《大江打卡考勤系统》

详细设计

Version 2.0

版本历史

版本/状态	作者	参与者	日期	备注
1.0.0	廖启凡		2018-08-24	创建
1.0.1	张诚天		2018-08-27	修订
1.0.2	李煜炜		2018-08-30	修订
1.0.3	江胤霖		2018-09-02	修订
2.0.0	廖启凡		2018-09-03	修订

目录

第一章引言

1.1 编写目的

编写本设计的目的是为了准确的阐述大江打卡考勤系统的具体实现思路和方法，即系统的详细架构和实现逻辑，主要包括程序系统的结构以及各层次中每个程序的设计考虑，实现方法。

1.2 项目背景

随着各种生物特征识别技术的普及和计算机算力的不断升级，企业对生物特征自动考勤的需求也与日俱增，其中对指纹识别、人脸识别的需求最为突出。为满足企业快捷、安全的考勤需求，针对企业上班打卡的实际需要，小组自主研发了大江打卡考勤系统，通过统一的 Web 界面实现人脸识别考勤，实现了在线人脸识别、自动打卡、考勤管理等方面的功能。

1.3 定义

- GUI

GUI 是 Graphic User Interface 的缩写，即图形用户界面，一种可视化的用户界面，它使用图形界面代替正文界面。

1.4 参考资料

a．属于本项目的其他已发表的文件；

《需求说明》

《概要设计》

本文件中各处引用到的文件资料，包括所要用到的软件开发标准。

《python 开发手册》

第二章总体设计

2.1、逻辑架构设计

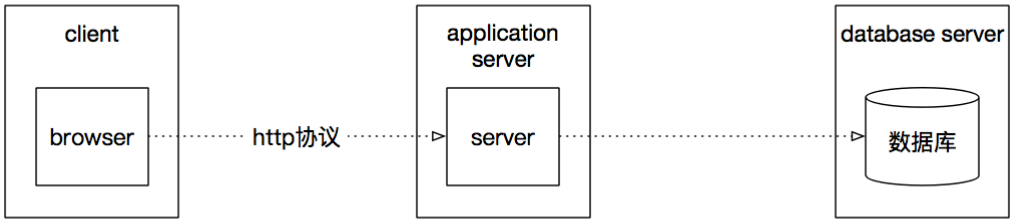
2.1.1、B / S 架构

B / S 架构即浏览器和服务器架构模式。用户工作界面是通过 WWW 浏览器来实现，极少部分事务逻辑在前端(Browser)实现，但是主要事务逻辑在服务器端(Server)实现，形成所谓三层 3-tier 结构。WEB 浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户端只需要安装一个浏览器，即可通过 web server 与服务器进行数据的交互。大大简化了客户端电脑载荷，减轻了系统维护与升级的成本和工作量。

2.1.2、web 界面应用

客户端在网络页面上，直接连接到网络，便于个人用户以及机构用户快速上传数据，方便快捷处理查询数据。支持所有操作系统，适用于大多数 PC 端环境，提供了更好的用户体验。

