史运动记录的搭配展示，为用户提供更清晰更直接的回顾自己往日的运动记录的方式

- 添加了消息播报功能。鉴于本作业是为北航学生服务的，故添加了消息播报功能，用于实时播报校内和运动相关的活动的通知（比如沙河冬季长跑节，各个体育馆的开放时间，体育相关考核的时间等）

三、总体设计方案

本部分我们首先阐释我们的主要功能的实现逻辑，核心代码和实现效果图，然后再讲解一下我们的数据库设计策略。

1. 支持运动记录

1.1 实现逻辑

- 运动记录的**存入**：

  用户点击"开始运动" ⇒ 用户选择需要进行的运动 ⇒ 开始运动计时 ⇒ 用户结束运动计时 ⇒ 将与本次运动有关的信息（当前用户用户名、运动种类、得分、公里数、重量级、运动时间、参与人、备注、本次运动评分、结束运动的时间等）存入数据库中，其中运动种类包括跑步、徒步、健身、TD、游泳、篮球、网球、乒乓球、台球、羽毛球、排球、足球和飞盘等。

- 运动记录的**展示**：
  - 用户可在首页的日历中查看当前月或者以往某个月的某一天的运动简要记录，每次查看都会从数据库中获取当前浏览月份的每一天的运动信息，并渲染在首页，包括当天的总运动时间以及当天的总运动次数，并且能够在日历中可视化地看到在一个月的哪些日子里有过运动，便于用户安排运动的频率。
  - 用户在自己的个人主页可以看到自己的历史运动记录（包括运动名，运动进行的时间，对当次运动的备注，当次运动持续的时长）。当用户在个人主页选择了历史记录日期后，会执行从数据库中获取对应日期下的运动记录项的函数，然后前端会调用函数updateHistoryUi(date)函数将这些信息更新到UI界面上从而展示在用户眼前。

1.2 核心代码

- **运动记录的存入**

  前端通过自己设计的 Timer 类来进行运动的计时以及对前端的实时渲染，从而达到用户友好的计时界面，同时在运动结束后会自动记录已经运动的时间，根据用户自身的需求来输入对应的运动信息，如参与人，备注等，然后存入数据库中，供后续历史记录的展示以及运动日历的维护使用

```
void readyToSave() {
    mBtnComplete.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Date date = new Date();   // 获取运动结束时间

            int score = 0;   // 用于存储 得分、重量级或公里数
            if (mEtScore.getText().toString().length() != 0) {
                score = Integer.valueOf(mEtScore.getText().toString());
            }

            String partner = "Single";  // 参与人
            if (mEtPartner.getText().toString().length() != 0) {
                partner = mEtPartner.getText().toString();
            }

            String note = "Default";  // 备注
```

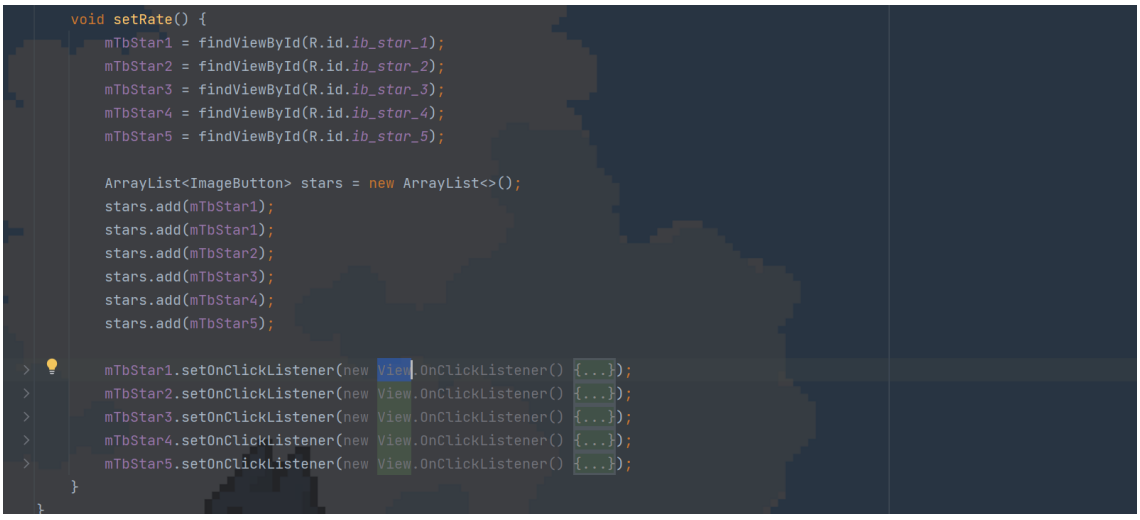
```
            if (mEtNote.getText().toString().length() != 0) {
                note = mEtNote.getText().toString();
            }

            SportRecord sportRecord = new SportRecord(sport_name
                    , sport_time, date, score
                    , partner, note, rate);

            // 将运动记录存入数据库
            addRecordToDB(MainActivity.getCurrentUsername(), sportRecord);
            Tools.toastMessageShort(CompleteSportActivity.this, "运动已记录");
            Intent intent = new Intent(CompleteSportActivity.this,
HomepageActivity.class);
            startActivity(intent);
        }
    });
}
```

除此之外，还做了类似星级评分的布局，可视化地实现对每次运动的评分

由于代码过长，故以图片形式展现大致框架

交互友好的评分逻辑：

- 倘若当前点击的星星未点亮，则点亮该星星以及更低级的星星，熄灭更高级的星星，评分为当前点击的星星的评分
- 倘若当前点击的星星已点亮，则熄灭该行星，评分为 $当前点击的星星评分 - 1$



- **运动记录的展示**
  - 运动日历的实现

    对于当天，日历上会有不同于其它样式的渲染，用以标注当前的日期，同时对于当前浏览的月份，若某天有运动记录，则会标红，以示这一天有过运动，并且当选择到这一天时，会显示这一天的运动次数以及运动总时间

```
void setCalendar(View view) {

    Date date = new Date();
    mCvCalendar.setDateSelected(date, true);
    calendarDay = new CalendarDay(date);

    Drawable drawable =
getResources().getDrawable(R.drawable.bg_current_day);
```

```
    CurrentDayDecorator currentDayDecorator = new
CurrentDayDecorator(calendarDay, drawable);
    setThisMonthDecorator(calendarDay.getYear(),
calendarDay.getMonth()%12 + 1);
    mCvCalendar.addDecorator(currentDayDecorator); // 设置当前日期提示的样式
信息

    // 获取选择的日期的运动次数
    int frequency =
getFrequencyInDate(MainActivity.getCurrentUsername(),
calendarDay.getYear(), calendarDay.getMonth()%12 + 1,
calendarDay.getDay());
    mTvDateFrequency.setText(String.valueOf(frequency));

    // 获取选择的日期的运动总时间
    String str = "%d h %d m %d s";
    int sumTime = getSportTimeInDate(MainActivity.getCurrentUsername(),
calendarDay.getYear(), calendarDay.getMonth()%12 + 1,
calendarDay.getDay());
    str = String.format(str, sumTime/3600, sumTime%3600/60, sumTime%60);

    mTvDateSumTime.setText(str);
    if (frequency == 0) {
        LL_LL.setVisibility(View.INVISIBLE);
        mTvEmpty.setText("今天还没有运动哦！健康生活，从现在开始！");
        LL_LL_empty.setVisibility(View.VISIBLE);
    } else {
        LL_LL.setVisibility(View.VISIBLE);
        LL_LL_empty.setVisibility(View.INVISIBLE);
    }

    mCvCalendar.setOnDateChangedListener(new OnDateSelectedListener() {
        @Override
        public void onDateSelected(@NonNull MaterialCalendarView widget,
@NonNull CalendarDay date, boolean selected) {
            int year = date.getYear();
            int month = date.getMonth()%12 + 1;
            int day = date.getDay();

            // 获取选择的日期的运动次数
            int frequency =
getFrequencyInDate(MainActivity.getCurrentUsername(), year, month, day);
            mTvDateFrequency.setText(String.valueOf(frequency));

            // 获取选择的日期的运动总时间
            String str = "%d h %d m %d s";
            int sumTime =
getSportTimeInDate(MainActivity.getCurrentUsername(), year, month, day);
            str = String.format(str, sumTime/3600, sumTime%3600/60,
sumTime%60); //

            mTvDateSumTime.setText(str);
            if (frequency == 0) {
                LL_LL.setVisibility(View.INVISIBLE);
                mTvEmpty.setText("今天还没有运动哦！健康生活，从现在开始！");
```

```
                if (calendarDay.getYear() != date.getYear() ||
calendarDay.getMonth() != date.getMonth() || calendarDay.getDay() !=
date.getDay()) {
                    mTvEmpty.setText("这一天没有运动哦！坚持才是胜利！");
                }
                LL_LL_empty.setVisibility(View.VISIBLE);
            } else {
                LL_LL.setVisibility(View.VISIBLE);
                LL_LL_empty.setVisibility(View.INVISIBLE);
            }
        }
    });
    mCvCalendar.setOnMonthChangedListener(new OnMonthChangedListener() {
        @Override
        public void onMonthChanged(MaterialCalendarView widget,
CalendarDay date) {
            int newYear = date.getYear();
            int newMonth = date.getMonth() % 12 + 1;
            setThisMonthDecorator(newYear, newMonth); // 日历月视图改变时，
重新从数据库获取新的月份的运动记录信息
        }
    });
}
```

- 个人主页历史记录的实现

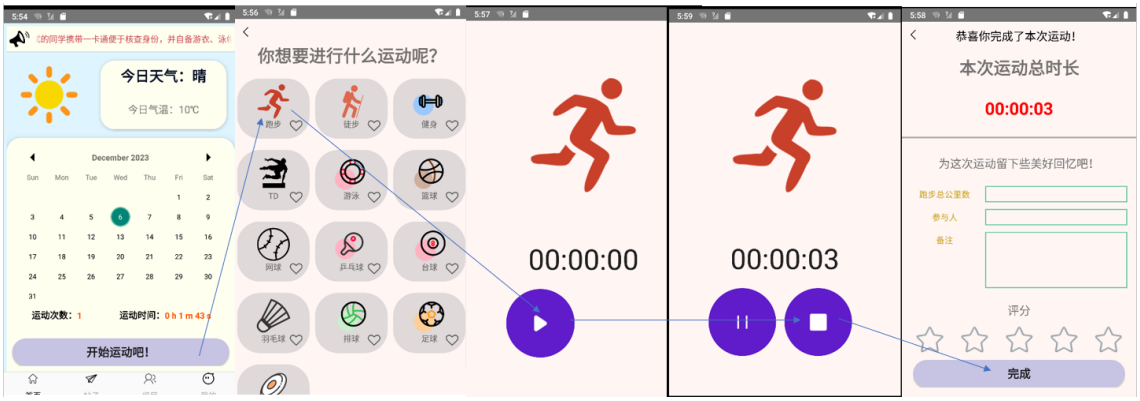
```
// 日期选择期器的点击监听器设置
DatePickerDialog datePickerDialog = new DatePickerDialog(getActivity(),
new DatePickerDialog.OnDateSetListener() {
        @Override
        // 当用户点击某个日期时，会调这个函数，然后后三个参数自动传进来用户点的日期
        public void onDateSet(DatePicker datePicker, int i, int i1, int
i2) {
            // 日期选择器的月份是从0开始算的，所以这里要加1来让他符合习惯
            String newDate = String.format("%04d-%02d-%02d", i, i1 + 1,
i2);
            dateTextView.setText(newDate);
            // 要更新适配器
            updateHistoryUi(newDate);
        }
    }, year, month, day); // 这里的三个年月日是在显示时日历的初始年月日
    datePickerDialog.show();
```

```
// 更新历史运动ui界面
// 该函数根据参数传入的日期，将日期选择器和历史记录展示区的内容全部都更新到与当前日期
对应的状态
public void updateHistoryUi(String date) {
    // 根据日期date更新 dateTextView 和 历史记录容器
    dateTextView.setText(date); // 默认时间为当前时间
    List<MyFragment.HistoryShowItem> items = DBFunction.getUserHistory(
        MainActivity.getCurrentUsername(), date);
    historyShowItems = new ArrayList<>(items);
    // 更新里面的imageId
    for (HistoryShowItem item : historyShowItems) {
```

```
        item.setItemImageId(Tools.sportNameMapLogoId.get(item.getItemTitle()));
        if (item.getItemDesc().equals("Default")) {
            item.setItemDesc("暂无备注");
        }

        item.setItemTitle(Tools.sportNameMapChinese.get(item.getItemTitle()));
        item.setItemTotalTime(translate(item.getItemTotalTime()));
    }
    setEmptyOrNot();
}
```

1.3 具体实现图

- **运动记录存入流程**



- **运动记录的展示**
  - 运动日历

  - 个人主页历史记录

2. 卡路里检测

2.1 实现逻辑

大体逻辑为，我们搜集资料提供了一些食物的卡路里数值（我们搜集了多达数百种食物），用户可以选择我们提供的食物，参考其卡路里值。我们在代码中用HashMap等数据结构将和食物卡路里相关的信息存下来。

当用户点击前端选择食物类型或者具体食物的下拉框并作出选择后，在下拉框上设置的监听器中的代码会执行操作更新ui界面上（具体地，比如用`TextView.setText()`方法来更新文本框的内容）显示的卡路里值数，相关的健康建议以及类型图片等。

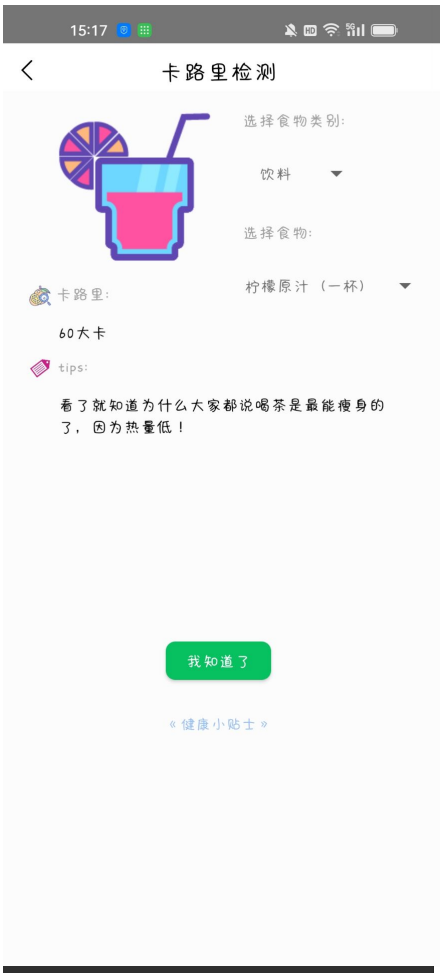
2.2 核心代码

```
// 依次添加卡路里信息
calorieInfo.get("主食").add(new CalorieInfoItem("白饭(140g)", "210大卡"));
typeMapFoodNames.get("主食").add("白饭(140g)");
foodNameMapCalorie.put("白饭(140g)", "210大卡");
calorieInfo.get("主食").add(new CalorieInfoItem("白馒头（一个）", "280大卡"));
typeMapFoodNames.get("主食").add("白馒头（一个）");
foodNameMapCalorie.put("白馒头（一个）", "280大卡");
calorieInfo.get("主食").add(new CalorieInfoItem("煎饼（100g）", "333大卡"));
typeMapFoodNames.get("主食").add("煎饼（100g）");
foodNameMapCalorie.put("煎饼（100g）", "333大卡");
... // 后面还有很多，这里省略了
```

