第十届大学生服务外包创新创业大赛作品

智能会议室系统·智会

Intelligent Conference Room System

软件详细设计

组　　名：FTD HHU

团队编号：1801947

团队成员：张亦昕 侯晓妍 刘伦豪杰 陆建华 张袁

目录

1、前言

1.1 编写目的

随着市场经济发展，会议室的使用需求日益增多，传统会议室主要基于人为管理，无法为用户提供高效的人性化的服务，显现出大量的弊端，如会议冲突、管理低效、费时费力等。因此，智能会议室管理系统显现出巨大的市场潜力和发展空间。

在需求规格说明书中，已经将用户对本系统的需求做了详细叙述，本文档为开发人员和用户提供智能会议室管理系统的功能、总体设计、数据库设计等概要说明，方便进行软硬件测试和使用推广。

1.2 项目背景

随着深度学习技术的兴起，智能交互逐渐深入到生产生活的方方面面，智能化的设备也越来越受到市场的欢迎，于此同时，随着市场经济发展，会议室的使用需求日益增多，传统会议室主要基于人为管理，无法为用户提供高效的人性化的服务，显现出大量的弊端，如会议冲突、管理低效、费时费力等。因此，智能会议室管理系统显现出巨大的市场潜力和发展空间。

1.3 技术背景

近年来，立足于引领计算摄影与视觉人工智能未来，创造“睛”彩智慧生活的虹软公司运用最新的神经网络、深度学习方法，改进开发出世界领先的视觉算法，推出了全球首个免费视觉人工智能开发平台，并致力于推动视觉 AI 技术的全行业应用。基于此，FTD HHU 团队在虹软人工智能开发平台的基础上开发设计出一套智能会议室管理系统。

2、创意描述

针对会议室智能化管理的诸多分析，团队在如下方面进行了一定的创意分析：

1. 开发方式与平台上：采用虹软人工智能开放平台的人脸识别技术结合物联网智能硬件实现会议室自动化、智能化。

2. 系统功能方面：增加了智能匹配会议室、会议延迟警告、人脸识别门禁与自动签到、环境监测等功能，并采取线上与线下相结合管理制度。

3. 其他方面：满足了大部分会议室系统基本要求，实用性和可移植性强；巧妙地将虹软人脸识别技术拓展于室内环境检测，无人自动关灯关门既提高了安全性又节约了能源，系统的自动化和智能化程度高。

3、功能简介

3.1 目标

本智能会议室管理系统旨在将虹软人工智能平台与物联网智能硬件技术相结合，开发出一套能够满足绝大部分应用场景的人性化、智能化、自动化的会议室管理系统，为用户提供良好体验。

基本技术目标包括:

(1) 基于虹软人工智能平台开发设计
(2) 支持多个会议及会议室管理，及多人同时预定
(3) 支持人脸门禁会议、签到打卡
(4) 系统易用、可拓展、人性化、智能化
(5) 硬件成本适宜

3.2 运行环境

应用环境:

网页客户端：Edge、火狐、谷歌等浏览器

微信端用户：微信客户端 2.6.2 及以上版本

会议室端：带前置摄像头平板、stm32f4 或其他嵌入式系列开发板

3.3 功能需求概述

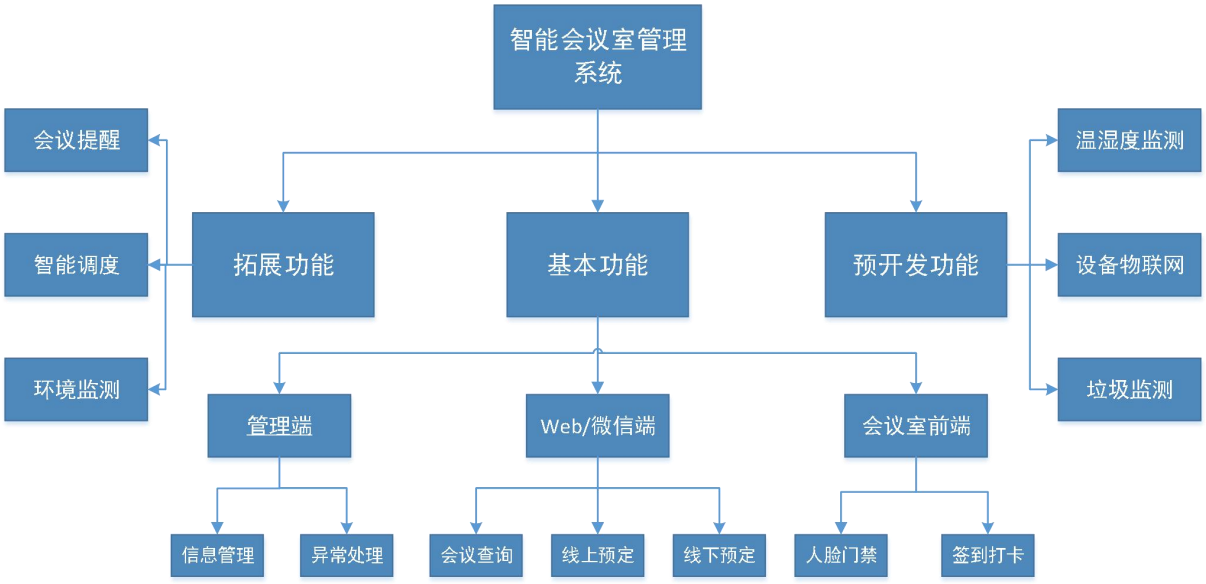

图 3.3.1 功能结构图

3.3.1 会议室查询

在网页端和微信端，可预览所有会议室及排期情况；查询预定满足条件的会议室。

3.3.2 会议室预定

1. 线下预定

可在网页端、微信端登陆后，填写会议的详细信息后，对空闲时段的会议室进行预定或修改。

2. 线下预定

在会议室门口处，操作平板对此会议室进行预定；打开微信端的扫一扫，对会议室门口处的二维码进行扫描后亦可预定。

3.3.3 人脸门禁

在会议室前端配备平板，通过人脸核验才可开启电磁锁门禁。

3.3.4 签到打卡

与会者通过人脸核验后系统自动签到打卡，可在前端查看。

3.3.5 会议室管理

增删人员信息，录入人员的照片信息；对会议室的异常情况进行处理。

3.3.6 会议提醒

1.开始提醒

与微信小程序关联的公众号会在会议预定后，通知所有的与会者，并会发送短信；在会议开始 15 分钟前，再次提醒，以免迟到。

2.结束提醒

距会议结束时间 5 分钟，提醒预定者按时结束会议。

3.3.7 智能调度

根据预定者输入的参会信息，使用智能调度算法，智能推荐匹配会议室。

3.3.8 环境检测

会议室内部单片机系统通过摄像头监测室内环境，在非会议时段检测到会议室内无人时控制断电关闭门禁。

3.4 条件与限制

系统使用的条件与限制主要有：除了离线功能外，本作品的客户端功能均需要连接系统的服务器。

4、总体设计

4.1 平台设计原则与思想

4.1.1 设计原则

智能会议室系统系统设计遵循以下原则:

（1）良好的用户界面
（2）操作简便性
（3）系统可拓展性
（4）人性化、自动化设计
（5）硬件成本适宜
（6）快速性，人脸门禁核验速率达到使用要求

4.1.2 设计思想

良好的用户界面可以极大地提高用户的使用体验，前端界面设计尽量简洁大方，并保持一致的审美效果。

为了使系统操作简便，设计中模块分区合理，符合用户的使用习惯，并显示提示语言与用户进行人机交互。

为了使系统功能具有一定的可拓展性，采用分层模式的思想，将系统分为三层：数据层、逻辑层、表现层。对于可能变化的文件内容，系统只需要修改数据层与数据库的交互代码即可，无需改动逻辑层和表现层。

