華中科技大學

软件课程设计报告 (个人)

基于人脸识别技术门禁考勤系统设计与开发

院　　系　电子信息与通信学院

专　　业　电子信息工程

班　　级　提高 2101 班

姓　　名　杨筠松

学　　号　U202115980

指导教师　刘文予

日　　期　2024 年 3 月 2 日

2024 年 3 月 2 日

目　　录

1 项目描述

1.1 项目背景

1.1.1 门禁系统

门禁系统，作为一种重要的安全控制手段，已经成为现代社会不可或缺的一部分。它通过限制未经授权的人员进入特定区域，来保护人员、资产和信息的安全。随着技术的不断进步和安全需求的日益增长，门禁系统从简单的物理锁和钥匙演变为今天高度集成的电子安全解决方案。早期的门禁系统主要依赖于传统的机械锁和钥匙，这种方式简单直接，但存在着安全隐患，比如钥匙的丢失或复制。随着电子技术的发展，电子卡（如磁卡、接触式和非接触式智能卡）开始被广泛应用于门禁系统中，提高了安全性和便捷性。进一步地，为了应对更高级别的安全威胁，生物识别技术（包括指纹识别、面部识别、虹膜识别等）被引入门禁系统，实现了更加严格和个性化的安全验证。

门禁系统的应用非常广泛，几乎涵盖了所有需要安全管理的场所：

1. 办公楼和商业中心：管理员可以精确控制员工和访客的进出，保护企业资产和信息安全。
2. 住宅小区：为居民提供安全便捷的生活环境，防止非法入侵。
3. 学校：保护学生和教职工的安全，管理校园的进出口。
4. 政府机构和重要基础设施：确保关键区域的安全，防止未授权访问。

1.1.2 人脸识别

人脸识别技术是一种先进的生物识别技术，它通过分析和比较个人的面部特征来识别和验证身份。这种技术在门禁系统中的应用尤为广泛，因为它能够提供一种安全、便捷且非侵入式的身份验证方法。人脸识别系统首先利用相机或其他成像设备捕捉到个人的面部图像，然后通过复杂的算法分析图像中的面部特征，如眼睛、鼻子、嘴巴的位置和形状，以及脸型等。这些特征被转换成一个数字代码，即面部的唯一“指纹”。

随后，系统将这个“指纹”与数据库中预先存储的面部数据进行比对。如果找到了匹配项，身份验证就成功了，门禁系统将允许用户进入。这一过程几乎是瞬间完成的，极大地提高了进出效率。与传统的身份验证方法相比，如密码或物理钥匙，人脸识别技术不仅减少了被复制或盗用的风险，还因其非接触式的特点，在提高用户体验的同时，也降低了传染病传播的可能性。

人脸识别技术的另一个优点是其自适应性强，能够在不同的光照条件下准确识别面部，甚至在部分面部被遮挡的情况下也能工作。随着人工智能和机器学习技术的发展，人脸识别技术的准确性和安全性正在不断提高，使其成为门禁系统中越来越受欢迎的身份验证选项。

1.1.3 人脸跟踪

人脸跟踪技术是基于人脸识别技术进一步发展的，它不仅能够识别出图像中的人脸，还能在视频流中实时追踪移动的人脸。这种技术通过分析视频帧之间的连续变化，精确捕捉并跟踪面部在空间中的移动轨迹。人脸跟踪的关键挑战在于如何在各种环境条件下（如不同光照、面部遮挡、不同角度和表情变化）准确地识别和追踪人脸。现代人脸跟踪算法利用机器学习和深度学习技术，通过大量的数据训练，提高了在复杂环境下的识别准确性和跟踪稳定性。

人脸跟踪技术的应用非常广泛，它不仅可以用于安全监控，提高公共场所的安全水平，还可以应用于人机交互、智能广告、虚拟现实等领域，为用户提供更加个性化和互动的体验。在安全监控领域，人脸跟踪技术能够帮助监控系统实时追踪特定目标，即使在人群中也能快速定位，大大提高了安全管理的效率和响应速度。此外，结合人脸识别技术，人脸跟踪还能实现更加精准的身份验证，为安全领域带来了革命性的改变。

1.1.4 心率估计

心率估计技术通过分析人脸视频图像中的微小颜色变化来无接触地监测个体的心率，这一技术基于远程光电容积描记术（rPPG）。rPPG 技术利用相机捕捉到的面部图像中血液流动引起的微小颜色变化，通过特定算法处理这些信息，从而估计出心率。这种技术的优势在于其非侵入性，不需要与皮肤直接接触，就能实时监测心率，适用于需要快速、无干扰监测生理状态的场合。

在门禁系统中引入心率估计技术，可以为身份验证系统增加一层生物识别保护。与传统的生物识别技术如指纹或虹膜扫描相比，心率估计提供了一种更为隐蔽的验证方式，增强了安全性。此外，这种技术还可以用于健康监测和紧急情况预警，例如，通过持续监测进入特定场所人员的心率，可以及时发现健康异常，采取必要的预防或救治措施。