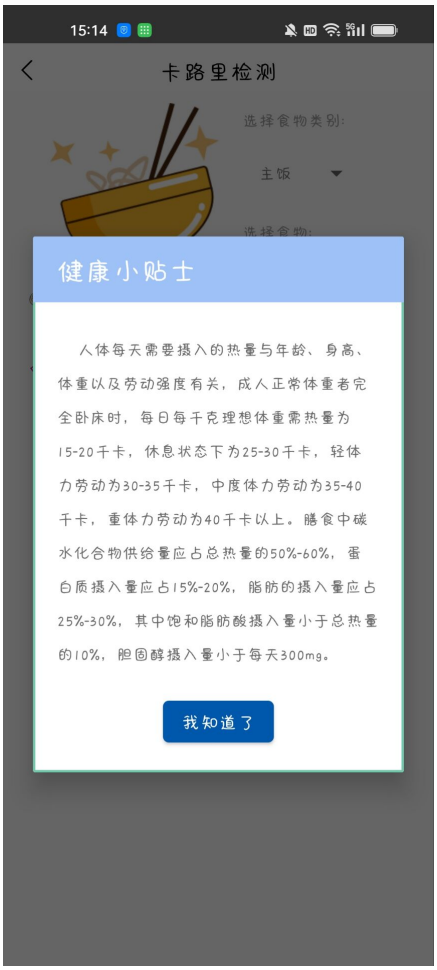
```
// <食物类型>选择下拉框的监听器设置
selectTypeSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long
l) {
        // 更新选择食物的spinner的素材
        String type = (String) selectTypeSpinner.getSelectedItem();
        // 更新food列表并将初始项设置为0项
        updateFoodAdapter(type);
        String name = (String) selectFoodSpinner.getSelectedItem();
        setImageCalorieTips(type, name);
    }
});
```

```
// <食物名字>选择下拉框的监听器设置
selectFoodSpinner.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
@Override
public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l)
{
    // 更新食物卡路里文本框
    String foodName = (String) selectFoodSpinner.getSelectedItem();
    calorieText.setText(foodNameMapCalorie.get(foodName));
}
```

2.3 具体实现图


15:17
卡路里检测
选择食物类别:
饮料
选择食物:
柠檬原汁（一杯）
卡路里:
60大卡
tips:
看了就知道为什么大家都说喝茶是最能瘦身的了，因为热量低！
我知道了
« 健康小贴士 »


15:14
卡路里检测
选择食物类别:
主饭
选择食物:
咖喱饭
卡路里:
640大卡
tips:
带炒的饭或者面的热量都很高，想减肥的孩纸还是尽量少吃煎炒类的食物最好！
我知道了
« 健康小贴士 »


15:14
卡路里检测
选择食物类别:
主饭
健康小贴士
人体每天需要摄入的热量与年龄、身高、体重以及劳动强度有关，成人正常体重者完全卧床时，每日每千克理想体重需热量为15-20千卡，休息状态下为25-30千卡，轻体力劳动为30-35千卡，中度体力劳动为35-40千卡，重体力劳动为40千卡以上。膳食中碳水化合物供给量应占总热量的50%-60%，蛋白质摄入量应占15%-20%，脂肪的摄入量应占25%-30%，其中饱和脂肪酸摄入量小于总热量的10%，胆固醇摄入量小于每天300mg。
我知道了

3. 个人信息管理

3.1 实现逻辑

这个的实现逻辑比较简单，就是在个人信息界面用`TextView`等控件展示用户的身高体重生日，以及头像等个人信息。并为显示这些信息的控件设置点击监器，在点击时可以以对话框或者跳转到新`activity`的形式为用户提供对个人信息修改的机会，比如：

- 对于个性签名的修改，在用户点击保存后，代码会读取`EditView`组件的内容来得到用户的输入信息，然后调用数据库函数`setPersonalSign`来更新该用户在数据库中存的个性签名的信息），用户在前端修改了个人信息后，会调用数据库的相关函数将用户的修改写入数据库中。
- 对于身高的修改，在用户点击显示身高的控件后会弹出来一个对话框，对话框是上面有数字选择器，用户可以滑动数字选择器来选择要设置的身高，选择后点击保存，这时候会触发监听器逻辑调用后端函数将设置的身高写入内存。
- 对于个人头像的管理。我们准备了几十张精美的头像可供用户选择。实现逻辑为：用户在个人主页点击自己的头像，触发选择头像的监听逻辑，进入到相应的界面，在选择界面，我们用`RecyclerView`的瀑布流效果来进行待选头像的排布，用户随意点击一款头像，触发设置头像的监听逻辑，此时调用数据库函数`setUserHeadImage`更新数据库中该用户的头像信息并同时对个人主页上的用户头像进行实时更新。

3.2 核心代码

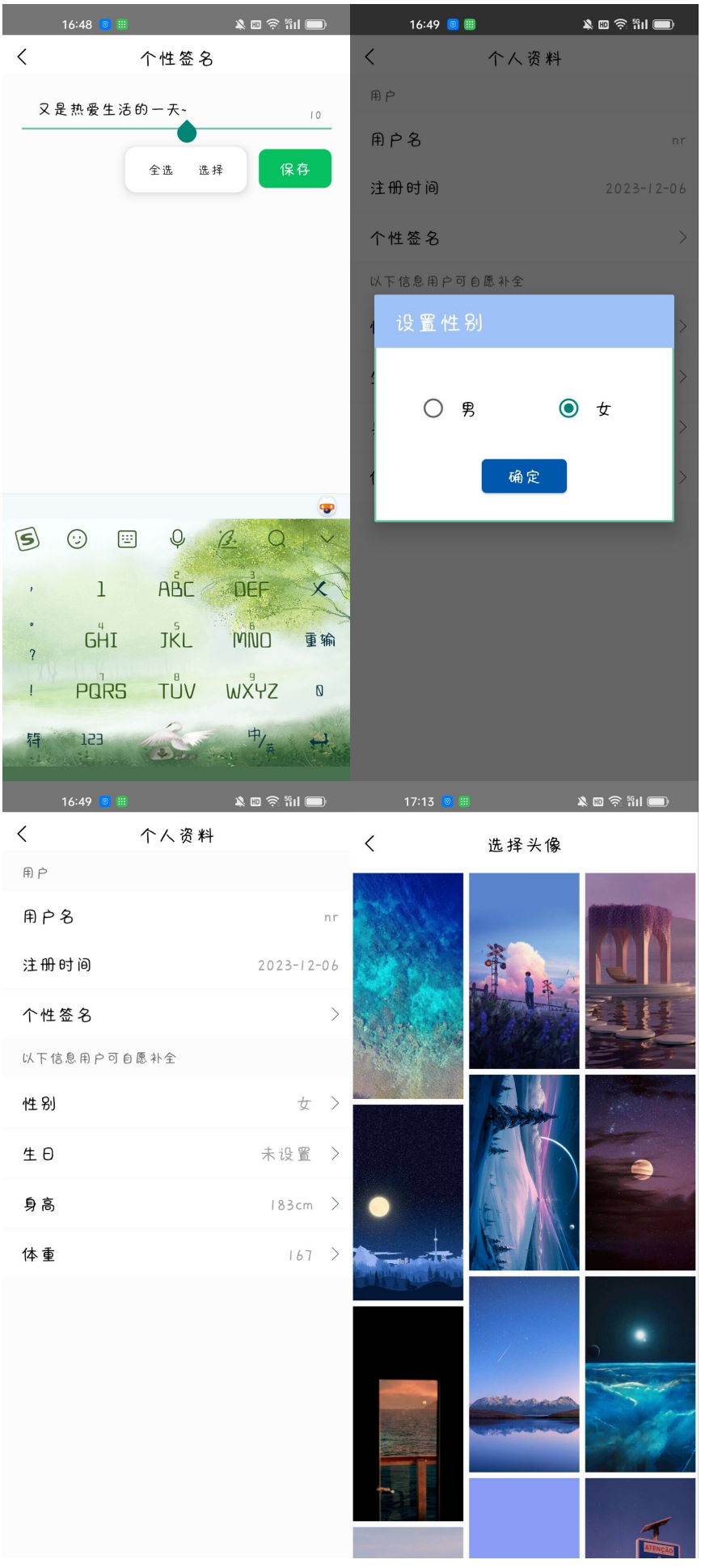
考虑到代码逻辑较为一致，这里仅仅挑选和部分信息相关的代码进行展示。

```
// 保存修改个性签名的核心逻辑。
saveButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        String newSign = inputEditText.getText().toString();
        DBFunction.setPersonalSign(MainActivity.getCurrentUsername(), newSign);
        // 回显新的个性签名
        MyFragment.setPersonalSignTextView(newSign);
        finish();
    }
});
```

```
// 这是点击<保存>后进行的有关身高的函数
void pickHeightAction(int curNumber) {
    heightTextView.setText(curNumber + "cm");
    DBFunction.setUserHeight(MainActivity.getCurrentUsername(), curNumber);
}
```

```
// 这是在选择头像界面任意点击一款头像后触发的逻辑
holder.getItemHeadImage().setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // FIXME-1 将此用户的头像更新为点击到的 void setHeadImage(String username, int
imageId)
        DBFunction.setHeadImage(MainActivity.getCurrentUsername(),
headImageItem.getItemImageId());
        Bitmap originalBitmap =
BitmapFactory.decodeResource(parent.getResources(),
            headImageItem.getItemImageId());
        Bitmap circularBitmap = Tools.getCircularBitmap(originalBitmap);
        MyFragment.getHeadImage().setImageBitmap(circularBitmap);
        parent.finish();
    }
});
```

3.3 具体实现图


16:48
个性签名
又是热爱生活的一天~
10
全选
选择
保存
16:49
个人资料
用户
用户名
nr
注册时间
2023-12-06
个性签名
以下信息用户可自愿补全
设置性别
男
女
确定
16:49
个人资料
用户
用户名
nr
注册时间
2023-12-06
个性签名
以下信息用户可自愿补全
性别
女
生日
未设置
身高
183cm
体重
167
17:13
选择头像

4. 用户账号管理

4.1 实现逻辑

登陆，注册，修改密码，注销账号。（前三者都有检查用户输入格式，或者输入是否为空的功能，如果不符合格式或者输入为空，在点击相关按钮时会弹窗提醒，这一点下面不再重复赘述）

- 登陆：用户在登陆界面的帐号密码文本框中输入自己的信息，点击登陆，触发登陆按钮监听器逻辑，调用后端函数`isAllowLogin`验证输入的账号密码是否对应正确，正确则可成功跳转到应用内部界面。
- 注册：用户输入注册的用户名和密码，然后点击注册按钮，触发点击监听器逻辑，首先用后端函数`isUserNameExist`来判断该用户名是否存在，如果不存在，那么就用后端函数`addUser`来将这个用户名和密码存入数据库中，完成注册的效果。
- 修改密码：首先要求用户在界面上输入当前的密码以便能验证用户的身份，然后要求用户输入新密码并确认新密码，然后点击"修改密码"，这将触发此按钮上的监听器逻辑，其会调用后端函数`changPassword`将该用户在数据库中存的密码值做出相应的修改以达到目的。
- 注销账号：用户点击注销账号，触发点击监听器事件，调用后端函数`cancelUser`将用户的信息从数据库中彻底删除。

4.2 核心代码

```
// 尝试进行登陆的函数
public void tryLogin(String username, String password) {
    LoginCheckResult res = checkLoginUser(username, password);
    switch (res) {
        case NO_USERNAME: Tools.toastMessageShort(MainActivity.this, "请输入用户
名"); break;
        case NO_PASSWORD: Tools.toastMessageShort(MainActivity.this, "请输入密码");
break;
        case USER_PASSWORD_NOT_MATCH: Tools.toastMessageShort(MainActivity.this,
"用户名和密码不匹配"); break;
        default: {
            // 登录成功，跳转到目标界面
            currentUsername = username;
            startActivity(new Intent(MainActivity.this, HomepageActivity.class));
            Tools.toastMessageShort(MainActivity.this, "登录成功!");
            finish();
        }
    }
}
```