采用虹软人脸识别技术使得人脸门禁核验快速准确，满足绝大部分应用场景使用需求。

通过拓展虹软人脸识别技术，将其衍生应用于室内环境监测，使系统更加智能化、人性化。

4.2 系统总体设计方案

4.2.1 系统结构

智会系统结构如图 4.2.1 所示

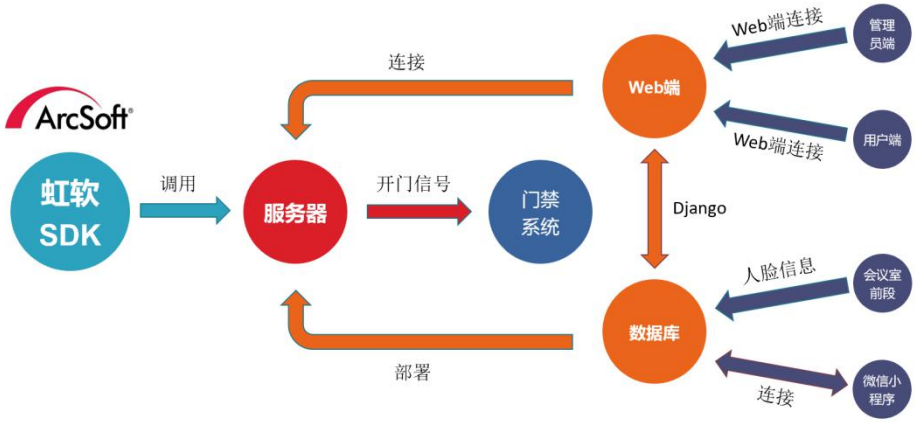

图 4.2.1 智会系统结构图

4.2.2 功能分配

1、核心功能设计

智会系统的核心功能如图 4.2.2.1 所示。

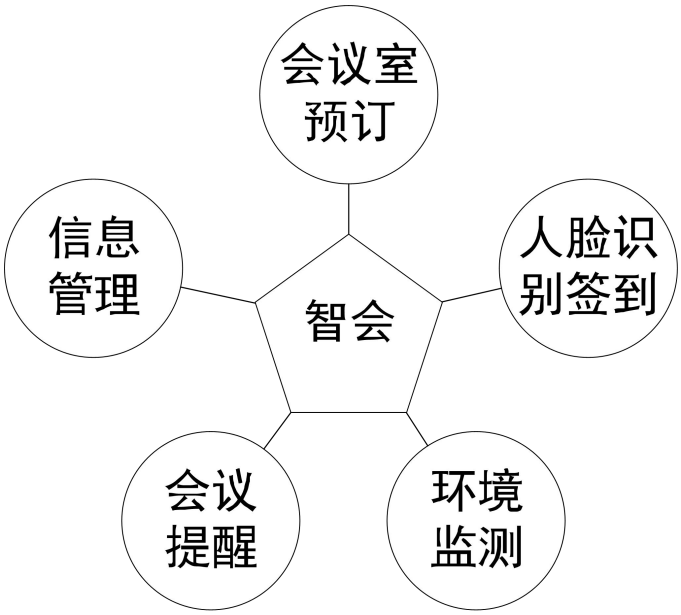

图 4.2.2.1 智会核心功能设计

2、分模块设计

智会系统主要包括 Web 客户模块、微信用户模块、后台管理模块和会议室前端模块。

（1）Web 客户模块中，普通用户可以根据公司给定的工号和初始密码进行登录，同时第一次登录后需要绑定微信号。用户可以通过访问 Web 端预定界面查看个人即将参加

的会议、创建的会议、已经参加的会议等。同时可以预览公司所有会议室详情按照自己的喜好选择指定的空闲会议室，也可以通过指定会议时间并通过智能匹配最佳会议室。

此模块设计如图 4.2.2.2 所示。

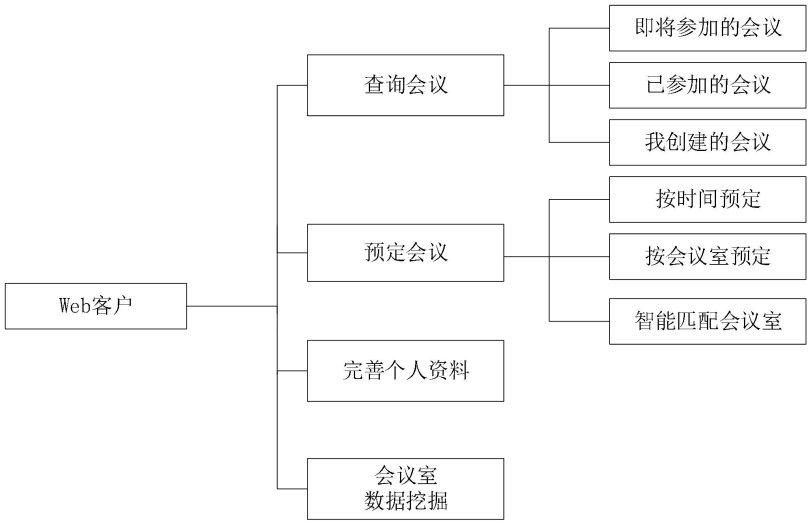

图 4.2.2.2 Web 客户模块

（2）微信用户模块中，用户通过预先网页端登录时绑定的微信号登录小程序，即可实现查询、预定、修改会议室等功能，并可以直接在小程序上发起会议。在会议预定成功后，小程序关联的公众号会给用户发来会议提醒，同时在会议召开前半个小时还会再发送一次会议提醒，防止用户遗忘，并且在会议结束前 10 分钟会发来会议即将结束的信息，防止影响下一场会议的召开。

此模块设计如图 4.2.2.3 所示。

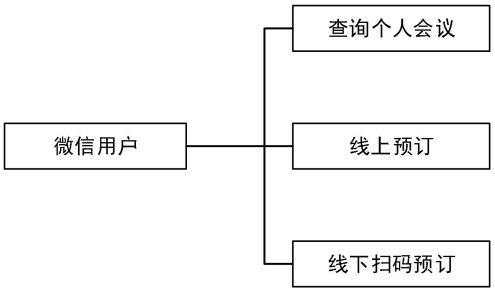

图 4.2.2.3 微信用户模块

（3）后台管理模块部分提供基于 Web 端的会议室后台管理系统，我们使用了 HTML5+CSS3+JavaScript，Bootstrap 框架，Jquery 构建前端。管理员身份信息一开始就被注册到数据库当中，管理员通过初始账号和密码登录管理员端后可以对使用人员进行管理、人脸注册等功能，具体包括对公司部门进行增删改查，对员工 ID、所属部门等个人信息进行增删改查，对会议室编号、硬件设施等进行编辑，同时对申请发起的会议进行实时审核功能。

管理员可以通过部门管理对部门信息进行修改，同时结合快速搜索功能方便管理员操作，员工管理可以对员工照片、姓名、归属部门、电话以及绑定的微信号进行编辑。同时管理员可以及时的对会议室的硬件设施情况进行实时编辑。

此模块设计如图 4.2.2.4 所示

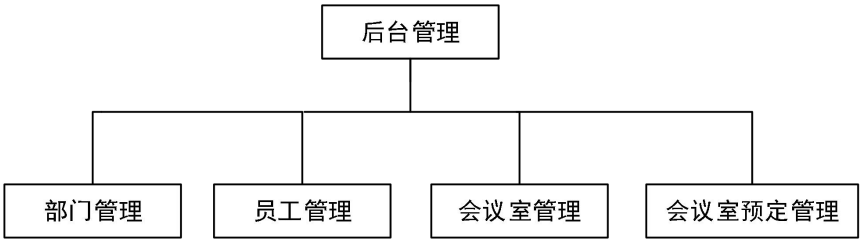

图 4.2.2.4 后台管理模块

（4）会议室前端通过放置平板，在平板上滚动播放当前会议室的预定情况，同时提供刷脸签到和实时预定功能，用户可以通过查看会议室当前预定情况后选择该会议室的空余时间进行预定，同时也可以通过微信小程序扫描该会议室的二维码进行预定。

此模块设计如图 4.2.2.5 所示。

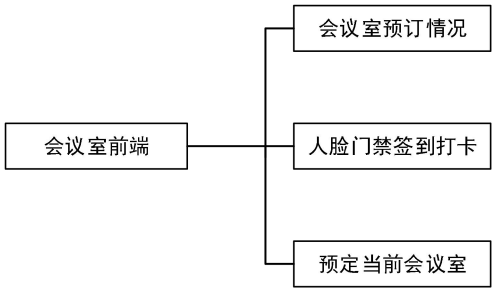

图 4.2.2.5 会议室前端模块

4.3 系统部署方案

智能会议室管理系统的基础架构由个人计算机，Web 服务器，平板设备，数据库服务器，单片机操作系统以及之间相连的网络和防火墙构成。用户使用个人计算机或会议室前端平板设备访问外部网络，并向 Web 服务器发送数据和操作请求，应用服务器与 Mysql 数据集直接相连，其根据个人计算机或前端设备发送的请求，返回来自数据库的内容，或对数据库进行读写操作，处理完成后给前端响应和单片机系统响应，其部署方案如下图所示。

