基于人脸识别的考勤系统

详细设计

[V1.0.0]

拟 制 人\_\_\_\_\_\_刘万顺禧、黄勇强\_\_\_\_\_\_\_\_\_\_

审 核 人\_\_\_\_\_\_\_\_钟志鹏、李进杰\_\_\_\_\_\_\_\_\_\_

批 准 人\_\_\_\_\_\_\_\_\_\_\_陈嘉昕\_\_\_\_\_\_\_\_\_\_\_\_\_\_

[ 二零二一 年 五 月 十四 日 ]

目录

[基于人脸识别的考勤系统 1](#_Toc72266474)

[1 引言 7](#_Toc72266475)

[1.1 编写目的 7](#_Toc72266476)

[1.2 背景 7](#_Toc72266477)

[1.3 参考资料 8](#_Toc72266478)

[1.4 术语定义及说明 9](#_Toc72266479)

[2 设计概述 10](#_Toc72266480)

[2.1 任务和目标 10](#_Toc72266481)

[2.2 需求概述 10](#_Toc72266482)

[2.3 运行环境概述 10](#_Toc72266483)

[2.4 条件与限制 11](#_Toc72266484)

[2.5 详细设计方法和工具 11](#_Toc72266485)

[3 系统详细需求分析 11](#_Toc72266486)

[3.1 详细需求分析 11](#_Toc72266487)

[3.1.1 教师客户端功能模块 11](#_Toc72266488)

[3.1.2 人脸识别模块 12](#_Toc72266489)

[3.1.3 详细性能需求分析 12](#_Toc72266490)

[3.1.4 详细系统运行环境及限制条件分析 12](#_Toc72266491)

[3.2 详细接口需求分析 13](#_Toc72266492)

[3.2.1 外部接口 13](#_Toc72266493)

[3.2.2 内部接口 13](#_Toc72266494)

[3.3 教师模块 13](#_Toc72266495)

[3.3.1 查询学生用户信息 13](#_Toc72266496)

[3.3.2 删除目标学生用户 14](#_Toc72266497)

[3.3.3 修改学生用户信息 14](#_Toc72266498)

[3.3.4 查询学生用户考勤日志 14](#_Toc72266499)

[3.4 学生模块 15](#_Toc72266500)

[3.4.1 查询个人信息 15](#_Toc72266501)

[3.4.2 修改个人信息 15](#_Toc72266502)

[3.4.3 查询个人考勤日志 15](#_Toc72266503)

[3.4.4 删除账户 16](#_Toc72266504)

[3.5 人脸识别模块 16](#_Toc72266505)

[3.5.1 人脸图像采集 16](#_Toc72266506)

[3.5.2 人脸特征提取 16](#_Toc72266507)

[3.5.3 人脸特征数据比对 17](#_Toc72266508)

[3.6 考勤模块 17](#_Toc72266509)

[3.6.1 考勤日志 17](#_Toc72266510)

[3.6.2 出勤人数统计 17](#_Toc72266511)

[3.6.3 出勤率统计 18](#_Toc72266512)

[4 系统详细设计 18](#_Toc72266513)

[4.1 总体架构设计 18](#_Toc72266514)

[4.1.1 功能模块设计 18](#_Toc72266515)

[4.1.2 系统部署图 19](#_Toc72266516)

[4.2 人脸识别模块 21](#_Toc72266517)

[4.2.1 模块描述 21](#_Toc72266518)

[4.2.2 处理流程 22](#_Toc72266519)

[4.2.3 MTCNN网络 23](#_Toc72266520)

[4.2.4 FaceNet网络 24](#_Toc72266521)

[4.2.5 性能 25](#_Toc72266522)

[4.2.6 输入项 25](#_Toc72266523)

[4.2.7 输出项 25](#_Toc72266524)

[4.2.8 限制条件 25](#_Toc72266525)

[4.3 学生管理模块 26](#_Toc72266526)

[4.3.1 模块描述 26](#_Toc72266527)

[4.3.2 处理流程 26](#_Toc72266528)

[4.3.3 性能 26](#_Toc72266529)

[4.3.4 输入项 26](#_Toc72266530)

[4.3.5 输出项 26](#_Toc72266531)

[4.3.6 限制条件 26](#_Toc72266532)

[4.4 教师管理模块 27](#_Toc72266533)

[4.4.1 模块描述 27](#_Toc72266534)

[4.4.2 处理流程 27](#_Toc72266535)

[4.4.3 性能 27](#_Toc72266536)

[4.4.4 输入项 27](#_Toc72266537)

[4.4.5 输出项 27](#_Toc72266538)

[4.4.6 限制条件 27](#_Toc72266539)

[4.5 教室管理模块 27](#_Toc72266540)

[4.5.1 模块描述 27](#_Toc72266541)

[4.5.2 处理流程 28](#_Toc72266542)

[4.5.3 性能 28](#_Toc72266543)

[4.5.4 输入项 28](#_Toc72266544)

[4.5.5 输出项 28](#_Toc72266545)

[4.5.6 限制条件 28](#_Toc72266546)

[4.6 课程管理模块 28](#_Toc72266547)

[4.6.1 模块描述 28](#_Toc72266548)

[4.6.2 处理流程 28](#_Toc72266549)

[4.6.3 性能 29](#_Toc72266550)

[4.6.4 输入项 29](#_Toc72266551)

[4.6.5 输出项 29](#_Toc72266552)

[4.6.6 限制条件 29](#_Toc72266553)

[4.7 考勤记录模块 29](#_Toc72266554)

[4.7.1 模块描述 29](#_Toc72266555)

[4.7.2 处理流程 29](#_Toc72266556)

[4.7.3 性能 29](#_Toc72266557)

[4.7.4 输入项 30](#_Toc72266558)

[4.7.5 输出项 30](#_Toc72266559)

[4.7.6 限制条件 30](#_Toc72266560)

[5 数据结构设计 30](#_Toc72266561)

[5.1 数据库部署结构图 30](#_Toc72266562)

[5.2 数据库物理结构设计 30](#_Toc72266563)

[5.2.1 学生表 30](#_Toc72266564)

[5.2.2 课程表 31](#_Toc72266565)

[5.2.3 教师表 31](#_Toc72266566)

[5.2.4 教室表 31](#_Toc72266567)

[5.2.5 考勤记录表 31](#_Toc72266568)

[5.2.6 图片表 32](#_Toc72266569)

[5.3 数据库逻辑结构设计 32](#_Toc72266570)

[5.4 数据结构与程序的关系 33](#_Toc72266571)

[5.5 数据模型 33](#_Toc72266572)

[5.5.1 models.Teacher 33](#_Toc72266573)

[5.5.2 models.Student 34](#_Toc72266574)

[5.5.3 result.Result 35](#_Toc72266575)

[6 接口详细设计 36](#_Toc72266576)

[6.1 GET 查询课程 36](#_Toc72266577)

[6.1.1 请求参数 36](#_Toc72266578)

[6.1.2 返回结果 36](#_Toc72266579)

[6.2 GET 查询课程 36](#_Toc72266580)

[6.2.1 请求参数 36](#_Toc72266581)

[6.2.2 返回结果 37](#_Toc72266582)

[6.3 GET 查询记录 37](#_Toc72266583)

[6.3.1 请求参数 37](#_Toc72266584)

[6.3.2 返回结果 37](#_Toc72266585)

[6.4 POST 创建记录 38](#_Toc72266586)

[6.4.1 请求参数 38](#_Toc72266587)

[6.4.2 返回结果 38](#_Toc72266588)

[6.5 GET 获取学生列表 38](#_Toc72266589)

[6.5.1 请求参数 38](#_Toc72266590)

[6.5.2 返回结果 39](#_Toc72266591)

[6.6 GET 获取学生 39](#_Toc72266592)

[6.6.1 请求参数 39](#_Toc72266593)

[6.6.2 返回结果 39](#_Toc72266594)

[6.7 PUT 更新学生信息 39](#_Toc72266595)

[6.7.1 请求参数 39](#_Toc72266596)

[6.7.2 返回结果 40](#_Toc72266597)

[6.8 GET 获取老师列表 40](#_Toc72266598)

[6.8.1 请求参数 40](#_Toc72266599)

[6.8.2 返回结果 41](#_Toc72266600)

[6.9 PUT 更新老师信息 41](#_Toc72266601)

[6.9.1 请求参数 41](#_Toc72266602)

[6.9.2 返回结果 41](#_Toc72266603)