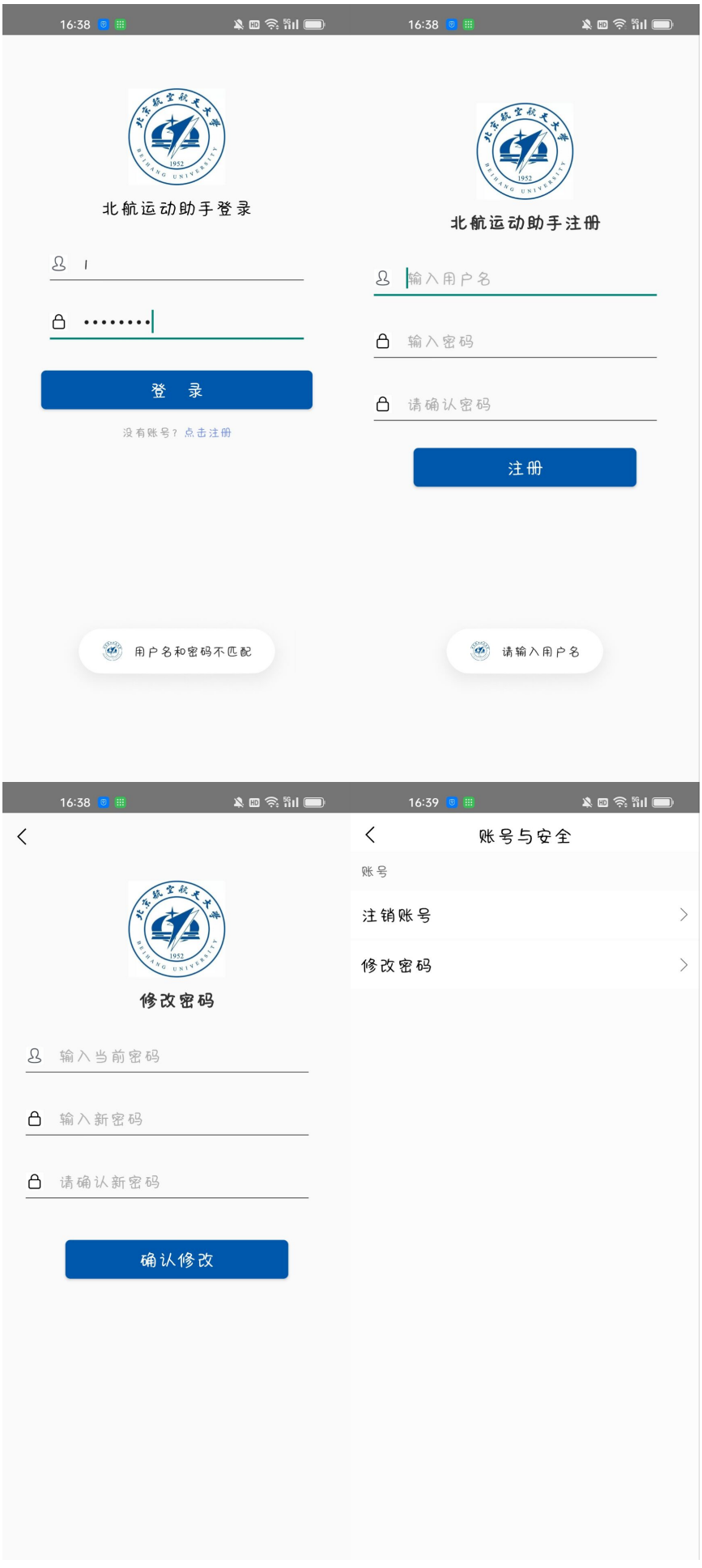
```
public void tryRegister(String username, String password1, String password2) {
    RegisterCheckResult res = checkRegisterUser(username, password1, password2);
    switch (res) {
        case NO_USERNAME: toastThis(NO_USERNAME_TOAST); break;
        case USERNAME_FORMAT_ERROR: toastThis(USERNAME_FORMAT_ERROR_TOAST);
break;
        case NO_PASSWORD: toastThis(NO_PASSWORD_TOAST); break;
        case NO_AGAIN_PASSWORD: toastThis(NO_AGAIN_PASSWORD_TOAST); break;
        case PASSWORD_NOT_SAME: toastThis(PASSWORD_NOT_SAME_TOAST); break;
        case PASSWORD_TOO_SHORT: toastThis(PASSWORD_TOO_SHORT_TOAST); break;
        case USERNAME_EXIST: toastThis(USERNAME_EXIST_TOAST); break;
```

```
        default: {
            // 跳转到登录界面
            Calendar calendar = Calendar.getInstance();
            int year = calendar.get(Calendar.YEAR);
            int month = calendar.get(Calendar.MONTH) + 1;
            int day = calendar.get(Calendar.DAY_OF_MONTH);
            String date = String.format("%04d-%02d-%02d", year, month, day);
            DBFunction.addUser(username, password1, date);
            startActivity(new Intent(RegisterActivity.this, MainActivity.class));
            toastThis("注册成功!");
            finish();
        }
    }
}
```

```
// 尝试修改密码的函数
public void tryChange(String currentPassword, String password1, String password2)
{
    ChangePasswordCheckResult res = checkChangePassword(currentPassword,
password1, password2);
    switch (res) {
        case NO_CURRENT_PASSWORD: toastThis(NO_CURRENT_PASSWORD_TOAST); break;
        case CURRENT_PASSWORD_ERROR: toastThis(CURRENT_PASSWORD_ERROR_TOAST);
break;
        case NO_PASSWORD: toastThis(NO_PASSWORD_TOAST); break;
        case NO_AGAIN_PASSWORD: toastThis(NO_AGAIN_PASSWORD_TOAST); break;
        case PASSWORD_NOT_SAME: toastThis(PASSWORD_NOT_SAME_TOAST); break;
        case PASSWORD_TOO_SHORT: toastThis(PASSWORD_TOO_SHORT_TOAST); break;
        default: {
            DBFunction.changePassword(MainActivity.getCurrentUsername(),
password1);
            // 跳转到登录界面
            Intent intent = new Intent(ChangePasswordActivity.this,
HomepageActivity.class);
            intent.putExtra("fragmentToShow", "我的");
            startActivity(intent);
            toastThis("修改成功!");
        }
    }
}
```

```
// 注销账号的函数
cancelAccountButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Intent intent = new Intent(AccountSafeActivity.this, MainActivity.class);
        // 从数据库中删除用户信息
        DBFunction.cancelUser(MainActivity.getCurrentUsername());
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK |
Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
        finish();
    }
});
```

4.3 具体实现图


16:38
北航运动助手登录
登 录
没有账号？点击注册
用户名和密码不匹配
16:38
北航运动助手注册
输入用户名
输入密码
请确认密码
注册
请输入用户名
16:38
修改密码
输入当前密码
输入新密码
请确认新密码
确认修改
16:39
账号与安全
账号
注销账号
修改密码

5. 运动收藏管理

5.1 实现逻辑

为了更好的为推荐运动服务，以及为用户提供更加个性化的功能，所以实现了运动收藏的功能，对于每一项运动，用户都可以进行收藏，同时也可以取消收藏。

- 添加收藏：当用户在开始运动界面点击“红心”，就会调用数据库函数将该红心对应的运动添加到该人的收藏运动清单中。
- 查看收藏：在个人主页可以点击相关图标进入到《我的收藏》界面查看相关运动的信息。具体地，在进入到该界面后，会调用后端函数`getUserStars`从数据库中获取到此用户收藏的运动的信息，然后我们采用`RecyclerView`这一常用的控件来将收集到的信息展示出来。此外，在开始运动的界面也可以清晰地看到自己对哪些活动进行了收藏。
- 取消收藏：在《我的收藏》界面可以实现对收藏运动的取消收藏，具体实现逻辑为：用户点击红心时，红心消失，然后在退出《我的收藏》页面时，onDestroy方法被调用，其会检测出所有红心消失了的运动，并且将调用数据库函数`cancelStar`将这些活动依次从该用户的收藏列表中除去。此外，在开始运动界面，也可以通过这样的"熄灭红心"的方式来取消收藏。

5.2 核心代码

```
// 开始运动界面有关收藏的逻辑（添加或取消收藏），以徒步为例
void setWalk() {
    mLLwalk = findViewById(R.id.ll_walk);
    mLLwalk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(SportkindActivity.this,
WalkSportActivity.class);
            intent.putExtras(sendBundle("walk"));
            startActivity(intent);
        }
    });

    mIbwalk = findViewById(R.id.heart_walk);
    if (ifStared(MainActivity.getCurrentUsername(), "walk")) { // 获取初始收藏信息，
用于前端渲染
        mIbwalk.setBackgroundResource(R.drawable.ic_heart);
    } else {
        mIbwalk.setBackgroundResource(R.drawable.ic_heart_empty);
    }
    mIbwalk.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (ifStared(MainActivity.getCurrentUsername(), "walk")) {
                cancelStar(MainActivity.getCurrentUsername(), "walk"); // 若已收
藏，则取消收藏
            } else {
                addStar(MainActivity.getCurrentUsername(), "walk"); // 若未收藏，则
收藏
            }
            if (ifStared(MainActivity.getCurrentUsername(), "walk")) {
                mIbwalk.setBackgroundResource(R.drawable.ic_heart);
            } else {
                mIbwalk.setBackgroundResource(R.drawable.ic_heart_empty);
```

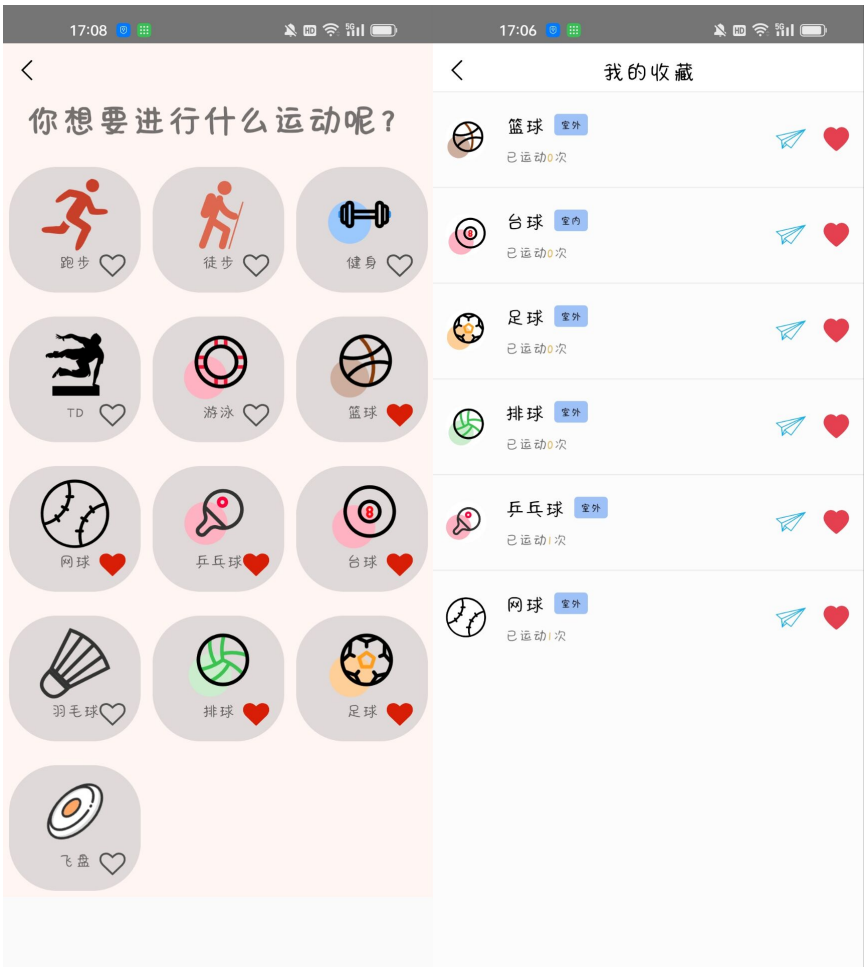
```
            }
        }
    });
}
```

```
// 个人主页界面有关收藏的逻辑：展示收藏列表
starItems = new ArrayList<>
(DBFunction.getUserStars(MainActivity.getCurrentUsername()));
// 上面执行过后主要执行这两句代码：
starsViewAdapter = new StarAdapter(starItems);
starsView.setAdapter(starsViewAdapter);
```

```
// 个人主页界面有关收藏的逻辑：取消收藏
@Override
protected void onDestroy() {
    super.onDestroy();
    System.out.println("退出了收藏界面！");
    // 这里要保存收藏的设置
    ArrayList<StarShowItem> afterItems = starsViewAdapter.getShowItems();
    // 找到其中取消收藏的项目(以运动名为唯一标志)
    for (StarShowItem item : afterItems) {
        if (!item.isRed()) {
            DBFunction.cancelStar(MainActivity.getCurrentUsername(),
                Tools.sportChineseNameMapEnglish.get(item.getItemTitle()));
        }
    }
}
```

5.3 具体实现图

点击前图中的小红心可以添加收藏，点两图中的红心都可以取消收藏。



6. 团体组局

这部分对应我们的 APP 的"组局开团"模块（GroupFragment），点击下边栏第三个按钮即可进入该页面。

6.1 实现逻辑

- **发起组局**
  该模块的逻辑链如下：
  - 用户点击 GroupFragment 页面的发起组局按钮
  - 触发 CreateGroupActivity 进入发起组局的页面
  - 填写组局的详细信息（活动名称、时间、地点、描述...）
  - 点击发起组局按钮，将信息写入数据库
  - 回到 GroupFragment 页面，依据数据库更新页面视图，较新的组局会显示在上部
- **浏览组局**
  在 GroupFragment 页面上，组局的简要信息显示在以 GridLayout 安排的色块视图上，用户可以点击感兴趣的色块，触发 GroupDetailActivity，进入组局详情页面；详情页面显示组局的活动主题、描述、地点、时间、人数限制、参与者和有意向组局者的交流信息。
- **组局相关操作**
  在 GroupDetailActivity 页面：
  - 用户可以点击加入组局按钮：
    - 若人数已达上限，则将被提示无法加入
    - 若人数未满，则用户加入该组局

        - 设置加入组局按钮为退出组局
        - 更新数据库，同步更新该页面的参与者视图
    - 用户可以点击退出组局按钮：
        - 若用户为组局的组织者
            - 若当前只有用户一人在该组局中，则用户可以退出，该活动的组织者设为"虚位以待"
            - 否则，组织者顺延到下一个参与者
        - 若用户不是组织者，则正常退出
        - 设置退出组局按钮为加入组局
        - 更新数据库、刷新视图
        - 返回到 GroupFragment 刷新视图
    - 若用户为组织者，可以点击取消组局按钮
        - 更新数据库
        - 返回到 GroupFragment 刷新视图
    - 若用户为组织者，可以点击完成按钮
        - 更新数据库
        - 返回到 GroupFragment 刷新视图
- **局内交流**
  用户可以在 GroupDetailActivity 页面，通过输入栏发布评论与其他人进行交流协商
  用户点击提交评论，数据库更新，GroupDetailActivity 的组局评论视图会进行刷新，较新的评论会显示在上部。
- **搜索组局**
  用户可以在 GroupFragment 页面上的搜索框键入关键词，点击搜索按钮，触发 SearchGroupResultActivity，在该页面上显示从数据库数据中筛选后的含关键词的活动，该页面上显示的色块也可点击进入，行为和进入 GroupDetailActivity 页面相同。
- **推荐组局**
  用户可以在 GroupFragment 页面上点击推荐组局按钮，触发 RecommendGroupActivity，在该页面上显示从数据库数据中依据推荐算法（见下文）筛选后的活动，该页面上显示的色块也可点击进入，行为和进入 GroupDetailActivity 页面相同。
- **我的组局**
  用户可以在 MyFragment 页面上点击我的组局按钮，触发 MyGroupActivity，在该页面上显示从数据库中筛选出的当前用户参与的组局，该页面上显示的色块也可点击进入，行为和进入 GroupDetailActivity 页面相同。

6.2 核心代码

- **发起组局**

```
// 按钮触发
    fabRecGroup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Start CreateGroupActivity
            Intent intent = new Intent(view.getContext(),
RecommendGroupActivity.class);
            startActivityForResult(intent, 2);
```

```
        }
    });

// 输入收集
    int joinerLimit = 1;
    if(editJoinerLimit.getText() != null &&
!editJoinerLimit.getText().toString().equals("")) {
        joinerLimit = Integer.parseInt(editJoinerLimit.getText().toString());
    }
    String sport = editSport.getText().toString();
    String description = editDescription.getText().toString();
    String location = editLocation.getText().toString();

// 返回更新
    @Override
    public void onActivityResult(int requestCode, int resultCode, @Nullable
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if(requestCode == 1 && resultCode == RESULT_OK){
            String newGroupString = data.getStringExtra("newGroup");
            GroupItem groupItem = new Gson().fromJson(newGroupString,
GroupItem.class);
            // 加入数据库
            DBFunction.addGroupToDB(groupItem);
        }
        groupItemList = DBFunction.getAllGroups();

        Log.d("TAG", "GroupItem in Adapter: " + "--------------------------------
");
        for (GroupItem groupItem : groupItemList) {
            Log.d("TAG", "GroupItem in Adapter: " + groupItem.getSport());
        }

        groupAdapter.updateAdapter(groupItemList);
        recyclerView.setAdapter(groupAdapter);
    }
```

