《Python 语言课程设计》报告

题目：人脸识别软件

专业：计算机科学与技术

班级：20 计算机 1 班

学号：2000310212

姓名：郑钧元

指导教师：叶敏超

完成日期： 2022 年 2 月 21 日

人脸识别软件

1 背景概述

随着社会进入自动化，人脸识别自动考勤系统应运而生。在一些需要打卡签到的环境下，人脸扫描后自动打卡变成了主流的打卡方式，代替了手动签到，并且防止了代替签到这种可能性的出现。所以通过借助这次课程设计的机会，我用 Python 设计开发了一款“人脸识别自动考勤系统”与“人脸识别自动考勤后台系统”。

2 系统设计

2.1 系统设计

1. 人脸识别自动考勤系统
该模块由 4 个部分组成：LCD 时钟、日历、点击签到、添加用户
Ⅰ.LCD 时钟：显示当前时间
Ⅱ.日历：展示日历，自动锁定当前日期
Ⅲ.点击签到：点击后拍照上传打卡
Ⅳ.添加用户：注册

2. 人脸识别自动考勤后台系统
该模块主要由 4 个部分组成：人数统计、打卡情况展示、刷新按钮、退出按钮
Ⅰ.人数统计：展示在职人员、已打卡、未打卡的人数
Ⅱ.打卡情况展示：分别展示已打卡与未打卡的名单
Ⅲ.刷新按钮：更新数据
Ⅳ.退出按钮：退出

2.2 模块详细设计

业务逻辑：
Ⅰ.人脸识别自动考勤系统
1. 点击“点击签到”按钮，开启摄像头拍照并保存至本地，将照片 URL 地址传到百度云接口，百度云会返回给用户一个 json 格式的数据。Json 数据中包括 score（用户的匹配得分），user_id（用户的 user_id）等等
   A. 如果被识别的用户已经注册（即百度云人脸库中已经有该用户的照

片），页面弹窗：“XXX（用户的姓名），打卡成功”。

B. 如果被识别的用户没有注册，那么返回的 score 一定是远远小于 80 的，通过 score 来设限，页面弹窗：“您还没有注册，请先注册！”

C. 已经打卡过的在 20 秒内继续打卡（为方便演示将打卡更新时间设置为 20 秒），则会返回“XXX（用户的姓名）,您今天已经打过卡了”。

D. 如果在拍照视野内没有发现人脸，则页面弹窗：“没有检测到人脸”。

2. 点击“添加用户”按钮，跳转到注册界面，填写姓名与 UserId，再点击“录入人脸”，将图片的 URL 传给百度云接口进行注册。

A. 如果在拍照视野内没有发现人脸，则页面弹窗：“没有检测到人脸”。

B. 如果已经注册的用户继续录入人脸，则页面弹窗：“您已经在系统中了，继续录入人脸将提高识别准确度！”

C. 如果用户填入了已经存在的 UserId，则页面弹窗：“该用户 ID 已被占用，请更换！”

Ⅱ.人脸识别自动考勤后台系统

在数据库中有一个 role 字段，0 代表不是管理员，1 代表是管理员，只有 role=1 的用户才可以登录到人脸识别自动考勤后台系统中。关于数据展示，即向数据库中按条件检索数据。刷新按钮功能的实现为再次连接数据库进行相同的操作。

