HCI小组项目中期报告

周一凡组

2020 年 5 月 11 日

目录

摘要

本报告主要汇报了本小组在准备与初步实现该项目过程中所做的准备工作以及具体的实现方法。详细地介绍了完成的过程及这样实现的原因。

1 课程设计题目简介

小组主要想要设计一个基于姿态识别的动作纠正软件，可以纠正用户的错误动作，提高训练的效果。

1.1 主要目标

能够识别用户的关键部位的位置，根据这些位置坐标得到用户当前动作与标准动作的差别，并且返回提示信息帮助用户纠正。

1.2 创新简述

1.2.1 动作纠正

相比于市面上的其他软件，最大的创新点还是在纠正用户动作这个功能上，这个功能是非常有需求的，尤其是考虑到疫情，一些健身房等专业

的训练场所并不能正常运作。需要锻炼的用户急需一个软件来替代一些专业的指导，帮助自己获得更好的健身效果。

1.2.2 运动效果计算

此外我们还可以根据用户实际的运动效果来计算用户的实际消耗，而不是像其他软件一样靠用户自己填写的运动量来估计，这样的计算是更加精确的。比如用户做了十个俯卧撑，可能一般的软件都是按照标准姿势的十个算的，但是我们可以根据姿态识别过程中用户实际的完成情况来计算，比如用户可能这十个俯卧撑中五个在偷懒，并没有起来，这五个我们就不予计算。

2 主要的交互方式与设计思路

2.1 交互背景

主要考虑用户应用的环境:

- 室内(客厅经济模式、[1])
- 空闲/忙中偷闲时,但是注意力无法集中于产品(用户需要关注自身动作或是身体状态，无法及时关注手机)。

其次是用户的视觉注意模式: 视觉注意模式更倾向于“什么”假说，用户有意识地关注感兴趣的部分。但是在“使用手机APP”这一点上，用户视觉注意是会收到相当干扰的，不论从眼球模型还是人在运动时的注意力分布上，用户很难在健身时将眼球/注意力完全放在APP提供的信息上。至于工作状态，本组APP是娱乐方向，工作影响较少。
但是如果未来真的可以实现APP投屏，在大屏幕上播放健身教程和自己的实时录像，受到的干扰就会大大变小，人们不需要转动身体或是眼球就可以在正面方向观察到侧面的动作信息。至于这种投屏模式，也许会是未来的一个趋势。
此外，由于用户在使用产品时需要专注于自身的动作，可能无法及时并且准确的观看屏幕，因此我们必须采用听觉交互的方式传达动作的改正信息，方便用户修正。

2.2 交互方式与具体设计

2.2.1 视觉交互

小组选题是实时动作纠错APP，对于这样一款APP，主要分为运动时

界面与非运动时界面，两者用户环境不同，视觉交互上也有所区别。主要的交互界面有:

a. 运动时界面:提供健身教程视频和配套文字提示:同时实时反馈用户健身视频。

b. 非运动界面:该手机端APP功能复杂度不高也不能设计得很高。所以界面可以采用简洁美观的图标/图片按钮设计。

c. 登录后界面:界面下方分栏(运动广场 & 排行榜 & 个人运动消耗记录)。

具体说来，运动广场需要健身视频内容吸引眼球，所以使用图片按钮，视频封面grid形式显示，点击即可开始训练。排行榜用列表形式呈现，按照每周好友运动calorie倒序排列，可以有最强的视觉冲击力。个人运动消耗记录类似排行榜。
其中重要的视觉信息有:

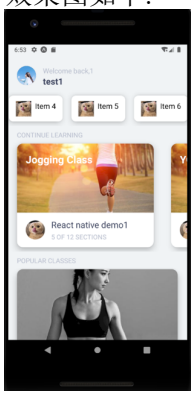
a. 运动时:运动提示和动作示范，要求高的用户会希望看到自己的动作，并和标准动作进行对比。

b. 非运动时：个人运动消耗数值，好友运动消耗数值。（数据是减肥最直观的呈现）

相应的视觉界面也有所不同: 运动时界面:

- 提供健身教程视频和配套文字提示
- 同时实时反馈用户健身视频
- 若动作有误，将通过语音和视觉文字提示问题

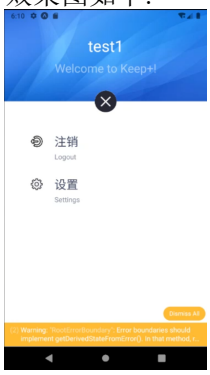
效果图如下:



非运动界面:

- 运动广场主页面→点击教程card→教程详细信息面板
- 菜单栏
- 个人中心页面

效果图如下:



2.2.2 听觉交互

听觉交互的主要应用情景就是健身时(此时用户注意力不在视觉上) 具体说来有以下两个步骤:

- 健身前，进行对焦
  用户在远处根据语音提示“太近/太远/调整方向左转/右转15度”，手机调整至合适角度。
- 健身时
  若用户动作不标准，将进行语音提醒帮助用户纠正

3 技术方案实现情况

3.1 姿态识别

在对openpose-android源码进行了一些拆解研究以后，发现其仍然调用了服务器端神经网络模型，本质上仍然是一个非本地的人体关键点检测工具。且其样例代码与我们的开发工具react-native不符，很难整合到我们的项目中，于是弃用，换成了较为方便的face++API接口，在网络较为通畅的情况下，可以达到满足开发条件的延迟和并发数。

3.2 前端

3.2.1 camera组件

使用reactnative的camera组件,完成拍摄、上传图片，获取返回值的操作。

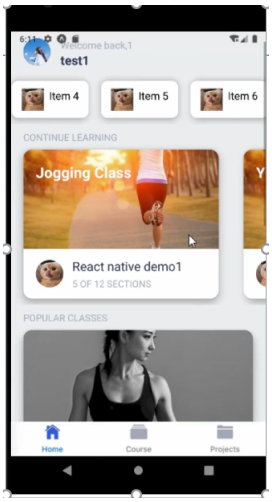
3.2.2 语音合成

语音合成的使用较简单，使用react-native-baidu-vtts包，将需要合成为语音的文字发送给百度API的接口，然后播放即可。但由于在不同的电脑上配置了环境，没有进行项目的整合，故未实装。

```
componentDidMount() {
    // 填写百度语音官网申请的appid, apikey, secretkey
    const appid = '19810843'
    const apikey = '9B2IHu4xxO3dNViNn3iXRgjG'
    const secretkey = 'MRWX8qpUIjj9INDxgWirI32uVmXYfgYU'
    RNBaiduvoice.initBaiduTTS(appid,apikey,secretkey)

}

_speechText = () => {
    RNBaiduvoice.speak('百度语音')
}

render() {
    return (
        <View style={styles.container}>
            {/*<TwoList/>*/}
            <TouchableOpacity onPress={this._speechText}>
                <Text style={{fontSize: 20, height: 30}}>测试语音</Text>
            </TouchableOpacity>
        </View>
    );
}
```

3.2.3 UI界面

利用react-animation保证界面动画流畅度。这一部分技术难度并不大，但是需要花费很多时间去调整界面美观度，以及设置符合审美的流畅界面过渡效果。利用redux，设计前端所需存储状态。效果图如下:



3.3 核心功能流程

从最基本、较简单的动作来验证技术方案的可行性。我们项目选取了俯卧撑，拍摄角度从正面拍摄。

3.3.1 准备阶段

指导用户打开系统音量，以接受后续的指导。该提示为文字，在打开该动作学习时优先弹框提示。后续提示都为语音提示（暂未实装）
指导用户将手机放置在合适的位置，在该例中，其提示语为“请将手机置于高于人体60厘米左右，并向人体方向倾斜约15度”。

```
componentDidMount(){

  setTimeout (this.alert2.bind(this), 1000 * 7);
}

alert2(){
  alert("请打开系统音量，以接收后续语音指导。");
  setTimeout(this.alert3.bind(this),1000*7);

}

alert3(){
  alert("请将手机置于高于人体60厘米左右，并向人体方向倾斜约15度。");
  setTimeout (this.start.bind(this), 1000 * 7);
}
```

此时，用户放置好手机以后，应该已经离开手机前往被拍摄区域。软件开

启检测功能；为了提高检测的准确性，软件会提示用户和摄像头保持一定距离。若人体与摄像头过近，软件会提示“请稍微远离手机”，直到软件可以连续3秒内获得正确的信息。此时软件会提示“开始运动”。

```
if(res.skeletons[0]){
  if(res.skeletons[0].body_rectangle.height > 0.75*height && res.skeletons[0].body_rectangle.width > 0.75*width)
  //if(false)
  {
    this.count =0;
    this.setState({info : "请稍微远离手机，并保证镜头内没有他人"});
  }else{
    let body = res.skeletons[0].landmark;
    this.count ++;
    if(this.count >= 2)
      this.setState({info : "开始运动"});
    alert(1);
    /*
    alert("肩:" + body.left_shoulder.x +"  " + body.left_shoulder.y +"  "+ body.left_shoulder.score + '\n'
        +"肘:" + body.left_elbow.x +"  " + body.left_elbow.y +"  "+ body.left_elbow.score+ '\n'
        +"手 :" + body.left_hand.x +"  " + body.left_hand.y +"  "+ body.left_hand.score+ '\n');*/
  }


}else {
  this.count =0;
  this.setState({info : "请到镜头前方"});
}
```

