啪啪打卡设计文档

1 引言

* 1. 系统概述

教学实验管理系统（啪啪打卡）是实现老师管理成绩实现现代化和信息化的重要系统。该教学实验管理系统能够为老师提供充足的信息和快捷的查询登记的手段，并支持学生老师简单交互的功能。该系统充分考虑到各种小的功能需求，面对实验项目的不断丰富、学生人数不断增多、潜在的推广价值，完整地、无bug地、高效率地实现了所有的系统功能。

* 1. 文档概述

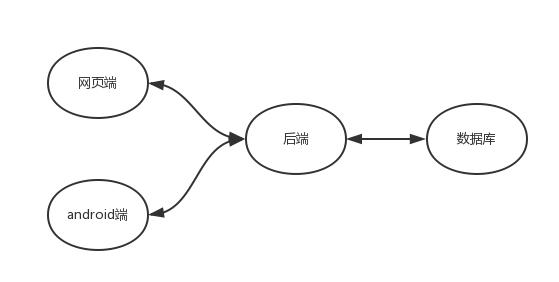
本文档按照模块划分的顺序编写，分为网页端，android端，后端与数据库三个模块，每模块中详细列举了细化功能。

本文档旨在让读者更好了解啪啪打卡系统的设计结构，方便系统的二次开发和维护。使用者具体操作流程详见用户指南。

2 引用文件

devise文档 <https://github.com/plataformatec/devise>

3 系统级策略设计

1. 前后端解耦合，网页端、android端分别同后端接口进行交互。  
   设计如下：  
   
2. 整套系统的交互基于HTTP协议。  
   网页端、后端、android 端预留了 SSL 的接口。今后可以轻松支持 HTTPS 协议传输。
3. 用户登录使用token和cookie系统同时管理，分别对应android端和网页端。用户登录使用devise库管理密码加密等操作。
4. 用户上传文件需要UTF-8编码。

4 系统体系结构设计

4.1 运行环境

* 网页端使用chrome、Safari、firefox、360、搜狗等主流浏览器访问。
* 后端要求使用 x86 架构 CPU， 1G 以上内存，需要有Internet访问。
* Android端运行环境：最低Android 4.0，体验好要求Android 4.4以上，最佳要求Android 6.0以上。Android 4.4以下可能发生显示异常，但是可以使用。  
  需要配备 Wi-Fi 模块以连接到数据库。  
  签到功能需要使用 GPS。  
  拍摄实验照片/录像的功能需要使用摄像头。

4.2 设计思想与关键技术

* 异步通信  
  后端同网页端或android端通过异步通信的方式传输。网页端所有数据都是通过ajax操作从约定的接口地址处获取JSON解析得到。
* 防止过度重绘  
  网页使用ajax实现异步刷新，Android通过convertView实现不重绘已有图形。提高效率。
* 自动获取手机号  
  使用 android 内置的获取手机号的接口，加速用户的登录过程。
* 防止GPS漂移  
  由于硬件限制，一次GPS定位难免会出现位置不准确的情况。为避免之，我们要以某个时间间隔连续进行若干次定位，当前仅当这些位置相差不超过某个阈值时，我们才认可这一次的定位结果，并将其发送至服务器。