- **浏览组局**

```
// 按钮触发
holder.itemView.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Start GroupDetailActivity and pass the groupItem
        Intent intent = new Intent(v.getContext(), GroupDetailActivity.class);
        if(op == 1) {
            intent = new Intent(v.getContext(),
GroupDetailForSearchActivity.class);
        } else if (op == 2) {
            intent = new Intent(v.getContext(), GroupDetailForMyActivity.class);
        } else if (op == 3) {
            intent = new Intent(v.getContext(), GroupDetailForRecActivity.class);
        }
        intent.putExtra("groupItem", new Gson().toJson(groupItem));
```

```
        ((Activity)v.getContext()).startActivityForResult(intent, 2);
    }
});


// 显示组局详细信息
TextView organizerTextView = findViewById(R.id.detailOrganizerTextView);
organizerTextView.setText("组织者: " + groupItem.getOrganizer());

TextView joinerLimitTextView = findViewById(R.id.detailJoinerLimitTextView);
joinerLimitTextView.setText("已加入人数: " + groupItem.getJoiners().size()
        + "/" + groupItem.getJoinerLimit());

TextView sportTextView = findViewById(R.id.detailSportTextView);
sportTextView.setText(groupItem.getSport());

// Add more TextViews and set their values based on groupItem
RecyclerView joinersRecyclerView = findViewById(R.id.joinersRecyclerView);
LinearLayoutManager layoutManager = new LinearLayoutManager(this);
joinersRecyclerView.setLayoutManager(layoutManager);
List<JoinerItem> joiners = new ArrayList<>();

for (String joinerName : groupItem.getJoiners()) {
    int image = DBFunction.getUserHeadImage(joinerName);
    Bitmap originalBitmap = BitmapFactory.decodeResource(getResources(), image);
    Bitmap circularBitmap = Tools.getCircularBitmap(originalBitmap);
    JoinerItem joinerItem = new JoinerItem(joinerName, circularBitmap);
    joiners.add(joinerItem);
}

JoinersAdapter joinersAdapter = new JoinersAdapter(joiners);
joinersRecyclerView.setAdapter(joinersAdapter);

TextView descriptionTextView = findViewById(R.id.descriptionTextView);
descriptionTextView.setText("活动简述: \n" + groupItem.getDescription());

TextView planStartTimeTextView = findViewById(R.id.planStartTimeTextView);
planStartTimeTextView.setText("开始时间: " + groupItem.getPlanStartTime());

TextView planEndTimeTextView = findViewById(R.id.planEndTimeTextView);
planEndTimeTextView.setText("结束时间: " + groupItem.getPlanEndTime());

TextView locationTextView = findViewById(R.id.locationTextView);
locationTextView.setText("活动地点: " + groupItem.getLocation());

Button btnJoinGroup = findViewById(R.id.btnJoinGroup);
Button btnCancelGroup = findViewById(R.id.btnCancelGroup);
Button btnCompleteGroup = findViewById(R.id.btnCompleteGroup);
```

- **组局相关操作**

```
// 按钮和动作
String currentUser = MainActivity.getCurrentUsername();
if (groupItem.hasJoiner(currentUser)) {
    btnJoinGroup.setText("💧 退出活动");
```

```
} else if (groupItem.getJoiners().size() >= groupItem.getJoinerLimit()) {
    btnJoinGroup.setText("人数已满");
}

btnJoinGroup.setOnClickListener(new View.OnClickListener() {

    @SuppressLint({"SetTextI18n", "NotifyDataSetChanged"})
    @Override
    public void onClick(View view) {

        // 判断用户是否已加入组局
        if (groupItem.hasJoiner(currentUser)) {
            boolean removed = groupItem.removeJoiner(currentUser);

            if (removed) {
                // 退出成功，更新界面
                DBFunction.removeJoinerFromGroup(currentUser,
groupItem.getGroupKey());
                List<JoinerItem> joinerItems = new ArrayList<>();
                for (User joiner :
DBFunction.getJoinersFromGroup(groupItem.getGroupKey())) {
                    Bitmap originalBitmap =
BitmapFactory.decodeResource(getResources(), joiner.getHeadImage());
                    Bitmap circularBitmap =
Tools.getCircularBitmap(originalBitmap);
                    JoinerItem joinerItem = new JoinerItem(joiner.getUsername(),
circularBitmap);
                    joinerItems.add(joinerItem);
                }
                joinersAdapter.updateData(joinerItems);

                if (Objects.equals(groupItem.getOrganizer(), "Wait for an
organizer...")) {
                    organizerTextView.setText("组织者: " +
groupItem.getOrganizer());
                }
                joinerLimitTextView.setText("已加入人数: " +
groupItem.getJoiners().size()
                        + "/" + groupItem.getJoinerLimit());
                // 更新按钮文字
                btnJoinGroup.setText("🔥 加入活动");
                btnCancelGroup.setVisibility(View.INVISIBLE);
                btnCompleteGroup.setVisibility(View.INVISIBLE);
                GroupFragment.groupItemList = DBFunction.getAllGroups();

 GroupFragment.groupAdapter.updateAdapter(GroupFragment.groupItemList);
            } else {
                // 退出失败，显示提示信息或执行相应操作
                Toast.makeText(GroupDetailActivity.this, "退出活动失败",
Toast.LENGTH_SHORT).show();
            }
        } else if (groupItem.getJoiners().size() >= groupItem.getJoinerLimit()) {
            btnJoinGroup.setText("人数已满");
            Toast.makeText(GroupDetailActivity.this, "该活动人数已满",
Toast.LENGTH_SHORT).show();
        } else {
```

```
            boolean newOrg = false;
            if (Objects.equals(groupItem.getOrganizer(), "Wait for an
organizer...")) {
                newOrg = true;
            }

            // 用户未加入，尝试加入组局 //todo 修改数据库
            boolean joined = groupItem.addJoiner(currentUser);

            if (joined) {
                DBFunction.addJoinerToGroup(currentUser,
groupItem.getGroupKey());
                // 加入成功，更新界面
                if (newOrg) {
                    organizerTextView.setText("组织者: " +
groupItem.getOrganizer());
                    btnCancelGroup.setVisibility(View.VISIBLE);

                    btnCancelGroup.setOnClickListener(new View.OnClickListener()
{
                        @Override
                        public void onClick(View view) {
                            // TODO: 在数据库中删除该活动

 DBFunction.removeGroupFromDB(groupItem.getGroupKey());
                            Toast.makeText(GroupDetailActivity.this, "活动已取消",
Toast.LENGTH_SHORT).show();
                            List<GroupItem> newlist = DBFunction.getAllGroups();
                            GroupFragment.groupAdapter.updateAdapter(newlist);
                            setResult(RESULT_OK);
                            finish();
                        }
                    });

                    btnCompleteGroup.setVisibility(View.VISIBLE);
                    btnCompleteGroup.setOnClickListener(new
View.OnClickListener() {
                        @Override
                        public void onClick(View view) {
                            // TODO: 在数据库中删除该活动

 DBFunction.removeGroupFromDB(groupItem.getGroupKey());
                            Toast.makeText(GroupDetailActivity.this, "活动已完成",
Toast.LENGTH_SHORT).show();
                            List<GroupItem> newlist = DBFunction.getAllGroups();
                            GroupFragment.groupAdapter.updateAdapter(newlist);
                            setResult(RESULT_OK);
                            finish();
                        }
                    });
                }
                joinerLimitTextView.setText("已加入人数: " +
groupItem.getJoiners().size()
                        + "/" + groupItem.getJoinerLimit());

                List<JoinerItem> joinerItems = new ArrayList<>();
```

```
                for (User joiner :
DBFunction.getJoinersFromGroup(groupItem.getGroupKey())) {
                    Bitmap originalBitmap =
BitmapFactory.decodeResource(getResources(), joiner.getHeadImage());
                    Bitmap circularBitmap =
Tools.getCircularBitmap(originalBitmap);
                    JoinerItem joinerItem = new JoinerItem(joiner.getUsername(),
circularBitmap);
                    joinerItems.add(joinerItem);
                }
                joinersAdapter.updateData(joinerItems);

                // 更新按钮文字
                btnJoinGroup.setText("💧 退出活动");
                GroupFragment.groupItemList = DBFunction.getAllGroups();

 GroupFragment.groupAdapter.updateAdapter(GroupFragment.groupItemList);
            } else {
                // 人数已满，显示提示信息或执行相应操作
                Toast.makeText(GroupDetailActivity.this, "该活动人数已满",
Toast.LENGTH_SHORT).show();
            }
        }
    }
});

if (currentUser.equals(groupItem.getOrganizer())) {
    // 当前用户是组织者，显示取消活动按钮
    btnCancelGroup.setVisibility(View.VISIBLE);

    btnCancelGroup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // 在数据库中删除该活动
            DBFunction.removeGroupFromDB(groupItem.getGroupKey());
            Toast.makeText(GroupDetailActivity.this, "活动已取消",
Toast.LENGTH_SHORT).show();
            List<GroupItem> newlist = DBFunction.getAllGroups();
            GroupFragment.groupAdapter.updateAdapter(newlist);
            setResult(RESULT_OK);
            finish();
        }
    });

    btnCompleteGroup.setVisibility(View.VISIBLE);
    btnCompleteGroup.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // 在数据库中删除该活动
            DBFunction.removeGroupFromDB(groupItem.getGroupKey());
            Toast.makeText(GroupDetailActivity.this, "活动已完成",
Toast.LENGTH_SHORT).show();
            List<GroupItem> newlist = DBFunction.getAllGroups();
            GroupFragment.groupAdapter.updateAdapter(newlist);
            setResult(RESULT_OK);
            finish();
```

```
        }
    });

}
```

- **局内交流**

```
// 按钮触发
public void onAddCommentButtonClick(View view) {
    // 获取评论内容
    EditText commentEditText = findViewById(R.id.groupCommentEditText);
    String commentContent = commentEditText.getText().toString();

    if (!commentContent.isEmpty()) {
        commentContent = SensitiveWordFilter.Filter(commentContent);
        // 创建新的评论对象
        GroupComment newComment = new GroupComment(commentContent,
MainActivity.getCurrentUsername()
                , groupItem.getGroupKey(), -1);

        //Fixme: 将评论加入数据库中
        DBFunction.addCommentToDBForGroup(newComment);
        // 刷新评论列表

 commentAdapter.updateData(DBFunction.getAllCommentsOfGroup(groupItem.getGroupKey
()));
        // 清空评论输入框
        commentEditText.getText().clear();
    }
}
```

- **搜索组局**

```
//按钮触发
SearchView searchView = view.findViewById(R.id.searchView);
ImageButton searchButton = view.findViewById(R.id.searchButton);
searchButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Get the user input from the search view
        String searchString = searchView.getQuery().toString().trim();

        Intent intent = new Intent(requireContext(),
SearchResultsGroupActivity.class);
        intent.putExtra("searchString", searchString);
        startActivity(intent);
    }
});

//筛选
private List<GroupItem> filterGroupItems(String searchString) {
    List<GroupItem> filteredList = new ArrayList<>();
```

```
    for (GroupItem groupItem : GroupFragment.groupItemList) {
        // Check if the groupItem's title contains the search string (case-
insensitive)
        String groupStr = new Gson().toJson(groupItem);
        if (groupStr.toLowerCase().contains(searchString.toLowerCase())) {
            filteredList.add(groupItem);
        }
    }
    return filteredList;
}
```

- **推荐组局**

```
// 按钮触发
FloatingActionButton fabRecGroup = view.findViewById(R.id.fabRecGroup);
fabRecGroup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        // Start CreateGroupActivity
        Intent intent = new Intent(view.getContext(),
RecommendGroupActivity.class);
        startActivityForResult(intent, 2);
    }
});

// 调用推荐算法
List<GroupItem> searchResults =
    Recommend.recommendGroup(MainActivity.getCurrentUsername(),
HomeFragment.getWeather());
```

- **我的组局**

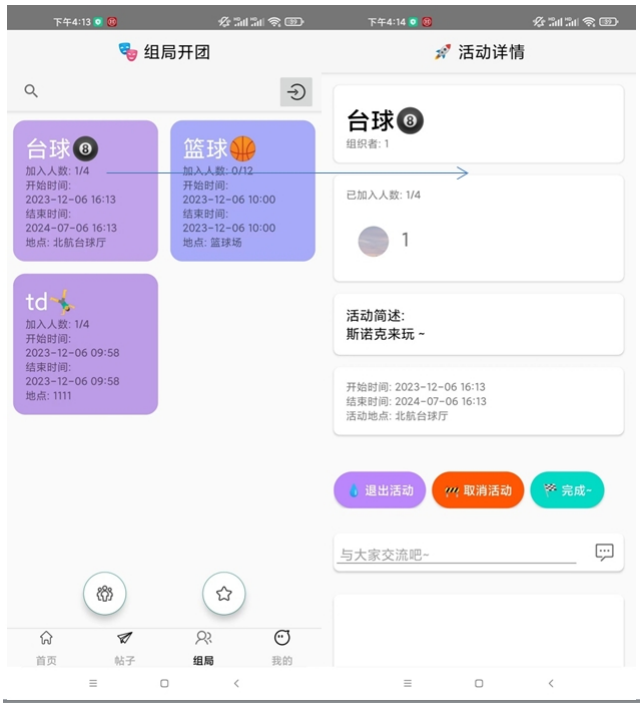
```
//按钮触发
goToMyGroup.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        startActivity(new Intent(getActivity(), MyGroupActivity.class));
    }
});

// 调用后端筛选
List<_Group> myResults =
DBFunction.findUserByName(MainActivity.getCurrentUsername()).getGroups();
```

6.3 具体实现图

- **发起组局**


组局开团
编辑组局
组局开团
台球
4
斯诺克来玩 ~
北航台球厅
开始时间
结束时间
发起组队
篮球
td
加入人数: 0/12
加入人数: 1/4
开始时间:
2023-12-06 10:00
2023-12-06 09:58
结束时间:
地点: 篮球场
地点: 1111
首页
帖子
组局
我的

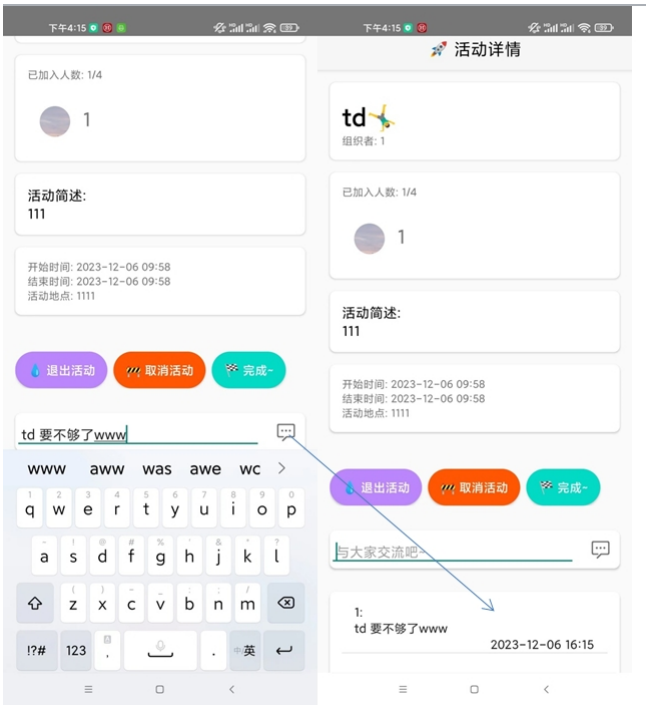
- **浏览组局**


组局开团
活动详情
台球
篮球
td
加入人数: 1/4
加入人数: 0/12
开始时间:
2023-12-06 16:13
2023-12-06 10:00
结束时间:
2024-07-06 16:13
地点: 北航台球厅
地点: 篮球场
地点: 1111
组织者: 1
已加入人数: 1/4
活动简述:
斯诺克来玩 ~
开始时间: 2023-12-06 16:13
结束时间: 2024-07-06 16:13
活动地点: 北航台球厅
退出活动
取消活动
完成~
与大家交流吧~
首页
帖子
组局
我的