心率估计技术的挑战主要在于如何提高测量的准确性和稳定性，尤其是在不同光照条件和面部表情变化时。随着图像处理和人工智能技术的发展，心率估计的准确度和应用场景都在不断扩展，预计未来将在门禁系统以及更广泛的健康监测领域发挥重要作用。

1.2 系统描述

1.2.1 系统功能

基于人脸识别的门禁系统设计旨在通过利用面部识别技术，实现对于特定场所的安全和便捷的访问控制。系统的主要功能包括：

- 用户注册：允许新用户通过录入个人信息和面部特征来注册系统。
- 面部信息录入：为系统中注册的用户录入其面部识别信息，以便后续的识别验证。

- 面部识别验证：通过摄像头捕捉访问者的面部图像，并与数据库中存储的面部信息进行匹配，以验证其访问权限。
- 权限管理：根据用户的不同角色和权限，控制其访问特定区域的能力。
- 用户心率估计：根据用户心率此类生物信息来完成对访客身份的认证。

1.2.2 系统架构

本系统采用模块化的设计理念，主要分为以下几个部分：

- 人机交互界面：设计良好的用户界面，确保用户能够方便快捷地进行操作，包括注册、登录和访问请求等。
- 开发工具和语言选择：根据项目需求选择合适的开发工具和编程语言，以支持高效的系统开发和维护。
- 容错性和可扩展性：确保系统能够处理各种异常情况，同时支持后续功能的扩展和升级。
- 算法优化：优化面部识别算法，确保其在时间和空间复杂度上满足实际应用需求，提高系统的响应速度和准确率。
- 代码管理和备份：在开发过程中注意代码的版本管理和备份，避免因代码整合等问题导致的数据丢失或系统故障。

1.3 模型描述

1.3.1 前端模型

前端界面主要包括以下几个部分：

- 侧边栏（Sidebar）：提供主菜单、ID 输入、查看和删除人脸信息、更改密码和脉搏检测的选项。
- 考勤列表（Attendance Frame）：显示考勤记录。
- 注册新成员区域（Register Frame）：用于注册新成员。
- 操作区域（Option Frame）：包括进行考勤和清除历史记录的操作按钮。
- 时间显示区域（Time Frame）：展示当前时间。

用户可以通过 GUI 进行一系列操作，如注册新的人脸信息、查看和删除人脸数据、进行考勤等。

1.3.2 后端模型

后端模型核心是 Control 类和 Data 类，其中 Control 类作为应用程序的控制中心，实现了以下主要功能：

- 密码验证
- 图像的获取与删除
- UI 缩放调整
- 考勤记录的生成

该类通过与 Data 类交互，来管理用户和考勤信息。Data 类负责管理所有与考勤系统相关的数据，其主要功能包括：

- 管理人脸数据
- 更新和获取考勤记录
- 维护用户信息

数据存储在 CSV 文件中，通过 Data 类提供的方法进行访问和处理。

1.3.3 算法模型

局部二值模式直方图（LBPH）算法是一种广泛应用于人脸识别领域的算法，以其高效性和对光照变化的鲁棒性而受到青睐。LBPH 算法通过以下步骤对人脸图像进行分析和识别：

1. 将人脸图像划分为若干小的局部区域。
2. 对每个局部区域计算局部二值模式（LBP）特征。
3. 根据 LBP 特征构建每个区域的直方图。
4. 合并各个局部区域的直方图，形成全局特征直方图。
5. 通过比较不同人脸图像的全局特征直方图进行人脸识别。

LBPH 算法具有以下主要特点：

- **鲁棒性**：算法对于光照变化和面部表情变化具有较好的鲁棒性。
- **效率**：由于算法主要基于局部区域的 LBP 特征计算，因此具有较高的计算效率。
- **实时性**：高效的特性使得 LBPH 算法适用于实时的人脸识别应用。

局部二值模式直方图（LBPH）算法以其独特的特点和优势，在人脸识别领域中占有重要地位。其对光照和表情的鲁棒性，以及高效的计算性能，使其成为实时人脸识别应用的理想选择。

2 软件设计

2.1 模块层次

面部识别考勤系统基于三个主要模块构建：前端用户界面 (UI)、后端控制逻辑和数据管理。这些模块协同工作，提供一套完整的考勤管理解决方案。

前端用户界面模块负责与用户直接交互，提供友好的操作界面。它包括以下子模块：

- 侧边栏：实现功能导航。
- 考勤列表：展示考勤记录。
- 注册新成员区：添加面部识别数据。
- 操作区：提供考勤记录和历史记录的管理功能。
- 时间显示区：显示当前时间。