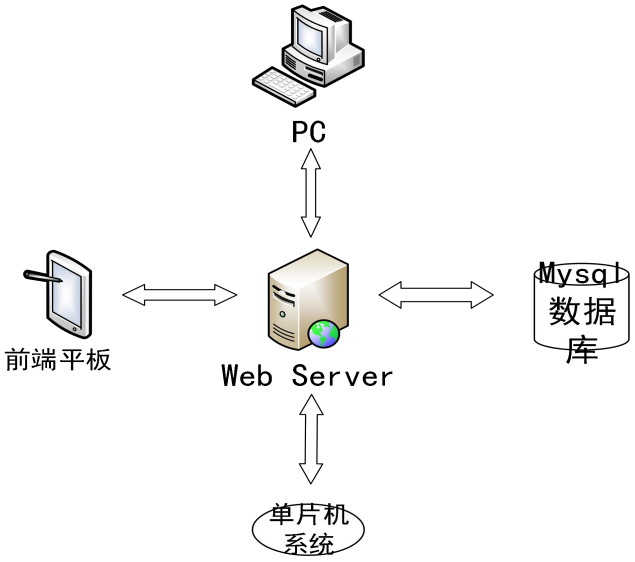

图 4.3.1 系统部署方案

4.4 系统界面设计

4.4.1Web 客户端

客户通过在浏览器地址栏输入 URL，即可进入 Web 端智会系统登录界面。

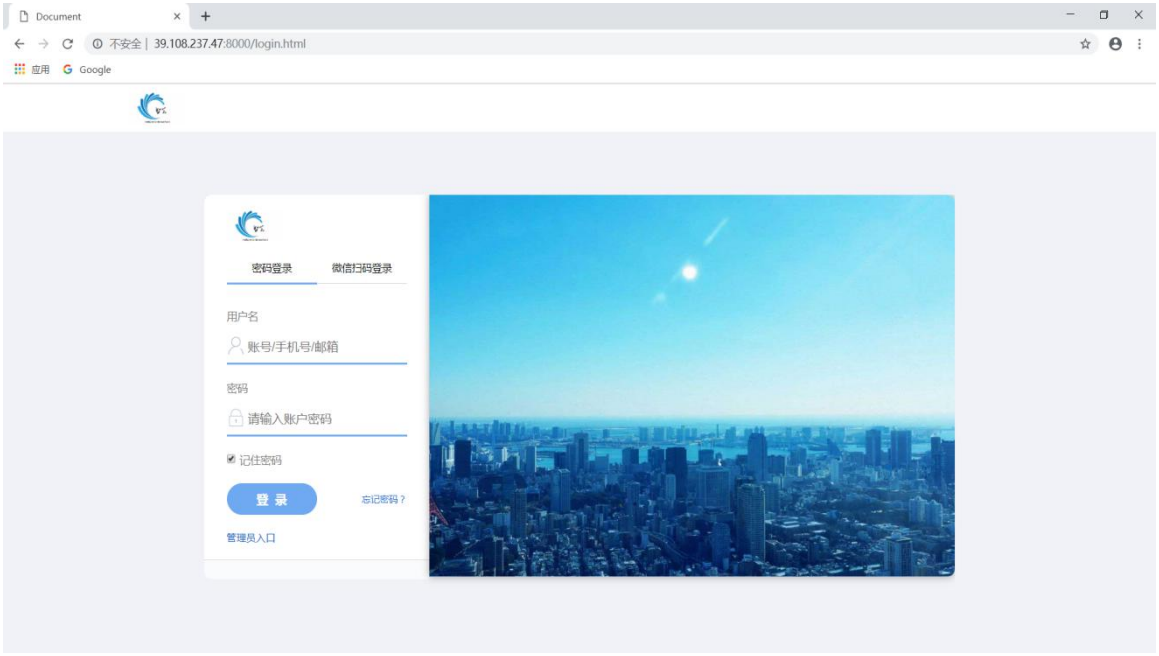

图 4.4.1 登录界面

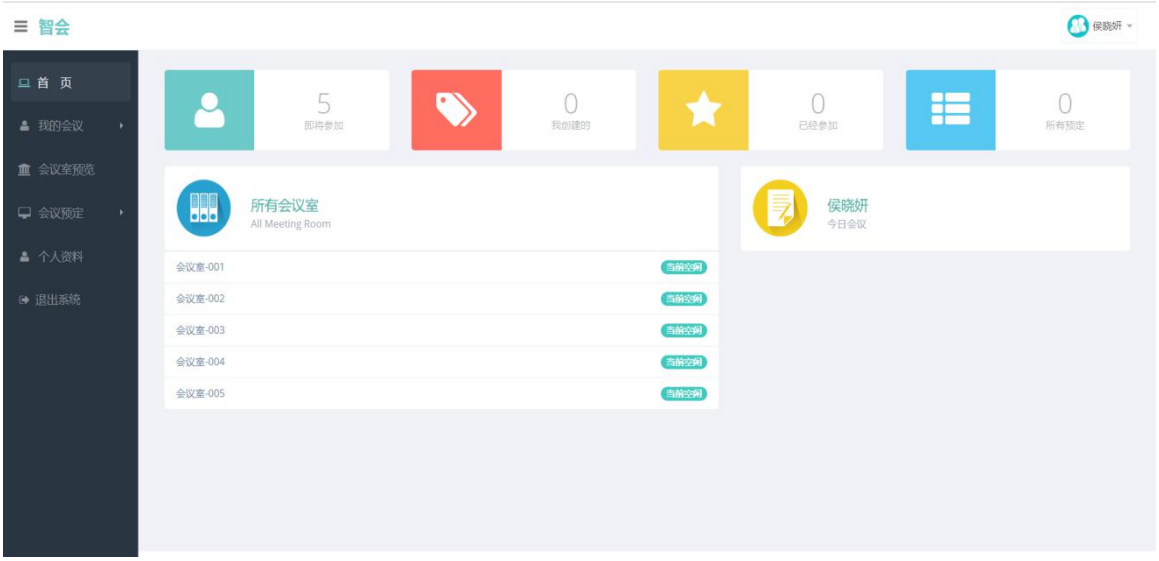

图 4.4.2 客户端主界面

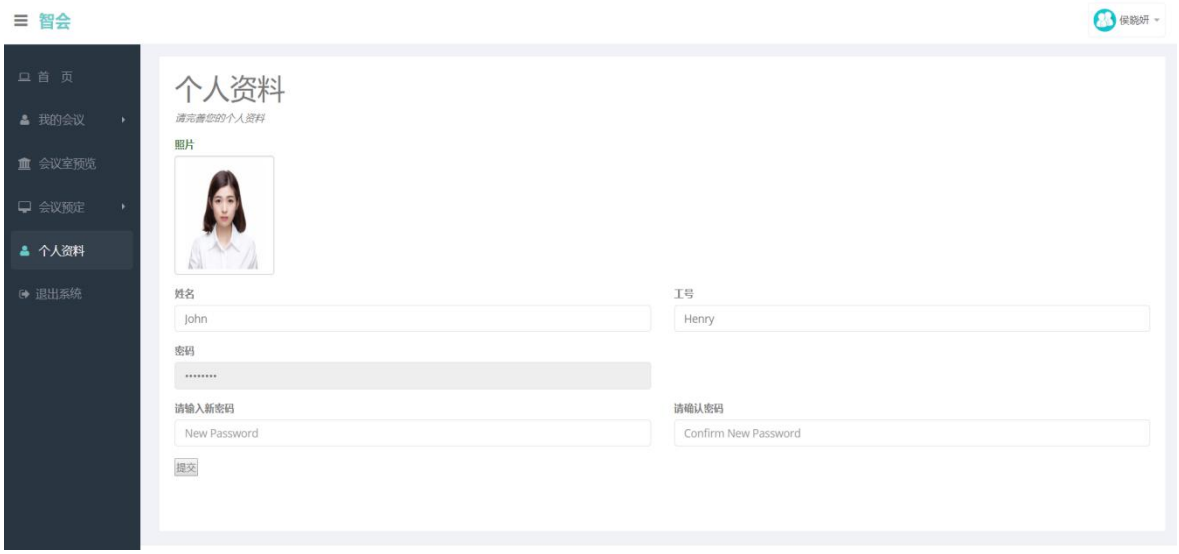

图 4.4.3 完善个人资料

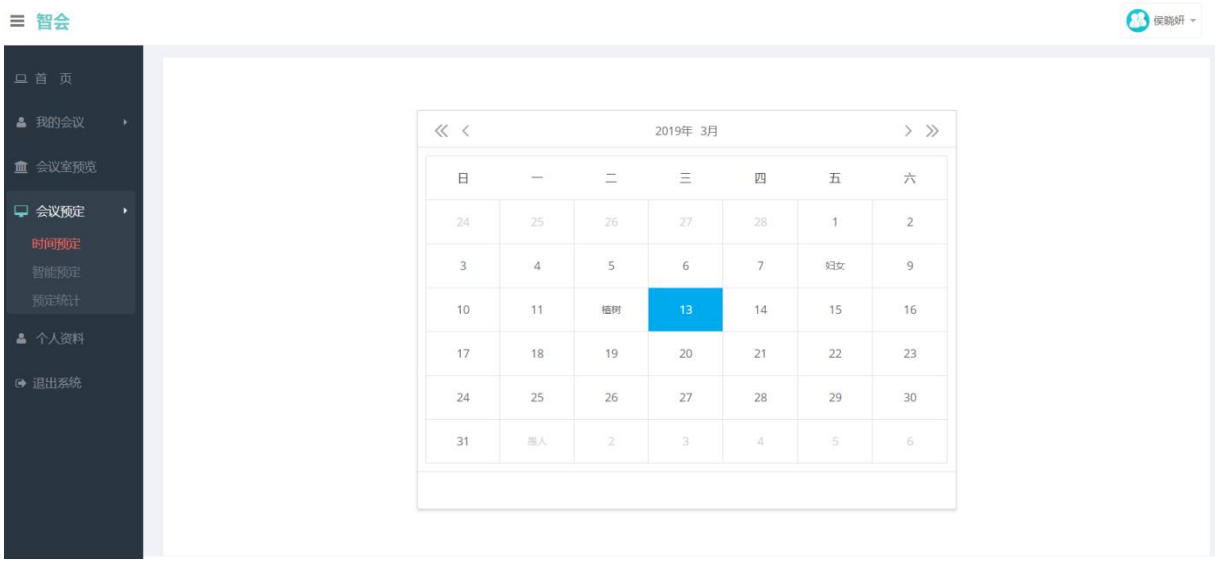

图 4.4.4 按时间预定界面

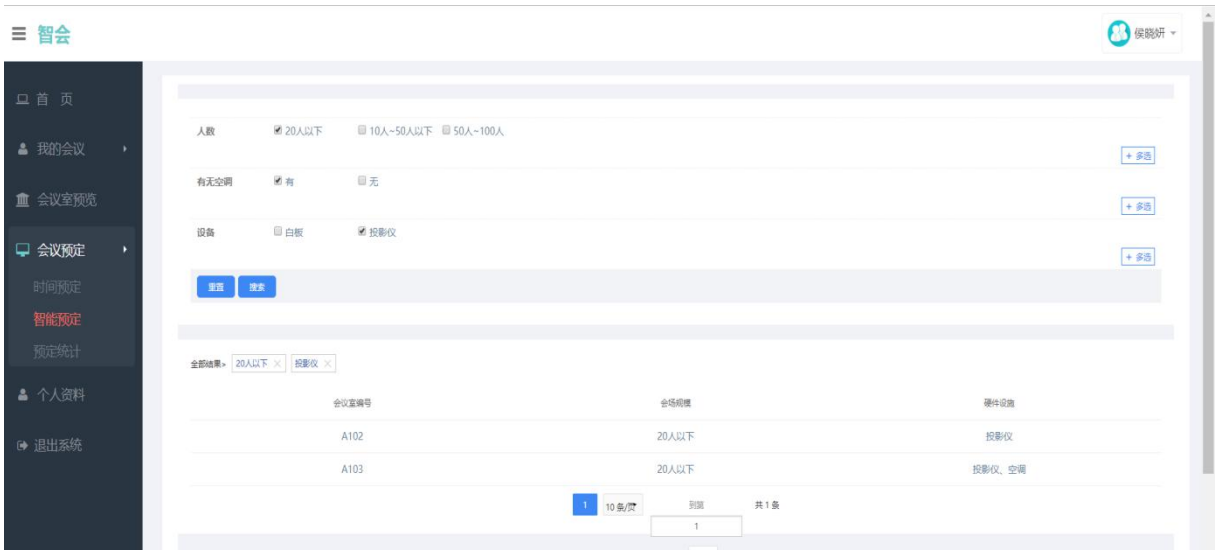

图 4.4.5 智能匹配预定界面

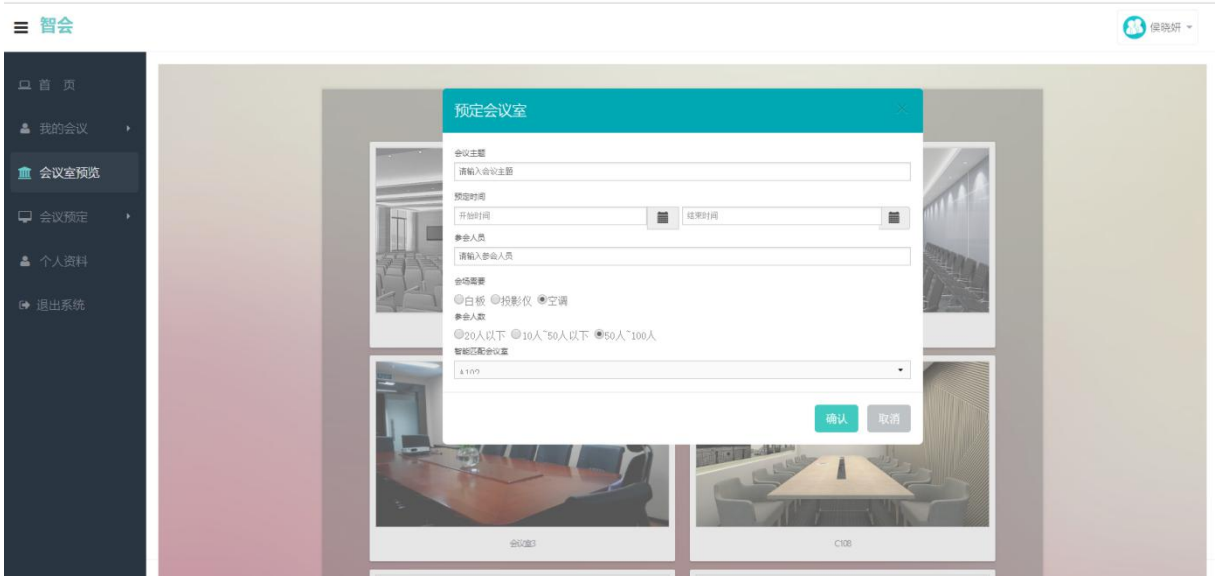