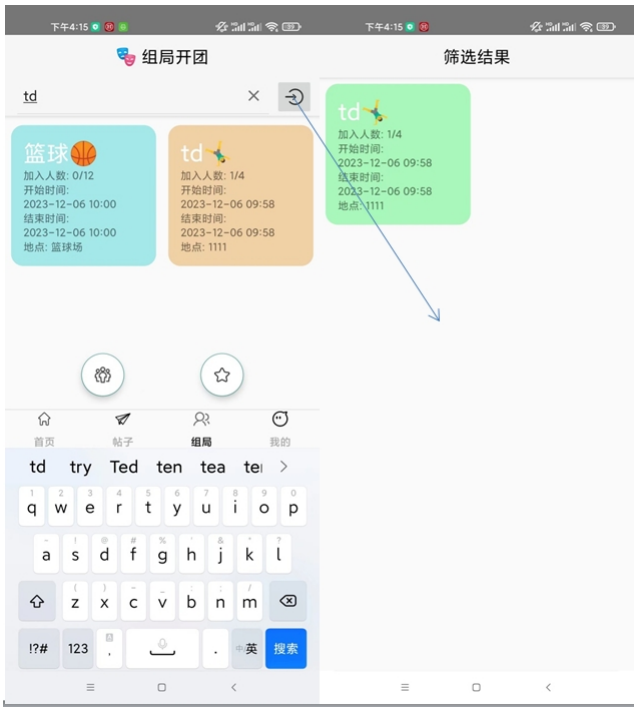
- **组局相关操作**


活动详情
活动详情
活动详情
组局开团
台球
组织者: 1
组织者: Wait for an organizer...
已加入人数: 1/4
已加入人数: 0/4
活动简述:
斯诺克来玩 ~
开始时间: 2023-12-06 16:13
结束时间: 2024-07-06 16:13
活动地点: 北航台球厅
加入活动
退出活动
取消活动
完成~
与大家交流吧~
篮球
td
加入人数: 0/12
加入人数: 1/4
开始时间:
2023-12-06 10:00
2023-12-06 09:58
结束时间:
地点: 篮球场
地点: 1111
首页
帖子
组局
我的

- **局内交流**

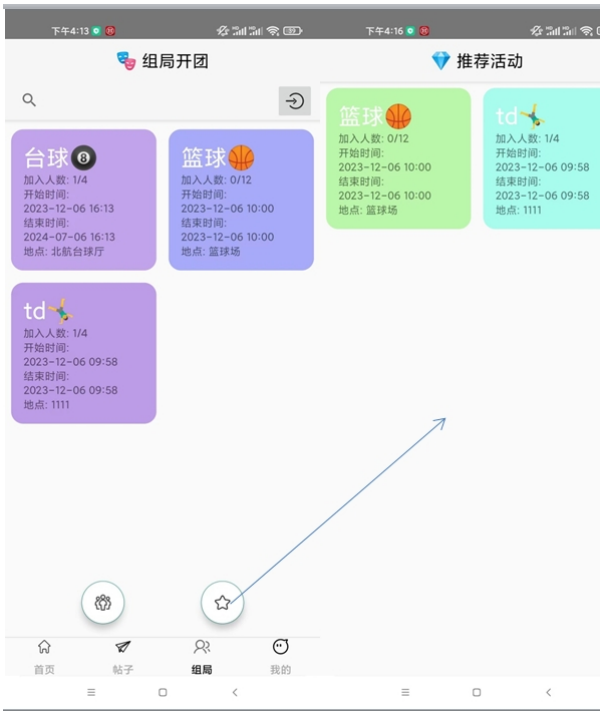


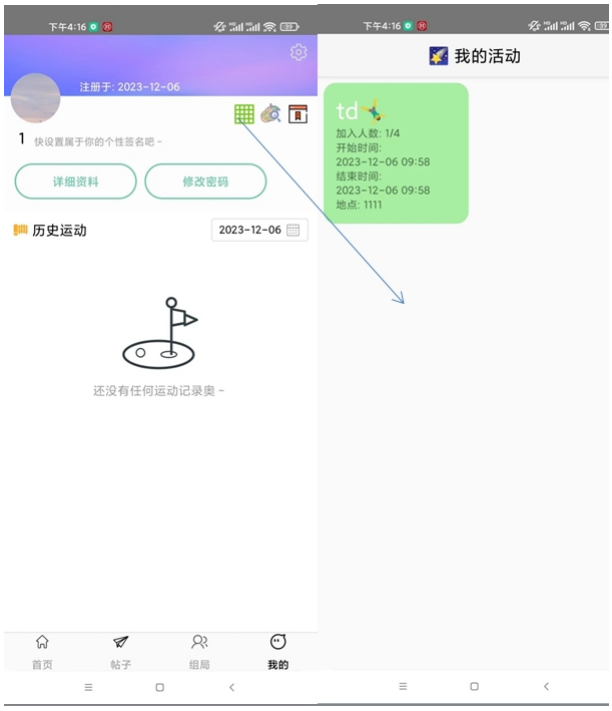
- **搜索组局**



- **推荐组局**

- **我的组局**





7. 攻略分享

这部分对应我们的 APP 的"帖子广场"模块（PublishFragment），点击下边栏第二个按钮即可进入该页面。

7.1 实现逻辑

- **发布攻略**
  该模块的逻辑链为下：
  - 用户点击 PublishFragment 的编辑帖子的按钮
  - 触发 BlogNewActivity，进入编辑帖子页面
  - 填写博客信息（标题、上传图片、内容）
  - 点击发布按钮，将信息写入数据库

    - 返回 PublishFragment，更新帖子视图，较新的帖子会展示在上部
- **浏览攻略**
  在 PublishFragment 页面上，帖子的简要信息以条状展示在页面上，用户可以点击相应的帖子，触发 BlogDetailActivity，进入帖子详情页面，展示帖子的标题、图片、内容、作者、发帖时间的信息。
- **评论攻略**
  在 BlogDetailActivity 页面上，用户通过输入框键入评论，点击评论按钮，将评论写入数据库，同时更新视图，将较新的评论显示在上部。
- **搜索攻略**
  在 PublishFragment 页面上，用户可在输入框键入关键词，点击搜索按钮，触发 SearchResultBlogActivity，在该页面上显示从数据库中筛选出的含有关键词的帖子，显示在页面上，该页面显示的帖子条目点击也可进入，行为和进入 BlogDetailActivity 相同。
- **CityWalk 攻略**
  从属于攻略搜索模块，用户键入 citywalk 和任意地点，点击搜索按钮，触发 SearchResultBlogActivity，会根据已有的 citywalk 攻略向用户推荐一些攻略和路线。
- **敏感词屏蔽**
  在用户发帖的过程中，可能有意无意触发一些敏感词汇，我们的 APP 通过敏感词过滤模块 SensitiveWordFilter，对敏感词表进行高效搜索，对用户输入的字符串进行过滤，敏感词汇用 * 代替，在不影响阅读的情况下屏蔽了敏感词。

7.2 核心代码

- **发布攻略**

```
// 触发按钮
FloatingActionButton addPostButton = view.findViewById(R.id.addPostButton);
addPostButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // 点击按钮跳转到 NewPostActivity
        Intent intent = new Intent(getActivity(), BlogNewActivity.class);
        startActivityForResult(intent, 1); // 使用 startActivityForResult 启动活
动，以便获取返回结果
    }
});

// 输入获取
publishButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        String title = titleEditText.getText().toString();
        String content = contentEditText.getText().toString();

        if (TextUtils.isEmpty(title) || TextUtils.isEmpty(content)) {
            Toast.makeText(BlogNewActivity.this, "标题和内容不能为空",
Toast.LENGTH_SHORT).show();
        } else {
            title = SensitiveWordFilter.Filter(title);
            content = SensitiveWordFilter.Filter(content);

            BlogPost newPost = new BlogPost(title,
MainActivity.getCurrentUsername(),
```

```
                    new Date(), content, -1);
            if (imageUri != null) {
                // Save the image to internal storage and get the new URI
                Uri savedImageUri = saveImageToInternalStorage(imageUri);
                newPost.setImageID(savedImageUri.toString());
            }
            DBFunction.addBlogToDB(newPost);
            setResult(RESULT_OK);
            finish();
        }
    }
});

//图片收集
private Uri saveImageToInternalStorage(Uri sourceUri) {
    try {
        // Get the content resolver for the app's internal storage
        ContentResolver resolver = getContentResolver();

        // Create a file in the internal storage directory
        File internalStorageDir = new File(getFilesDir(), "images");
        if (!internalStorageDir.exists()) {
            internalStorageDir.mkdirs();
        }

        String timeStamp = new SimpleDateFormat("yyyyMMdd_HHmmss",
Locale.getDefault()).format(new Date());
        String imageFileName = "JPEG_" + timeStamp + "_";
        File destinationFile = new File(internalStorageDir, imageFileName +
".jpg");

        // Copy the image to the internal storage
        try {
            InputStream inputStream = resolver.openInputStream(sourceUri);
            OutputStream outputStream = new FileOutputStream(destinationFile);

            byte[] buffer = new byte[8192];
            int bytesRead;

            while ((bytesRead = inputStream.read(buffer)) != -1) {
                outputStream.write(buffer, 0, bytesRead);
            }

            inputStream.close();
            outputStream.close();

            // Notify the system that a new image has been added to the gallery
            ContentValues values = new ContentValues();
            values.put(MediaStore.Images.Media.DATA,
destinationFile.getAbsolutePath());
            values.put(MediaStore.Images.Media.MIME_TYPE, "image/jpeg");
            resolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI,
values);

            // Return the URI of the saved image
            return Uri.fromFile(destinationFile);
```

```
        } catch (IOException e) {
            e.printStackTrace();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }

    return null;
}

//视图更新

protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data !=
null && data.getData() != null) {
        imageUri = data.getData();
        ImageView selectedImageView = findViewById(R.id.selectedImageView);
        selectedImageView.setImageURI(imageUri);
    }
}
```

- **浏览攻略**

```
// 触发动作
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, final int position,
long id) {
        // 获取点击的帖子
        final BlogPost selectedBlogPost = blogPosts.get(position);

        // 创建一个Intent，将帖子信息传递给帖子详情页面
        Intent intent = new Intent(getActivity(), BlogDetailActivity.class);
        intent.putExtra("blogPost", new Gson().toJson(selectedBlogPost));
        startActivity(intent);
    }
});

// 展示视图
titleTextView.setText(blogPost.getTitle());
authorTextView.setText("作者: " + blogPost.getAuthor());
contentTextView.setText(blogPost.getContent());

if (blogPost.getImageID() == null) {
    postImageView.setVisibility(View.INVISIBLE);
} else {

    postImageView.setVisibility(View.VISIBLE);
    Uri imageUri = Uri.parse(blogPost.getImageID());
    postImageView.setImageURI(imageUri);
    Glide.with(this)
```

```
            .load(imageUri)
            .into(postImageView);
}
```

- **评论攻略**

```
// 触发按钮
public void onAddCommentButtonClick(View view) {
    // 获取评论内容
    EditText commentEditText = findViewById(R.id.commentEditText);
    String commentContent = commentEditText.getText().toString();

    if (!commentContent.isEmpty()) {
        commentContent = SensitiveWordFilter.Filter(commentContent);
        // 创建新的评论对象
        BlogComment newComment = new BlogComment(commentContent,
MainActivity.getCurrentUsername()
                , blogPost.getBlogKey(), -1);

        //Fixme: 将评论加入数据库中
        DBFunction.addCommentToDB(newComment);

        // 将评论添加到帖子

 blogPost.setComments(DBFunction.getCommentsByBlog(blogPost.getBlogKey()));
        // 刷新评论列表
        commentAdapter.updateData(blogPost.getComments());
        // 清空评论输入框
        commentEditText.getText().clear();
    }
}
```

- **搜索攻略 & CityWalk 攻略**

```
// 触发按钮
SearchView searchView = view.findViewById(R.id.searchPubView);
ImageButton searchButton = view.findViewById(R.id.searchPubButton);
searchButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Get the user input from the search view
        String searchString = searchView.getQuery().toString().trim();

        Intent intent = new Intent(requireContext(),
SearchResultsBlogActivity.class);
        intent.putExtra("searchString", searchString);
        startActivity(intent);
    }
});

// 查找逻辑
private List<BlogPost> filterBlogPosts(String searchString) {

    if (!searchString.toLowerCase().contains("citywalk") &&
```

```
    !searchString.toLowerCase().contains("city walk")) {
        List<BlogPost> filteredList = new ArrayList<>();
        String[] tokens = searchString.split(" ");
        for (BlogPost blogPost : PublishFragment.blogPosts) {
            String blogStr = new Gson().toJson(blogPost);
            for (String token : tokens) {
                if (blogStr.toLowerCase().contains(token.toLowerCase())) {
                    filteredList.add(blogPost);
                    continue;
                }
            }
        }
        return filteredList;
    } else {
        String[] tokens = searchString.split(" ");
        List<String> tokenIn = new ArrayList<>();
        for (String token : tokens) {
            if (token.equalsIgnoreCase("city") ||
                    token.equalsIgnoreCase("walk") ||
                    token.equalsIgnoreCase("citywalk")) {
                continue;
            }
            tokenIn.add(token);
        }

        List<BlogPost> filteredList = new ArrayList<>();
        for (BlogPost blogPost : PublishFragment.blogPosts) {
            String blogStr = new Gson().toJson(blogPost);
            if (!blogStr.toLowerCase().contains("citywalk") &&
                    !blogStr.toLowerCase().contains("city walk")) {
                continue;
            }
            if(tokenIn.isEmpty()) {
                filteredList.add(blogPost);
                continue;
            }
            for (String token : tokenIn) {
                if (blogStr.toLowerCase().contains(token.toLowerCase())) {
                    filteredList.add(blogPost);
                    break;
                }
            }
        }
        return filteredList;
    }
}
```

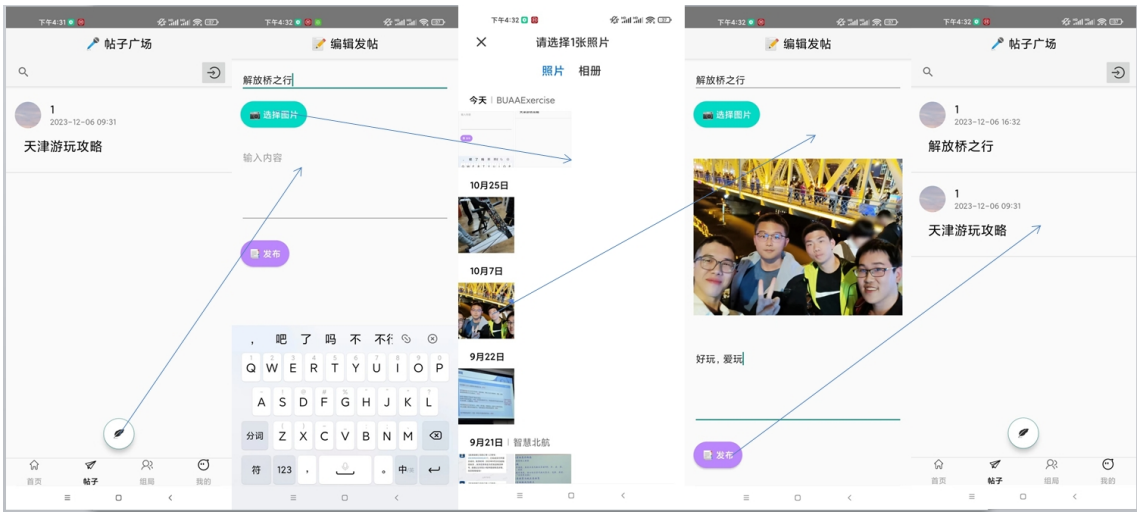
- **敏感词屏蔽**

```
//核心逻辑
 public static String Filter(String text) {
    if (wordList == null || wordList.size() == 0) return text;
    char[] __char__ = text.toCharArray(); // 把String转化成char数组，便于遍历
    int i, j;
    Word word;
```