后端控制逻辑模块负责处理前端发起的请求，执行面部识别、数据加密、考勤记录生成等任务。主要包括：

- 面部识别：利用 OpenCV 实现面部识别。
- 数据加密：对敏感信息进行加密处理。
- 考勤管理：生成和管理考勤记录。

数据管理模块负责存储和维护系统中的所有数据，如用户信息、考勤记录等。主要功能包括：

- 数据存储：使用文件系统或数据库存储数据。
- 数据访问：提供接口访问和修改存储的数据。
- 数据维护：更新和清理旧数据。

各模块间通过明确定义的接口进行交互。前端用户界面通过调用后端控制逻辑的 API 执行操作，后端控制逻辑再与数据管理模块交互，以访问和更新所需的数据。本系统的模块层次设计旨在提高软件的可维护性、可扩展性和用户体验。通过将系统划分为明确的模块，我们能够更容易地管理和更新系统的各个部分。

2.2 技术选型

在本项目中，我们精心挑选了一系列技术和工具，以确保软件系统的高性能、可靠性和用户友好性。以下是我们的主要技术选型及其理由：

2.2.1 前端界面：customtkinter

为了提供一个现代化且易于使用的图形用户界面(GUI),我们选择了基于Python的customtkinter库。这个选择基于以下考虑：

- **用户体验**：customtkinter提供了比标准tkinter库更美观、更可定制的控件，可以创建更吸引用户的界面。
- **易于开发**：作为Python的一个扩展库，customtkinter易于学习和使用，有助于加速开发过程。
- **跨平台兼容性**：customtkinter支持Windows、macOS和Linux，确保了应用的广泛可用性。

2.2.2 人脸识别：OpenCV

对于面部识别功能，我们采用了开源库OpenCV中的LBPHFaceRecognizer算法。选择这个技术的理由包括：

- **性能**：LBPHFaceRecognizer以其高效的运算和对光照变化的鲁棒性，提供了快速且可靠的面部识别功能。
- **开源**：OpenCV是一个广泛使用的开源计算机视觉库，拥有强大的社区支持和丰富的学习资源。
- **灵活性**：OpenCV提供了广泛的图像处理功能，可用于未来扩展本项目的其他视觉处理需求。

2.2.3 数据存储：CSV 文件

对于数据管理，我们选择使用CSV文件作为数据存储方案，原因如下：

- **简单性**：CSV文件易于创建和读取，不需要复杂的数据库管理系统，适合快速开发和小规模数据管理。
- **兼容性**：CSV格式被广泛支持，可以轻松导入到Excel等表格软件中进行分析和处理。
- **可移植性**：CSV文件是纯文本格式，确保了数据的可移植性和长期存储。

通过上述技术选型，我们构建了一个高效、可靠且用户友好的面部识别考勤系统。这些技术的选择不仅基于当前的项目需求，也考虑了未来的可扩展性和维护性。

2.3 技术实现

本节详细介绍了面部识别考勤系统的技术实现，包括算法、后端和前端的设计。

2.3.1 算法设计

本系统的核心功能之一是面部识别，这一功能基于 OpenCV 库的 LBPHFaceRecognizer 算法实现。该算法通过分析和比较人脸图像的局部二值模式（LBP），实现对个体面部的识别。算法流程如下：

1. 将输入的人脸图像转换为灰度图像，以简化后续处理。
2. 将灰度图像划分为多个小区域，计算每个区域的 LBP 特征。
3. 对每个区域的 LBP 特征进行直方图统计，形成特征直方图。
4. 将得到的特征直方图与训练集中的图像特征进行比较，以识别人脸。

这种方法的优势在于其对光照变化的鲁棒性，以及相对较低的计算复杂度，适合实时应用场景。

2.3.2 后端设计

后端部分主要负责处理数据管理、面部识别逻辑和用户请求。使用 Python 编程语言，结合 OpenCV 库处理图像识别任务，以及自定义的数据管理逻辑来维护考勤记录和用户信息。后端设计遵循模块化原则，分为数据处理、面部识别和用户认证等模块，以支持系统的可扩展性和维护性。

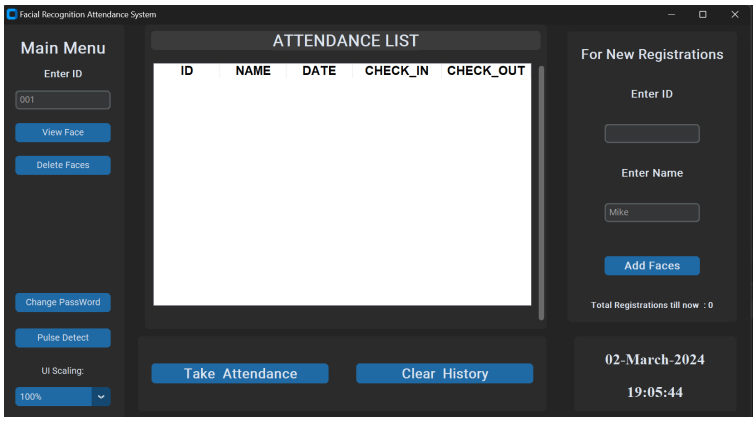
2.3.3 前端设计

前端界面基于 customtkinter 库设计，提供了一个直观且友好的用户界面。前端部分包括多个视图和控件，如侧边栏、考勤记录视图、用户注册和设置选项等。通过与后端模块的交互，前端不仅展示考勤信息，还支持用户通过图形界面进行面部数据的注册和管理操作。前端设计注重用户体验，通过简洁明了的布局和直观的操作流程，降低用户的使用门槛。