3.3.2 检测阶段

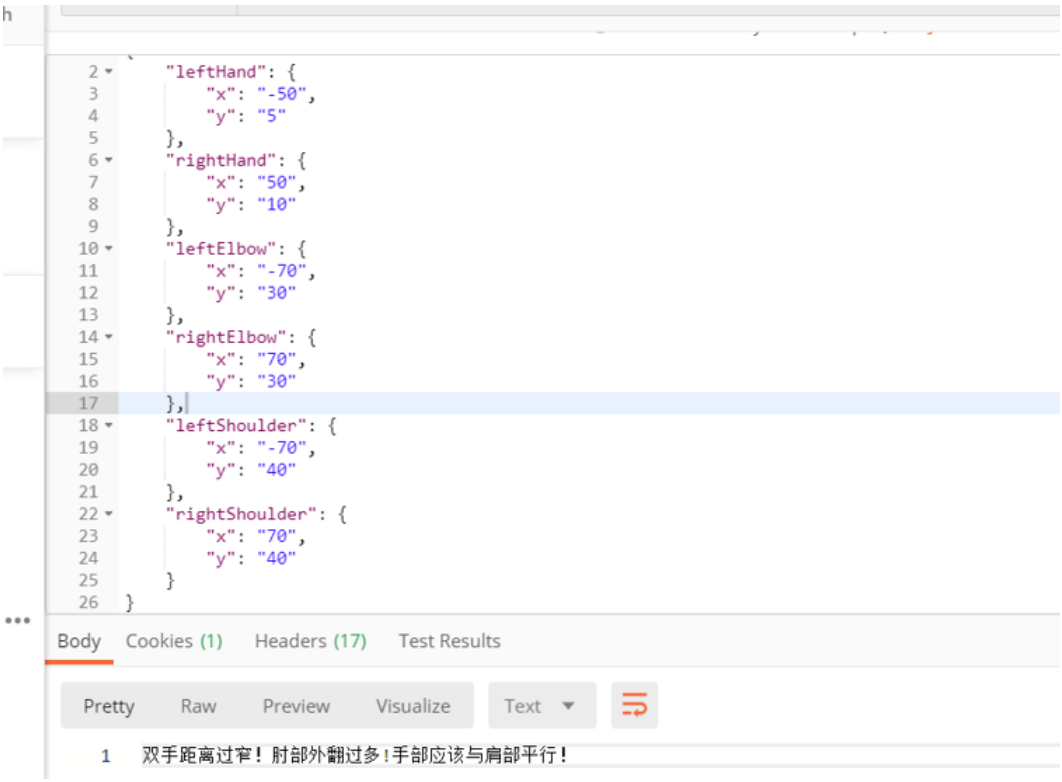
为了较全面的验证可行性，在俯卧撑实例中，我们检测两种姿势：一是双手间距过宽，此为较静态的运动姿势检测（因为用户在做俯卧撑的时候一般不会主动移动手掌撑的位置）；二为手肘过度外翻，此为动态的运动姿势检测。将获取到的参数传递给后端进行计算:

- 如果检测到双手间距过宽，软件会提示“请保持双手间距比肩略宽”。

- 如果检测到手肘过度外翻，软件会提示“请将手肘向身体收缩”。

3.3.3 计算并返回结果阶段

后端根据前端识别姿态传回的六个关键点的数据，进行了数据的处理，根据数据来判断用户的姿态是否标准，若不标准，返回对应的提示帮助用户改善效果如下图:



目前采用的还是我们人工设定参数进行计算与判断，后续将考虑利用深度学习，通过研究大量标准姿势的关键点位置来生成一个模型，借助此模型判断这个姿势是否标准。后续将与前端相结合，通过语音的方式返回这些提示，帮助用户改善姿态。