功能模块图：

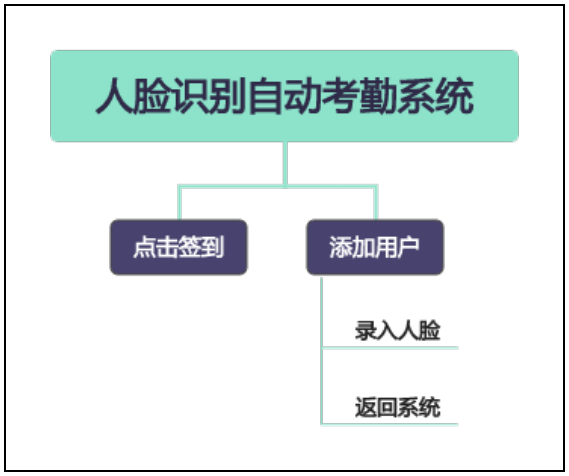

图 1 人脸识别自动考勤系统功能模块图

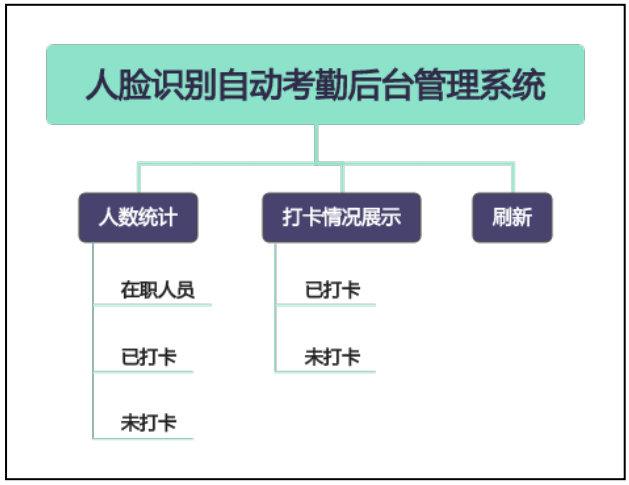

图 2 后台管理系统功能模块图

程序流程图：

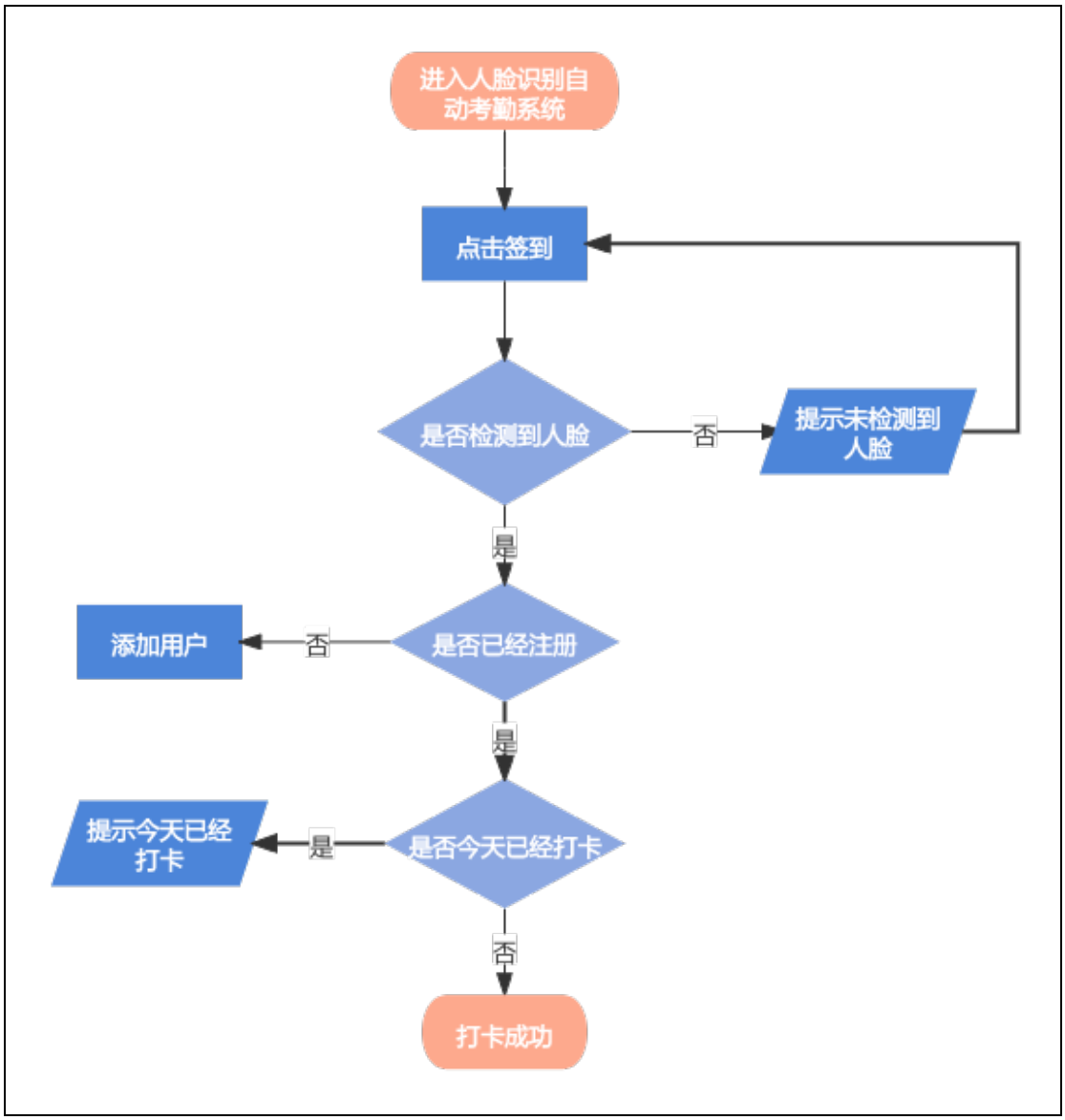