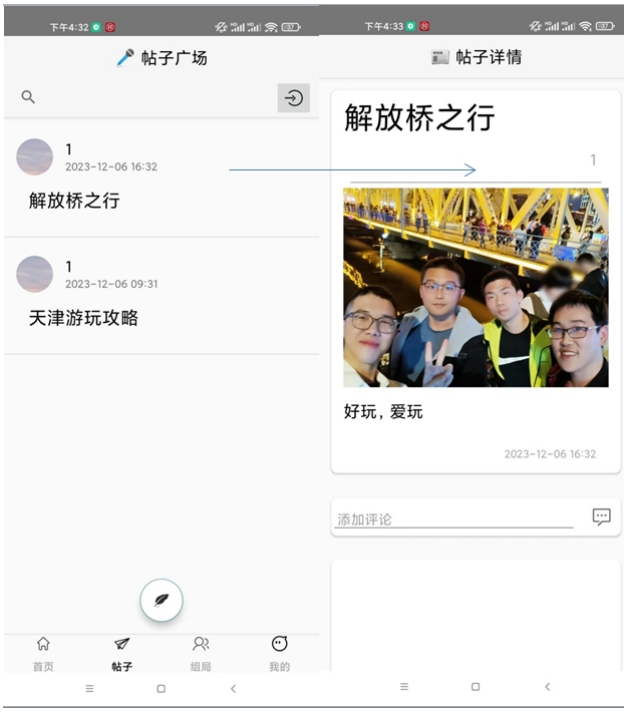
```
    boolean flag; // 是否需要替换
    for (i = 0; i < __char__.length; i++) { // 遍历所有字符
        char c = __char__[i];
        word = wordList.binaryGet(c); // 使用二分查找来寻找字符，提高效率
        if (word != null) { // word != null说明找到了
            flag = false;
            j = i + 1;
            while (j < __char__.length) { // 开始逐个比较后面的字符
                if (skip(__char__[j])) { // 跳过空格之类的无关字符
                    j++;
                    continue;
                }
                if (word.next != null) { // 字符串尚未结束，不确定是否存在敏感词
                    word = word.next.get(__char__[j]);
                    if (word == null) {
                        break;
                    }
                    j++;
                } else { // 字符串已结束，存在敏感词汇
                    flag = true;
                    break;
                }
            }
            if (word != null && word.next == null) {
                flag = true;
            }
            if (flag) { // 如果flag==true，说明检测出敏感粗，需要替换
                while (i < j) {
                    if (skip(__char__[i])) {
                        // 跳过空格之类的无关字符，如果要把空格也替换成'*'，则删除这个if语
句
                        i++;
                        continue;
                    }
                    __char__[i] = replace;
                    i++;
                }
                i--;
            }
        }
    }
    return new String(__char__);
}
```

7.3 具体实现图

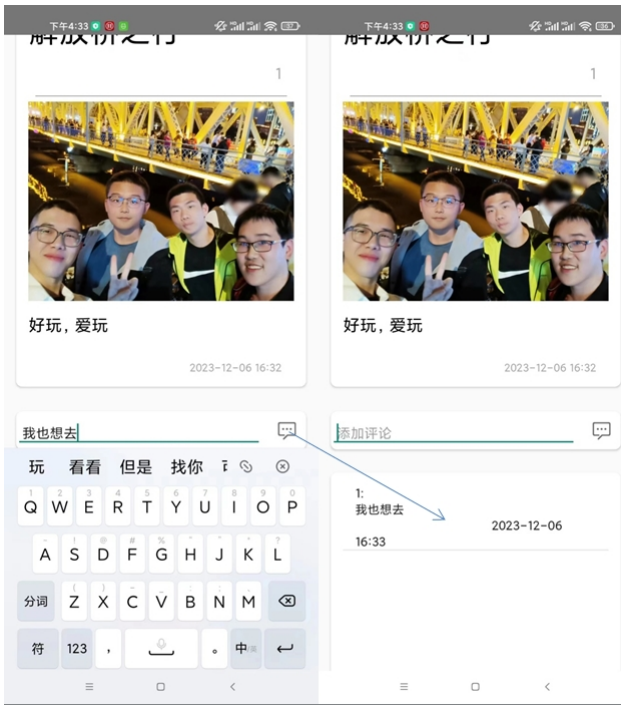
- **发布攻略**



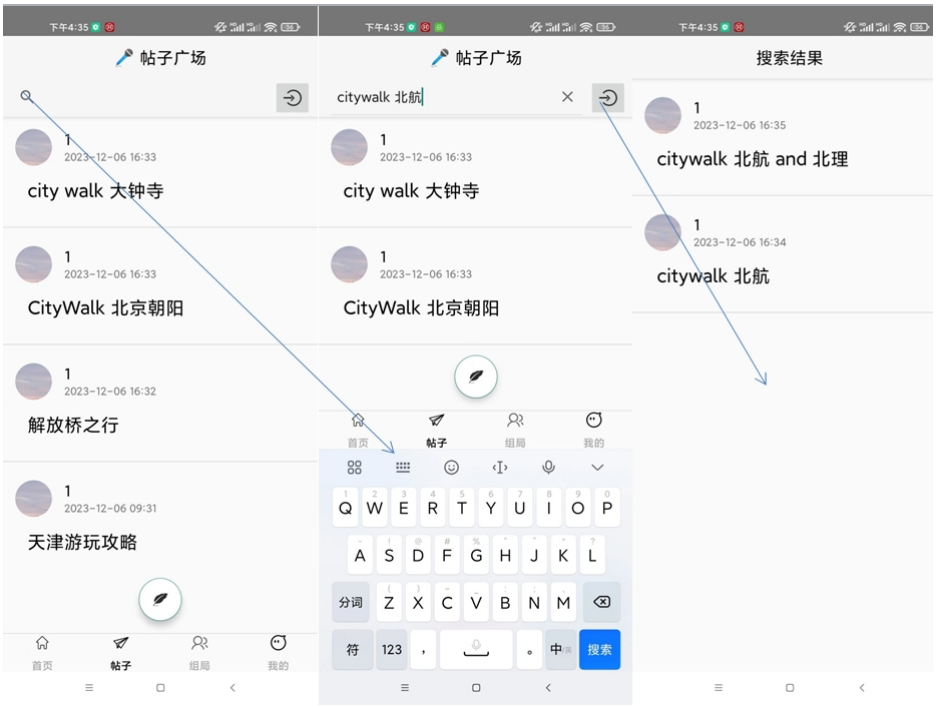
- **浏览攻略**



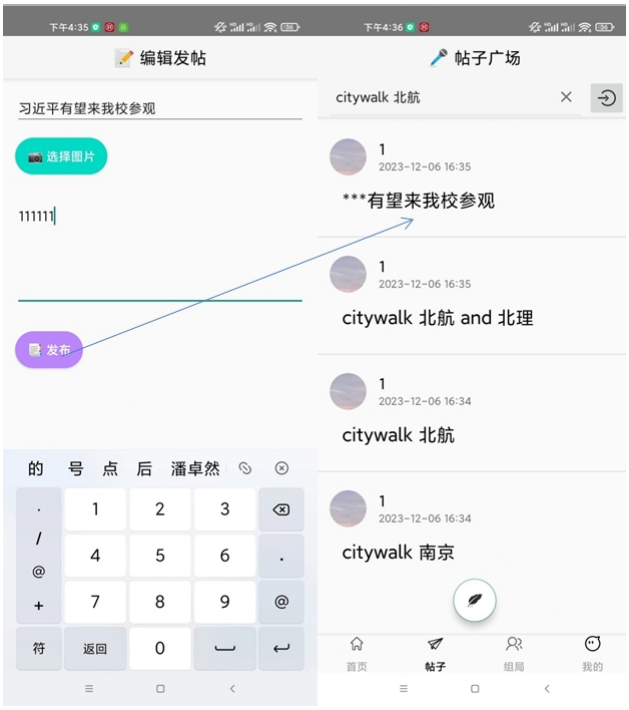
- **评论攻略**



- **搜索攻略 & CityWalk 攻略**



- **敏感词屏蔽**



8. 健身小计划

8.1 实现逻辑

- 推荐健身计划的逻辑我们主要借鉴了 keep 运动软件的推荐算法，从 APP 数据库后端中，获取用户信息，根据用户的近 10 日运动状况，包括：
  - 卡路里消耗
  - 运动时间
  - 运动频率
  - 以及该用户的性别，身高，体重
- 推荐相应的健身套餐，比如：
  - 对于高强度有氧训练，我们有“增肌训练计划”和“塑形训练计划”；

    - 对于中强度有氧运动方案，我们有“核心训练计划” 和 “减脂训练计划”
    - 对于“低强度训练方案”，我们主要推荐“保持健康运动方案”
- 对具体的健身方案：我们制定了详细的健身计划，包括每周每天的健身方法，训练部位，训练动作及其训练次数和组数，比如高强度增肌计划：
    - 周一和周四（胸部和三头肌）
        - 卧推（杠铃或哑铃）：3组，每组8-12次
        - 上斜卧推（杠铃或哑铃）：3组，每组8-12次
        - 坐姿飞鸟：3组，每组10-15次
        - 三头肌下压（杠铃或哑铃）：3组，每组10-15次
        - 颈后臂屈伸（杠铃）：3组，每组10-15次
    - 周二和周五（背部和二头肌）
        - 拉力器划船：3组，每组8-12次
        - 引体向上：3组，每组8-12次
        - 哑铃弯举：3组，每组10-15次
        - 杠铃弯举：3组，每组10-15次
        - 集中弯举：3组，每组10-15次
    - 并且我们会给出贴心的小提示：
        - 每组的重量应该适中，能够完成8-12次或10-15次的动作，但又不至于太轻
        - 每组之间休息1-2分钟，以便肌肉得到恢复
        - 注意正确的姿势和动作执行技巧，避免受伤。
        - 配合饮食，确保蛋白质和营养的摄入，提供肌肉生长所需的营养物质。

8.2 核心代码

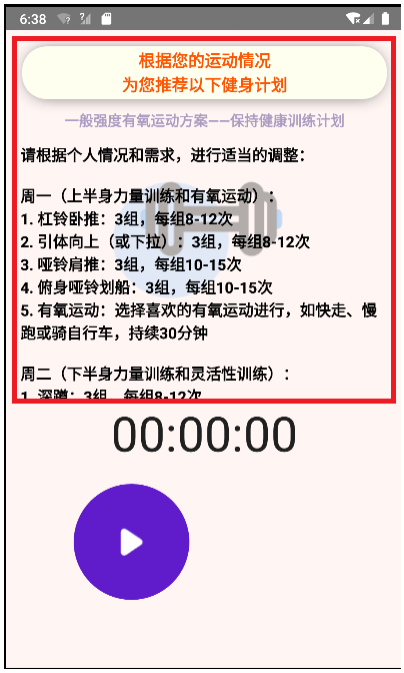
此部分代码较长，这里只展示核心逻辑部分

```
public ArrayList<String> recommendFitPlan(String userName, int year, int month,
int day) {
    double totalCalorie = 0;
    double sportTime = 0;
    double frequency = 0;

    /*
    计算该用户近 10 日的卡路里消耗，运动时间，频率等
    */
    for (int idx = 1; idx <= 10; idx++) {
        totalCalorie += getTotalCalorieInDate(userName, year, month, day);
        sportTime += getSportTimeInDate(userName, year, month, day);
        frequency += getFrequencyInDate
        day--;
    }

    /*
     根据用户信息和统计数据进行健身计划推荐
      这里用到了 性别，身高，体重
      以及上面收集到的卡路里消耗，运动时间，频率
      下面的代码将以男性健身推荐计划为例进行说明
```

```
        */

    if (sex.equals("男")) {
        // 男性推荐计划
        if (height >= 180 && weight >= 70 && totalCalorie > 2500 && sportTime >
60 && frequency > 4) {

            /*
            查阅相关资料，我们可以得到相应强度的训练方案
            此处我们的推荐计划是预先准备好的方案
            根据用户的个人情况进行套餐推荐
            */
            res.add(planRecommendations.get("高强度有氧运动方案"));
            res.add(planRecommendations.add("增肌训练计划"));
        }
        else if (height >= 170 && weight >= 60 && totalCalorie > 2000 &&
sportTime > 45 && frequency > 3) {
            res.add(planRecommendations.get("中强度有氧运动方案"));
            res.add(planRecommendations.add("核心训练计划"));
        }
        else {
            res.add(planRecommendations.get("一般强度有氧运动方案"));
            res.add(planRecommendations.add("保持健康训练计划"));
        }
    }
    else if(sex.equals("女") ) {
        /*
        女性推荐计划同理
        */
    }
    return planRecommendations;
}
```

8.3 具体实现图

- 内嵌到健身运动的 Activity 中



9. 组局推荐逻辑

9.1 实现逻辑

组局推荐算法的主体实现逻辑分为两大部分

- 一为检查活动组局是否符合用户的喜好
- 二为计算匹配分数，并更新最高匹配分数和推荐的组局

9.2 核心代码

- 检查活动组局是否符合用户的喜好

```
private static double groupMatchesUserPreferences(_Group group, List<Hobby>
usrHobbyLst) {
        // 检查活动组局与用户的爱好列表是否匹配
        // 返回权重
        for (Hobby hobby : usrHobbyLst) {
            if (hobby.getSportName().equals(group.getSport())) {
                return 10;
            }
        }
        return 0;
}
```

- 考虑到用户可能并没有收藏喜好的活动，所以若仅在爱好列表中匹配查找，可能存在无活动推荐的情况，因此我们进行了权重设置。考虑到用户爱好的影响力较大，对于用户爱好的活动我们增大权重至 10，非爱好活动的权值为 0；
- 计算活动匹配分数

```
private static double calculateMatchScore(_Group group, List<ExerciseRecord>
usrERLst, String weather) {
        /* 根据用户的历史活动记录、天气情况等计算活动组局的匹配分数
        返回一个0到1之间的分数，表示匹配程度
        计算历史活动记录的匹配分数
        */
        double historyMatchScore = calculateHistoryMatchScore(group, usrERLst);

        // 计算天气匹配分数
        double weatherMatchScore = calculateWeatherMatchScore(group, weather);

        // 综合考虑历史活动记录和天气的匹配程度
        double overallMatchScore = (historyMatchScore + weatherMatchScore) / 2;

        return overallMatchScore;
}
```

- 计算历史活动记录的匹配分数

```
private static double calculateHistoryMatchScore(_Group group,
List<ExerciseRecord> usrERLst) {
        int totalMatchCount = 0;
```

```
        for (ExerciseRecord record : usrERLst) {
            if (groupMatchesActivity(record.getSportName(), group)) {
                totalMatchCount++;
            }
        }

        // 计算匹配度得分
        double matchScore = (double) totalMatchCount / usrERLst.size();

        return matchScore;
}
```

- 计算天气匹配分数
  此部分的代码较长，我们只展示函数标签，描述核心逻辑：

```
private static double calculateWeatherMatchScore(_Group group, String weather)
```