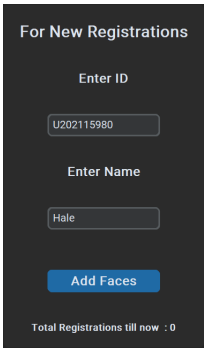
3 设计结果

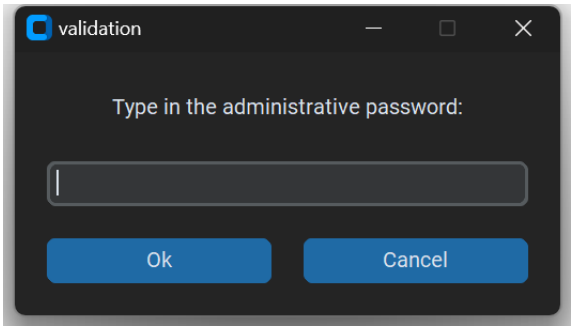
3.1 前端页面结果展示

启动程序得到如下页面：

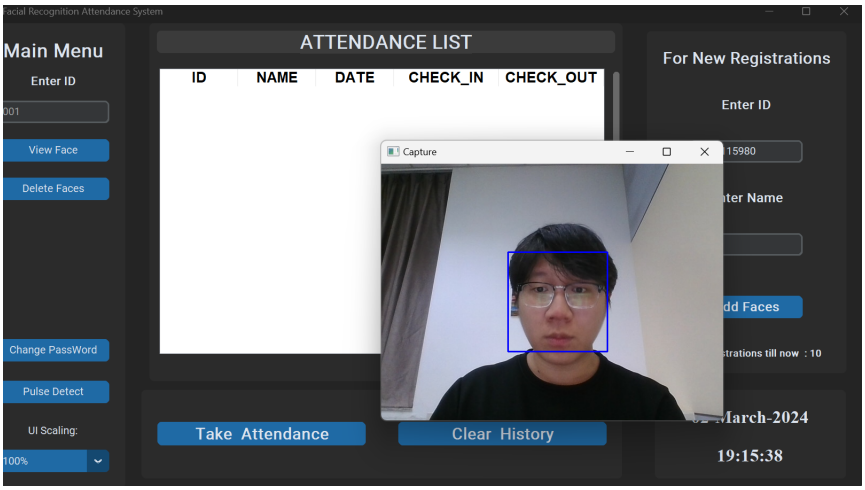


在右侧输入 ID(eg: U202115980), Name(eg: hale)，并且点击 Add face 按钮可以得到，如下的提示页面, 以希望获取管理员密码

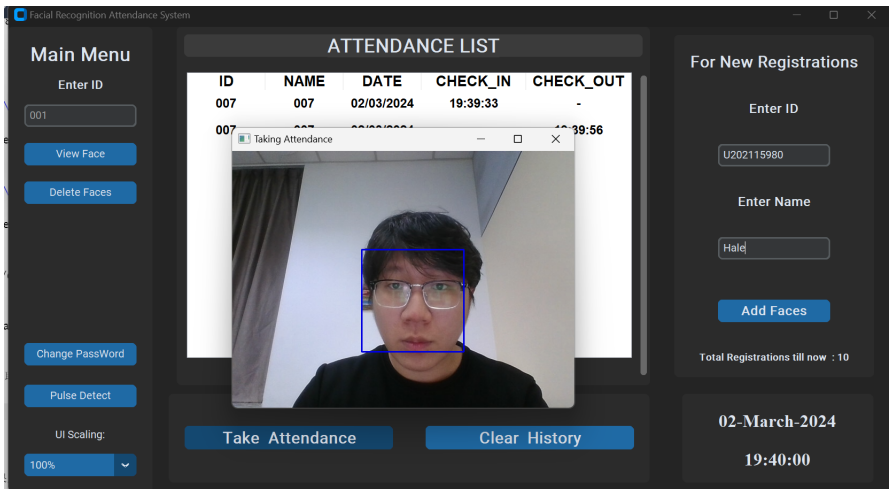




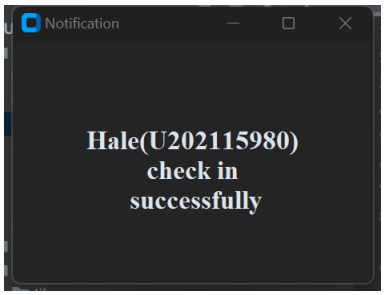
完成管理员密码输入后（可在./data/pwd.txt 文件中看到当前密码），调起前置摄像头完成对人像的采集