图 3 点击打卡程序流程

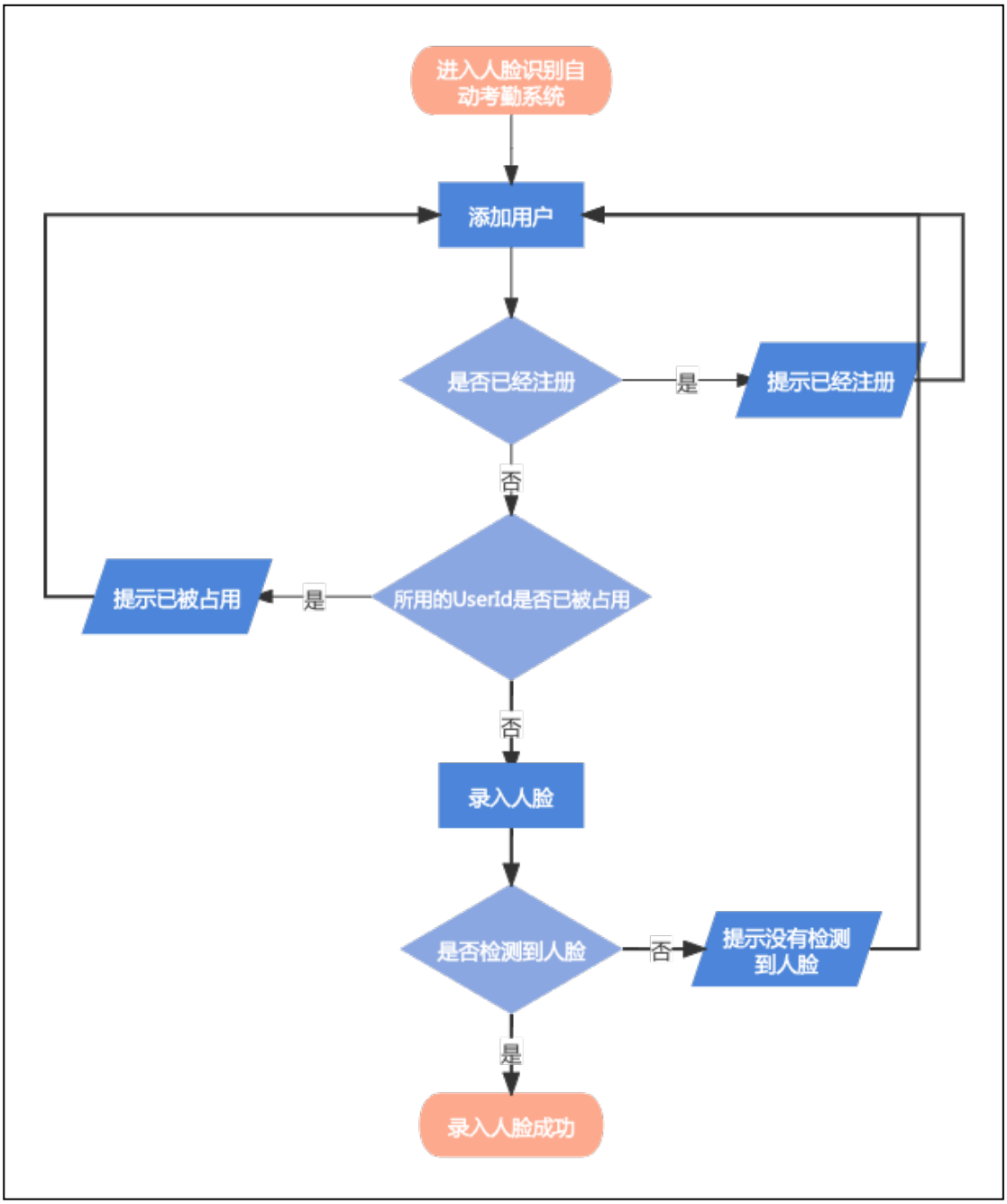
进入人脸识别自动考勤系统
添加用户
是否已经注册
是
提示已经注册
否
所用的UserId是否已被占用
是
提示已被占用
否
录入人脸
是否检测到人脸
否
提示没有检测到人脸
是
录入人脸成功