[6.10 GET 获取老师 41](#_Toc72266604)

[6.10.1 请求参数 41](#_Toc72266605)

[6.10.2 返回结果 42](#_Toc72266606)

# 引言

## 编写目的

本概要设计说明书用于描述基于人脸识别的考勤系统项目的整体系统设计方案，本说明书以《用户需求说明书》和《系统需求规格说明书》作为依据，本说明书是系统详细设计的依据，也作为系统测试需求和系统测试计划和测试程序的重要参考。

## 背景

当下课堂考勤仍旧是各大高校学生教育管理的基本手段，也直接影响着教学质量的好坏以及课堂效率的高低。但目前高校中的校园考勤系统普遍使用纸质签到、APP签到、指纹识别等良莠不齐的方式；传统的手机签到过于依赖于移动工具且远程签到的“取巧”带来的疏漏，较为优质的指纹识别则存在着卫生隐患与经济因素带来的普及难问题；大量弊端和缺点难以避免。

高校目前比较常用的考勤方式有以下几种:

1）考勤本考勤。这是最传统也是最常见的考勤方式，由教师按照考勤本（学生名单）点名考勤，或由学生在考勤本上签字。优点是简单，缺点是比较麻烦，需要教师配备助教。

2）打卡考勤机。在固定位置设置考勤机，由学生将自己的考勤卡片放入考勤机中考勤。优点是简单，缺点是麻烦，这个方法几乎被淘汰了。

3) “一卡通”考勤。利用“一卡通”射频卡在考勤机上进行考勤登记是一种相对成熟的考勤手段，优点是部署成本较低，缺点是对于“一卡通”丢失的学生来说不是很友好。

4）手机考勤。手机考勤有多种方式：基于射频卡、GPS、APP等。优点在于考勤数据易于采集和统计，形式友好，易于推广，缺点是个体手机存在差异性（手机操作系统、卡片型号等差异)，也不能防范考勤作弊。

4）指纹考勤。指纹识别是古老的生物识别技术，国内外从事自动指纹识别技术研究的机构和公司众多，其中具有代表性的公司有Intel、IBM等。优点是简单，缺点是技术复杂。

6）人脸考勤。与指纹考勤方式类似，人脸考勤也基于生物识别技术，常见的人脸考勤系统只是将指纹考勤系统的指纹采集设备换成了摄像头，它不像指纹考勤一样具有接触性，考勤方式比较友好。

面对困境，人脸识别技术脱颖而出，人脸识别是一项生物鉴定技术。与其他识别技术相比，人脸识别技术具有应用广泛、误警率低、方便快捷、性能稳定、直观易操作等优点，而近年来随着人脸识别算法大量采用卷积神经网络模型，人脸识别的速度和准确率也已经得到极大提升，不少针对特定场景的人脸识别应用已经出现在我们的生活中，例如支付宝人脸付款、火车站人脸验证等，为人脸识别应用在校园考勤系统提供强力的理论、现实支持。

而借助于现代信息技术，将人脸识别应用在考勤系统中，借助校园中常见的系统摄像头，就可以实现考勤全程无接触无干扰，同时智能化生成考勤记录，可以很大程度上解决当前校园考勤系统普遍存在的弊端，保证考勤的准确性和高效率。

## 参考资料

1. 《可行性分析报告\_基于人脸识别的考勤系统》
2. 《需求规格说明书\_基于人脸识别的考勤系统》
3. 郭琪,王志慧,范道尔吉,武慧娟.基于深度学习的多人脸识别系统的设计实现[J/OL].内蒙古农业大学学报(自然科学版):1-7[2021-04-22]
4. Wang Mei, Deng Weihong Deep face recognition: A survey[J] Neurocomputing, 2021, 429
5. Liu Yanfei, Chen Junhua Unsupervised face Frontalization for pose-invariant face recognition[J] Image and Vision Computing, 2021, 106
6. 李欣,黄镕,陈哲辉,龙宇翔,胥亮.一种改进的MTCNN和FaceNet人脸检测及识别算法研究[J].广东石油化工学院学报,2021,31(01):44-47+43.
7. 邓健.基于迁移学习和MTCNN的人脸识别方法[J].科技风,2020(34):77-78.
8. 庞殊杨,芦莎.基于改进MTCNN网络的多尺度安全帽识别[J/OL].计算机应用研究:1-8[2021-04-22].
9. 徐竟泽. 基于MTCNN与SVM算法的人脸检测与识别[D].山东科技大学,2020.
10. 李林峰,李春青,田博源,廖晓霞.基于MTCNN的FaceNet架构的人脸识别考勤系统设计与实现[J].电脑知识与技术,2020,16(27):181-183.
11. 李慧颖,李薇,邢艳芳,何光威.基于TensorFlow的人脸识别系统设计[J].计算机时代,2020(09):61-63.
12. 贾小硕,曾上游,潘兵,周悦.基于改进MTCNN网络的目标人脸快速检测[J].计算机工程与科学,2020,42(07):1262-1266.
13. 李天昊. 基于OpenCV的实时人脸识别系统的设计与实现[D].东北石油大学,2020.
14. 何玲.基于人脸识别的高校考勤系统研究[J].中小企业管理与科技(中旬刊),2020(12):86-87.
15. R. He, J. Cao, L. Song, Z. Sun and T. Tan, "Adversarial Cross-Spectral Face Completion for NIR-VIS Face Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 42, no. 4, pp. 1024-1037, 1 May 2020, doi: 10.1109/TPAMI.2019.2961900.
16. C. Ding and D. Tao, "Trunk-Branch Ensemble Convolutional Neural Networks for Video-Based Face Recognition," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 4, pp. 1002-1014, 1 April 2018, doi: 10.1109/TPAMI.2017.2700390.
17. X. Bai, F. Jiang, T. Shi and Y. Wu, "Design of Attendance System Based on Face Recognition and Android Platform," 2020 International Conference on Computer Network, Electronic and Automation (ICCNEA), 2020, pp. 117-121, doi: 10.1109/ICCNEA40244.2020.00033.
18. P. Patil and S. Shinde, "Comparative analysis of facial recognition models using video for real time attendance monitoring system," 2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2020, pp. 840-844, doi: 10.1109/ICECA49313.2020.9297374.

M. Othman, S. N. Ismail and H. Noradzan, "An adaptation of the web-based system architecture in the development of the online attendance system," 2012 IEEE Conference on Open Systems, 2012, pp. 1-6, doi: 10.1109/ICOS.2012.6417619.

## 术语定义及说明

* MTCNN：Multi-task convolutional neural networks，多任务级联卷积神经网络。
* FaceNet：是谷歌提出的一种新的人脸识别的方法，该方法在LFW数据集上的准确度已经达到了99.6%，目前是该数据集上检测的最好记录。
* Golang： Google 的 Robert Griesemer，Rob Pike 及 Ken Thompson 开发的一种静态强类型、编译型语言。Go 语言语法与 C 相近，但功能上有：内存安全，GC（垃圾回收），结构形态及 CSP-style 并发计算。
* PyQt4：将Python编程语言和Qt开发框架完美融合，极为高效地促进了开发人员python桌面图形界面的开发。
* opencv-python：将OpenCV和python语言结合，便于我们实现计算机视觉中对图像的各种处理。
* Keras：一个基于python的深度学习框架，它封装了很多深度学习的有关函数，简化了我们构建神经网络的过程。
* Nginx：一个高性能的HTTP和反向代理web服务器。
* CDN：Content Delivery Network，内容分发网络。
* Kafka：一种高吞吐量的分布式发布订阅消息系统，它可以处理消费者在网站中的所有动作流数据。
* Redis：Remote Dictionary Server，即远程字典服务。
* Zipkin：一款开源的分布式实时数据追踪系统。

ELK：ELK是三个开源软件的缩写，分别表示：Elasticsearch , Logstash, Kibana。Logstash 负责日志收集，Elasticsearch 负责存储日志，并对日志进行处理，便于之后分析与查询，Kibana负责展示日志并进行查询显示。

# 设计概述

## 任务和目标

* 能够实现人脸检测和人脸识别，完成基础的打卡操作。
* 完成数据库的设计。
* 完成接口的设计。
* 完成桌面QT界面的设计。
* 完成网页信息管理界面的设计。
* 搭建后台服务器。
* 完成接口测试。