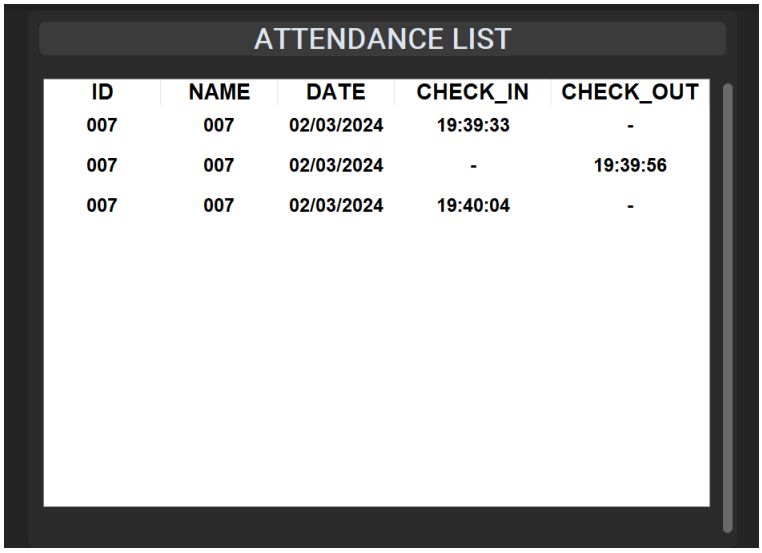
当获取操作完成后，弹框显示录取信息成功，便可进入到人脸检测的环节，点击 (Take Attendance) 按钮调起前置摄像头，进行人脸的检测和跟踪



将会展示如下的提示信息，表示签到成功：

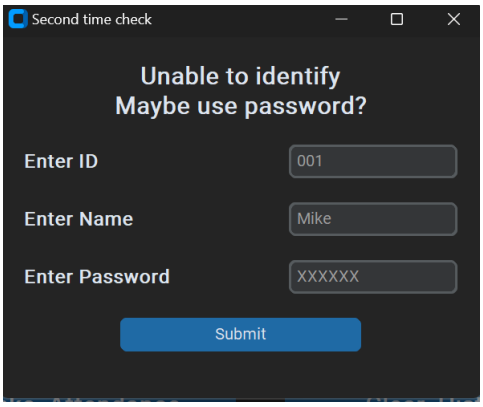


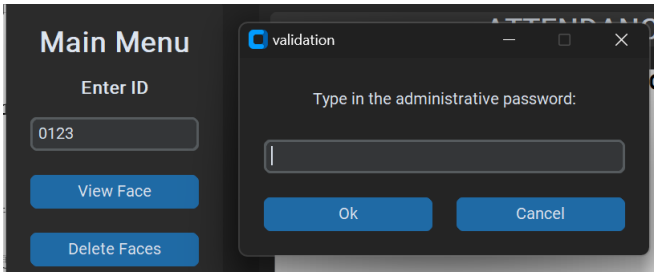
系统将会显示你的签到记录在屏幕上，如果再次进行签到便会自动进入签出模式，并且实时更新数据到后台



这样的记录，将会由后台打包成 csv 文件，以便于后期对于数据进行分析与解释。当人脸识别失败时，可由管理员手动加入记录，同时也会在出勤记录表中完成对人员记录的更新。

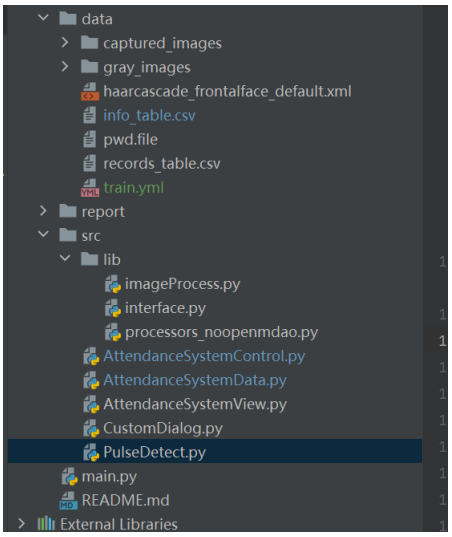
同时我们可以完成库内人脸的查看，通过输入 id 和二级管理员密码验证来完成查看和删除





3.2 后端数据库结果展示

整体代码框架层级结构及具体数据存储状态展示



	record_id	user_id	name	date	check_in	check_out
1	0	U202115980	yys	05/01/2024	-	10:00:08
2	1	U202115980	yys	05/01/2024	10:00:16	-
3	2	U202115980	yys	05/01/2024	-	10:00:23
4	3	U202115980	yys	05/01/2024	10:00:29	-
5	5	U202115980	yys	05/01/2024	-	10:16:57
6	6	U202115980	yys	05/01/2024	10:17:19	-
7	7	U202115980	yys	05/01/2024	-	10:17:28
8	8	U202115980	yys	05/01/2024	10:39:44	-
9	9	U202115980	yys	05/01/2024	11:20:06	-
10	10	U202115980	yys	05/01/2024	-	11:20:17
11	11	U202115980	yys	05/01/2024	12:42:56	-
12	12	U202115980	yys	05/01/2024	-	12:43:35
13	14	U202115980	yys	05/01/2024	12:50:53	-