- 该函数的功能是：根据传入的Group 组局对象和天气字符串判断活动组局是否适合当前天气，返回一个0到1之间的分数，表示匹配程度。

  ①首先，我们定义了一系列变量来表示不同天气和运动类型的权重。例如，`sandstorm`表示沙尘暴的权重，`snow`表示小雪的权重，以此类推。同时，`gpVal`变量用于存储根据运动类型获取的权重值。

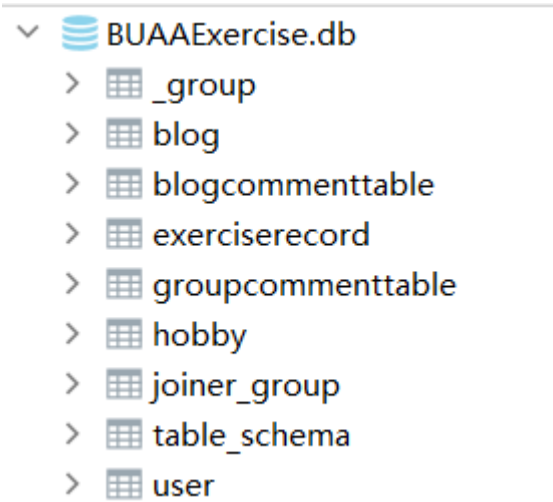
  ②接下来，通过switch语句，根据传入的_Group对象中的运动类型（`group.getSport()`），将对应运动类型的权重值赋给`gpVal`变量。

  ③然后，根据传入的天气字符串（`weather`），通过switch语句将对应的天气的权重值赋给`weVal`变量。

  ④最后，通过将天气权重值除以运动类型权重值，得到匹配得分`val`。并且使用`Math.min()`函数将得分限制在0到1之间，确保得分不超过1.0。

附: 数据库设计

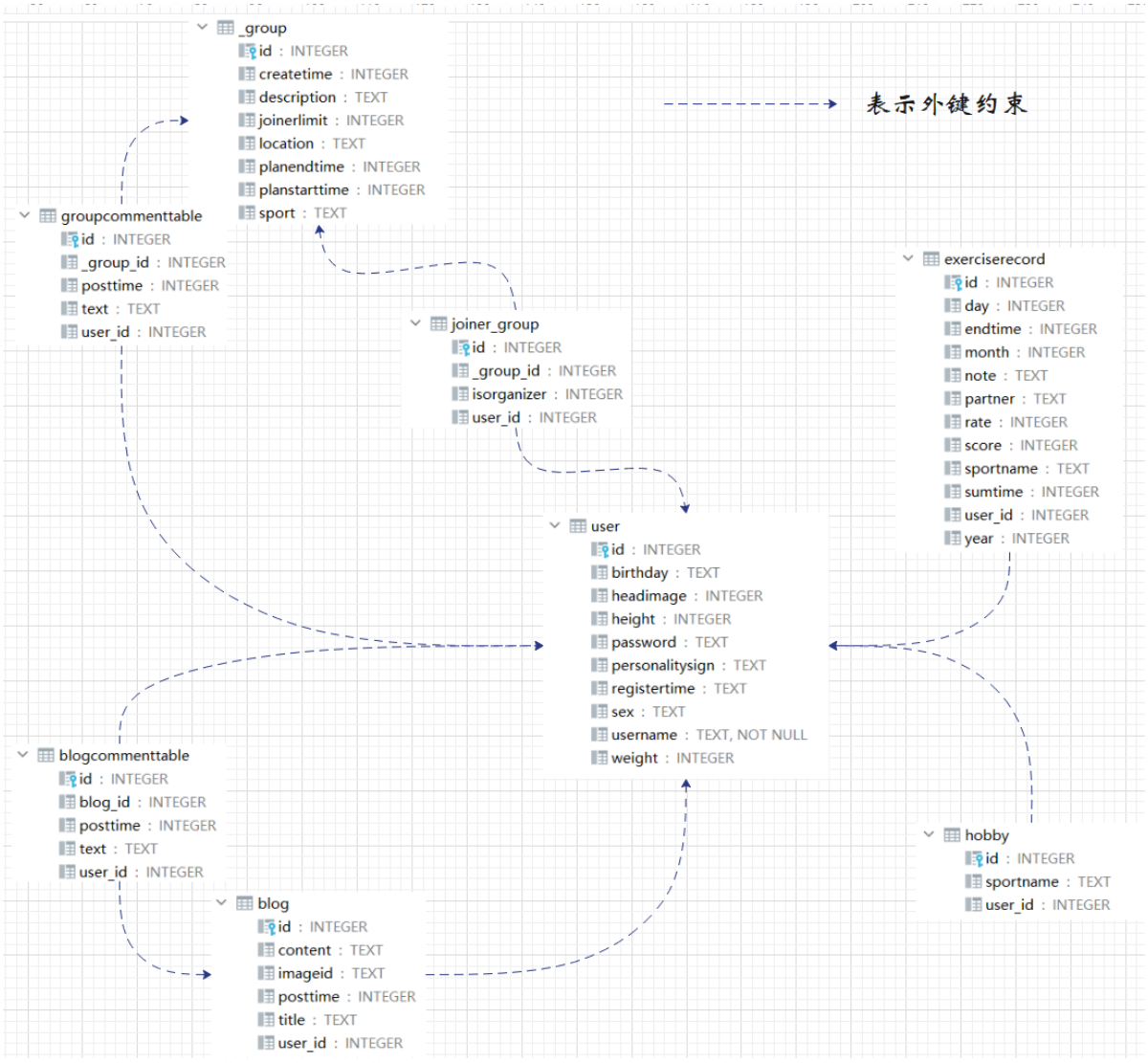
按照题目要求以及自定义选做任务的功能，进行数据需求分析，我们设计出了8张表，来存储android app的所有用户数据。



user表存储用户个人基本信息，exerciserecord存储用户运动记录数据，_group表和joiner_group存储活动组局的相关数据，groupcommenttable表支持用户可以对组局进行讨论评论，blog表和blogcommenttable表存储了用户攻略分享的博客和博客下面的评论。hobby表，我们自定义添加了用户收藏运动的功能，存储用户青睐的运动，按照用户喜好帮助我们更精准的为用户推荐团体活动组局，生成健身小计划。

在设计上我们充分考虑了数据库的完整性约束，表自身具有实体完整性，表与表之间有着严格的外键约束关系，保证了数据的正确性，相容性，可靠性。

下图展示了表的具体属性，以及表间的约束关系。



SQLite & LitePal

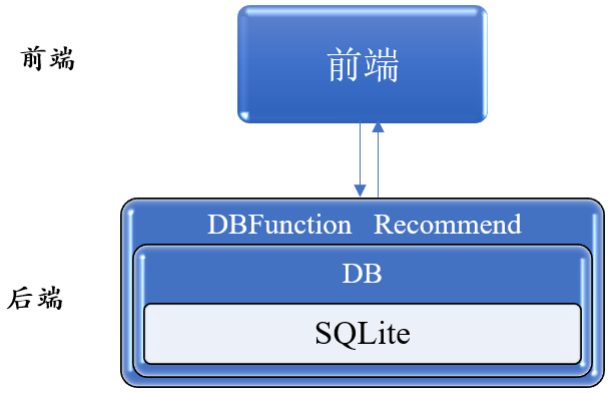
我们使用安卓原生数据库SQLite存储数据，并选用开源LitePal框架来操作SQLite数据库。

LitePal是一个轻量级的Android数据库框架，它具有以下几个优势：

1. 对象关系映射（ORM）：LitePal支持对象关系映射，可以将Java对象映射到数据库中的表。
2. 轻量级：LitePal的库文件非常小巧，不会增加太多的包体积，适合在Android应用中使用。
3. 支持多种数据类型：LitePal支持多种常见的数据类型，包括整型、浮点型、文本型等，满足了一般应用程序的数据存储需求。
4. 数据库升级方便：LitePal支持数据库版本升级，可以通过修改实体类的方式进行数据库结构的更新，避免了手动编写复杂的数据库升级代码。

后端架构

- 后端封装了两个包DBFunction，Recommend作为前后端交互的调用接口。
- 包DBDunction，Recommend对数据对象类组成的包DB进行封装。
- 包DB使用LitePal框架来实现对象关系映射。



举例：活动组局中加入参与者

- **前端调用后端包DBFunction接口**

```
// public class GroupDetailActivity

protected void onCreate(Bundle savedInstanceState) {
    ...
    boolean joined = groupItem.addJoiner(currentUser);
    if (joined) {
        DBFunction.addJoinerToGroup(currentUser, groupItem.getGroupKey());
        // 加入成功，更新界面
        if (newOrg) {
            organizerTextView.setText("组织者: " + groupItem.getOrganizer());
            btnCancelGroup.setVisibility(View.VISIBLE);
            btnCancelGroup.setOnClickListener(new View.OnClickListener() {
               ...
}
```

- **包DBFunction接口调用包DB中的函数**

```
// public class DBFunction

public static void addJoinerToGroup(String joiner, int groupKey) {
    _Group group = LitePal.find(_Group.class, groupKey);
    if (group != null) {
        // 没有组织者<==>人数为0。当且仅当这种情况，才会addOrganizer
        boolean foreignKeyOK = group.addJoinerByName(joiner,
group.getCurrentPersonNum() == 0);
        if (!foreignKeyOK) {
            Log.w(DBFunction.TAG, "不存在joiner用户 , addJoinerToGroup失败");
        }
    } else {
        Log.w(DBFunction.TAG, "不存在该组局groupKey , addJoinerToGroup失败");
```

```
    }

}
```

- **包DB中的函数建立与SQLite的映射**

```
//_group表

public boolean addJoinerByName(String joinerName, boolean isOrganizer) { //已保证，不加入重复的数据
    Joiner_Group joiner_group = new Joiner_Group();
    joiner_group.setIsOrganizer(isOrganizer ? 1 : 0);
    boolean foreignKeyOK = joiner_group.setJoinerByName(joinerName);
    //
    joiner_group.setGroup(this);
    if (foreignKeyOK) {
        //防止加入重复数据
        User user = DBFunction.findUserByName(joinerName);
        if (!LitePal.isExist(Joiner_Group.class, "user_id=? and _group_id=?", String.valueOf(user.getId()), String.valueOf(id))) {
            joiner_group.save();
        }
    }
    return foreignKeyOK;
}
```

```
//joiner_group表

public boolean setJoinerByName(String JoinerName) {
    User user = DBFunction.findUserByName(JoinerName);
    if (user != null) {
        this.joiner = user;
        return true;
    } else {
        Log.w(DBFunction.TAG, "数据不符合外键约束！插入数据失败");
        return false;
    }
}
```

数据库实现

实现难点-表的关联关系

表的创建

采用面向对象的思想来处理外键约束：

一对多举例：

```
public class Blog extends LitePalSupport {
    ...
    private long user_id;   //表中自动生成列名为user_id的foreign key,此处拿出来是为了方便子表查询
    private User user;  //foreign key refer to User
    ...
}
```

```
public class User extends LitePalSupport {
    ...
       private List<Blog> blogList = new ArrayList<Blog>(); // used by Blog
    ...
}
```

多对多举例:

```
public class User extends LitePalSupport {
    ...
    private List<Joiner_Group> joinerGroupList = new ArrayList<>();
    ...
}
```

```
public class Joiner_Group extends LitePalSupport {
    private int isOrganizer = 0;
    //
    private long user_id;
    private User joiner;
    private long _group_id;
    private _Group group;
    ...
}
```

```
public class _Group extends LitePalSupport {
    ...
    public List<GroupCommentTable> groupCommentTables = new ArrayList<>(); // used by GroupCommentTable
    ...
}
```

这些字段仅仅作为LitePal识别外键的标识，需要注意，无论是一对一，一对多，多对多关联关系、两个表之间的"傀儡"属性字段都不可以少！

外键相关属性的赋值与查询

外键相关属性仅仅作为LitePal识别外键的标识，在保存时需要为其赋值，才可以建立外键约束。

举例:

```
// Blog表

public boolean setAuthorByName(String authorName) {
    User user = DBFunction.findUserByName(authorName);
    if (user != null) {
        this.user = user;
        return true;
    } else {
        Log.w(DBFunction.TAG, "数据不符合外键约束！插入数据失败");
        return false;
    }
}
```

在实际查询过程中，有两种方式。激进查询和懒惰查询。

激进查询的用法简单，但是这种查询方式并不推荐，因为如果一旦关联表中的数据很多，查询速度可能就会非常慢。而且激进查询只能查询出指定表的关联表数据，但是没法继续迭代查询关联表的关联表数据。

因而我们实现上使用懒惰查询。一条数据记录的相关外键属性，并不能直接从类中获取，类中对应字段的值为null。需要单独查表获取与外键相关的字段。

举例：

```
// Blog表

public User getAuthor() {
    User user = LitePal.find(User.class, user_id);
    if (user != null) {
        return user;
    } else {
        throw new NullPointerException("外键约束异常：没有找到对应的user");
    }
}
```

后端接口实现

举例：组局组织者退出逻辑处理（自定义任务）

```
//    为对应组局添加参与者
public static void addJoinerToGroup(String joiner, int groupKey) {
    _Group group = LitePal.find(_Group.class, groupKey);
    if (group != null) {
        // 没有组织者<==>人数为0。当且仅当这种情况，才会addOrganizer
        boolean foreignKeyOK = group.addJoinerByName(joiner,
group.getCurrentPersonNum() == 0);
        if (!foreignKeyOK) {
            Log.w(DBFunction.TAG, "不存在joiner用户 , addJoinerToGroup失败");
        }
    } else {
        Log.w(DBFunction.TAG, "不存在该组局groupKey , addJoinerToGroup失败");
    }

}
```

```
//    将参与者移出组局
public static void removeJoinerFromGroup(String joinerName, int groupKey) {
    User joiner = findUserByName(joinerName);
    List<Joiner_Group> joiner_groups = LitePal.where("user_id=? and _group_id=?",
String.valueOf(joiner.getId()),
String.valueOf(groupKey)).find(Joiner_Group.class);
    if (!joiner_groups.isEmpty()) {     //存在groupKey--joinerName记录
        Joiner_Group joiner_group = joiner_groups.get(0);
        boolean isJoinerOrganizer = joiner_group.getIsOrganizer() == 1;
        //  执行remove
        LitePal.deleteAll(Joiner_Group.class, "user_id=? and _group_id=?",
String.valueOf(joiner.getId()), String.valueOf(groupKey));
        //  如果joiner是group的Organizer
        if (isJoinerOrganizer) {
            List<Joiner_Group> joiner_groups1 = LitePal.where("_group_id=?",
String.valueOf(groupKey)).find(Joiner_Group.class);
            if (!joiner_groups1.isEmpty()) {  // if(还有人)
                joiner_groups1.get(0).setIsOrganizer(1);
            }
        }
    } else {
        Log.w(DBFunction.TAG, "不存在groupKey--joinerName , removeJoinerFromGroup
失败");
    }
}
```

四、创新之处

1. 天气获取

1.1 内容

由于运动类型的不同，其对天气的依赖程度也不同，显然，对于室外和室内活动，不同的天气影响着当前的运动选择，为了让用户能够更好更全面地选择运动，添加了对天气情况的获取功能，根据不同的天气信息，传递给运动推荐的模块，从而实现更合理的运动规划以及提醒；除此之外，也获取了相应的气温信息，提醒了用户要对运动的安全问题关注。