图 4 添加用户程序流程

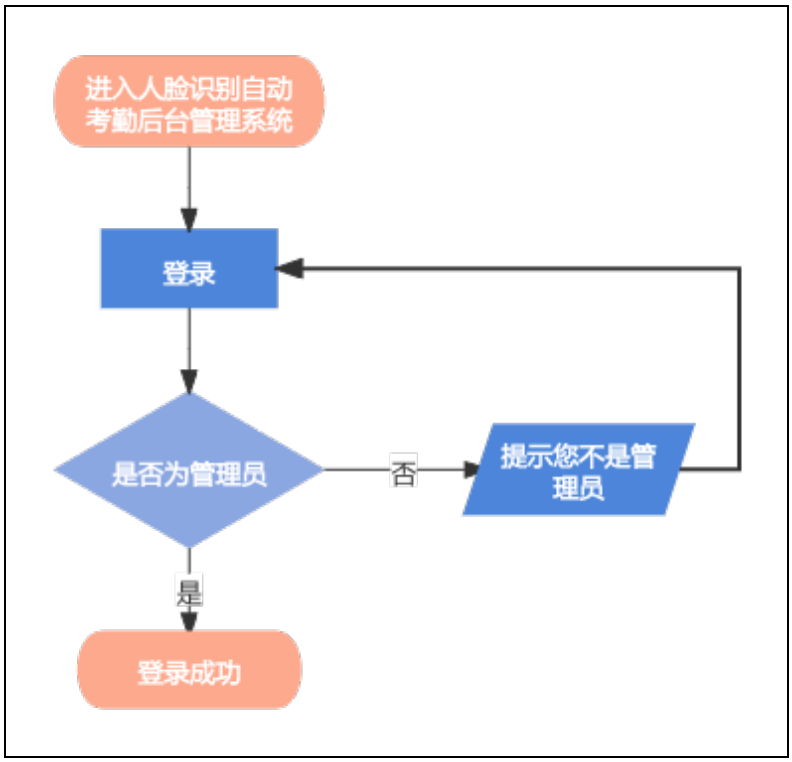

图 5 后台管理登录程序流程

3 技术实现细节

3.1 开发环境

开发硬件：1.4 GHz 四核 Intel Core i5
系统环境：macOS Monterey 12.1
Python 版本：Python 3.8
使用的模块：uuid,base64,json,pymysql,requests,cv2,PyQt5,sys
集成开发环境:PyCharm 2020.3