## 需求概述

为了提高老师的上课效率，减少用于签到的不必要时间开销，我们使用基于人脸识别的考勤系统识别在某节课时间进入每间教室的学生，将识别的学生信息发送到服务器，服务器完成打卡并记录下考勤情况。要求人脸识别系统能准确识别在某时间内进入教室的学生，并将识别的学生信息，时间信息，课程信息发送到服务器，要求服务器能准确接收并处理这些信息，并记录下考勤情况。

## 运行环境概述

系统分为三个部分，运行在本地人脸识别工具、可以浏览器访问的信息管理网页以及后台服务器。本地人脸识别工具运行在win7或者win10的PC机上；信息管理网页适配主流的浏览器，例如Google Chrome、Microsoft Edge、火狐等；我们的服务器部署在阿里云的linux(centos7)上。

## 条件与限制

* 性能要求：服务端的响应时间在 30 ms 内；网络响应时间应在 4ms 内；客户端响应时间应在 14ms 内；用户并发数为 400；吞吐量能达到 2000 请求；桌面应用程序初始化的时间控制在4秒，其中MTCNN模型搭建、模型权值载入的时间控制在3秒，打开摄像头并检查连接的时间控制在1秒，更新脸部缓存特征数据的时间控制在1秒。
* 工具要求：pycharm、Vscode、Goland、MySql、Navicat
* 环境：windows 10
* 时间：在6周以内完成项目的开发与测试

## 详细设计方法和工具

流程图，E-R图，部署图。

# 系统详细需求分析

主要对系统级的需求进行分析。首先应对需求分析提出的企业需求进一步确认，并对由于情况变化而带来的需求变化进行较为详细的分析。

## 详细需求分析

### **教师客户端功能模块**

主要包括学生管理、教师管理、课程管理、教室管理、考勤记录管理几个方面的功能。学生管理功能包括学生信息查询，学生考勤记录更改，学生信息修改（课程变动等）；教师管理功能包括教师信息查询、教师信息修改、添加课程；教室管理功能包括教室信息查询；课程管理功能包括课程信息查询、课程信息修改；考勤记录管理包括考勤率统计，考勤人数统计，考勤日志等。

### 人脸识别模块

人脸识别模块是本项目的核心模块。人脸检测是对人脸进行识别和处理的第一步，主要用于检测并定位图片中的人脸，返回高精度的人脸框坐标及人脸特征点坐标。人脸识别会进一步提取每个人脸中所蕴涵的身份特征，并将其与已知的人脸进行对比，从而识别每个人脸的身份。

早期传统的人脸识别算法主要依赖人脸特征，例如，基于LDA的Fisherf ace是把人脸特征向量从高维降到低维。但是传统的识别算法，对背景要求高，处理更复杂的数据时，识别效果和速度都会大幅度下降。在人脸识别任务中，如果采用深度学习的研究方法，可以提取层次更加丰富的特征，适用于多人脸识别的研究。

本项目我们准备使用深度学习模型 MTCNN 进行人脸检测和深度学习模型 FaceNet进行人脸识别。

### 详细性能需求分析

* 桌面应用程序初始化的时间控制在8秒，其中MTCNN模型搭建、模型权值载入的时间控制在4秒，打开摄像头并检查连接的时间控制在2秒，更新脸部缓存特征数据的时间控制在1秒。
* 接口响应的时间控制在3秒，其中网络传输200毫秒，sql查询控制在3秒。
* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 详细系统运行环境及限制条件分析

1. 系统运行环境：

系统分为三个部分，运行在本地人脸识别工具、可以浏览器访问的信息管理网页以及后台服务器。本地人脸识别工具运行在win7或者win10的PC机上；信息管理网页适配主流的浏览器，例如Google Chrome、Microsoft Edge、火狐等；我们的服务器部署在阿里云的linux(centos7)上。

2. 限制条件：

* 在电脑端使用时，必须按照指定的文件夹排列模式，各个文件在运行时才能够顺利找到（因为使用了相对路径和文件夹分类）。
* 需要的python3库： os、sys、datatime、ffmepg、xlrd、matplotlib、keras、numpy、pandas、sklearn、yaml、opencv-python、PyQt4、pymysql。
* 需要电脑麦克风、声卡、驱动。

3. 客户端模块：

接口的响应时间和网络的带宽有关。

## 详细接口需求分析

### 外部接口

|  |  |
| --- | --- |
| 接口 | 说明 |
| 网络连接服务接口 | 开启软件的时候，调用网络服务，使得软件可以联网操作 |
| HTTP通讯协议 | 使用此协议来完成各种信息的通讯与交互 |
| 图像处理接口 | 图像人脸识别库接口的使用 |
| 信息系统的身份查询接口 | 查询学生的身份ID的时候，返回该ID是否存在 |
| 系统数据库接口 | 人脸识别查勤数据通过该接口与数据库进行信息的通讯和交互 |
| 校园信息数据库接口 | 学校信息数据库通过该接口提供学生、课程等详细信息源给系统数据库 |
| 考勤数据查询接口 | 在web管理前端输入查询关键词，数据库通过接口返回相关联的查寝数据 |

### 内部接口

系统各个模块之间采用松耦合紧内聚的连接方式，将系统中的公用数据类分离出来，单独当作一个接口来使用，这就提高了系统的容错率，将服务器和客户端的通信也抽离出来，这样使得程序功能模块与通信模块相分离，降低了系统代码的复杂性。服务器端的数据库使用MySql来创建，人脸识别功能的实现调用当前流行的人脸识别的接口在客户端来实现。

## 教师模块

### 查询学生用户信息

1. 需求编号：001
2. 功能说明：教师可以查看学生用户的信息。
3. 前置条件：教师用户已经登录了。
4. 输入：教师输入学生的姓名或者学号。
5. 处理：将用户的姓名或者学号和数据库中的数据做对比，将配对的上的学生数据展示出来；如果没有查询到目标用户，就显示“没用这个学生的数据”。
6. 输出：展示目标学生的数据。

### 删除目标学生用户

1. 需求编号：002
2. 功能说明：教师可以删除目标学生用户。
3. 前置条件：教师用户已经登录了；进入删除用户界面。
4. 输入：需要删除的学生用户的用户名或者学号。
5. 处理：将用户的姓名或者学号和数据库中的数据做对比，将配对的上的学生数据展示出来，并再一次询问是否删除；如果没有查询到目标用户，就显示“没用这个学生的数据”。
6. 输出：无。

### 修改学生用户信息

1. 需求编号：003
2. 功能说明：教师可以修改目标学生用户的信息。
3. 前置条件：教师用户已经登录了；进入修改学生用户信息的界面。
4. 输入：需要修改信息的学生用户的用户名或者学号。
5. 处理：将用户的姓名或者学号和数据库中的数据做对比，将配对的上的学生数据展示出来，选择目标信息字段直接进行修改；如果没有查询到目标用户，就显示“没有这个学生的数据”。
6. 输出：无。

### 查询学生用户考勤日志

1. 需求编号：004
2. 功能说明：教师可以查看学生用户的考勤记录。
3. 前置条件：已经登录了教师用户，进入查看用户考勤记录的界面。
4. 输入：无
5. 处理：从数据库中调取学生的考勤记录。
6. 输出：学生用户的考勤记录。

## 学生模块

### 查询个人信息

1. 需求编号：005
2. 功能说明：学生用户可以查看自己的个人信息。
3. 前置条件：学生用户已经登录了。
4. 输入：无
5. 处理：从数据库中调取学生用户的个人信息，展示到网页上。
6. 输出：学生的个人信息。

### 修改个人信息

1. 需求编号：006
2. 功能说明：学生用户可以修改自己的个人信息。
3. 前置条件：学生用户已经登录，进入修改用户信息的个人界面。
4. 输入：需要修改的信息。
5. 处理：点击修改修改的信息字段，直接修改。
6. 输出：无。

### 查询个人考勤日志

1. 需求编号：007
2. 功能说明：学生用户可以查看自己的考勤记录。
3. 前置条件：学生用户已经登录。
4. 输入：无
5. 处理：进入考勤界面，从数据库显示自己的考勤结果。
6. 输出：无。

### 删除账户

1. 需求编号：008
2. 功能说明：学生用户可以删除自己的账户。
3. 前置条件：学生用户已经登录。
4. 输入：无。
5. 处理：点击删除账户的按钮，并再次确定，就可以删除当前账户了。
6. 输出：无。