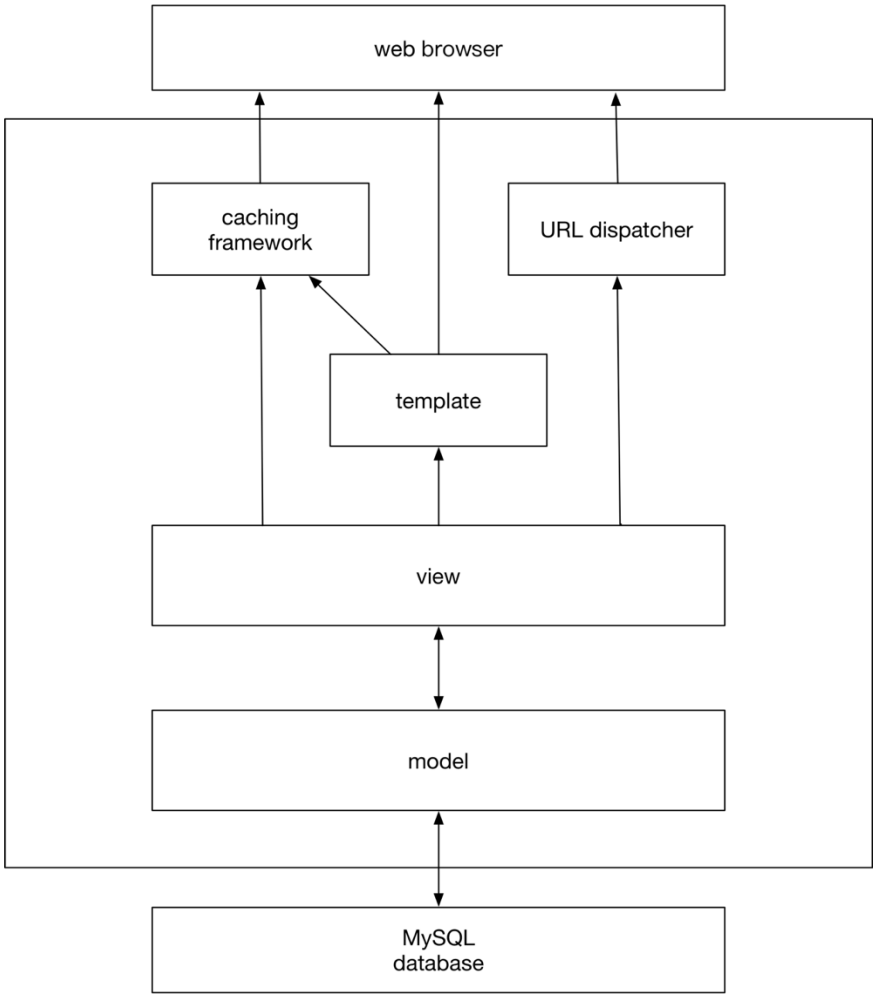
2.2.物理架构设计



1）服务器端：系统服务端部署在服务器上，系统使用者通过浏览器实现业务操作；

2) 客户端：系统采用 B / S 模式，所以 PC 客户端只要求 Chrome 浏览器支持。

2.3.技术架构设计



2.3.1、分层模型设计

系统在 Django 框架下进行开发，采用的是 MTV 设计模式，实现分层模型设计，详细分为模型层（model），模板层（template），视图层（view），层与层间互相联系、传输，紧密不可拆分，同时在分层的过程中使得创建使用思路清晰明确。