1.2 实现方式

- 获取方式：采用访问开放的 api 的方式，从心知天气中获取天气控制台，通过私钥 url 的形式，对网络进行访问（免费版的限制是每日 1k 次访问，足以满足一般的需求）。访问对应的实时天气的 api，该 url 能够返回 json 格式的数据，其中包含的信息有：地址信息、温度信息、天气名字、天气代码，根据返回的信息，实时渲染对应的前端布局，并且提供外部访问的接口，为推荐运动等功能提供天气信息
- 交互优化：利用心知天气自身推荐的图标以及代码对应（V3天气现象代码说明 (yuque.com)），对原图标名称进行重新设计，使得能够根据当前天气代码来渲染对应天气的图标，梅花用户界面，使画面不单调
- 实现代码：

```
void setOnlineWeather(View view) throws IOException, JSONException {
        new Thread(new Runnable() {
            @Override
            public void run() {
```

```
                try {
                    OkHttpClient client = new OkHttpClient();//创建一个OkHttp
实例
                    Request request = new
Request.Builder().url(urlApi).build();//创建Request对象发起请求

                    Response response = client.newCall(request).execute();//
创建call对象并调用execute获取返回的数据
                    String responseData = response.body().string();
                    String jsonData = responseData;
                    try {
                        JSONObject jsonObject = new JSONObject(jsonData);
                        JSONArray results =
jsonObject.getJSONArray("results");    //得到键为results的JSONArray
                        JSONObject now =
results.getJSONObject(0).getJSONObject("now");//得到键值为"now"的JSONObject

                        weatherStr = now.getString("text");//得到"now"键值的
JSONObject下的"text"属性,即天气信息
                        weatherCode = now.getString("code"); //获取天气代码
                        temperature = now.getString("temperature"); //获取温度

                        setWeather(view);

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
```

- 页面展示：

2. 健身计划推荐

2.1 内容

- 个性化推荐：该算法根据用户的个人信息和近期运动数据，为用户提供个性化的健身计划推荐。这样可以更好地满足用户的需求和目标。
- 综合考虑多因素：算法综合考虑了用户的卡路里消耗、运动时间、频率以及性别、身高和体重等因素，以更全面的方式进行推荐。
- 健身套餐的提供：算法不仅提供训练方案的推荐，还给出具体的健身套餐，包括每周每天的健身方法、训练部位、动作次数和组数，帮助用户更好地实施健身计划。

2.2 实现方式

- 数据收集和分析：通过记录用户的运动数据或使用智能设备收集运动数据，然后对这些数据进行统计和分析，计算卡路里消耗、运动时间和频率等指标。
- 推荐逻辑和套餐制定：根据收集到的数据和用户的个人信息，制定推荐逻辑并为用户推荐适合的健身计划。同时根据推荐的计划，制定具体的健身套餐，包括训练方法、部位、动作次数和组数。
- 用户提示：为用户提供贴心的小提示，包括重量选择、休息时间、姿势和动作执行技巧，以及饮食的配合。这些提示可以通过文字、图像或视频等形式呈现给用户。

3. 组局推荐

3.1 内容

这个组局推荐算法的创新点主要体现在以下几个方面：

- 综合考虑因素：该算法综合考虑了用户的爱好、历史活动记录以及天气条件等多个因素，通过权重设置和综合计算匹配分数，更全面地评估了活动组局的推荐程度。这种综合考虑可以提高推荐的准确性和用户满意度。
- 多维度匹配：通过计算历史活动记录的匹配分数和天气匹配分数，算法在匹配过程中考虑了用户的过往行为和环境因素，而不仅仅是静态的用户偏好。这样的多维度匹配可以更好地适应用户变化的需求和外部环境的变化。
- 动态权重调整：通过天气匹配分数的计算中的权重值设定，算法可以根据不同的天气情况对活动的适宜程度进行动态调整。这种动态权重调整能够使得推荐更加贴近实际情况

3.2 实现方式

- 综合考虑因素：该算法从后端数据库中获取的历史活动记录，从前端获取当前天气情况，不同活动和天气赋以不同的权值，实现综合考量。
- 多维度匹配：算法检查用户的爱好列表，当前活动组局的运动类型进行匹配，用户的历史活动记录和当前天气情况，实现多维度匹配。
- 动态权重调整：算法中使用了权重值来表示不同因素对匹配的重要程度。通过根据运动类型和天气情况动态调整权重，可以根据不同的情况对匹配分数进行合理的加权。例如，在用户历史活动记录匹配中，如果用户之前参加过类似的活动，则该因素的权重会增加。通过动态调整权重，可以更好地适应不同用户和不同情况的需求。

4. 组局机制

这个实现起来有一些小讲究，所以在这里简单写几句。

4.1 内容

活动组局模拟了实际情况，更符合用户使用逻辑如果组局者：如果组局者（群主）退出，活动并不解散，如果还有其他参与者，群主递补到下一位参与者；如果没有其他参与者，保留活动，等待其他感兴趣用户加入。

4.2 实现方式

前端逻辑：

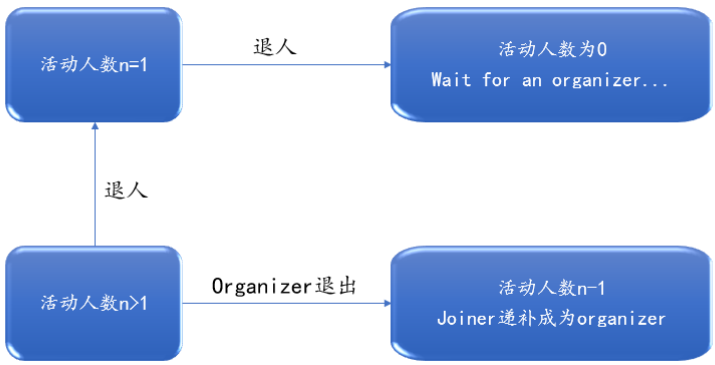
- 用户可以点击加入组局按钮：
  - 若人数已达上限，则将被提示无法加入
  - 若人数未满，则用户加入该组局
  - 设置加入组局按钮为退出组局
  - 更新数据库，同步更新该页面的参与者视图
- 用户可以点击退出组局按钮：
  - 若用户为组局的组织者
    - 若当前只有用户一人在该组局中，则用户可以退出，该活动的组织者设为"虚位以待"
    - 否则，组织者顺延到下一个参与者
  - 若用户不是组织者，则正常退出
  - 同步更新数据库和参与者、按钮视图

后端逻辑：
组局活动创建：组局者创建活动时，创建活动的同时，将组局者写入活动的Joiner列表中，写入数据库

Joiner退出组局活动：在移除Joiner记录的同时，如果Joiner是活动的组局者：如果有其他Joiner，组局者身份顺延至其他Joiner；如果没有其他Joiner，保留活动，等待其他感兴趣用户加入。

为组局活动添加Joiner：在添加Joiner记录的同时，如果活动没有组局者（等价为空活动，人数为0，仅在组局者退出活动后出现），自动将Joiner设置为组局者身份。

组局活动删除：删除活动的同时，在数据库中联级删除活动对应所有Joiner记录



五、项目总结

我们的项目致力于开发一款北航运动app来为大家提供便利。在本项目中，我们定期开展团队讨论，解决开发时遇到的各种问题，在明确的分工下我们的项目得到不断的推进直到基本完成项目，基本完成后我们又精益求精，尽量地去完善我们的作品，为其设计了启动动画，美化各个角落的ui设计，学习获取真实天气的接口的使用并将其应用到我们的app中……最后尽管由于考期和编译实验等压力我们的app还有可以改善的地方，但是做到这一步也已经基本符合了我们的开发预期，我们很高兴也很自豪能在短期之内一同完成了这样一款可以真正运行在我们的智能手机上的应用，这次开发经历真的让我们每个同学都有了很多的收获。

六、课程学习总结

1. 课程收获和难点分析（是否有安卓基础，做完这个大作业自我感觉是否有提高等其他收获，本次项目感觉最困难的地方在哪里）

1.1 课程收获

本小组均基本无安卓开发基础，我们通过课堂听讲以及查资料自学等方式来完成大作业的设计。完成大作业之后我们能深深感觉到开发一款能运行在我们天天用着的智能手机上的app所带来的成就感。同时我们也都有了或同或异的收获，比如：

- 负责前端的同学学到了安卓ui的设计（熟悉了安卓各种各样的控件等）与美化技巧，学到了安卓前端业务逻辑代码的设计，学到了安卓前端开发的过程中与后端进行对接的具体接口形式……
- 负责后端的同学学到了如何在数据库中合理地创建关系表的知识以及在安卓开发的背景下应用数据库的基本方法，学到了在开发过程中和前端对接的具体形式……
- 我们都学到了团队协同开发的“技巧”：在开发前要统一开发的环境，比如工具版本等，并在写的时候提前考虑与其他成员负责的代码的合并，留好简洁的合并代码接口，这将对代码合并和版本管理起到莫大的帮助。同时我们体会到整个团队约着线下一起开发，既能相互监督，又能在有对接问题时及时和其他成员进行高效率的沟通，这些协同开发的“技巧”经过我们的亲身体会认为是在专业知识之外的一个非常有价值的收获。
- 我们的创新意识获得了提高：通过开发安卓APP，我们可以自行设计和实现创意功能，这极大程度上激发了我们的创新意识和创造力，在开发过程中，我们在如何设计更好的用户体验、添加更多的功能等方面进行了大量的思考，收获了想象力和创新意识的提升。
- 宝贵的实践经验：开发安卓APP是一项实践性很强的工作，团队中的每个人在实践中积累了大量的经验和实践技巧，这是只有亲身经历了才能收获到的体会，也将让我们在以后的学习工作中更加游刃有余。

1.2 难点分析

负责不同部分的同学有不同的难处，很难在这不同同学的种种难处之间评出一个“最”字，因此这里将可能的比较困难的地方罗列出来。

- 界面错位：由于不同版本的安卓模拟器，或者不同的安卓手机设备存在界面尺寸等方面的差异，app运行在不同的设备上会出现组件错位、显示异常等不可预期的问题。
- 版本兼容性，在实际开发过程中，不同的安卓版本存在兼容性问题，甚至是功能缺失：某些功能在较老版本的安卓系统上可能不可用，而在较新版本的系统上则可以正常使用，这导致我们花费了大量时间进行调试和环境的配置。
- 获取用户输入的图片：一开始简单地将用户选择的图片的 URI 作为数据存储，但在另一个 Activity 访问时会遇到权限不足的错误；解决方式是复制用户的图片到 APP 内部路径，生成内部的 URI，避免安全性问题。

- 用户权限管理：开发到后期，对于加入的新功能，往往需要获取用户手机的权限，比如网络访问（用于获取实时的天气信息）、手机内存访问（用于上传图片并保存至本地），在线离线管理。在最初尚未解除过权限管理的时候，会出现难以解决的问题，而且对于不同的安卓版本，针对同一权限的管理方式也有所不同，所以需要仔细地查阅资料并调试，才能完成权限的使用。

2. 教师授课评价（老师上课过程的一些建议，以及希望老师之后能够介绍一些什么东西）

2.1 授课评价及建议

- 授课评价
  - 老师的授课很好，非常细致，讲解的知识覆盖面积很大。
  - 老师授课非常细致，认真负责，课堂风格风趣幽默，经常和大家一起分享日常生活的点滴，平易近人。
  - 老师除了课内的安卓开发技能之外，也会讲授大模型，深度学习，神经网络的科研领域前沿知识，很好地拓广了我们作为计算机专业学生的思维。
  - 课下咨询老师问题时总能够得到老师的耐心积极回复，这更让我们觉得老师是对学生很认真负责，很尊重学生的。
- 简单建议
  - 建议老师在授课时能够稍稍做一些代码演示，比如，以从0开始一步步迭代进行代码演示的方式，不断地“抛出需求-解决需求”，在解决需求的过程中讲解课程的知识点，我们觉得这样或许可能会有更好一些的授课效果～
  - 考虑结合实际案例进行教学：通过结合实际案例进行教学，可以帮助学生更好地理解安卓APP的开发过程和相关技术，同时也可以提高学生的兴趣和积极性。

2.2 希望之后介绍的内容

- 介绍关于安卓的实践开发**技巧**相关的内容，比如：
  - 如何设计出美观的ui界面
  - 如何设计出高效的数据传输方案以降低用户操作的延迟
- 介绍如何真正地上市自己的app（就像是介绍论文投稿的流程一样）
- 介绍使得自己开发的安卓应用能够适配各种安卓设备的属性的具体实现方法（比如：可能有的组件在不同口径的安卓上机上会有不可预期的显示效果，如何避免这种情况的出现）
- 希望老师考虑下讲授队友之间如何高效率地进行协作和版本管理，以有效引导同学们在团队合作中并行开发项目，并使用版本控制工具（如Git）进行代码管理，提高学生的团队协作能力和代码管理能力。

3. 助教评价

- 无论是之前的python全英课还是这学期的安卓课，我们均认为助教学长比较随和友善，我们小组成员有问题和学长沟通的时候会得到积极回应，我们认为这种品质对于助教工作来说是非常可贵的，也很幸运可以遇到这样的学长作为我们的课程助教。

4. 当前课程教授内容评价与课程进一步改进建议

4.1 课程内容评价

我们认为当前的授课内容覆盖了安卓很多方面的知识，并且老师邀请了华为的相关负责人为我们介绍了鸿蒙系统下的开发，拓宽了我们的视野，这些无疑是令我们很受益的。

4.2 课程进一步改进建议

- 课程可以适当留一些小代码作业，来及时检测同学们基础知识的掌握情况，就类似于python课程中对python基础语法进行考察的作业一样。
- 可以提供一些针对于大作业中比较复杂一些的功能的实现建议，来为同学们指明方向，比如可以撰写大作业指导手册。或者可以考虑每个月开展线下答疑，为学生提供技术支持和指导，帮助学生及时解决开发过程中遇到的问题，提高开发效率。
- 希望老师可以提供丰富的资源和支持，比如开发工具、教程、案例等资源，方便学生学习。
- 开发安卓APP需要学生具备一定的实践操作能力，因此，希望能够提供足够的实验和实践机会，让学生能够循序渐进地，阶段性地开发APP，课上知识与课下实验进行同步，平时将开发进度抓紧点，也可以及时地巩固和加深同学们对课上相关知识的理解。

七、主要参考资料

1. 心知天气 API 使用手册（V3版） (yuque.com)
2. Android实现-心知天气API接口开发(天气预报app) 知心天气 api_Yang2023.的博客-CSDN博客
3. Customization · prolificinteractive/material-calendarview Wiki (github.com)
4. bilibili安卓开发视频教程 —— up主天哥在奔跑
5. 7种经典推荐算法模型及应用
6. keep 的 AI 健身推荐算法
7. sqlite语句合集-百度文库 (baidu.com)
8. Android数据库高手秘籍_guolin的博客-CSDN博客
9. Litepal使用详解_litepal-CSDN博客
10. Android使用LitePal插入数据不重复_litepal 插入-CSDN博客
11. Developmc关于litepal方法的小结*litepal设置主键*不惭_的博客-CSDN博客
12. Android Litepal多表关联数据查询*android 多表查询*liulicuican的博客-CSDN博客
13. LitePal学习(四)——表关联及相关查询 - 简书 (jianshu.com)

八、实际展示视频（不超过5min）

见文件夹.