3.2 界面设计

图 6 人脸识别自动考勤系统主界面

在主界面上除了两个功能按钮外，还有 LCD 时钟与日历，方便使用者查看当时日期与时间。

图 7 添加用户界面

图 8 后台管理系统登录界面

中间的 logo 来自阿里矢量库，点击登录按钮后进行身份认证，只有管理员可以登录到后台系统。

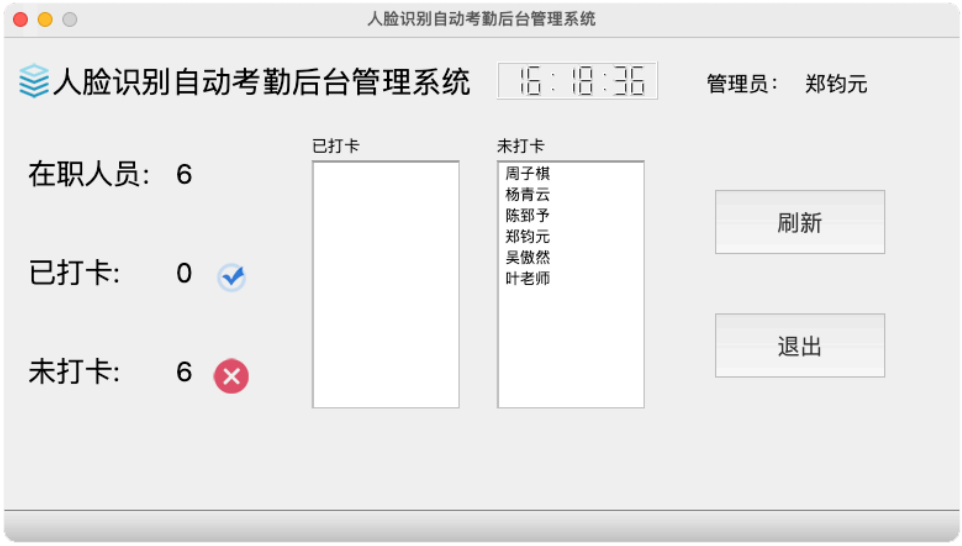

图 9 后台管理系统界面

该模块主要由 4 个部分组成：人数统计、打卡情况展示、刷新按钮、退出按钮。右上方管理员姓名为登录者姓名，为动态 label。

3.3 功能编程

3.3.1 拍照

核心代码：

```
def take_photo(self):
    cap = cv2.VideoCapture(0)
    url_pre = "/Users/zhengjunyuan/Desktop/face_test/"
    url_aft = str(uuid.uuid4()) + '.jpeg'
    url = url_pre + url_aft
    while True:
        ret, frame = cap.read()
        cv2.imshow("FaceRecognition", frame)
        if cv2.waitKey(1) & 0xFF == ord('p'):
            cv2.imwrite(url, frame)
            break
    cap.release()
    cv2.destroyAllWindows()
    self.name = self.lineEdit.text()
    self.userId = self.lineEdit_2.text()
    return url
```

说明：在 python 中导入 opencv、uuid 模块，调用 uuid.uuid4()就可以生成随机字符串，通过 url_pre + url_aft 拼接为图片的 url，拍照完毕后 take_photo()将会返回这个 url。cv2.waitKey(1) & 0xFF == ord('p')是在对键盘操作的监听，按下按键 p 就会拍照。

3.3.2 点击签到

核心代码：

```
client_id = "N5GHjXHSFQHtVMOyrHWSK1o4"
client_secret = "nIDUqoQq3446Z15Gz8A8d5K7mai1b4ve"
token_url = "https://aip.baidubce.com/oauth/2.0/token"

host=f"{token_url}?grant_type=client_credentials&client_id={clien
t_id}&client_secret={client_secret}"
response = requests.get(host)
access_token = response.json().get("access_token")
request_url = "https://aip.baidubce.com/rest/2.0/face/v3/search"
with open(url, 'rb') as f:
    image = base64.b64encode(f.read())
body = {
       "image": image,
       "image_type": "BASE64",
       "group_id_list": "N1",
       "quality_control": "NONE",
       "liveness_control": "NONE",
      }
headers = {"Content-Type": "application/json"}
request_url = f"{request_url}?access_token={access_token}"
response = requests.post(request_url, headers=headers, data=body)
result = json.loads(response.content.decode("UTF-8"))
```

