关键技术和技术难点

## 整体软件框架结构难点与技术

整个软件采用底部导航栏，控制中间viewpager的内容显示，而viewpager的显示是由fragment填充的，所以fragment方面数据的填充非常重要，软件中出现了数据加载太多导致内存溢出的现象，针对这种情况，我们团队决定在以后采用网络缓存异步加载的方式来改善，并且，软件页面的适配工作是重点要改进的地方，尽量的多用percent布局，以力争能够在多种机型中能尽可能的达到美观的效果。

## 登陆注册功能相关技术与难点

1. 调用了mob公司的SDK，通过获取mob公司提供的免费验证码，获取验证码成功之后返回验证，再通过验证结果进行下一步操作，在这里的注册和更改密码都是用的同一个activity跳转，实现功能的不一样，是在登陆模块传入按钮的信息，从而发送不同的网络请求来进行修改密码和注册的行为。
2. 因为mob公司的新版sdk存在问题，通过人工咨询客服得到旧版sdk并使用，同时在获取验证码的时候，采用异步线程的方式，设置倒计时，同时与服务器连接，保证了验证码的可靠性。接着登陆依然是通过比对自建服务器端中的账号密码信息，从而通过验证并加载数据。

## 比赛创建加入功能相关技术与难点

1. 基于网络请求的方式，通过dialog中信息的填写，并且获得登陆用户的id，一起传输到服务器并新建比赛，比赛能否新建成功会通过网络请求的返回值进行判断，逻辑设计分布在服务器端以及安卓端。
2. 通过网络请求，获取已有比赛的队伍信息并加载在spinnner控件上，从而以供用户选择，若用户选择新建队伍，则软件会获取输入的名称以及队伍信息并一起上传到服务器，同时服务器端也会返回创建队伍的结果。

## 服务器的选择

1. Django+uwsgi+nginx
2. Django:自带WSGI协议，用于处理Web服务器和应用程序的交互，但是性能一般，只能用于测试，它强调代码复用，多个组件可以很方便的以“插件”形式服务于整个框架通过django的特性，可以简便快速的开发出数据库驱动的网站，同时提供admin可视化数据查看界面，方便数据的管理。
3. Uwsgi：中间件
   1. uWSGI本身是内网接口，开启多个work和processes可能也不够用，而nginx可以代理多台uWSGI完成uWSGI的负载均衡；
   2. 支持的并发量更高;方便管理多进程；发挥多核的优势；
   3. 提升性能，因为uwsgi协议比WSGI协议有优势
4. Nginx：HTTP和方向代理服务器：
   1. 负载均衡
   2. 安全：客户端对Web服务器的访问需要先经过反向代理服务器。这样可以防止外部程序对Web服务器的直接攻击。
   3. 提升Web服务器IO性能，在反向代理服务器完整的接受请求，然后传给web服务器，保证服务器性能；
   4. 单独处理静态文件的访问，这并不是uwsgi，django的强项

## 数据模型的设计

数据的模型，决定了之后检索数据时的逻辑，怎么设计可以减少存储空间，怎样可以加快检索数据的速度，这都是我们在设计数据存储的格式前需要考虑的。我们在设计数据模型之前进行了多次交流，保证数据的检索速度的同时，数据占用的空间也保持在较低的水平。

## 保证程序和服务器数据的准确性

服务器和APP在通信时，由于讲的“语言”不通，服务器的数据需要可以被APP看懂，我们使用了JSON格式传输数据，并在刚开始定下输入输出，在APP制作中间增删查改，保证APP收到的数据准确无误。

## 回传数据

使用个推的API，在用户加载地图界面的时候，上传用户的Client ID，并在用户队友打点时，回传信息，方便队员知道比赛完成情况，如果这里轮询服务器的话，则会占用大量的资源，这是我们所不愿意看到的