4.3 模块具体设计及运行过程

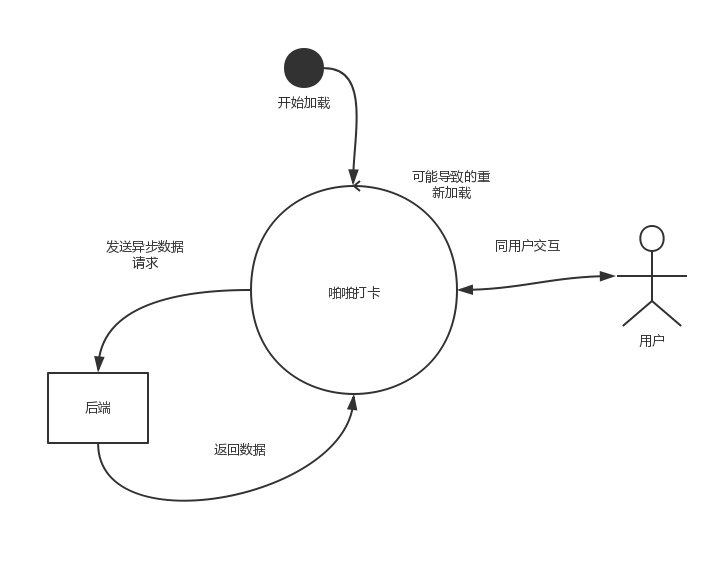
4.3.1 后端及服务器

 见后端概述文档 <../db/doc/overview.md>

4.3.2 网页端

网页端是为了教师管理课程和学生设计的，包括教师登录、修改个人信息、添加删除修改课程、管理学生及助教、查看实验课成绩和到课情况、打分并评价、推送消息和与学生私信联系等多种功能。