1）模型层

即数据存取层，该层处理与数据相关的所有事务： 如何存取、如何验证有效性、包含哪些行为以及数据之间的关系等。封装了对数据库等基础数据的操作和人脸识别的基本算法；

2）模板层

即表现层，该层处理与表现相关的决定：如何在页面和其他类型的文档中进行显示，前端使用 HTML+CSS+JavaScript 实现；

3）视图层

即业务逻辑层，该层包含存取模型及调取恰当模板的相关逻辑，是模板层和模型层之间的桥梁。

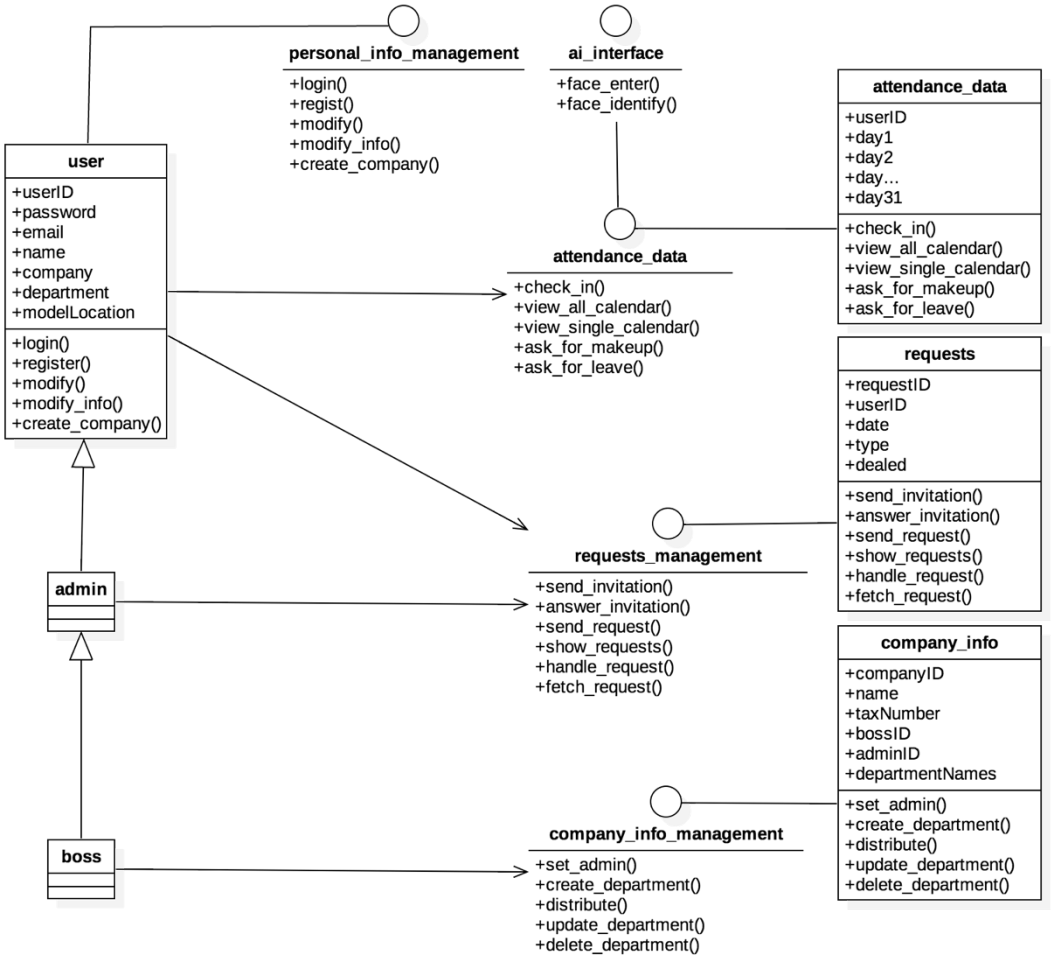
2.3.2、web service 技术

客户端的所有数据请求采用 Web Service 技术，在后台应用设计一个 web service 服务器，提供数据增删改查以及人脸识别等服务。

第三章 系统数据结构详细设计

3.1、系统数据结构类图

3.1.1、类图



3.1.2、类说明

1)、user 类

```
@interface Department : NSObject {
    String userID;
```

```
    String password;
    String email;
    String name;
    String company;
    String department;
    String model_location;//人脸存储模型位置
}
@end
```

2)、admin 类

```
//无数据类型，只有特殊方法
```

3)、boss 类

```
//无数据类型，只有特殊方法
```

4)、company 类

```
@interface SaleProductsList : NSObject {
    String companyID;
    String tax_number;//企业税号
    String bossID;//老板的 ID
    String adminID;
    String department_name;
    List company_attendance_list;//公司员工考勤列表
}
@end
```

5)、attendance_data 类

```
@interface SaleProductsList : NSObject {
    String userID
    String day1;
```

```
    String day2;
…
    String day31;
}
@end
```

6)、requests 类

```
@interface SaleProductsList : NSObject {
    String requestID;
    String userID;
    String date;
    String type;
    Bool dealed;
}
@end
```

3.2、后台有关数据表说明

3.2.1、个人信息管理模块

（1）注册

名称	*代码*	*数据类型*	*文本说明*
用户名	userID	varchar(20)	
姓名	name	varchar(20)	
邮箱	email	varchar(20)	
密码	password	varchar(20)	

（2）登录

名称	*代码*	*数据类型*	*文本说明*
用户名	userID	varchar(20)	
密码	password	varchar(20)	

（3）信息修改

名称	*代码*	*数据类型*	*文本说明*
待修改信息	status	int	
修改后内容	info	varchar(20)	
用户名	userID	varchar(20)	