## 人脸识别模块

### 人脸图像采集

1. 需求编号：009
2. 功能说明：我们使用的设备是一个摄像头，需要连续拍摄多张用户的照片，并在照片中找到用户的头像，将用户的头像照片裁剪出来。
3. 前置条件：进入考勤界面，摄像头已经打开。
4. 输入：摄像头采集的照片。
5. 处理：在照片中定位出用户的头像位置。
6. 输出：用户头像。并将用户的头像保存到本地。

### 人脸特征提取

1. 需求编号：010
2. 功能说明：利用人脸特征提取的方法提取目标头像的特征，将这些特征数值化。
3. 前置条件：已经采集了用户的头像照片。
4. 输入：无。
5. 处理：提取头像照片的特征。
6. 输出：数值化的特征。

### 人脸特征数据比对

1. 需求编号：011
2. 功能说明：利用已经提取到的用户的脸部特征，将这些数值的特征和数据库中的人脸特征集做比对，找出匹配度最高的用户，且匹配度超过80%，我们就可以认为当前进行人脸识别的用户就是他。
3. 前置条件：已经获取了数值化的脸部特征。
4. 输入：数值化的脸部特征。
5. 处理：将数值化的脸部特征和数据库的数据做比对，找出对应的用户；如果没有识别出来，就显示“识别失败”。
6. 输出：对应的用户。

## 考勤模块

### 考勤日志

1. 需求编号：012
2. 功能说明：用户面对摄像头，识别出了当前用户或者没有识别出来，将这一次识别结果保存到数据库和本地。
3. 前置条件：经过一次识别。
4. 输入：无。
5. 处理：进行了一次数据库保存操作和本地保存操作。
6. 输出：无。

### 出勤人数统计

1. 需求编号：013
2. 功能说明：以时间为单位，访问数据库，查询当前时间段的出勤人数。
3. 前置条件：数据库连接成功。教师账户已经登录，学生用户没用查看记录的权限。
4. 输入：无
5. 处理：进入出勤日志的界面。
6. 输出：考勤情况。

### 出勤率统计

1. 需求编号：014
2. 功能说明：这个是在出勤人数的基础上计算出来的，教师可以设定当前签到的应到人数，实到人数除以应到人数就是出勤率。
3. 前置条件：数据库连接成功。教师账户已经登录，学生用户没用查看记录的权限。
4. 输入：无。
5. 处理：实到人数除以应到人数。
6. 输出：出勤率。

# 系统详细设计

## 总体架构设计

### 功能模块设计

本系统主要提供人脸识别、学生管理、教师管理、课程管理、教室管理、考勤记录管理六个方面的功能。人脸识别功能包括人脸图像采集、多人脸检测、人脸对齐；学生管理功能包括学生信息查询，学生考勤记录更改，学生信息修改（课程变动等）；教师管理功能包括教师信息查询、教师信息修改、添加课程；教室管理功能包括教室信息查询；课程管理功能包括课程信息查询、课程信息修改；考勤记录管理包括考勤率统计，考勤人数统计，考勤日志等。

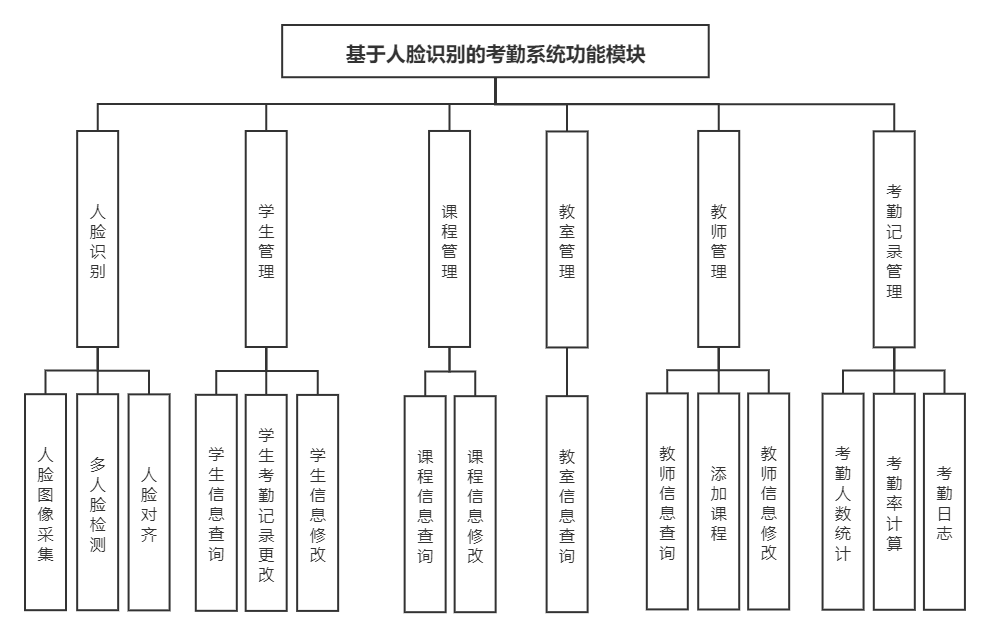


图 1 功能模块图

### 系统部署图

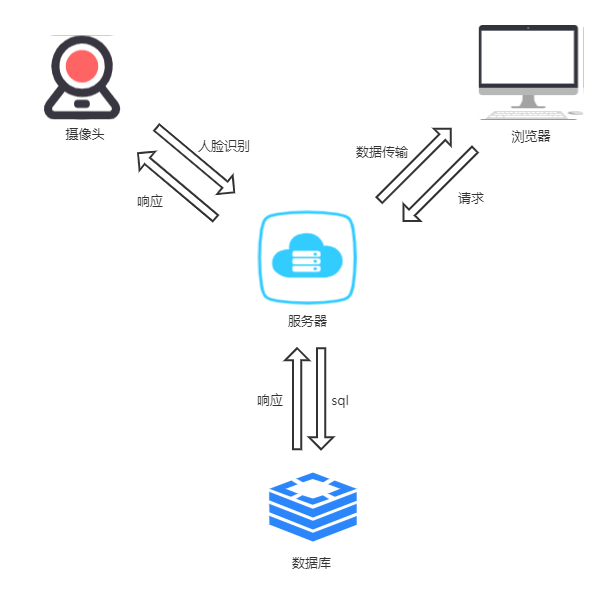


图 2 精简架构图

功能强大的编程语言Python含有众多功能实用的第三方库。比如PyQt库将Python编程语言和Qt开发框架完美融合，极为高效地促进了开发人员python桌面图形界面的开发；opencv-python库将OpenCV和python语言结合，便于我们实现计算机视觉中对图像的各种处理；Keras是一个基于python的深度学习框架，它封装了很多深度学习的有关函数，简化了我们构建神经网络的过程。所以，在人脸识别模块，我们准备使用PyQt5设计出界面良好的桌面应用程序，使用opencv-python进行图像采集和图像处理，使用Keras框架搭建MTCNN和FaceNet网络模型完成人脸检测、人脸识别的功能。

在Web开发中，我们主要开发一个用于信息管理系统，用于记录学生的签到信息并供学校、老师查询及修改。我们后端采用的语言为 golang ，它天然支持高并发，能较好的实现单服务器承担多个教室、多端的同时请求带来的压力，而 web 框架为 gin，因为其路由匹配采用压缩前缀树实现、并且提供了中间件、异常等的处理，简介而高效，ORM 则使用 gorm，它是当前使用较多的一个 golang 的 ORM 框架，性能较高，也无什么较大的 bug，后端的技术栈就是这些。在前端网页设计方面，基础技术为前端三件套，即 HTML、CSS、JavaScript，采用的框架为 Vue，UI 组件库为 Element UI 与 Vuetify，路由则为 Vue-Router，请求套件则为 Axios，前端网页主要负责记录信息的查询、老师与学院对信息的管理，前端网页端的技术栈就是这些。