图 4.4.6 指定会议室预定界面

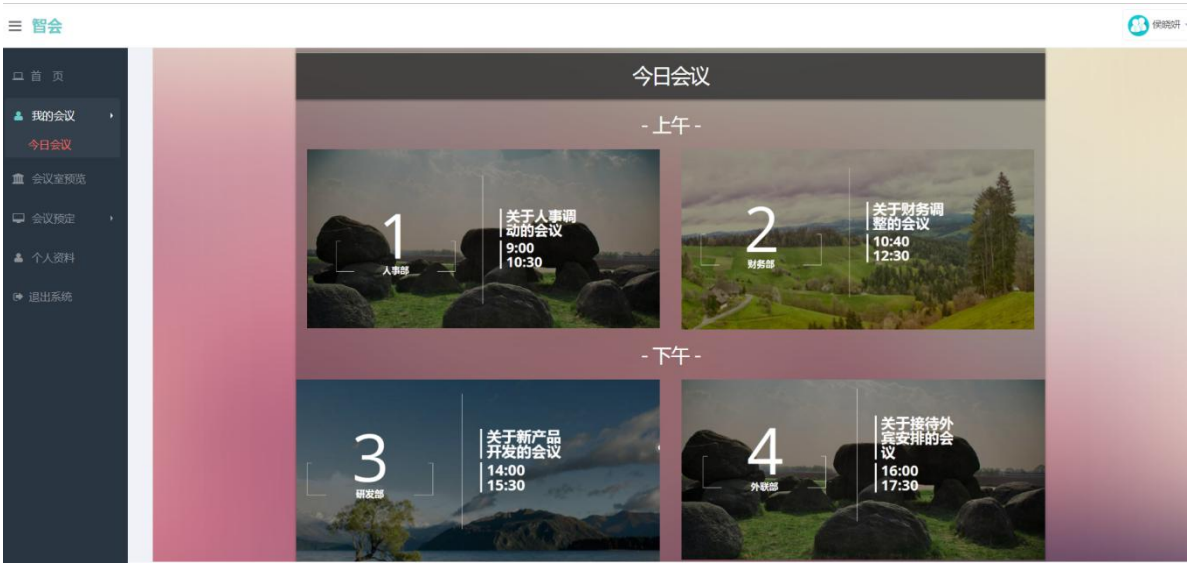

图 4.4.7 用户今日会议界面

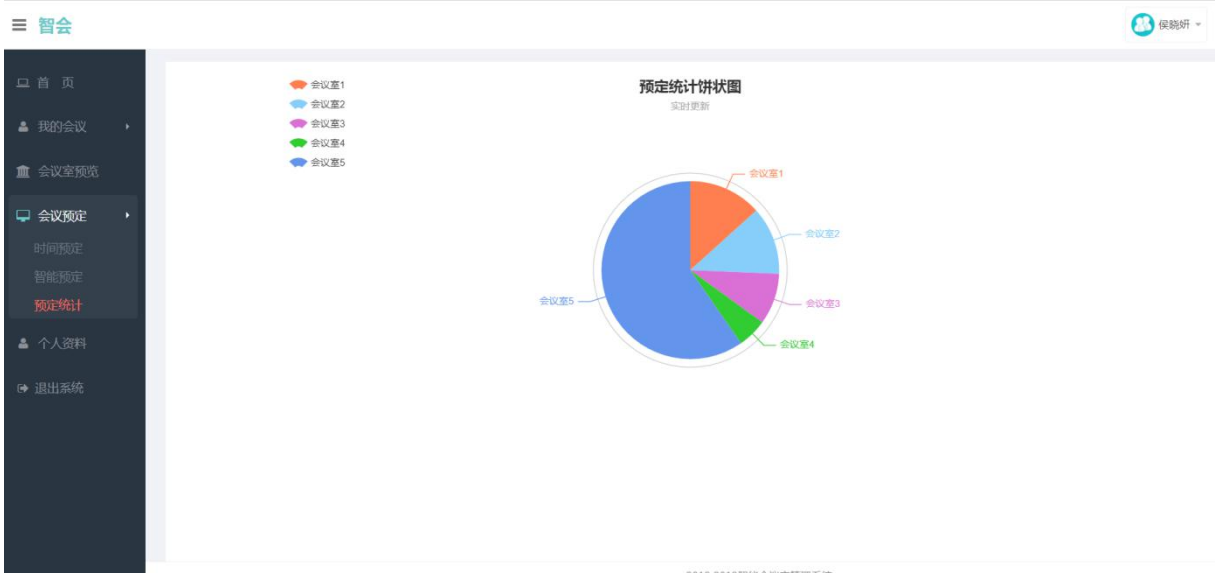

图 4.4.8 数据挖掘界面

4.4.2 微信用户端

图 4.4.9 小程序登录界面

图 4.4.10 小程序主页界面

图 4.4.11 会议室预定界面

图 4.4.12 我的日程界面

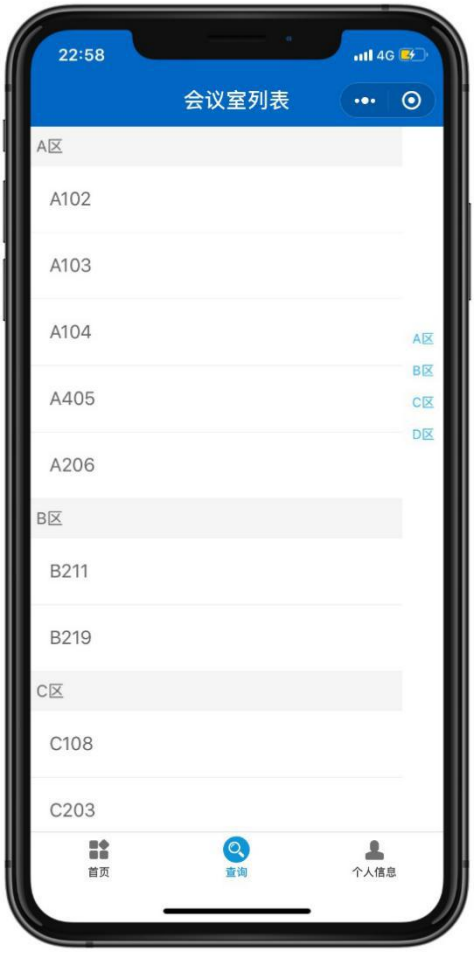

图 4.4.13 会议室查询界面

图 4.4.14 各会议室排期界面

图 4.4.15 预约详情界面

图 4.4.16 线下扫码预定界面

图 4.4.17 联系人界面

图 4.4.18 个人信息界面

4.4.3 后台管理端

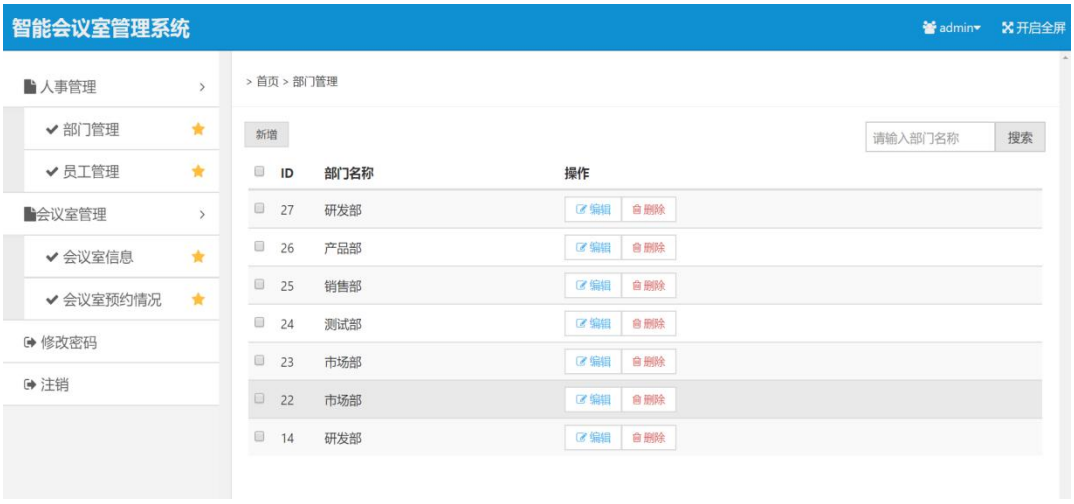

图 4.4.19 部门信息界面

图 4.4.20 员工信息界面