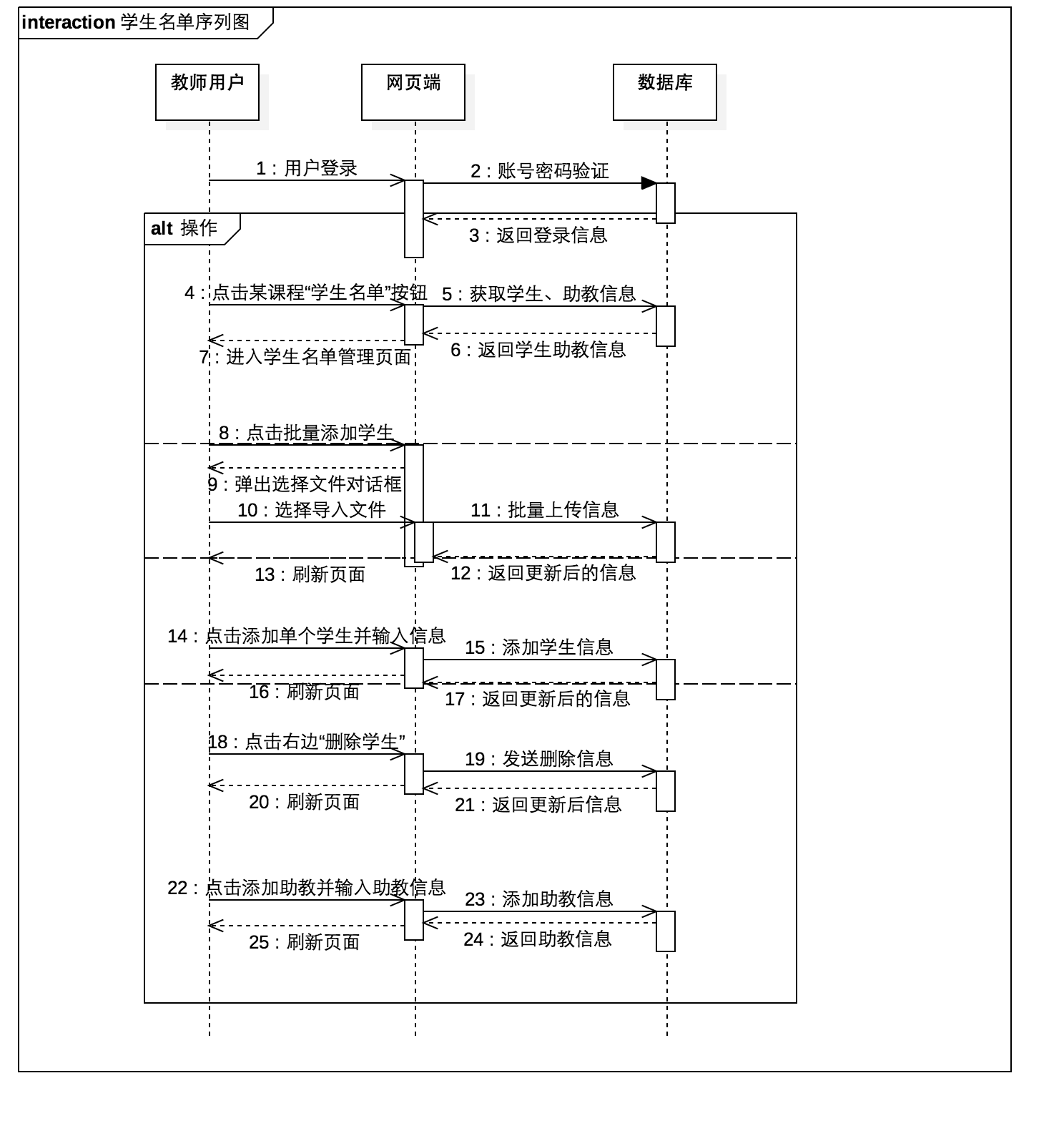
网页端的页面加载流程如下：



下面具体介绍一下各具体功能的运行序列：

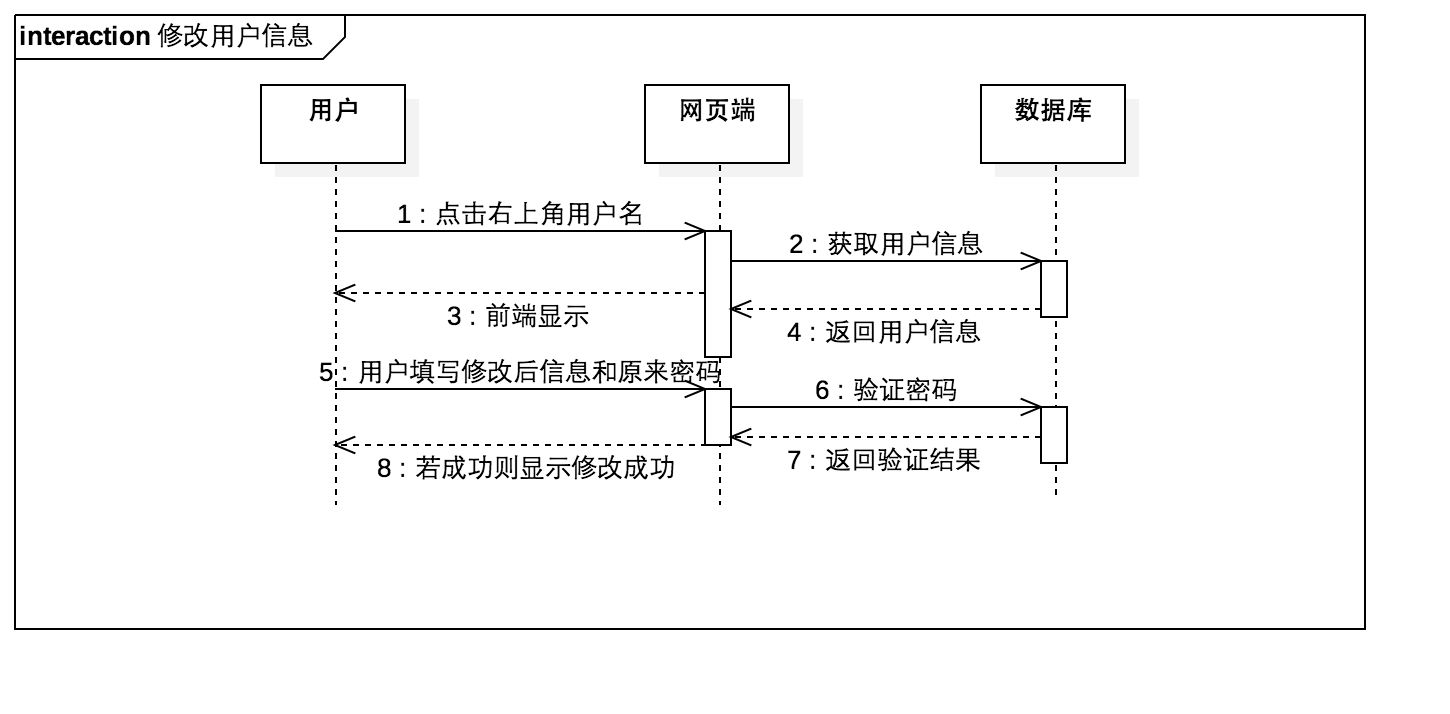
1. 查看课程学生助教名单。

序列图如下：



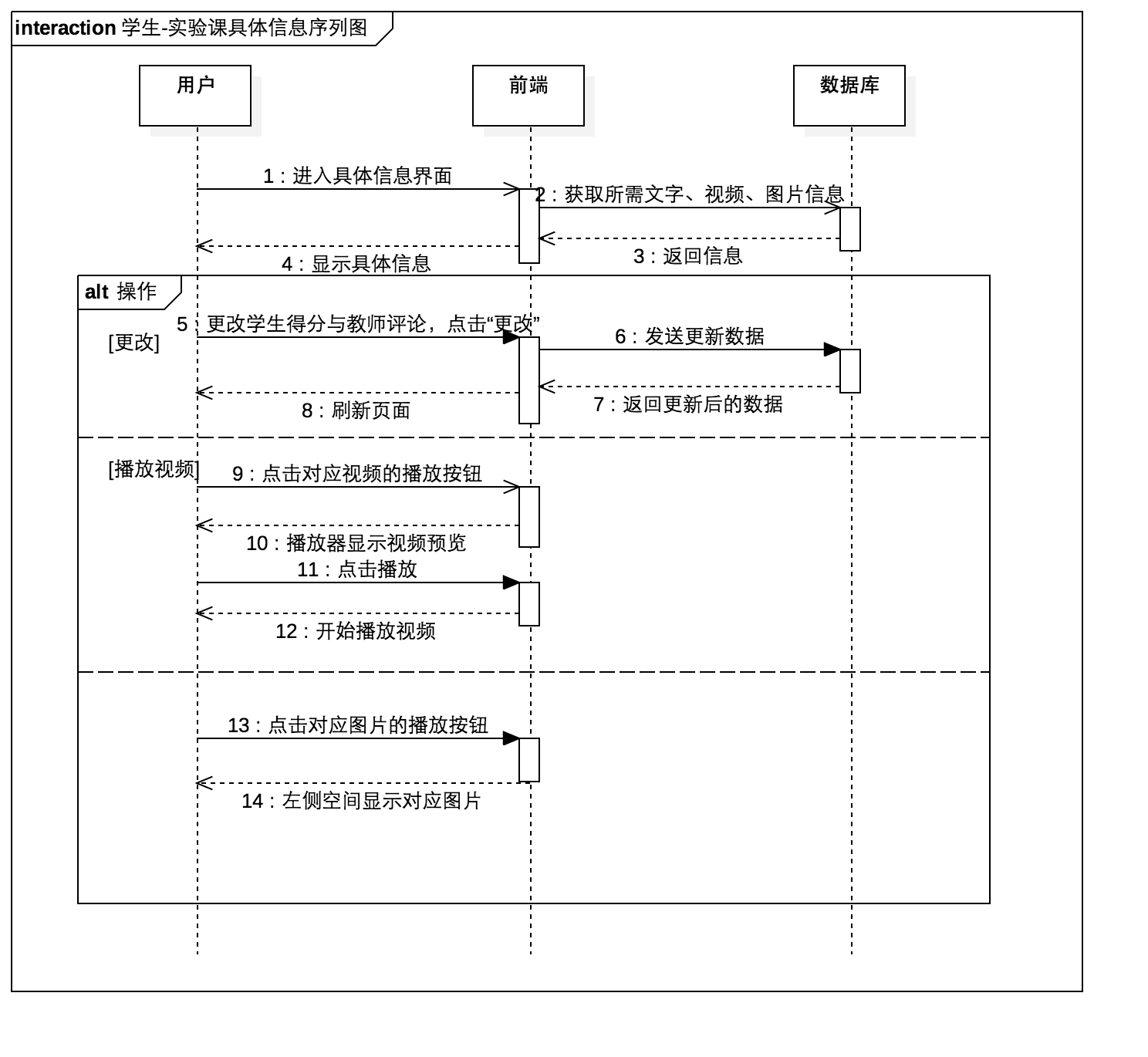
1. 修改用户信息。

序列图如下：



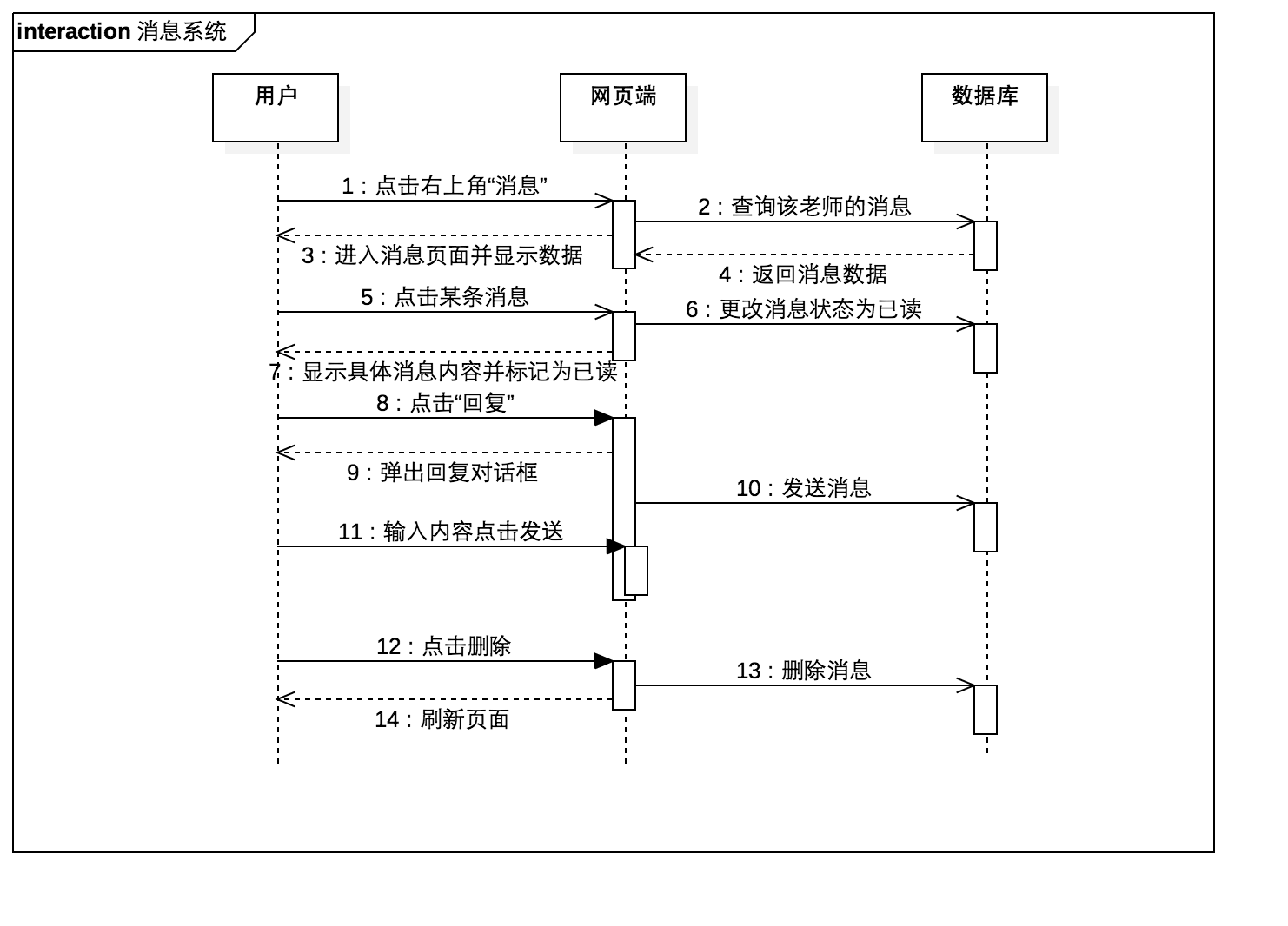
1. 查看具体学生-实验课信息。

序列图如下：

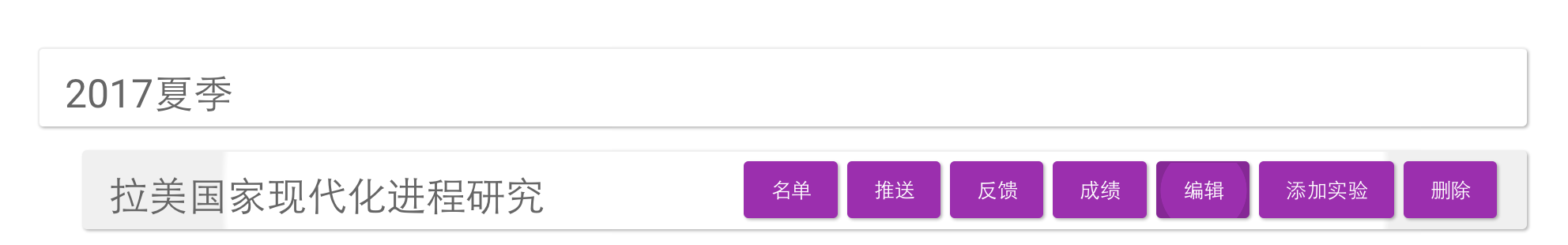


1. 消息系统

序列图如下：



1. 其他功能如导出成绩，修改课程信息等，只需要点击界面对应按钮即可。