（4）创建公司

名称	*代码*	*数据类型*	*文本说明*
公司名	company	varchar(20)	
税号	tax_number	varchar(20)	公司唯一标识符
创建者	boss_id	varchar(20)	

3.2.2、人工智能模块

（1）人脸录入

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	
人脸图片	imgs	list	传递 200 张人脸图片

（2）人脸识别

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	
人脸图片	img	image	传递 1 张人脸图片以识别

3.2.3、打卡信息管理模块

（1）打卡

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	传递用户名，调用人工智能模块进行操作
用户人脸	user_img	img	传递用户人脸

（2）查看公司员工考勤日历

名称	*代码*	*数据类型*	*文本说明*
查询者用户名	query_user_id	varchar(20)	检测该员工是否为公司管理员，以防数据泄露
公司名	company	varchar(20)	

（3）查看单个员工考勤日历

名称	*代码*	*数据类型*	*文本说明*

查询者用户名	query_user_id	varchar(20)	检测该员工是否为本身或公司管理员，以防数据泄露
待查询用户名	user_id	varchar(20)	

（4）补卡申请

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	补卡者用户名
补卡理由	content	varchar(100)	

（5）请假申请

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	
请假理由	content	varchar(100)	

（6）审批申请（调用消息处理模块）

名称	*代码*	*数据类型*	*文本说明*
申请 id	request_id	varchar(20)	
状态码	status	bool	表示同意与否

3.2.4、站内信模块

（1）发出邀请信息

名称	*代码*	*数据类型*	*文本说明*
收件人	receiver_id	varchar(20)	
发件人	sender_id	varchar(20)	
邀请内容	content	varchar(100)	

（2）接受邀请（调用消息处理模块）

名称	*代码*	*数据类型*	*文本说明*
邀请信息 id	request_id	varchar(20)	
状态码	status	bool	

（3）查看消息列表

名称	*代码*	*数据类型*	*文本说明*
用户名	user_id	varchar(20)	

3.2.5、公司管理模块

（1）设置公司管理员

名称	*代码*	*数据类型*	*文本说明*
公司名	company	varchar(20)	
管理员 id	admin_id	varchar(20)	

（2）创建部门

名称	*代码*	*数据类型*	*文本说明*
部门名称	department	varchar(20)	
公司名	company	varchar(20)	

（3）分配员工部门

名称	*代码*	*数据类型*	*文本说明*
员工用户名	staff_id	varchar(20)	
公司名	company	varchar(20)	
分配部门名	department	varchar(20)	

3.3、数据接口说明

3.3.1、接口调用说明

1）人脸识别模块：

1. 矩阵 face2matrix（图片）
2. 矩阵 s faces2matrices（图片 s） 调用 1
3. bool train（userID,矩阵 s）
4. userID identify（矩阵）
5. bool face_enter（userID，图片 s） 调用 2 和 3,调用数据库模块
6. userID face_identify(one picture) 调用 1，

2）个人信息管理模块：

1. bool login（账号，密码）
2. bool register（账号，密码，名字，邮箱）
3. bool modify（账号，条目 column，String 内容） 调用数据库
4. bool create_company（账号，公司名 ，税号） 调用数据库

3）考勤打卡模块:

1. userID check_in（图片）调用 人脸识别模块函数 face_identify，调用 假设 check_in_info { info[][2] 第一列是 userID，第二列是时间 格式为 yy/mm/dd-hh:mm:ss@yy/mm/dd-hh:mm:ss,分别为上班打卡和下班打卡}
2. 所有员工打卡信息 view_all_calendar（date）
3. 单个员工打卡信息 view_single_calendar（userID,month）
4. bool ask_for_makeup(userID,date)
5. bool ask_for_leave(userID,date)

4）站内信模块:

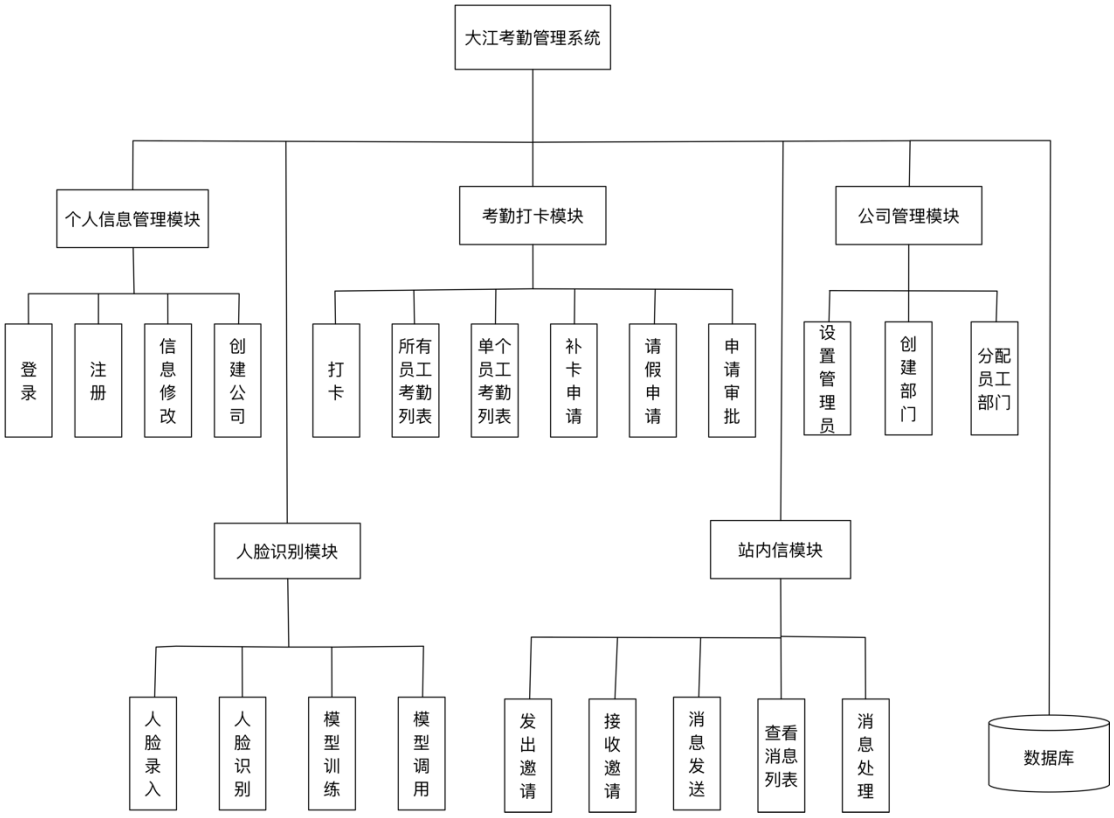
1. bool send_invitation(userID,companyID)
2. bool answer_invitation(userID,companyID ,bool)
3. bool send_request(userID,date,type)
4. List<requestID> show_requests()
5. bool handle_request(requestID)
6. < userID,companyID ,bool > fetch_request(requestID)

5）公司管理模块

1. bool set_admin(userID, companyID)
2. create_department(department_name, companyID)
3. distribute(userID, department_name, companyID)
4. update_department(companyID, origin_department, new_department)
5. delete_department(companyID, department_name)

第四章 系统业务详细设计

4.1、系统功能结构



4.2、系统功能设计

4.2.1、个人信息管理模块：

4.2.1.1、登录：

用户输入用户名密码，点击登录，将登录请求转交给服务器，返回登录结果，若成功，则根据用户类型跳转到相应的界面；

4.2.1.2、注册：

用户输入用户名、邮箱、密码，后台检测其用户名是否重复，若为合法用户名，则返回注册成功，跳转到登录界面；

4.2.1.3、信息修改：

用户在修改信息界面选定更改的信息并点击保存，之后将修改的请求以及内容传递至后台，若更改成功，返回成功，则跳转到用户登录后界面，若更改失败，跳转到信息修改界面并提示；

4.2.1.4、创建公司：

用户若未加入公司，则有创建公司选项，可以输入公司名、税号，提交创建公司请求。将用户名、公司名、税号作为表单传至服务器，若成功，则返回成功界面，反之返回创建公司失败；