4 已完成工作

4.1 前端

4.1.1 确定界面风格

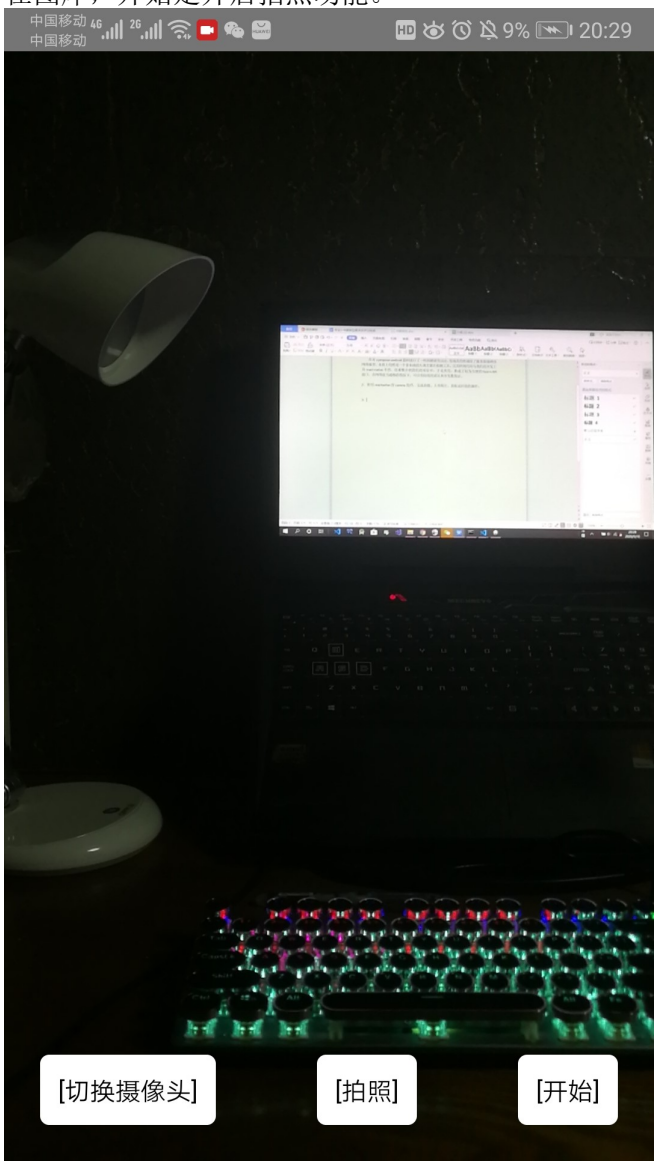
本项目作为用户向App，需要有符合大众审美的界面UI以及流畅的操作手感。采用欧美化的扁平化界面风格，主体采用蓝绿色调，体现青春健身的主题。

此外，集成React-Animation和styled-components 组件。React-Animation能够方便实现界面组件平移、缩放等功能，并且与react-native内置功能相结合，可以适配不同大小、分辨率的android设备，在视觉上给扁平化界面带来3D跳转效果。在弹出菜单栏和教程详细页面时，带来流畅舒适的视觉感受。Styled-components[2]可以非常方便地定义组件样式，并且不同文件中

的同一组件在渲染时会自动生成不同名称，防止样式表出现覆盖错误。
在此基础上，前端已经实现了运动广场页面、简单菜单栏及登录/注销。

4.1.2 摄像头的测试与使用

为了完成用户的录像等功能，我们使用reactnative的camera组件，完成拍摄、上传图片，获取返回值的操作。react-native-camera组件的作用是在rn项目开发中调用手机摄像头进行拍照和录像。其难度本身并不大，但是配置和集成的步骤比较繁琐。在一系列配置完毕后，可以得到如下的效果图。其中切换摄像头是切换前置、后置摄像头，拍照即拍下一张照片存在图库，开始是开启拍照功能。



```
<View style={styles.container}>
  <RNCamera
    ref={(cam) => {
      this.camera = cam;
    }}
    style={styles.preview}
    type={this.state.cameraType}
    >
    <Text style={styles.button} onPress={this.switchCamera.bind(this)}>[切换摄像头]</Text>
    <Text style={styles.button} >[{this.state.info}]</Text>
  </RNCamera>
</View>
```

rn-camera相比于手机自带的摄像机软件，在性能方面的落后是比较大的，即拍摄一张照片需要的时间比较久，大约在3秒左右。在经过资料的查找和分析以后，我发现其主要耗时并不是拍照这一动作，而是高精度图像数据由于不像手机原生的图库有直接访问硬件的接口，需要经过多步转储这将近2-3M的数据，最终传递到软件中供我使用。