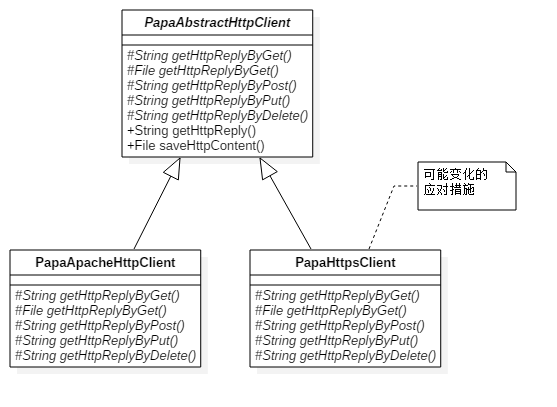


4.3.2 Android端

App 端是为了给学生和助教提供一个课程成绩管理、课程评价管理的平台而设计的。在实现过程中，我们充分利用了当今智能手机普遍具有的GPS、摄像、网络等功能，完成了一键式签到、精彩实验瞬间回放、与教师沟通互动、课程通知提醒等功能。

啪啪打卡 App 端在设计之初就充分考虑到了将来可能会应对到的变化。对于访问数据库的过程，我们将其划分为了几个不同类型的类之间的交互，以尽可能的实现将来的复用。

4.3.2.1 底层HTTP(S) 协议接口



目前本系统所有的信息交互都是基于 HTTP 协议，但众所周知，其安全性令人堪忧。因此，一个可能的变化就是需要使用 HTTPS协议。