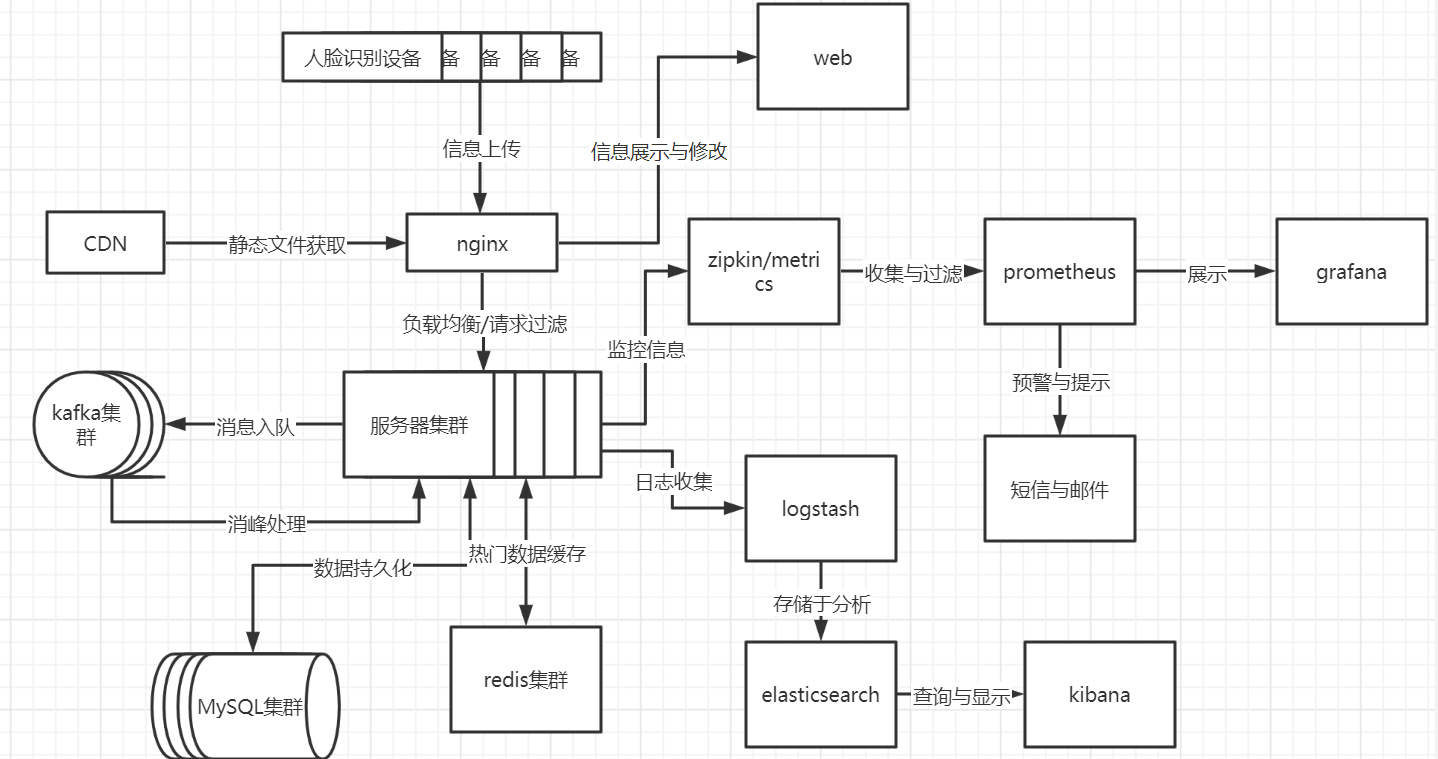


图 3 详细架构图

整个川大大概有一千多个教室，总量不是很大，再加上我们的功能比较明确单一，无需太多拆分，所以我们总体采用 MVC 架构来构建这个项目，而非 RPC 或 SOA，细节如下：

* 人脸识别设备：负责基于人脸识别学生信息，并将匹配的学生其信息上传至服务器，人脸识别设备会与教室相绑定。
* Web：用于展示学生的上课信息、人脸信息等，同时也能对这些信息进行修改。
* Nginx：负责进行负载均衡，并解决跨域问题，此外还能进行一定的限流。
* CDN：静态文件存储的位置，降低人脸数据加载对带宽的压力。
* 服务器集群：负责记录数据、处理请求。
* Kafka集群：负责消峰，防止短时间大量请求送达服务器而使得服务器无法及时处理、请求堆积，服务器资源耗尽而崩溃。
* MySQL集群：负责存储数据，如学生上课相关数据、课程与教室相关映射数据。
* Redis集群：用于缓存热门而不经常修改的数据，如今日教室与课程的映射数据，减少对数据库的压力。
* Zipkin：负责进行链路监控。
* Metrics：负责节点信息监控。
* Prometheus：负责收集信息，并通过短信进行预警，通过 grafana 进行图形化显示。
* ELK：logstash 负责日志收集，elasticsearch 负责存储日志，并对日志进行处理，便于之后分析与查询，kibana 负责展示日志并进行查询显示。

## 人脸识别模块

### 模块描述

人脸识别模块是本项目的核心模块。人脸检测是对人脸进行识别和处理的第一步，主要用于检测并定位图片中的人脸，返回高精度的人脸框坐标及人脸特征点坐标。人脸识别会进一步提取每个人脸中所蕴涵的身份特征，并将其与已知的人脸进行对比，从而识别每个人脸的身份。

早期传统的人脸识别算法主要依赖人脸特征，例如，基于LDA的Fisherf ace是把人脸特征向量从高维降到低维。但是传统的识别算法，对背景要求高，处理更复杂的数据时，识别效果和速度都会大幅度下降。在人脸识别任务中，如果采用深度学习的研究方法，可以提取层次更加丰富的特征，适用于多人脸识别的研究。

本项目我们准备使用深度学习模型 MTCNN 进行人脸检测和深度学习模型 FaceNet进行人脸识别。

### 处理流程

第一步，在电脑桌面打开我们使用PyQt5开发的桌面应用，初始化应用程序，搭建MTCNN模型，读入模型训练权值，最后检查电脑和摄像头是否成功连接。

第二步，让目标人物站到摄像头面前，摄像头自动拍照，并将拍摄的图片预处理（调节颜色通道、修改大小、镜像翻转），最后将图片传入MTCNN模型。

第三步，图片经过MTCNN三层网络结构处理后，截取出人脸框图和五点标记。

第四步，将人脸框图传入FaceNet模型，输出128个人脸特征点，将128个特征值和本地数据库中的人脸数据进行特征比对，计算欧式距离，认定欧式距离最小且置信度大于0.7的目标人物就算是所识别的人物。

第五步，将识别结果通过接口上传到服务器，服务器获取结果，查询当前课堂是否存在这个学生，如果存在，打卡成功，将打卡记录保存到远程数据库；当前课堂不存在目标用户则打卡失败。