这个步骤是比较耗时的，所以我认为最简单也最直接的方式是减小图片大小。从两个方面下手，一是分辨率；二是图片拍摄较靠近中央的区域。幸运的是，在查看源码时发现拍照的接口暴露了这两个参数供我调整。

```
  const options = { quality: 0.5, base64: true,width : 960 };
photo = await this.camera.takePictureAsync(options);
```

分别是quality和width。在不断的调参下，找出了一组合适的参数，既能优化性能，又能保证后续人体关键点检测可以正常进行。优化后，这一步骤大约在1-1.5秒。

最后是设置软件在开始运动后（精确地说是用户放置好手机并离开后）定时拍照。由于免费API的QPS（每秒并发数）限制和上述camera流程时间的限制，频率不能过快。为了保证软件能正常运作，设置了1.8-2.2秒的间隔拍摄一张照片并上传。

4.1.3 face++ API调用

先配置调用API所必需的appkey等密钥。随后将camera获得的图片打包进文件，用POST方法调用API。

```
let data = {
    api_key: 'xqYKV2nwumRpfBxbF0THCxW0mNzZqUnC',
    api_secret: '73DpLjoIhxTsq-NN56we4NM-7Kf-lc8b',
    image_base64: photo.base64,
}
```

```
const formData = new FormData();
formData.append('api_key', data.api_key);
formData.append('api_secret', data.api_secret);
formData.append('image_base64', data.image_base64);
```

```
await axios({
    url: url,
    method: 'POST',
    cache: false,
    data: formData,
    processData: false,
    contentType: false,
```

采用异步方法获得返回值，这一步骤大概在2-2.5秒左右。返回值为JSON类型字符串，里面包含了人体各关键点的坐标数据。按格式将数据提取出来发送给后端即可。

4.2 交互技术

交互技术的部分主要是百度语音包的初步研究与运用。

首先，在百度Ai官网注册，创建应用项目，获得appId、appKey、secretKey。然后分析Github开源项目 [3]，配置react-native-baidu-vtts,需要事先在android文件夹下配置gradle信息。运行代码，测试百度语音。

```
componentDidMount() {
    // 填写百度语音官网申请的appid, apikey, secretkey
    const appid = '19810843'
    const apikey = '9B2IHu4xxO3dNViNn3iXRgjG'
    const secretkey = 'MRWX8qpUIjj9INDxgWirI32uVmXYfgYU'
    RNBaiduvoice.initBaiduTTS(appid,apikey,secretkey)

}

_speechText = () => {
    RNBaiduvoice.speak('百度语音')
}

render() {
    return (
        <View style={styles.container}>
            {/*<TwoList/>*/}
            <TouchableOpacity onPress={this._speechText}>
                <Text style={{fontSize: 20, height: 30}}>测试语音</Text>
            </TouchableOpacity>
        </View>
    );
}
```

但是此开源项目对react-native本身的版本有要求，所以后续还需要完成百度离线Android语音SDK的构建[4]。现阶段对官网下载的SDK demo进行分析，将按照百度集成文档[5]配置so文件、apk文件以及jar包。

4.3 后端

4.3.1 基本框架的搭建

在后端框架方面，我们选用了有丰富实战经验的的springboot + mysql作为基本框架，由于不需要后续的负载均衡，因此可以直接使用内置的tomcat，总体来说十分方便。确定好框架以后，我们就开始着手编写基本的框架，第一步是在springboot的生成网站生成一个我们需要的demo[6]。在根据使用的依赖生成了demo后，就有了基本的maven结构和一个基础主方法以及测试类。然后我们就开始着手进行最基本的功能的实现，首先便是配置好数据库。由于我们决定采用的是jwt的权限管理系统，因此并不需要使用其余数据库来存储session，我们只需要将mysql配置好即可。使用mysql首先需要在计算实例上安装mysql server，安装完以后设置好root的密码即可使用，创建好数据库hci_local,随后springboot便可通过url连接到这个数据库。在springboot上配置DataSource的url为localhost:3306/hci_local，并且配置好用户名以及密码，即可正常使用。
在配置好了mysql数据库后，基本框架就搭建完成了。

4.3.2 用户权限管理的实现

我们在商讨之后决定采用jwt来实现权限管理，主要原因是jwt token有以下的优点:

- 安全性高
  jwt使用密文签名进行权限控制，在大部分情况下都能保证用户权限信息的安全。
- 方便应用于移动端
  在许多原生的移动端，cookie的存储是不被支持的(需要自己对cookie容器进行处理)，这个时候使用token进行验证就会方便许多。
- 易于实现
  jwt在java中提供了大量的支持，可以直接调用许多jwt的库来实现，实现起来非常方便。