在设计之初，我们就考虑设计了一个 HTTP 抽象类，其涵盖了基本的 HTTP和HTTPS的信息交互接口，全部封装成

[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **abstract** **class** PapaAbstractHttpClient;

类。设计之初，我们是通过封装org.apache.http.client.HttpClient包实现的网络访问，故有一个子类

[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **class** PapaApacheHttpClient **extends** PapaAbstractHttpClient;

实现具体的网络访问过程。如果将来需求发生变化，要实现 HTTPS 协议，只需建一个新类

[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **class** PapaHttpsClient **extends** PapaAbstractHttpClient

2 {

3 //...

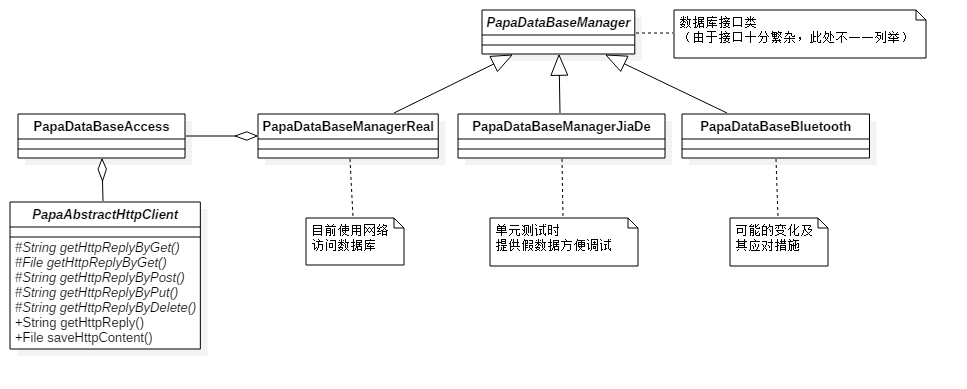
4 }

并实现相关接口即可。

注意到无论是对于用户还是对于我们程序其他部分的逻辑，我们都不关心信息是采用何种协议登录的。因此，这样的设计是合理的。

4.3.2.2 数据库访问接口

再退一步，目前本系统仍然是通过TCP/IP协议来与数据库交互的。可能将来我们会尝试走其他的协议（比如蓝牙协议等），来实现与数据库的交互。这时，整套HTTP(S) 协议的接口都可能无法继续使用。因此，我们设计了我们啪啪打卡标准数据库的相关接口。