图 4.4.21 会议室信息界面

图 4.4.22 会议室预约审核界面

4.4.4 会议室前端

图 4.4.23 会议室前端界面

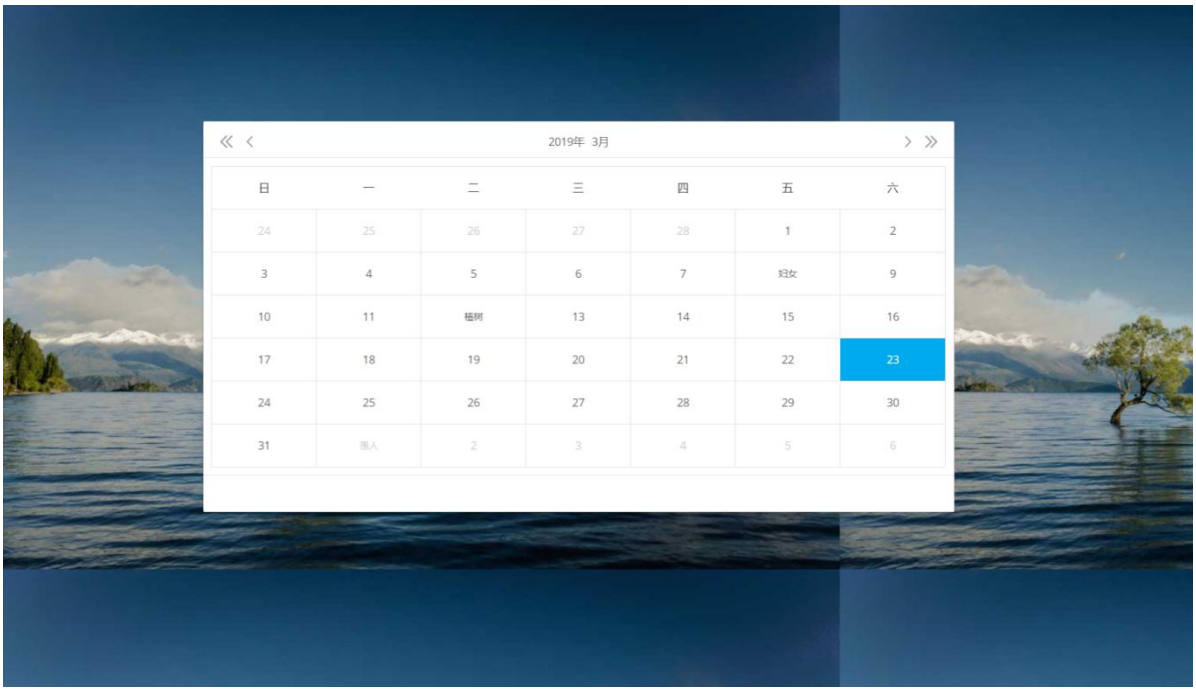

图 4.4.24 会议室前端预定界面

图 4.4.25 本次会议签到详情界面

5、数据库设计

5.1 数据库描述

表 5.1 数据表列表

序号	名称	描述
1	app_department	部门表存储部门的名称，id
2	app_meeting	会议表存储会议的 id、创建人、名称、会议日期、开始时间、结束时间、与会人员
3	app_meetingroom	会议室表存储会议室的 id、名称、设备、概貌、容纳人数
4	app_staff	职工表存储员工的 id、工号、名称、照片
5	app_sudo	Sudo 存储当前时段与会人员面部信息
6	app_userinfor	Userinfo 表存储摄像头检测识别的人脸信息
7	auth_user	存储员工密码、是否为管理员、id