实现的过程首先需要我们进行一些基本属性的配置，主要是jwt的expiration，考虑到是在移动端进行应用，我们的app可以在本地长时间存储token，因此我们将expiration设置为一个月，这也比较符合现在移动端登录长期有效的趋势。在设置完成后，我们首先着手实现了一个简单的jwt Provider类，这个类主要提供了token的生成方法与token的验证方法，以及根据token获取用户名的方法，这三个方法是jwt验证的基础。

具体的实现方法也比较简单，token生成只需要调用库函数Jwts.builder并且设置好几个参数即可，token获取用户名只需要parse后get subject即可，而token的验证也只需要调用库函数即可，总体来说没有什么特别复杂的工作，只需要根据需要调用库函数即可。

随后我们在provider的基础上实现了一个jwt监听器(Filter),用于验证每个http请求的权限，实现的方法就是在http请求的header中读取对应的key(Authorization),然后验证这个key对应的value是否Bearer Token，如果是，则将token对应的用户权限载入权限上下文中，完成权限的获取过程。

最后实现一个简单的入口函数，用于在用户权限不足时返回报错信息即可。

至此jwt的部分已经实现完成，当然这部分还是不够的，因为在权限控制的过程中还需要有一个类能够提供用户的信息，来帮助jwt进行验证与权限管理。这部分在spring security的UserDetail类以及UserService类中已经覆盖了，这也是spring对jwt的一部分支持。因此我们要做的工作就是实现这两个接口类，实现的过程也比较简单，将各个接口根据其语义依次实现并返回值即可。

实现完两个接口类以后，我们的工作就基本完成了，接下来只需要加入一个配置类就大功告成了，这个配置类只需要将前面实现的各种功能类注入即可，当然我们还可以在这个配置类中配置一些不需要权限的url，比如注册与登录的接口url，完成配置类后权限控制模块就彻底完成了。

当然在实现的过程中还是存在了一些不足，突出的问题是实现的过于机械化，导致中间多了一些不必要的无用函数，这也是因为jwt通常还带有权限级的管理控制，但是在我们这个应用中其实并没有这个需求，但是在实现的过程中还是将这部分耦合进入了代码中，导致了过程中的一些多余代码，而且因为这部分多余代码，后面又耦合地产生了更多多余代码，这是没有透彻理解运行机制的问题。不过好处就是这样实现会比较快一点，而且后续如果想要加入权限级控制也会比较直接。