[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **abstract** **class** PapaDataBaseManager;

是总的数据库封装类。

[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **class** PapaDataBaseManagerReal **extends** PapaDataBaseManager;

2 **public** **class** PapaDataBaseManagerJiaDe **extends** PapaDataBaseManager;

是通过网络真实访问数据库的类和提供假数据方便单元测试的类。核心代码通过调用接口类提供的接口，实现不同运行状态下的不同功能。

[1](http://stash.secoder.net/projects/PAPA/repos/papa/browse/app/kernel/app/src/main/java/com/Back/NetworkAccess/papa/PapaAbstractHttpClient.java?at=dev#1) **public** **class** PapaDataBaseAccess;

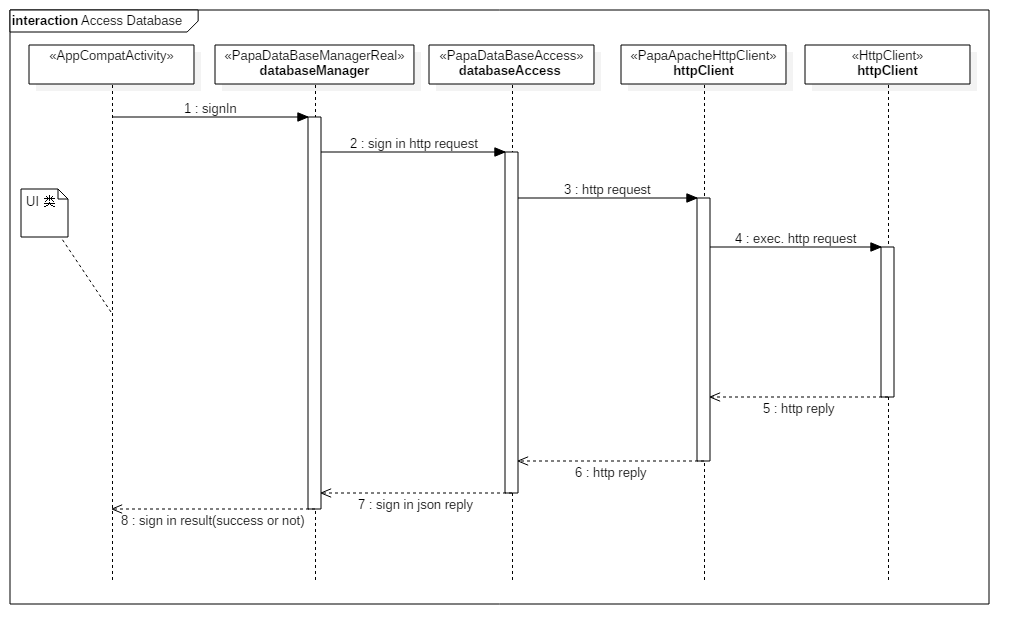
记录了服务器的相关信息，例如

* IP地址
* 端口号
* 等等

并对返回的文本内容进行 JSON 解析，使之能够直接通过org.json.JSONObject 提供的相关接口进行进一步的操作。

4.3.2.3 数据库访问过程

下面以登录为例，展示UI类访问数据库的过程



可以看到，用户的请求通过一层又一层的封装、嵌套，从最处的纯文本信息，最后打包成数据包在网上传输。服务器端收到后，会根据协议解包，执行用户请求。

上图只是以登录为例举了个例子。得益于对数据库接口的定义，对于其他的用户操作，其序列图相同。

5 接口设计

见接口文档 <../db/doc/api.md>

6 出错处理及维护设计

6.1出错处理：

* 如果网络不畅，在设定的时间内数据库没有回复，则会终止网络访问，并向用户提示。

6.2维护设计：

* Android端提供检查更新和错误反馈，在一定程度上实现后期的改进工作。

7 尚待解决的问题

1. 网页端视频播放器对部分格式的视频和部分浏览器支持不好。
2. 不能根据Android系统版本进行不同的适配，兼容性不是很好。
3. android视频使用标准的系统 API 录制。由于当今android 版本不一，因此录制出的视频的编码也不一。在网页端，可能因为出现不常见的编码而导致无法播放视频