4.2.2、公司管理模块

4.2.2.1、设置管理员：

老板用户选择员工并任命其为管理员，后台检测该员工是否为管理员，若是，则更改数据库字段并返回成功，反之则弹出提示信息；

4.2.2.2、创建、修改部门：

老板用户创建部门，选定部门名称即可提交表单；修改某部门名称也将公司名、原名、现名提交，若更改成功，则返回成功信息，反之弹出提示信息；

4.2.2.3、分配员工部门：

用户为员工分配部门，提交请求，将员工名、现部门传递至后台进行修改；

4.2.3、考勤打卡模块：

4.2.3.1、打卡：

员工可以在登录系统后进行打卡，也可以在登录之前进行打卡，打卡只传递用户的人脸，由人工智能模型自动识别用户信息，并将打卡信息存入系统。

4.2.3.2、查看所有员工考勤列表：

4.2.4、人脸识别（ai）模块：

4.2.4.1、人脸录入：

调用终端摄像头，提示用户将脸部置于摄像头内；等时间间隔截取 200 张用户的图片，传输至服务器，录入结束；在服务器端进行模型的训练；训练结束后，检测正确率：若达到要求值，即训练成功，则录入成功，分配给用户模型所在的位置并存入数据库；若未达到要求值，则重新录入；

4.2.4.2、人脸识别：

调用终端摄像头，提示用户将脸部置于摄像头内；等时间间隔截取用户图片发送至服务端；服务器端使用模型对图片进行人脸识别，返回识别结果；若成功，浏览器停止截取，识别成功。若超时，则识别失败，返回人脸识别界面。

4.3、系统参数设计

4.3.1、信息存储时间长度信息

用户 cookie 信息保存时间：3600

4.3.2、路径设计

attendance_data:存放员工打卡信息的 app

basic_info:存放员工和公司基本信息的 app

requests_data:存放员工请求、管理员处理请求的 app

backends\ai:后端的人脸识别部分

backends\attendance_checking:后端的签到打卡处理部分

backends\company_management:后端的公司管理部分

backends\mail_management:后端的请求处理部分

backends\personal_info_management:后端的个人信息管理部分

bigRiver:存放项目的基本信息，决定页面跳转逻辑等

static\css:存放前端所需的 css 文件

static\js:存放前端所需的 JavaScript 文件

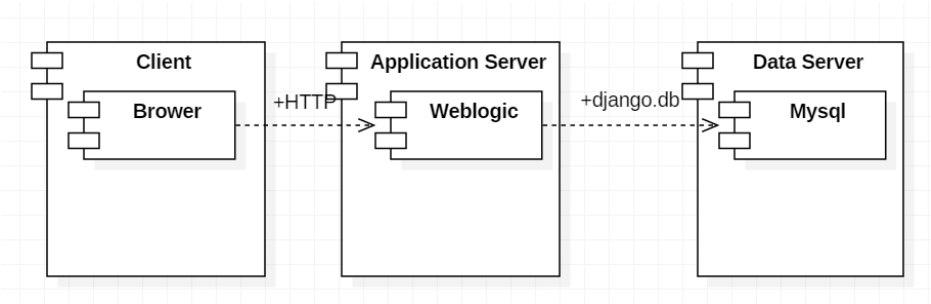
static\fonts:存放前端所需的字体文件

static\images:存放前端所需的图片文件

templates:模板文件夹，存放前端的 HTML 文件等

第五章 部署设计

5.1、系统部署图



5.2、运行环境

5.2.1 客户机器环境

1) 浏览器 chrome 版本 68.0.3440.106

5.2.2 开发环境要求

项目	名称	版本
开发平台	Windows	Windows10
开发工具	PyCharm	PyCharm 2017.3.3
代码管理工具	Git	2.18.0.windows.1
开发环境	Django，TensorFlow，Numpy，Opencv-python	Django 2.1.1，TensorFlow 1.10.1，Numpy 1.15.1,Opencv-python 3.4.2.17