4 总结与建议

本项目深入探讨了基于人脸识别技术的门禁考勤系统的设计与开发过程。主要成就和洞察包括：

- **技术选型的重要性：**通过精心选择的技术栈（包括 Python、OpenCV、customtkinter 等），展示了合适的技术对于实现系统功能和提高开发效率的重要性。
- **面部识别技术的应用：**项目成功地应用了 OpenCV 实现的 LBPH 算法，展示了面部识别技术在提升门禁系统安全性和用户体验方面的潜力。
- **用户界面设计：**利用 customtkinter 库开发了直观且友好的用户界面，有效降低了用户操作门槛。
- **数据管理：**采用 CSV 文件作为数据存储方案，在系统初期阶段满足了数据管理需求。
- **系统模块化：**模块化设计提高了代码的可维护性与可扩展性，为未来的升级和扩展奠定了基础。

为了进一步提升系统性能和用户体验，我们提出以下建议：

- **考虑使用数据库：**随着用户量增长，迁移到数据库管理系统（如 SQLite 或 MySQL）可以支持更复杂的数据操作和提高数据安全性。
- **增加多因素认证：**引入多因素认证机制（如 RFID 卡片、指纹或密码等），以增强身份验证过程的安全性。
- **优化算法性能：**持续优化和更新识别算法，提高识别准确率和处理速度。
- **扩展功能：**考虑增加更多功能，如实时数据分析、异常检测、健康状态监测等，以提升系统应用价值。
- **加强用户培训和支持：**提供详尽的用户手册和在线支持，定期举办培训研讨会，收集用户反馈以持续改进系统。

通过在这些方面的持续努力和改进，基于人脸识别技术的门禁考勤系统将能够更好地满足现代化的安全管理需求，为用户提供更安全、便捷、智能的体验。

参考文献

[1] OpenCV.org. (n.d.). *OpenCV Face Recognition*. Available at: https://opencv.org/opencv-face-recognition/

[2] Yu, S. (n.d.). *libfacedetection*. Available at: https://github.com/ShiqiYu/libfacedetection

[3] Ageitgey. (n.d.). *face_recognition*. Available at: https://github.com/ageitgey/face_recognition

[4] Zhang, K., et al. (n.d.). MTCNN face detection & alignment. Available at: https://github.com/kpzhang93/MTCNN_face_detection_alignment .

[5] DeepInsight. (n.d.). *InsightFace Detection*. Available at: https://github.com/deepinsight/insightface

[6] Zeusees. (n.d.). *HyperFT: Fast Video Face Tracking*. Available at: https://github.com/zeusees/HyperFT

[7] Thearn. (n.d.). *Webcam Pulse Detector*. Available at: https://github.com/thearn/webcam-pulse-detector