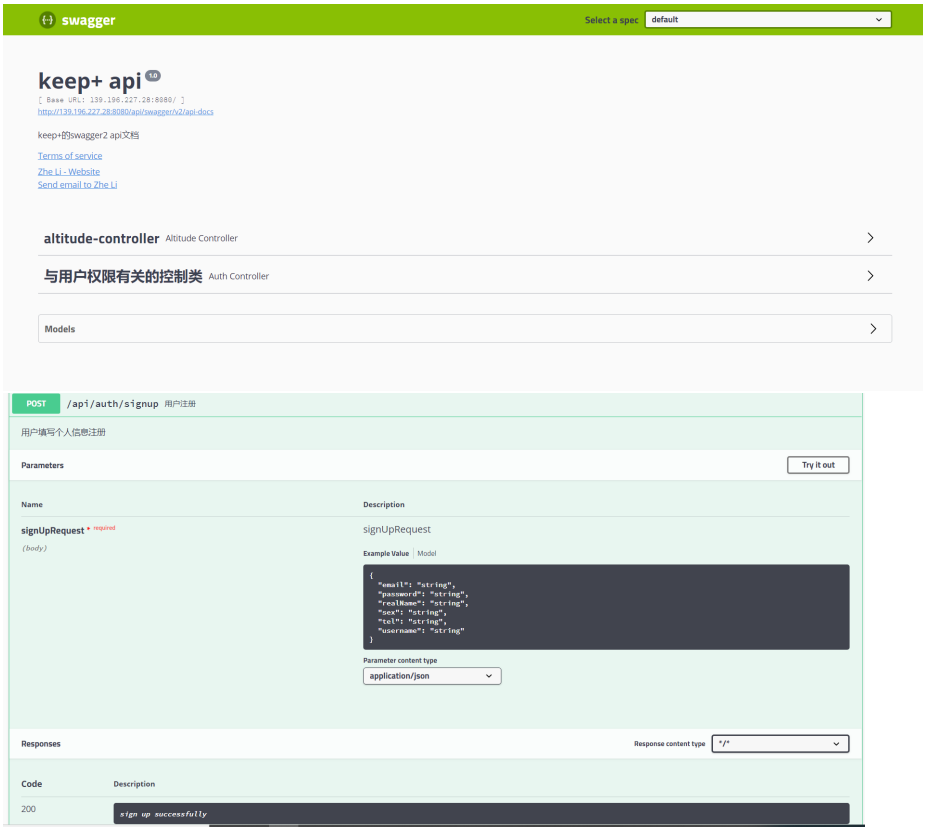
4.3.3 注册与登录的接口实现

完成了权限控制的基本框架以后，我们就要着手实现用户注册与登录的接口。这部分的实现需要与前端进行交互，因此我们需要设计一些request body作为表格，方便与前端交互传递信息。设计的过程也比较简单，只需要加入一个类，并且加入几个私有成员作为信息，如注册表格，私有成员就有用户名，密码，手机号码，邮箱等，然后生成相应的getter and setter，constructor即可。当然为了功能的鲁棒性，还加入了一些验证，比如手机号码使用正则表达式进行匹配，防止用户的非法输入，并且要求用户的一

些关键信息不能为空。
完成request body的设计后，我们遵循controller，service，serviceimpl的顺序依次实现这个接口，当然在这之前，还需要增加一个userRepository与数据库进行交互，存取与读取用户的数据，这个在spring中也非常容易实现，只需要调用jpa接口即可。 controller就非常简单了，只需要用request map对应好url，然后调用service的接口即可。 service也是一样，只要规定好注册与登录的接口即可。我们的业务逻辑主要集中在serviceImpl这一块，注册的业务逻辑就是首先进行验证，主要是验证用户名，手机号码有没有重复，如果没有重复我们就允许用户进行注册，验证完以后，我们出于安全性的考虑，使用了spring security的encoder接口对密码进行了加密，在数据库中存储了用户密码的密文，进一步的提高了账户的安全性，防止了sql注入等攻击。在验证完信息后，将用户信息与加密后的密码存入数据库，就完成了用户的注册。用户的登录需要根据用户的用户名和密码进行authentication的生成，生成后加入权限上下文，然后让jwt Provider生成jwt token并返回即可。返回后前端就可以将这个token存入本地，并且在后续进行http请求时将这个token带入头部，在有效期内可以一直使用这个token，无须再次登录。
至此，用户的注册与登录也完成了，我们的基本功能也就实现完成了，下面就可以进行接口文档的生成与部署了。

4.3.4 接口文档的生成

由于疫情的影响，我们小组的成员不得不各自在家完成这次的开发，因此交流上多有不便，需要一个更加方便的接口文档进行交互。我们在商量以后，决定采用swagger进行接口文档的自动生成，主要是考虑到了这个swagger2可以在springboot非常方便的集成，而且十分方便，只需要在代码中加入Annotation即可，接口文档也非常美观。
加入的过程也比较简单，只需要在几个关键的配置类和Main方法类加入Annotation @EnableSwagger2，然后写一个主类生成主界面即可。当然因为我们需要在服务器上部署这个，因此还需要加入一些配置来转发请求。这部分比较繁琐也没有什么技术价值，因此不多赘述，生成完成后只需要在对应的地方加入Annotation并且进行相应的注释即可。
最后的生成效果如图所示:



4.3.5 俯卧撑的标准姿势识别

最后是一个业务逻辑，这个接口需要我们根据传入的六个关键点的坐标，判断用户的姿态是否标准，主要的判断依据有两个，一是双手与双肩的相对位置，看双手是否过宽或者过窄，二是手肘是否外翻，主要看手肘与手的相对位置。现在的判断逻辑比较naive，主要是通过后端写死的一个算法进行粗略的判断，根据双手与双肩位置的相对宽度来判断，而双肘外翻方面则是根据手肘与手的位置计算斜率，根据斜率来判断手肘是否外翻。这部分功能相对容易实现，只需要进行一些简单的数学运算即可。

4.4 服务器的部署

4.4.1 服务器的选择

在比较了各家知名厂商的云计算资源后，我们最终选择了阿里云的学生服务器，主要是价格便宜，并且计算资源足够，包含一个CPU与2G内存，40G本地存储，基本足够满足我们的基本功能。