第六步，摄像头获取服务器的响应，获取打卡结果，打卡流程结束。

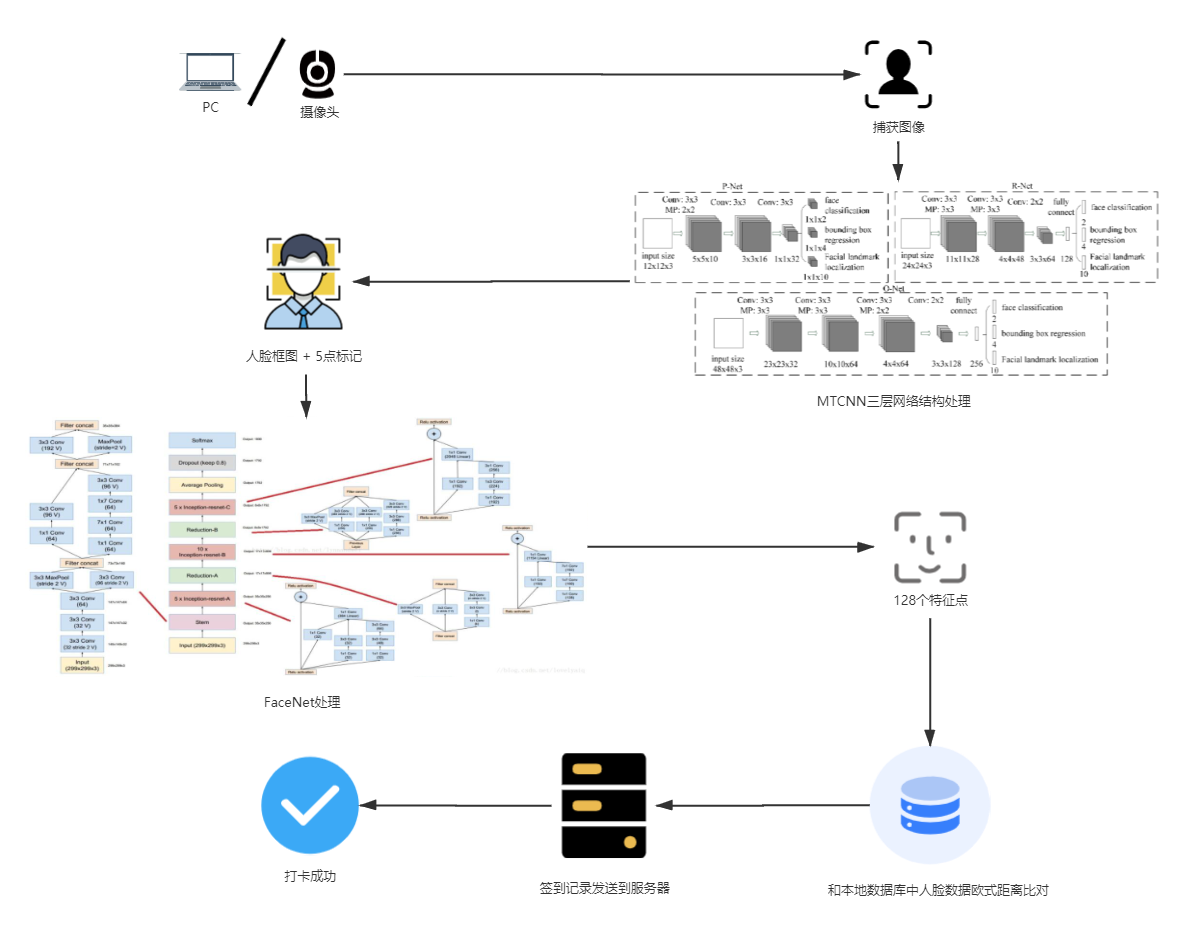


图 4 人脸识别流程图

### MTCNN网络

MTCNN的“MT”是指多任务学习(Multi-Task)，CNN（Convolutional Neural Networks, CNN）是指卷积神经网络。MTCNN多任务级联卷积神经网络，基于级联的特定目标检测器，它可以在一张图片中检测出多张人脸。

MTCNN的网络结构是由P-Net、R-Net、O-Net 3个传统的卷积子网络构成。P-Net模型训练过程中输入12x12的图像，输出置信度和人脸偏移量。R-Net模型输入图像的大小为24x24，对P-Net的输出图像和偏移值作进一步处理。O-Net模型输入的图像的大为48x48，对R-Net输出的图像进行最终的判别，最终确定人脸的位置。

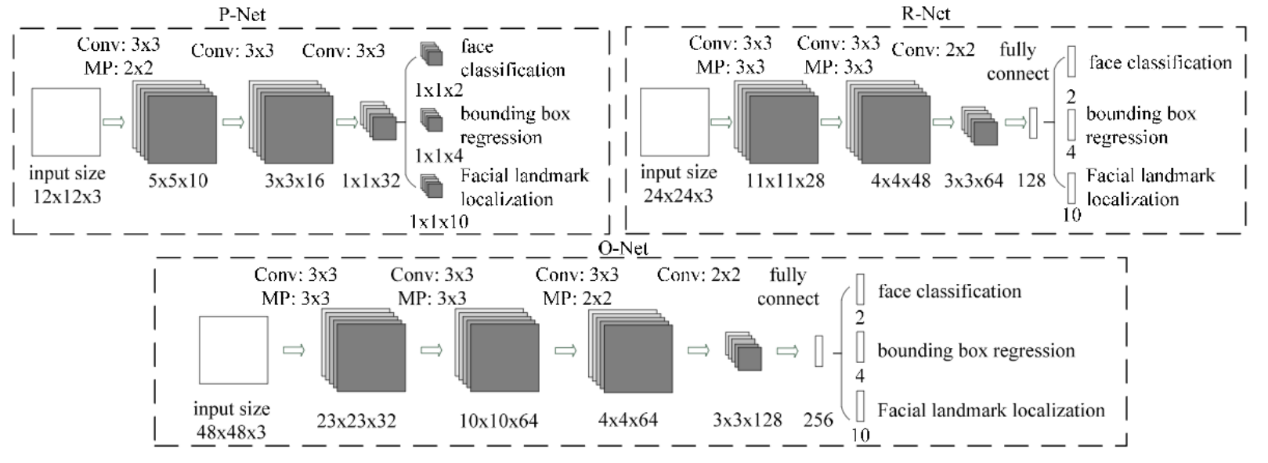


图 5 MTCNN模型结构

在 MTCNN算法3个网络结构中，P -Net网络结构主要是为了获取人脸区域的候选窗口和边界框的回归向量，利用该边界框进行回归，并对候选窗口进行校准，然后通过非极大值抑制(NMS)来合并高度重叠的候选框。R-Net 网络结构是利用非极大值抑制(NMS)和边界框回归来去除一些false - positive区域。其网络结构与Р-Net 网络相比，差异之处是多了个全连接层，所以可以更好地抑制false - positive。O-Net网络结构比R-Net多了一层卷基层，因此在处理的结果上与R–Net结构相比会精细很多，其作用和R-Net相同，但是O-Net网络结构会对人脸区域进行更多监督，同时还会输出5个地标(landmark)，这五个坐标是双眼、嘴角和鼻尖。

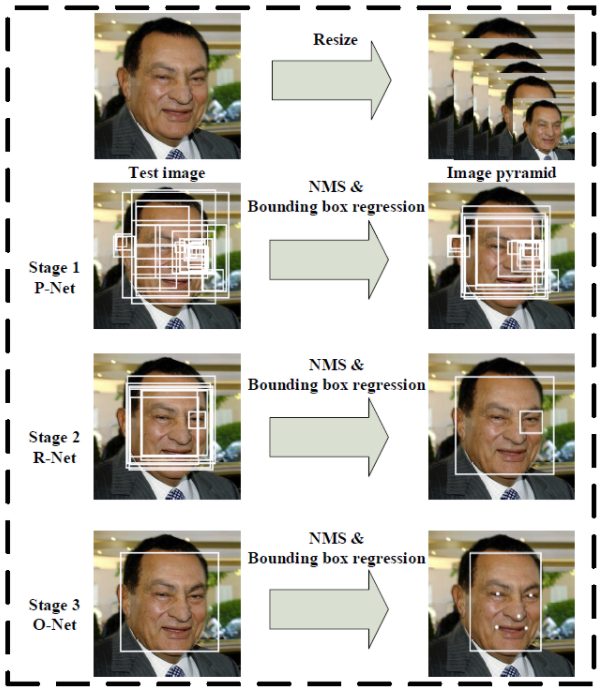


图 6 MTCNN人脸检测的流程图

### FaceNet网络

FaceNet是谷歌人脸检测算法，发表于 CVPR 2015，利用相同人脸在不同角度等姿态的照片下有高内聚性，不同人脸有低耦合性，提出使用 cnn + triplet mining 方法，在800万人，2亿多张样本集训练后，FaceNet 在 LFW 数据集上测试的准确率达到了99.63%，在YouTube Faces DB数据集上，准确率为95.12%。

FaceNet 的主要思想是把人脸图像映射到一个多维空间，通过空间距离表示人脸的相似度。同个人脸图像的空间距离比较小，不同人脸图像的空间距离比较大。这样通过人脸图像的空间映射就可以实现人脸识别，FaceNet 中采用基于深度神经网络的图像映射方法和基于 triplets（三联子）的 loss 函数训练神经网络，网络直接输出为 128 维度的向量空间。