说明：client_id 为官网获取的 AK，client_secret 为官网获取的 SK，在个人中心中可以找到，用户可以通过 AK 与 SK 获取 access_token，access_token 有效期为 30 天。百度云会返回一个 json 格式的 result 字符串，通过 python 的内置模块 json 中的 json.loads()方法就可以操作相应数据。主要使用 result 下的“user_id”“score”。

返回示例：

```
{
  "face_token": "fid",
  "user_list": [
     {
        "group_id" : "test1",
        "user_id": "u333333",
        "user_info": "Test User",
        "score": 99.3
     }
  ]
}
```

3.3.3 添加用户

核心代码：

```
client_id = "N5GHjXHSFQHtVMOyrHWSK1o4"
client_secret = "nIDUqoQq3446Z15Gz8A8d5K7mai1b4ve"
token_url = "https://aip.baidubce.com/oauth/2.0/token"
host=f"{token_url}?grant_type=client_credentials&client_id={clien
t_id}&client_secret={client_secret}"
response = requests.get(host)
access_token = response.json().get("access_token")
request_url="https://aip.baidubce.com/rest/2.0/face/v3/faceset/us
er/add"
with open(url, 'rb') as f:
    image = base64.b64encode(f.read())
    body = {
       "image": image,
       "image_type": "BASE64",
       "group_id": "N1",
       "user_id": userId,
       "quality_control": "NONE",
       "liveness_control": "NONE",
     }

 headers = {"Content-Type": "application/json"}
 request_url = f"{request_url}?access_token={access_token}"
 response = requests.post(request_url, headers=headers, data=body)
 result = json.loads(response.content.decode("UTF-8"))
```

说明：基本代码相同，仅更换下 request_url 的请求地址。

3.3.3 展示用户数量

核心代码:

```
 ver_sql = "SELECT COUNT(userId) FROM userAndphoto"
 cursor1.execute(ver_sql)
 num = cursor1.fetchall()
 self.allClerk = num[0][0]
 cursor2 = db.cursor()
 has_sql = "SELECT COUNT(userId) FROM userAndphoto WHERE pc = 1"
 cursor2.execute(has_sql)
 has_num = cursor2.fetchall()
 self.has = has_num[0][0]
 self.not_has = self.allClerk - self.has

 self.label_6.setText(_translate("MainWindow",
"{}".format(self.allClerk)))
 self.label_7.setText(_translate("MainWindow",
```

```
"{}".format(self.has)))
    self.label_8.setText(_translate("MainWindow",
"{}".format(self.not_has)))
```

说明:这里写了两条 sql 语句对数据库数据进行查询,"SELECT COUNT(userId) FROM userAndphoto"用来查询所有用户的数量，然后在 PyQt5 中将 label_6 的数据用用户的数量填充。"SELECT COUNT(userId) FROM userAndphoto WHERE pc = 1"用来查询已打卡的用户数量，self.allClerk - self.has 通过总数减已打卡的用户数量可以得出未打卡用户的数量，最后也同样把数据填充到 label 标签中。

3.3.4 用户列表展示

核心代码:

```
cursor3 = db.cursor()
whoNot_sql = "SELECT name FROM userAndphoto WHERE pc = 0"
cursor3.execute(whoNot_sql)
whoNot_list = cursor3.fetchall()
for i in range(len(whoNot_list)):
    self.list.append(whoNot_list[i][0])
self.listView.setObjectName("listView")
listModel = QStringListModel()
listModel.setStringList(self.notList)
```

说明："SELECT name FROM userAndphoto WHERE pc = 0"查询到未打卡的用户，然后 python 会得到一个具有用户姓名的元组，将元组转换为列表后传给 PyQt5,就可以在列表中呈现出用户姓名。

3.4 创新点或难点的技术解决方案

1.UI 界面制作麻烦，无法在短时间内写出规模较大的 UI 界面。

解决方案：使用 PyQt5 进行 UI 程序编写，使用用 QtDesigner 来设计 UI 界面，设计好 UI 界面后可以通过终端命令 python -m PyQt5.uic.pyuic xxx.ui -o xxx.py 自动在项目目录下生成对应的.py 文件，效率大大提升。