5.2 逻辑结构设计

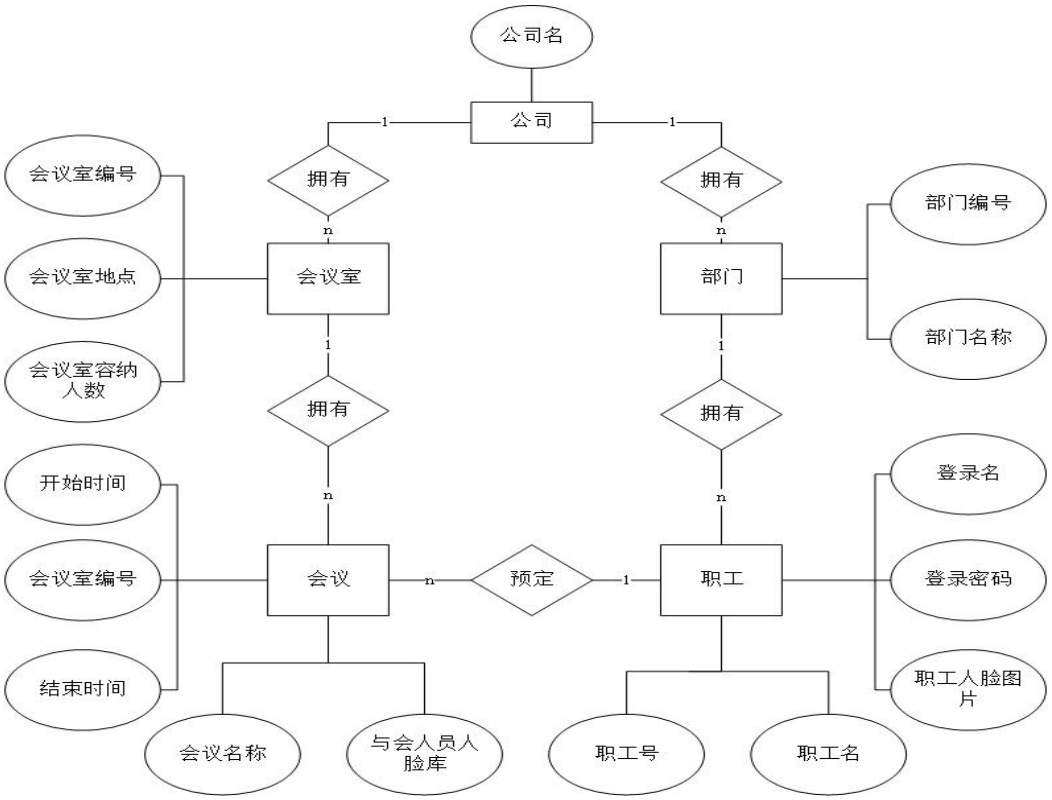

图 5.2 数据库实体-联系图

5.3 物理结构设计

1、用户信息表（auth_user）

字段	说明	类型	可空	备注
User_id	id	int	NO	主键，自增
User_name	名称	varchar	YES	
pwd	密码	char	YES	

表 5.3.1 用户信息表

2、检测信息表（auth_userinfo）

字段	说明	类型	可空	备注
Info_id	id	int	NO	主键，自增
Info_data	面部数据	Longtext	YES	人脸数据
Info_state	状态	Int	YES	更新状态

表 5.3.2 检测信息表

3、与会人员信息表（app_sudo）

字段	说明	类型	可空	备注
Sudo_id	id	int	NO	主键，自增
Sudo_name	名称	varchar	YES	
Sudo_data	面部数据	Longtext	YES	
Sudo_number	工号	int	YES	

表 5.3.3 与会人员信息表

4、全体员工信息表（app_staff）

字段	说明	类型	可空	备注
Staff_id	工号	int	NO	主键
Staff_name	姓名	varchar	YES	
Staff_image	照片	longtext	YES	
Staff_sex	性别	char	YES	
Staff_job	职业	varchar	YES	

表 5.3.4 全体员工信息表

5、会议室信息表（app_meetingroom）

字段	说明	类型	可空	备注
Room_id	id	int	NO	主键，自增
Room_name	名称	varchar	YES	
Room_number	容纳人数	Int	YES	
Room_shebei	设备	Longtext	YES	

表 5.3.5 会议室信息表

6、会议信息（app_meeting）

字段	说明	类型	可空	备注
Meet_id	id	int	NO	主键，自增
Meet_name	会议名称	varchar	YES	
Meet_owner	发起人	varchar	YES	
Meet_data	会议日期	char	YES	
Meet_btime	开始时间	char	YES	
Meet_etime	结束时间	char	YES	

表 5.3.6 会议信息

7、部门表(app_department）

字段	说明	类型	可空	备注
Dt_id	id	int	NO	主键，自增
Dt_name	名称	varchar	YES	

表 5.3.7 部门表

结语

本团队开发设计了一套智能会议室管理系统，具有会议室查询、管理、智能调度、会议签到等功能，配备人脸识别门禁，与智能硬件相结合，为用户提供人性化的良好体验，符合当前行业发展的潮流。