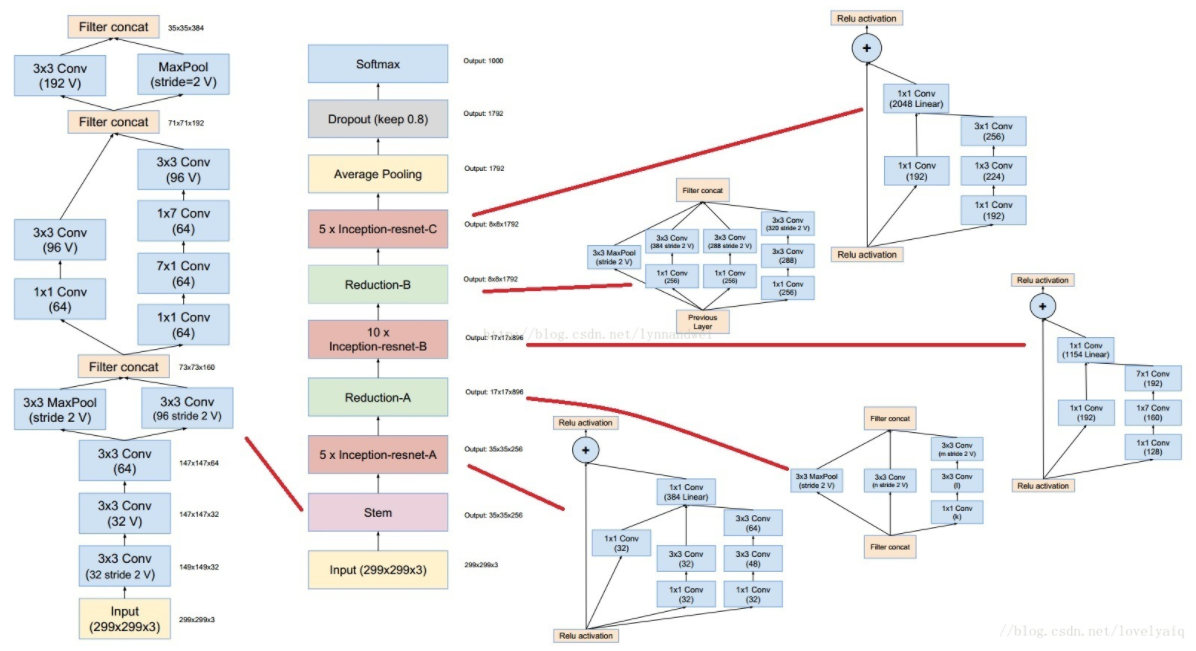


图 7 Inception-Resnet v2结构图

### 性能

* 桌面应用程序初始化的时间控制在8秒，其中MTCNN模型搭建、模型权值载入的时间控制在5秒，打开摄像头并检查连接的时间控制在2秒，更新脸部缓存特征数据的时间控制在1秒。
* 接口响应的时间控制在3秒，其中网络传输200毫秒，sql查询控制在3秒。

### 输入项

使用摄像头进行人脸捕获，输入的是一张图像。

### 输出项

如果人脸识别成功，显示人物姓名；如果人脸识别失败，显示Unknow。

### 限制条件

* 在电脑端使用时，必须按照指定的文件夹排列模式，各个文件在运行时才能够顺利找到（因为使用了相对路径和文件夹分类）。
* 需要的python3库： os、sys、datatime、ffmepg、xlrd、matplotlib、keras、numpy、pandas、sklearn、yaml、opencv-python、PyQt5、pymysql。
* 需要电脑麦克风、声卡、驱动。

## 学生管理模块

### 模块描述

学生管理模块，这里主要是对学生信息的管理，包括学生信息查询，学生考勤记录修改，学生信息修改等。学生信息查询，就是将目标学生的基本信息（年龄、专业、班级、学号等）都展示出来，同时显示这个学生在最近一段时间内的打卡记录。学生考勤记录修改，就是人工插入一条当前学生的打卡记录，打卡显示“病假”。学生信息修改，就是修改学生的基本信息。

### 处理流程

前提，已经进入了考勤信息管理界面。

第一步，点击学生管理模块，跳转到学生管理界面。

第二步，输入目标学生的学号，点击查询，等待服务器的响应。

第三步，就可以查看当前学生的基本信息了。

第四步，可以直接修改当前学生的基本信息。

第五步，可以给当前学生插入一条考勤记录。

### 性能

* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 输入项

目标查询学生的id。

### 输出项

浏览器获取服务器的响应，展示当前学生的信息。

### 限制条件

接口的响应时间和网络的带宽有关。

## 教师管理模块

### 模块描述

教师管理模块，这里主要是对教师信息的管理，包括教师信息查询、教师信息修改、添加课程等。教师信息查询，就是将目标教师的基本信息（年龄、学院、id等）都展示出来，同时显示当前教师教授的课程。教师信息修改，就是修改教师的基本信息。添加课程，添加一条课程记录，包括上课时间、下课时间、上课教室等。

### 处理流程

前提，已经进入了考勤信息管理界面。

第一步，点击教师管理模块，跳转到教师管理界面。

第二步，输入目标教师的id，点击查询，等待服务器的响应。

第三步，就可以查看当前教师的基本信息了。

第四步，可以直接修改当前教师的基本信息以及当前教师教授的课程。

第五步，可以直接添加一个课程纪律，包括上课时间、下课时间、上课教室。

### 性能

* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 输入项

目标查询教师的id。

### 输出项

浏览器获取服务器的响应，展示当前教师的信息。

### 限制条件

接口的响应时间和网络的带宽有关。

## 教室管理模块

### 模块描述

教室管理功能包括教室信息查询。

### 处理流程

前提，已经进入了考勤信息管理界面。

第一步，点击教室管理模块，跳转到教室管理界面，等待服务器的响应。

第二步，界面会显示四川大学所有的教室信息，包括教室id、教室描述信息。

第三步，点击目标教室，会显示当前教室所有的上课课程和上课时段。

### 性能

* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 输入项

无。

### 输出项

浏览器获取服务器的响应，展示教室的信息。

### 限制条件

接口的响应时间和网络的带宽有关。

## 课程管理模块

### 模块描述

课程管理功能包括课程信息查询、课程信息修改。

### 处理流程

前提，已经进入了考勤信息管理界面。

第一步，点击课程管理模块，跳转到课程管理界面。

第二步，输入目标课程的id或者名称，点击查询，等待服务器的响应。

第三步，就可以查看当前课程的基本信息了，包括上课教室、科任老师、上课时间、下课时间等。

第四步，可以直接修改当前课程的信息（只要sql不冲突）。

### 性能

* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 输入项

目标查询课程的id。

### 输出项

浏览器获取服务器的响应，展示当前课程的信息。

### 限制条件

接口的响应时间和网络的带宽有关。

## 考勤记录模块

### 模块描述

考勤记录管理包括考勤率统计，考勤人数统计，考勤日志等。

### 处理流程

前提，已经进入了考勤信息管理界面。

第一步，点击考勤管理模块，等到服务器的响应，跳转到考勤管理界面。

第二步，界面上会显示最近的考勤记录，例如：张三在10:00的时候在文科楼打卡了。

第三步，点击考勤率统计按钮，显示当前课程的考勤率，如果当前没有课程在上课，则给出提示“没有课程在进行”。

第四步，点击考勤人数统计的按钮，显示当前课程的已经打卡的学生名字；如果当前没有课程在上课，则给出提示“没有课程在进行”。

### 性能

* 每条sql的查询时间在1秒内。
* 网络接口的响应时间在200毫秒内。

### 输入项

无。

### 输出项

浏览器获取服务器的响应，展示近期考勤记录。

### 限制条件

接口的响应时间和网络的带宽有关。

# 数据结构设计

## 数据库部署结构图

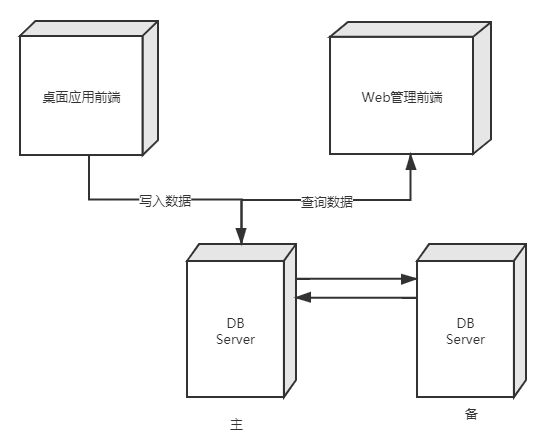


图 8 数据库部署图

## 数据库物理结构设计

### 学生表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | name | major | grade | age |
|  |  |  |  |  |

学生信息：主要用于学生模块信息修改与人脸验证是检测学生身份真实性；

ID：String

name：String

major：String

grade：int

age：int

### 课程表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | name | teacher\_id | class\_room\_id | start\_time | end\_time |
|  |  |  |  |  |  |

ID：String

name：String

teacher\_id：String

class\_room\_id：String

start\_time：date

end\_time：date

### 教师表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | name | age | major | gender |
|  |  |  |  |  |

ID：String

name：String

major：String

gender：int

age：int

### 教室表

|  |  |
| --- | --- |
| ID | descripition |
|  |  |

ID：String

Descripition：String

### 考勤记录表

|  |  |  |  |
| --- | --- | --- | --- |
| ID | student\_id | class\_room\_id | time |
|  |  |  |  |

ID：student\_id

class\_room\_id：String

time：date

### 图片表

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | name | URL | addressX | addressY | functionality |
|  |  |  |  |  |  |