4.4.2 springboot的部署

在租用得到云服务器后，我们开始着手进行前面已经写好的springboot的部署。首先在本地进行maven打包，然后将得到的jar包通过软件传输到云端服务器，然后安装好对应的jdk版本，安装mysql并完成配置，利用脚本进行不间断运行，完成部署，部署过程主要参考了这个网站的教程[7]，非常的详细。

4.4.3 版本管理

版本管理主要是借助github，由李哲创建了一个项目[8]并添加了其余两人作为collaborator，三人根据分工在各自的分支开发，定期merge。

4.4.4 持续集成与自动部署

为了方便后续的集成与部署，采用了jenkins来进行自动集成。 jenkins是一个非常高效实用的持续化集成平台，其主要的优势有[9]:

- 持续集成和持续交付
  作为一个可扩展的自动化服务器，Jenkins 可以用作简单的 CI 服务器，或者变成任何项目的持续交付中心。

- 简易安装
  Jenkins 是一个基于 Java 的独立程序，可以立即运行，包含 Windows、Mac OS X 和其他类 Unix 操作系统。

- 配置简单
  Jenkins 可以通过其网页界面轻松设置和配置，其中包括即时错误检查和内置帮助。

- 插件
  通过更新中心中的 1000 多个插件，Jenkins 集成了持续集成和持续交付工具链中几乎所有的工具。

- 扩展
  Jenkins 可以通过其插件架构进行扩展，从而为 Jenkins 可以做的事提供几乎无限的可能性。

- 分布式
  Jenkins 可以轻松地在多台机器上分配工作，帮助更快速地跨多个平台推动构建、测试和部署。

综上，我们最终采用了jenkins进行持续集成的工具。应用jenkins的过程主要参考了这篇教程:[10]，过程比较琐碎也没有什么技术含量，就不赘述了。

5 后续工作与分工安排

5.1 迭代周期安排

迭代周期	日期	具体任务
迭代1	5.11-5.24	完成语音功能的实装，完成UI的全部设计，后端着手准备深度学习相关的准备工作。
迭代2	5.25-6.7	优化延时等问题，完善提示功能的交互设计，开始训练一些简单的锻炼的标准姿势判断模型。
迭代3	6.8-6.30	进行最后的运行与维护，提高用户体验，改善软件质量

5.2 待解决的难题

- 深度学习模型的可用性
  因为不清楚该类的深度学习是否有人曾经尝试过，因此目前来说我们还不能确保这个一定能达到预期的效果。我们预期对大量的标准动作的图片进行识别得到大量训练集，以此为基础训练一个标准模型，并且期望这个标准模型可以涵盖大部分的体型情况，但是目前尚不清楚需要几层神经网络对这个模型进行计算，也不清楚能否利用这个模型进行标准验证。

- 调用api的高延时问题
  由于本地计算资源的匮乏，因此我们在姿态识别方面调用的是免费的api进行计算，由于这个api是免费的并且开放的性能有限，因此成为了我们整个项目主要的性能瓶颈，但是这个问题因为是个黑盒调用，我们也很难进行改善，只能在用户提示等方面完善来提高用户体验。

- 录像过程的用户体验问题
  因为我们进行姿态识别，对用户录像的角度有一定的要求，可能会导致用户体验很差，后续可能要进行一些不同角度的处理来完善。

5.3 分工

- 李哲:
  主要负责后端的业务逻辑编写与部署，服务器的性能提升与持续化集成。
- 周一凡:
  主要负责交互部分的实现，包括调用摄像头进行录像与姿态识别部分的实现。
- 黄雨瑒:
  主要负责交互部分的实现，包括UI的设计与语音提示等功能的实现。

参考文献

[1] https://www.sohu.com/a/125966299_223768, sohu, 客厅经济

[2] https://styled-components.com/ , styled-components, styled-components官方网站

[3] https://github.com/wayne214/react-native-baidu-vtts, wayne214, react-native-baidu-vtts开源代码

[4] https://blog.csdn.net/wayne214/article/details/85045229 , wayne214, ReactNative集成百度语音合成

[5] https://ai.baidu.com/ai-doc/SPEECH/Ok83w9anr , baidu, 百度语音合成集成指南

[6] https://start.spring.io , spring , springboot生成器

[7] https://blog.csdn.net/Mou_Yang/article/details/102137861, 某羊, 部署详细流程

[8] https://github.com/lz18064638846/hci_project , 李哲 , github地址

[9] https://www.jenkins.io/zh/ , jenkins , jenkins中文官网

[10] https://blog.csdn.net/boling_cavalry/article/details/78943061, 程序员欣宸, jenkins自动构造详细流程