附录

```python
import cv2
import os
import time, datetime
import customtkinter as cstk
import shutil
from collections import defaultdict
import numpy as np

from src.AttendanceSystemData import Data


class Control:
    # define the password to check the user or administrator
    password = '123456'
    images_path = './data/captured_images/'
    data_path = './data/'
    gray_images_path = './data/gray_images/'
    password_file_path = './data/pwd.file'


    def __init__(self):
        self.toplevel_window = None
        self.key = 10

    @staticmethod
    def validate_password(input_pwd: str):
        Control.read_password()
        if input_pwd == Control.password:
            return True
        else:
            return False

    @staticmethod
    def fetch_image_by_id(index):
        files_and_folders = os.listdir(Control.images_path)
        for item in files_and_folders:
            if not item == str(index):
                continue
            item_path = os.path.join(Control.images_path, item)
            for file in os.listdir(item_path):
                return os.path.join(item_path, file)


    @staticmethod
    def delete_image_by_id(index):
        files_and_folders = os.listdir(Control.images_path)
        for item in files_and_folders:
            if not item == str(index):
                continue
            item_path = os.path.join(Control.images_path, item)
            shutil.rmtree(item_path)
        files_and_folders = os.listdir(Control.gray_images_path)
        for item in files_and_folders:
            if not item == str(index):
                continue
            item_path = os.path.join(Control.gray_images_path, item)
            shutil.rmtree(item_path)
```

```
58
59      @staticmethod
60      def change_scaling_event(new_scaling: str):
61          new_scaling_float = int(new_scaling.replace("%", "")) / 100
62          cstk.set_widget_scaling(new_scaling_float)
63
64      @staticmethod
65      def sidebar_button_event():
66          print("sidebar_button click")
67
68      @staticmethod
69      def get_current_time():
70          mont = {'01': 'January',
71                  '02': 'February',
72                  '03': 'March',
73                  '04': 'April',
74                  '05': 'May',
75                  '06': 'June',
76                  '07': 'July',
77                  '08': 'August',
78                  '09': 'September',
79                  '10': 'October',
80                  '11': 'November',
81                  '12': 'December'
82                  }
83          ts = time.time()
84          date = datetime.datetime.fromtimestamp(ts).strftime('%d-%m-%Y')
85          day, month, year = date.split("-")
86          return f"{day}-{mont[month]}-{year}"
87
88      @staticmethod
89      def get_total_register_number():
90          return len(Data.keymap_label)
91
92      @staticmethod
93      def take_photos(id, name):
94          print('take the photos', id, name)
95          cap = cv2.VideoCapture(0)
96          cap.set(3, 640)
97          cap.set(4, 480)
98
99          output_path = os.path.join(Control.images_path, '{0}'.format(id))
100
101         # Check if the folder exists, if not, create it
102         if not os.path.exists(output_path):
103             os.makedirs(output_path)
104
105         count = 0
106
107         while count < 10:
108             ret, frame = cap.read()
109             face_cascade = cv2.CascadeClassifier('./data/haarcascade_frontalface_default.xml')
110
111             gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
112             faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=5)
113
114             for (x, y, w, h) in faces:
115                 if w * h < 150 * 150:
116                     continue
```

```
                cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)

                # Crop the face region and save
                face_roi = frame[y:y + h, x:x + w]
                file_name = os.path.join(output_path, "{}_{}.png".format(name, count))
                cv2.imwrite(file_name, face_roi)
                print("Captured face {} saved.".format(count))

            # show the frame
            cv2.imshow('Capture', frame)

            capture_interval = 1
            if time.time() % capture_interval < 0.1:
                # add the counter
                count += 1
            # wait for a second
            cv2.waitKey(1)

        cap.release()
        cv2.destroyAllWindows()


    @staticmethod
    def encrypt(pwd, key):
        data_list = [int(d) for d in pwd]
        encrypted_data = [str(int(d) ^ key) for d in data_list]
        return encrypted_data

    @staticmethod
    def train_images():
        Control.convert_gray_faces()
        faces, labels = [], []
        label = 0
        for index in os.listdir(Control.gray_images_path):
            index_path = os.path.join(Control.gray_images_path, index)
            for each in os.listdir(index_path):
                each_path = os.path.join(index_path, each)
                gray_np = np.array(cv2.imread(each_path, 0), 'uint16')
                name = each.split('_')[0]
                labels.append(label)
                faces.append(gray_np)
            print(index, label)
            label += 1

        data_path = os.path.join(Control.data_path, 'train.yml')
        if not os.path.exists(Control.data_path):
            os.makedirs(Control.data_path)
        recognizer = cv2.face.LBPHFaceRecognizer_create()  # 初始化LBPH识别器
        if faces is not None and len(faces) != 0:
            recognizer.train(faces, np.array(labels))
        recognizer.save(data_path)
        Data.update_info_table()


    @staticmethod
    def convert_gray_faces():
        if not os.path.exists(Control.gray_images_path):
            os.makedirs(Control.gray_images_path)

```

```
        files_and_folders = os.listdir(Control.images_path)
        for item in files_and_folders:
            gray_folder = os.path.join(Control.gray_images_path, item)
            if not os.path.exists(gray_folder):
                os.makedirs(gray_folder)
            for each in os.listdir(os.path.join(Control.images_path, item)):
                each_path = os.path.join(gray_folder, each)
                if not os.path.exists(each_path):
                    ori_path = os.path.join(Control.images_path, item, each)
                    img = cv2.imread(ori_path)
                    gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
                    cv2.imwrite(each_path, gray_img)

    @staticmethod
    def recognise_face():
        # check the model data exists and load the data
        yml_data = os.path.join(Control.data_path, 'train.yml')
        if not os.path.exists(yml_data):
            Control.train_images()
        recognizer = cv2.face.LBPHFaceRecognizer_create()
        recognizer.read(yml_data)
        face_cascade = cv2.CascadeClassifier('./data/haarcascade_frontalface_default.xml')

        confirmation_interval = 10
        last_confirmation_time = time.time()
        cap = cv2.VideoCapture(0)
        times = defaultdict(int)
        flag = True
        while flag:
            ret, frame = cap.read()
            cv2.imshow('Taking Attendance', frame)
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            faces = face_cascade.detectMultiScale(gray, scaleFactor=1.08, minNeighbors=5)
            for (x, y, w, h) in faces:
                if w * h < 150 * 150:
                    continue
                cv2.rectangle(frame, (x, y), (x + w, y + h), (225, 0, 0), 2)
                label, score = recognizer.predict(gray[y:y + h, x:x + w])
                if score < 50:
                    times[label] += 1
                    if times[label] > 20:
                        flag = False
            cv2.imshow('Taking Attendance', frame)
            cur_time = time.time()
            if cur_time - last_confirmation_time > confirmation_interval:
                break

            if cv2.waitKey(1) == ord('q'):
                break
        cap.release()
        cv2.destroyAllWindows()
        index, max_value = -1, 0
        for key, value in times.items():
            if value > 20 and value > max_value:
                index = key
                max_value = value
        return index

    @staticmethod
```

```
    def change_password(ori: str, new: str):
        if not Control.validate_password(ori):
            return False
        else:
            Control.password = new
            Control.write_password()
        return True

    @staticmethod
    def read_password():
        with open(Control.password_file_path, 'r') as file:
            pwd = file.readline()
            Control.password = pwd

    @staticmethod
    def write_password():
        if not os.path.exists(Control.password_file_path):
            open(Control.password_file_path, 'w')
        with open(Control.password_file_path, 'w') as file:
            file.write(Control.password)
```