ID：String

name：String

URL：String

addressX：double

addressY：double

functionality：String

## 数据库逻辑结构设计

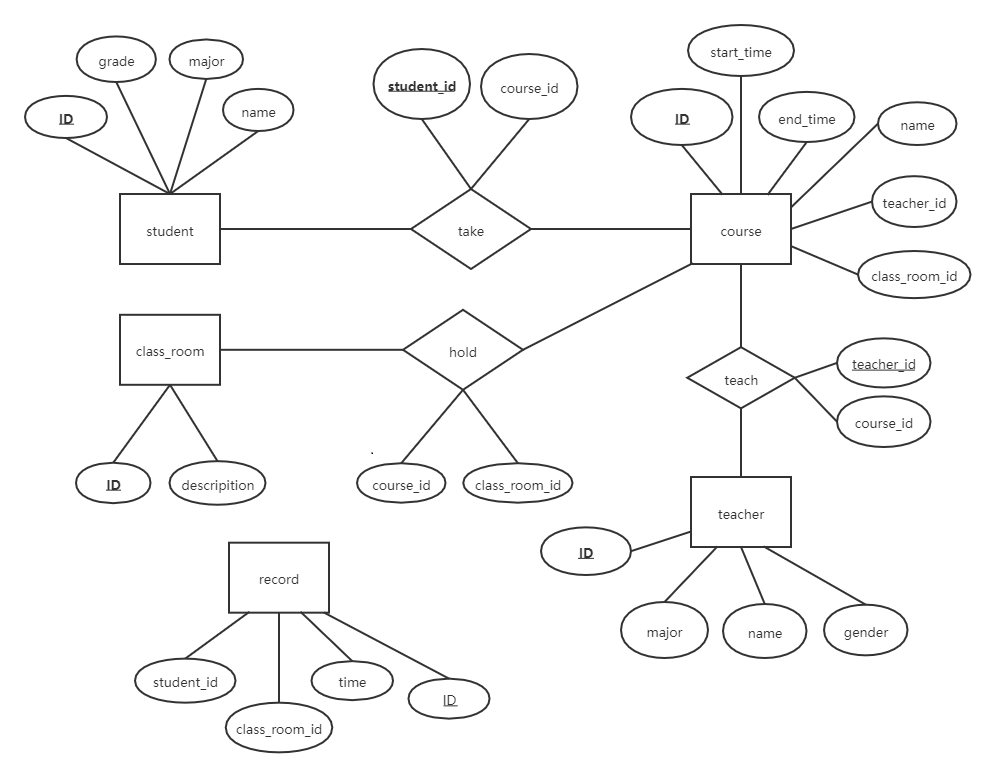


图 9 E-R图

## 数据结构与程序的关系

数据结构为关系型数据，采用标准SQL语句访问数据库；数据库对外暴露通用的数据访问接口。对数据库访问采用DAO设计模式实现，提高维护性于扩展性。数据结构，决定了接口的设计；而接口的设计，决定了前后端的交互。因此，数据结构是程序设计的前提基础，只有设计了详尽的数据库，才能进行下一步的开发。

## 数据模型

### models.Teacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 必选 | 约束 | 说明 |
| gender | boolean | false | none | 性别，false 表示男，true 表示女 |
| grade | integer | false | none | 年级 |
| id | integer | true | none | none |
| major | string | false | none | 专业 |
| name | string | false | none | none |

{  
 "properties": {  
 "gender": {  
 "description": "性别，false 表示男，true 表示女",  
 "type": "boolean"  
 },  
 "grade": {  
 "description": "年级",  
 "type": "integer"  
 },  
 "id": {  
 "type": "integer"  
 },  
 "major": {  
 "description": "专业",  
 "type": "string"  
 },  
 "name": {  
 "type": "string"  
 }  
 },  
 "required": [  
 "id"  
 ],  
 "type": "object"  
}

### models.Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 必选 | 约束 | 说明 |
| age | integer | false | none | none |
| grade | integer | false | none | 年级 |
| id | integer | true | none | none |
| major | string | false | none | 专业 |
| name | string | false | none | none |

{  
 "properties": {  
 "age": {  
 "type": "integer"  
 },  
 "grade": {  
 "description": "年级",  
 "type": "integer"  
 },  
 "id": {  
 "type": "integer"  
 },  
 "major": {  
 "description": "专业",  
 "type": "string"  
 },  
 "name": {  
 "type": "string"  
 }  
 },  
 "required": [  
 "id"  
 ],  
 "type": "object"  
}

### result.Result

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 必选 | 约束 | 说明 |
| code | integer | false | none | 状态码 2xx 表示请求成功，3xx 表示重定向，4xx 表示客户端错误，5xx 表示服务端错误 |
| data | object | false | none | 数据 |
| msg | string | false | none | 状态码描述信息 |

{  
 "properties": {  
 "code": {  
 "description": "状态码\n2xx 表示请求成功，3xx 表示重定向，4xx 表示客户端错误，5xx 表示服务端错误",  
 "type": "integer"  
 },  
 "data": {  
 "description": "数据",  
 "type": "object"  
 },  
 "msg": {  
 "description": "状态码描述信息",  
 "type": "string"  
 }  
 },  
 "type": "object"  
}

# 接口详细设计

## GET 查询课程

GET /course/student/list

基于学生id 查询课程。

老师根据学生的ID查询该学生的课程情况，主要用于辅导员在查看学生的学习情况时，以及作为系统的一个基础接口。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| studentId | query | string | false | 学生id |
| pageNumber | query | string | false | 页号 |
| pageSize | query | string | false | 页大小 |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## GET 查询课程

GET /course/list

基于老师id、教室id 查询课程

主要用于校务处在安排老师课程时，查看该老师已有的课程情况，单独列出该老师的课程情况。以及在安排教室的时候查看该教室已经安排了的课程情况，以及在根据两者查询具体某节课程，作为一个数据查询的基础接口。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| teacherId | query | string | false | 老师id |
| classRoomId | query | string | false | 教室id |
| pageNumber | query | string | false | 页号 |
| pageSize | query | string | false | 页大小 |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## GET 查询记录

GET /record/list

基于学生id、课程id 查询记录

用于查询某特定学生的所有课程的签到情况，以及查询某节课程所有学生的签到情况，以及根据两者查询某特定学生的签到情况。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| studentId | query | string | false | 学生id |
| courseId | query | string | false | 课程id |
| pageNumber | query | string | false | 页号 |
| pageSize | query | string | false | 页大小 |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## POST 创建记录

POST /record

基于学生id、教室id 插入记录

主要用于学生向老师请假后，老师手动的加入该学生的签到情况。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| studentId | query | string | false | 学生id |
| classRoomId | query | string | false | 教室id |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | [result.Result](#schemaresult.result) |

## GET 获取学生列表

GET /student/list

分页获取学生列表

用于系统内部获取学生列表，作为一个内部的基础功能。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| pageNumber | query | string | false | 页号 |
| pageSize | query | string | false | 页大小 |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## GET 获取学生

GET /student/{id}

基于学生 id 获取学生信息

用于查询某个学生的详细信息。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| id | path | string | true | 学生id |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## PUT 更新学生信息

PUT /student

基于学生 ID 更新学生信息

用于每年在有新学生进学校时添加学生信息，或者有学生转专业时更新学生信息。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| body | body | [models.Student](#schemamodels.student) | false | none |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | [result.Result](#schemaresult.result) |

## GET 获取老师列表

GET /teacher/list

分页获取老师列表

用于系统内部获取老师列表，作为系统内部的一个基础功能接口。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| pageNumber | query | string | false | 页号 |
| pageSize | query | string | false | 页大小 |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |

## PUT 更新老师信息

PUT /teacher

基于老师 ID 更新老师信息

用于有新老师加入或者老师的专业有所改变时更新老师的信息。

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| body | body | [models.Teacher](#schemamodels.teacher) | false | none |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | [result.Result](#schemaresult.result) |

## GET 获取老师

GET /teacher/{id}

基于老师 id 获取老师信息

用于获取老师的详细信息

### 请求参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 位置 | 类型 | 必选 | 说明 |
| id | path | string | true | 老师id |

### 返回结果

|  |  |  |  |
| --- | --- | --- | --- |
| 状态码 | 状态码含义 | 说明 | 数据模型 |
| 200 | [OK](https://tools.ietf.org/html/rfc7231#section-6.3.1) | OK | Inline |