附录

核心程序 1：

```
import os,django
os.environ.setdefault("DJANGO_SETTINGS_MODULE", "face_end.settings")
django.setup()
import glob
from lib import fd_clibrary, fg_clibrary, fa_clibrary, ff_clibrary
from lib.face_detect_sdk_library import *
from lib.face_age_sdk_library import *
from lib.face_gender_sdk_library import *
from lib.face_recognition_sdk_library import *
from util import image_loader
from conf.config import *
import traceback
import time
import base64
import sys
from app.models import UserInfor,sudo,staff
from socket import *
APP_ID = c_char_p(bytes(config.get('arcsoft', 'APP_ID'), encoding='utf-8'))
FD_SDK_KEY = c_char_p(bytes(config.get('arcsoft', 'FD_SDK_KEY'), encoding='utf-8'))
FASE_SDK_KEY = c_char_p(bytes(config.get('arcsoft', 'FASE_SDK_KEY'), encoding='utf-8'))
FSGE_SDK_KEY = c_char_p(bytes(config.get('arcsoft', 'FSGE_SDK_KEY'), encoding='utf-8'))
FAFR_SDK_KEY = c_char_p(bytes(config.get('arcsoft', 'FAFR_SDK_KEY'), encoding='utf-8'))

WORKBUF_SIZE = 50 * 1024 * 1024
MAX_FACE_NUM = 10
bUseYUVFile = False
bUseBGRToEngine = True

fd_engine = None
fa_engine = None
fg_engine = None
ff_engine = None

def init_face_engine():
```

```
    global fd_engine
    fd_work_memery = fd_clibrary.malloc(c_size_t(WORKBUF_SIZE))
    fd_engine = c_void_p()
    ret = AFD_FSDK_InitialFaceEngine(APP_ID, FD_SDK_KEY, fd_work_memery, c_int32(WORKBUF_SIZE),
byref(fd_engine),
                                     AFD_FSDK_OPF_0_HIGHER_EXT, 32, MAX_FACE_NUM)

    if ret != 0:
        fd_clibrary.free(fd_work_memery)
        print(u'AFD_FSDK_InitialFaceEngine ret 0x{:x}'.format(ret))
        exit(0)


def init_face_age_engine():
    global fa_engine
    fa_work_memery = fa_clibrary.malloc(c_size_t(WORKBUF_SIZE))
    fa_engine = c_void_p()
    ret = ASAE_FSDK_InitAgeEngine(APP_ID, FASE_SDK_KEY, fa_work_memery, c_int32(WORKBUF_SIZE),
byref(fa_engine))
    if ret != 0:
        fa_clibrary.free(fa_work_memery)
        print(u'ASAE_FSDK_InitAgeEngine ret 0x{:x}'.format(ret))
        exit(0)


def init_face_gender_engine():
    global fg_engine
    fg_work_memery = fg_clibrary.malloc(c_size_t(WORKBUF_SIZE))
    fg_engine = c_void_p()
    ret = ASGE_FSDK_InitGenderEngine(APP_ID, FSGE_SDK_KEY, fg_work_memery, c_int32(WORKBUF_SIZE),
byref(fg_engine))
    if ret != 0:
        fg_clibrary.free(fg_work_memery)
        print(u'ASGE_FSDK_InitGenderEngine ret 0x{:x}'.format(ret))
        exit(0)

def init_face_recognition_engine():
    global ff_engine
    ff_work_memery = ff_clibrary.malloc(c_size_t(WORKBUF_SIZE))
    ff_engine = c_void_p()
    ret = AFR_FSDK_InitialEngine(APP_ID, FAFR_SDK_KEY, ff_work_memery, c_int32(WORKBUF_SIZE),
byref(ff_engine))
    if ret != 0:
        ff_clibrary.free(ff_work_memery)
        print(u'AFR_FSDK_InitialEngine ret 0x{:x}'.format(ret))
        exit(0)
```

```
def do_face_detection(fd_engine, image):
    face_res = POINTER(AFD_FSDK_FACERES)()
    ret = AFD_FSDK_StillImageFaceDetection(fd_engine, byref(image), byref(face_res))

    if ret != 0:
        print(u'AFD_FSDK_StillImageFaceDetection 0x{0:x}'.format(ret))
        return 0, [], []

    faces = face_res.contents
    face_num = faces.nFace
    #print('{} 个人脸'.format(face_num))
    dan_face = faces.rcFace[0]
    dan_orient = faces.lfaceOrient[0]
    if face_num > 0:
        return face_num, faces.rcFace, faces.lfaceOrient, dan_face, dan_orient


def do_face_age_estimation(fd_engine, fa_engine, image):
    face_num, rect_info, orient_info, dan_face, dan_orient = do_face_detection(fd_engine, image)
    age_face_input = ASAE_FSDK_AGEFACEINPUT()  # 定義臉部信息
    age_face_input.lFaceNumber = face_num
    age_face_input.pFaceRectArray = rect_info
    age_face_input.pFaceOrientArray = orient_info
    age_result = ASAE_FSDK_AGERESULT()  # 定義年齡檢測結果信息
    ret = ASAE_FSDK_AgeEstimation_StaticImage(fa_engine, byref(image), byref(age_face_input),
                                              byref(age_result))

    if ret != 0:
        print(u'ASAE_FSDK_AgeEstimation_StaticImage 0x{0:x}'.format(ret))
        return []

    return age_result


def do_face_gender_estimation(fd_engine, fg_engine, image):
    face_num, rect_info, orient_info, dan_face, dan_orient = do_face_detection(fd_engine, image)
    gender_face_input = ASGE_FSDK_GENDERFACEINPUT()
    gender_face_input.lFaceNumber = face_num
    gender_face_input.pFaceRectArray = rect_info
    gender_face_input.pFaceOrientArray = orient_info
    gender_result = ASGE_FSDK_GENDERRESULT()
    ret = ASGE_FSDK_GenderEstimation_StaticImage(fg_engine, byref(image),
byref(gender_face_input),
                                                 byref(gender_result))
    if ret != 0:
```

```
        print(u'ASGE_FSDK_GenderEstimation_StaticImage 0x{0:x}'.format(ret))
        return []
    return gender_result

#feature
def do_face_feature(fd_engine,ff_engine, image):
    face_num, rect_info, orient_info, dan_face, dan_orient = do_face_detection(fd_engine, image)
    recognition_face_input = AFR_FSDK_FACEINPUT()
    recognition_face_input.rcFace = dan_face
    recognition_face_input.lOrient = dan_orient
    # print(dan_face.left,'aaa')
    # print(dan_orient,'bbb')

    recognition_feature_result = AFR_FSDK_FACEMODEL()
    ret = AFR_FSDK_ExtractFRFeature(ff_engine, image,
recognition_face_input,recognition_feature_result)
    #print(type(recognition_feature_result))
    if ret != 0:
        print(u'AFR_FSDK_ExtractFRFeature 0x{0:x}'.format(ret))
        return []

    try:
        return recognition_feature_result.deepCopy()
    except Exception as e:
        traceback.print_exc()
        print('error')
        return None

#recognition
def do_face_recognition(ff_engine, feature_ku, feature_bi):
    #score = AFR_FSDK_FACE_DUIBI()
    score = c_float()
    ret = AFR_FSDK_FacePairMatching(ff_engine, feature_ku, feature_bi,byref(score))
    if ret != 0:
        print(u'AFR_FSDK_FacePairMatching 0x{0:x}'.format(ret))
        return []
    return score


def main():

    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'face_end.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
```

```
        "Couldn't import Django. Are you sure it's installed and "
        "available on your PYTHONPATH environment variable? Did you "
        "forget to activate a virtual environment?"
    ) from exc
execute_from_command_line(sys.argv)
##open client
tcp_client_socket = socket(AF_INET, SOCK_STREAM)
tcp_client_socket.connect(("192.168.43.28", 20000))
####init sdk
init_face_engine()
init_face_age_engine()
init_face_gender_engine()
init_face_recognition_engine()



dir_image_ku = sudo.objects.all()
num = len(dir_image_ku)
for i in range(num):
    temp = sudo.objects.values().get(id=i)
    temp_str = temp['photo_ku']
    image_temp = base64.b64decode(temp_str)
    file = open('image_ku/{}.jpg'.format(i+1), 'wb')
    file.write(image_temp)
    file.close()

# time.sleep(3)
while True:
    ##jiancha station
    all_object = UserInfor.objects.all()
    num = len(all_object)
    if num == 1:
        sta = UserInfor.objects.values().last()

        station = sta['station']
        if station == 1:
            dir_image = UserInfor.objects.values().last()
            image_str = dir_image['photo']
            image = base64.b64decode(image_str)
            file = open('image/test.jpg', 'wb')
            file.write(image)
            file.close()

            file_path = 'image/test.jpg'
            image = image_loader.load_image(bUseBGRToEngine, file_path)
            image_feature = do_face_feature(fd_engine, ff_engine, image)
```

```
                file_ku_path = glob.glob(r"image_ku/*.jpg")
                len_ku = len(file_ku_path)
                image_ku = [0] * len_ku
                for i in range(len_ku):
                    # image_ku[i] = image_loader.load_image(bUseBGRToEngine, file_ku_path[i])
                    image_ku[i] = image_loader.load_image(bUseBGRToEngine,
"image_ku/{}.jpg".format(i+1))

                my_score = [0] * len_ku
                zhixingdu = [0] * len_ku
                for i in range(len_ku):
                    temp_feature = do_face_feature(fd_engine, ff_engine, image_ku[i])
                    my_score[i] = do_face_recognition(ff_engine, temp_feature, image_feature)
                    # print(my_score[i].value)
                    zhixingdu[i] = my_score[i].value

                grade = max(zhixingdu)
                weizhi = zhixingdu.index(max(zhixingdu))
                if (grade > 0.7):
                    print('yes')
                    meg = 'y'
                    temp2 = sudo.objects.values('name').get(id=weizhi)
                    name = temp2['name']
                    staff.objects.create(user_name=name)
                    tcp_client_socket.send(meg.encode())
                    UserInfor.objects.all().delete()

                else:
                    print('no')
                    UserInfor.objects.all().delete()

            else:
                pass
        else:
            time.sleep(0.1)

    ASAE_FSDK_UninitAgeEngine(fa_engine)
    ASGE_FSDK_UninitGenderEngine(fg_engine)
    AFD_FSDK_UninitialFaceEngine(fd_engine)
    AFR_FSDK_UninitialEngine(ff_engine)


if __name__ == '__main__':
    main()
```