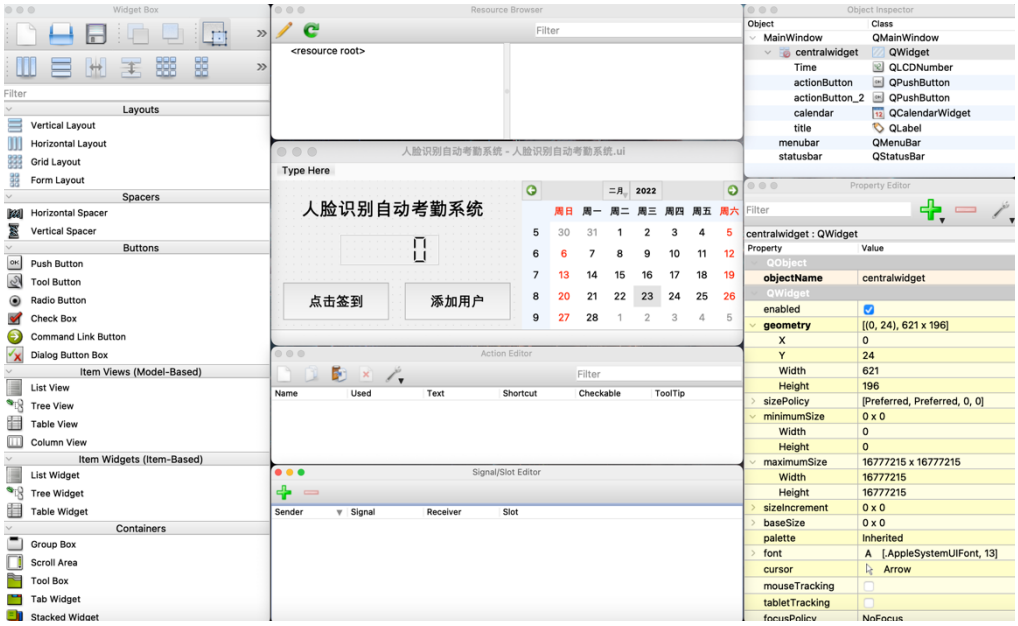

图 10 QtDesigner 界面

2. 人脸识别的实现与防止用户使用照片进行打卡

解决方案：使用百度云人脸搜索与库管理，可以往百度的人脸库中添加人脸，也可以通过接口的调用实现人脸的搜索，而且百度云人脸搜索与库管理的文档非常详细易懂，方便开发者开发。为防止防止用户使用照片进行打卡，百度云的人脸识别还提供了活体检测，照片不是活体，可以被检查出来，返回相应的数据可以防止这一情况的发生。拍照功能调用 OpenCV 的拍照方法。

3. 百度云的人脸库用户 Id 无法使用中文，导致页面中返回的用户姓名无法使用中文，而且每天的打卡情况无法保存，特定时间内可无限次打卡。

解决方案：使用 MySQL 数据库进行数据的保存，给百度云的人脸库用户 Id 对应相应的人名，并且添加是否打卡的逻辑字段，0 为未打卡，1 为已打卡。并给 MySQL 设置定时器事件，为方便演示将逻辑字段的更新为 0 的事件定为 20 秒一次。

4. 拍照后照片文件名的生成

解决方案：我使用了 uuid 模块，uuid 能够随机生成字符串，作为文件名的前缀，后缀为.jpeg，通过字符串拼接生成文件名。

例：58528bce-64e6-4484-a40b-be40e5ed8d75.jpeg

4 测试

4.1 黑盒测试

测试需求：用户在注册时，需要填写姓名、UserId，再进行拍照注册。没检测到人脸时，提示“未检测到人脸”。填写的姓名与 UserId 已经存在，提示“您已经在系统中，继续录入人脸会提高识别准确度”。填写的姓名不存在但是 UserId 存在，提示“UserId 已被占用，请更换”。测试用例如下：