核心程序 2：

```
# include "MICO.h"
# include "SocketUtils.h"
# include "tcp_server.h"
# include "camera_data_queue.h"
# include "bsp_sdram.h"
# include "bsp_beep.h"
# include "stm32f4xx.h"

# define fire_demo_log(M, ...) custom_log("WIFI", M, ##__VA_ARGS__)

// 路由名
const
char
SSID_NAME[] = "llhj";
// 路由密码
const
char
SSID_KEY[] = "77777777";

static
mico_semaphore_t
wait_sem = NULL;

static
void
micoNotify_ConnectFailedHandler(OSStatus
err, void * inContext)
{
    fire_demo_log("join Wlan failed Err: %d", err);
}

static
void
micoNotify_WifiStatusHandler(WiFiEvent
event, void * inContext)
{
    switch(event)
{
    case
NOTIFY_STATION_UP:
fire_demo_log("Station up");
mico_rtos_set_semaphore( & wait_sem );
break;
case
```

```
NOTIFY_STATION_DOWN:
fire_demo_log("Station down");
break;
default:
break;
}
}

OSStatus
wifi_station(void)
{
    OSStatus
err = kNoErr;
network_InitTypeDef_adv_st
wNetConfigAdv = {0};

MicoInit(); // 初始化网络协议栈

               / * Register
user
function
when
wlan
connection
status is changed * /
          err = mico_system_notify_register(mico_notify_WIFI_STATUS_CHANGED, (void *)
micoNotify_WifiStatusHandler, NULL );
require_noerr(err, exit);

/ *Register
user
function
when
wlan
connection is faile in one
attempt * /
err = mico_system_notify_register(mico_notify_WIFI_CONNECT_FAILED, (void *)
micoNotify_ConnectFailedHandler, NULL );
require_noerr(err, exit);

cli_init();
mico_rtos_init_semaphore( & wait_sem, 1);

/ *Initialize
wlan
parameters * /
```

```
strcpy((char *)
wNetConfigAdv.ap_info.ssid, SSID_NAME); / *wlan
ssid
string * /
strcpy((char *)
wNetConfigAdv.key, SSID_KEY); / *wlan
key
string or hex
data in WEP
mode * /
wNetConfigAdv.key_len = strlen(SSID_KEY); / *wlan
key
length * /

wNetConfigAdv.ap_info.security = SECURITY_TYPE_AUTO; / *wlan
security
mode * /
wNetConfigAdv.ap_info.channel = 0; / *Select
channel
automatically * /
wNetConfigAdv.dhcpMode = DHCP_Client; / *Fetch
Ip
address
from DHCP server * /
wNetConfigAdv.wifi_retry_interval = 100; / *Retry
interval
after
a
failure
connection * /

/ * Connect
Now! * /
fire_demo_log("connecting to %s...", wNetConfigAdv.ap_info.ssid);
micoWlanStartAdv( & wNetConfigAdv);

/ *Wait
for wlan connection * /
                mico_rtos_get_semaphore( & wait_sem, MICO_WAIT_FOREVER );
fire_demo_log( "wifi connected successful" );

exit:
    return err;
}

extern
```

```
CircularBuffer
cam_circular_buff;

int
application_start(void)
{
char
temp[10] = {0};
uint8_t
len = 0;
char * buf = NULL;
/ ** ** ** ** ** *llhj ** ** ** * /
BEEP_GPIO_Config();
/ ** ** ** ** ** *llhj ** ** ** * /
camera_data * cambuf;
OSStatus
err = kNoErr;

SDRAM_Init();

err = wifi_station();
require_noerr(err, exit);

err = camera_queue_init();
require_noerr(err, exit);

cambuf = cbWrite( & cam_circular_buff);
require_noerr( !cambuf, exit ); // cambuf = NULL 时表示有错误

err = open_camera((uint32_t *)
cambuf->head, CAMERA_QUEUE_DATA_LEN);

require_noerr(err, exit);

err = mico_rtos_create_thread(NULL, MICO_APPLICATION_PRIORITY, "TCP_server", tcp_server_thread,
0x1000, NULL);
require_noerr_string(err, exit, "ERROR: Unable to start the tcp server thread.");

while (1)
    {
    }
exit:
mico_rtos_delete_thread(NULL);
return err;
}
```

核心程序 3：

```
# include "MICO.h"

# include "camera_data_queue.h"
# include "bsp_beep.h"

extern
CircularBuffer
cam_circular_buff;

# define tcp_server_log(M, ...) custom_log("TCP", M, ##__VA_ARGS__)

# define SERVER_PORT 20000 /*set up a tcp server,port at 20000*/

# define NO_USED_BUFF_LEN    (1024)
# define TCP_MAX_SEND_SIZE   (1024*50)

OSStatus
jpeg_send(int
fd, const
uint8_t * inBuf, size_t
inBufLen )
{
    OSStatus
err = kParamErr;
ssize_t
writeResult;
int
selectResult;
size_t
numWritten;
fd_set
writeSet;
struct
timeval_t
t;

require(fd >= 0, exit);
require(inBuf, exit);
require(inBufLen, exit);

err = kNotWritableErr;

t.tv_sec = 5;
t.tv_usec = 0;
numWritten = 0;
```

```
while (numWritten < inBufLen)
    {
        FD_ZERO( & writeSet );
    FD_SET(fd, & writeSet );

    selectResult = select(fd + 1, NULL, & writeSet, NULL, & t );
    require(selectResult >= 1, exit);

    if (FD_ISSET(selectResult, & writeSet))
    {
    writeResult = write( fd, (void * )( inBuf + numWritten ), ( inBufLen - numWritten ) );
    require( writeResult > 0, exit );

    numWritten += writeResult;
    }
    }

    require_action(numWritten == inBufLen, exit,
                   tcp_server_log("ERROR: Did not write all the bytes in the buffer. BufLen: %zu,
Bytes Written: %zu",
                                  inBufLen, numWritten);
    err = kUnderrunErr );

    err = kNoErr;

exit:
return err;
}

OSStatus
jpeg_tcp_send(int
fd, const
uint8_t * inBuf, size_t
inBufLen )
{
uint32_t
i = 0, count = 0, index = 0;
OSStatus
err = kGeneralErr;

count = inBufLen / TCP_MAX_SEND_SIZE;

for (i = 0; i < count; i ++)
{
    err = jpeg_send(fd, inBuf + index, TCP_MAX_SEND_SIZE);
```

```
require(err == kNoErr, exit);
index = index + TCP_MAX_SEND_SIZE;
}

if ((inBufLen % TCP_MAX_SEND_SIZE) != 0)
    {
        err = jpeg_send(fd, inBuf + index, inBufLen % TCP_MAX_SEND_SIZE);
    require(err == kNoErr, exit);
    }

    exit:
    return err;
    }


    void
    jpeg_socket_close(int * fd)
    {
    int
    tempFd = *fd;
    if (tempFd < 0)
        return;
    *fd = -1;
    close(tempFd);
    }

    void
    Delay_beep(__IO
    uint32_t
    nCount)
    {
    for (;nCount != 0;nCount--);
    }

    void
    tcp_server_thread(void * arg)
    {
    OSStatus
    err = kNoErr;
    struct
    sockaddr_t
    server_addr, client_addr;
    socklen_t
    sockaddr_t_size = sizeof(client_addr);
    char
    client_ip_str[16];
```

```
int
tcp_listen_fd = -1, client_fd = -1;
IPStatusTypedef
para;
int32_t
camera_data_len = 0, i = 0;
uint8_t * in_camera_data = NULL;
uint8_t
packet_index = 0;
uint8_t * no_used_buff = NULL;
int
len = 0;
char
buf[5];
micoWlanGetIPStatus( & para, Station);
tcp_server_log("TCP server ip:%s, port:%d", para.ip, SERVER_PORT);

tcp_listen_fd = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP);
require_action(IsValidSocket(tcp_listen_fd), exit, err=kNoResourcesErr);

server_addr.s_ip = INADDR_ANY; / *Accept
conenction
request
on
all
network
interface * /
server_addr.s_port = SERVER_PORT; / *Server
listen
on
port: 20000 * /
err = bind(tcp_listen_fd, & server_addr, sizeof(server_addr) );
require_noerr(err, exit);

err = listen(tcp_listen_fd, 0);
require_noerr(err, exit);

no_used_buff = malloc(NO_USED_BUFF_LEN);
memset(no_used_buff, 0, NO_USED_BUFF_LEN);

require_noerr(err, exit);

while (1)
    {
        client_fd = accept(tcp_listen_fd, & client_addr, & sockaddr_t_size );
    if (IsValidSocket(client_fd))
```

```
        {
        inet_ntoa( client_ip_str, client_addr.s_ip );
        tcp_server_log( "TCP Client %s:%d connected, fd: %d", client_ip_str, client_addr.s_port,
client_fd );

    while (1)
        {

            len = recv(client_fd, buf, sizeof(buf), 0);
        tcp_server_log("fd: %d, recv data %c from client", client_fd, buf[0]);
        if (buf[0] == 'y')
        {
        tcp_server_log("yes");
        BEEP_ON;
        Delay_beep(0x2FFFFFF);
        // 13S
        BEEP_OFF;
        }
        else if (buf[0] == 'n') tcp_server_log("no");
        buf[0] = NULL;
        }

        tcp_server_log("TCP Client disconnect %s:%d connected, fd: %d", client_ip_str,
client_addr.s_port, client_fd);

        jpeg_socket_close( & client_fd );
        }
        }

        exit:
        if (err != kNoErr)
        {
        tcp_server_log( "Server listerner thread exit with err: %d", err );
        }

        jpeg_socket_close( & tcp_listen_fd );

        mico_rtos_delete_thread(NULL );
        }
```