表 1 黑盒测试测试用例

测试用例 Id	测试场景	测试步骤	预期结果	实际结果
Test_01	未检测到人脸	点击“录入人脸”拍照	添加失败，提示：“未检测到人脸”	添加失败，提示：“未检测到人脸”
Test_02	成功检测到人脸但是填写的姓名与 UserId 已经存在	点击“录入人脸”拍照	添加失败，提示：“您已经在系统中，继续录入人脸会提高识别准确度”	添加失败，提示：“您已经在系统中，继续录入人脸会提高识别准确度”
Test_03	成功检测到人脸但是填写的姓名不存在但是 UserId 存在	点击“录入人脸”拍照	添加失败，提示：“UserId 已被占用，请更换”	添加失败，提示：“UserId 已被占用，请更换”
Test_04	成功检测到人脸并且填写的姓名与 UserId 均不存在	点击“录入人脸”拍照	添加成功	添加成功

4.2 白盒测试

测试需求：用户进行打卡时，需对人脸拍照。没检测到人脸时，提示“未检测到人脸”；被识别的用户未注册，提示“您还没有注册，请先注册！”；被识别的用户在 20 秒内已经打过卡，提示“您今天已经打过卡了！”。测试用例如下:

表 2 白盒测试测试用例

测试用例 Id	测试场景	测试步骤	预期结果	实际结果
Test_01	成功检测到人脸且以注册且未打卡	点击“点击签到”拍照	打卡成功	打卡成功

Test_02	成功检测到人脸但是未注册	点击“点击签到”拍照	打卡失败，提示：“您还没有注册，请先注册！”	打卡失败，提示：“您还没有注册，请先注册！
Test_03	成功检测到人脸且已注册但是 20 秒内已经打过卡	点击“点击签到”拍照	打卡失败，提示：“您今天已经打过卡了！”	打卡失败，提示：“您今天已经打过卡了！”
Test_04	未检测到人脸	点击“点击签到”拍照	打卡失败，提示：“未检测到人脸”	打卡失败，提示：“未检测到人脸”

4.3 bug 与解决方案

Bug1:增加“您已经在系统中，继续录入人脸会提高识别准确度”提示后，系统无法正常注册新用户。

解决：调整 sql 语句顺序以及 if,elif 的顺序解决。

Bug2:因不可知的系统兼容性问题，tk 模块无法正常导入，导致无法制作 UI 界面。

解决：更换为 PyQt5 配合 QtDesigner 进行 UI 设计，更加高效。

5 总结与心得体会

课程设计总结：本次 Python 课程设计，完成了“人脸识别自动考勤系统”与“人脸识别自动考勤后台系统”。使用了 macOS 下的 Python 3.8 环境。

完成的功能：人脸识别签到与登录、人脸扫描注册、数据刷新、可视化打卡情况名单与数量展示。

主要掌握的技术：MySQL、百度云人脸识别、OpenCV 拍照、uuid 随机字符串生成、PyQt5、QtDesigner 工具

解决的技术难题：人脸识别、字符串随机生成、摄像、UI 界面快速设计

设计的（功能/技术）亮点：1.使用 QtDesigner 工具使 UI 界面快速成型 2.uuid 随机字符串生成 3.后台管理数据可随时刷新，数据可视化让用户体验更好 4.页面上有电子时钟与日历，方便使用者查看。

软件需要改进的地方：1.用户可以不拍照，系统自动抓拍人脸 2.有两个人在摄像头视野中时，系统需要自动识别出哪个是主要人物 3.系统可以做到实时更新，无需用户自己点击刷新按钮 4.页面可以再好看点

学会了使用什么 Python 的第三方模块:uuid,PyQt5,cv2,json,base64

参考文献

[1]王英英.MySQL8 从入门到精通[M].北京:清华大学出版社,2019:191.
[2]杨连贺.Python 程序设计实用教程[M].北京:清华大学出版社 2018:205
[3]百度云:人脸识别 API 文档人脸搜索与库管理，百度云官方网站，2022 年 1 月 05 日，https://cloud.baidu.com/doc/FACE/s/Gk37c1uzc。

教师评语
教师签名：