Listing 1: AttendanceSystemControl.py

```
import csv
import os
from collections import defaultdict
from datetime import datetime

from src.PulseDetect import PulseDetect


class Data:
    App = None
    data_path = './data'
    gray_images_path = './data/gray_images/'
    keymap_index = dict()
    keymap_label = dict()
    status_index = defaultdict(bool)

    # records table related
    records_count = 0
    records_table = []      # append not rewrite
    records_table_fields = ['record_id', 'user_id', 'name', 'date', 'check_in', 'check_out']
    records_table_path = './data/records_table.csv'

    # information table related
    info_table = []
    info_table_fields = ['user_id', 'name', 'label']
    info_table_path = './data/info_table.csv'

    @classmethod
    def initialize(cls):
        with open(Data.info_table_path, 'r') as csvfile:
            reader = csv.DictReader(csvfile)
            header = reader.fieldnames
            for row in reader:
                Data.keymap_label[int(row.get('label'))] = (row.get('user_id'), row.get('name'
                    ))
                Data.keymap_index[row.get('user_id')] = (row.get('name'), row.get('label'))

```

```
        if not os.path.exists(Data.records_table_path):
            file = open(Data.records_table_path, 'w')
        else:
            with open(Data.records_table_path, 'r') as csvfile:
                reader = csv.reader(csvfile)
                last_line = 0
                for line_number, row in enumerate(reader, start=1):
                    last_line = line_number
                Data.records_count = last_line


    @staticmethod
    def fetch_name_by_id(id):
        if id not in Data.keymap_index:
            return None
        else:
            return Data.keymap_index[id][0]

    @staticmethod
    def fetch_name_by_label(label):
        if label >= len(Data.keymap_label):
            return None
        else:
            return Data.keymap_label[label][1]

    @staticmethod
    def fetch_label_by_id(id):
        if id not in Data.keymap_index:
            return None
        else:
            return int(Data.keymap_index[id][1])

    @staticmethod
    def fetch_id_by_label(label):
        if label >= len(Data.keymap_label):
            return None
        else:
            return Data.keymap_label[label][0]

    @staticmethod
    def make_records(label) -> (bool, str, str):
        # return back True or False indicate check-in or out status
        new_record = dict()
        index, name = Data.keymap_label[label]
        right_datetime = datetime.now()
        cur_date = right_datetime.date().strftime('%d/%m/%Y')
        cur_time = right_datetime.time().strftime('%H:%M:%S')
        new_record['user_id'] = index
        new_record['name'] = name
        new_record['record_id'] = Data.records_count
        Data.records_count += 1
        new_record['date'] = cur_date
        new_record['check_in'], new_record['check_out'] = '-', '-'
        if not Data.status_index[index]:
            new_record['check_in'] = cur_time
        else:
            new_record['check_out'] = cur_time
        Data.status_index[index] = not Data.status_index[index]
        Data.records_table.append(new_record)
```

```
96          return Data.status_index[index], index, name
97
98      @staticmethod
99      def update_records_table():
100         if not Data.records_table:
101             return
102         with open(Data.records_table_path, 'a', newline='') as csvfile:
103             writer = csv.DictWriter(csvfile, fieldnames=Data.records_table_fields)
104             writer.writeheader()
105             writer.writerows(Data.records_table)
106         Data.records_table = []
107     @staticmethod
108     def update_info_table():
109         # read items in the directory one by one
110         label = 0
111         for item in os.listdir(Data.gray_images_path):
112             id, name = item, None
113             dir_path = os.path.join(Data.gray_images_path, item)
114             for file in os.listdir(dir_path):
115                 name = file.split('_')[0]
116                 break
117             Data.info_table.append({
118                 'user_id': id,
119                 'name': name,
120                 'label': label
121             })
122             label += 1
123         # open the csv file and write back
124         with open(Data.info_table_path, 'w', newline='') as csvfile:
125             writer = csv.DictWriter(csvfile, fieldnames=Data.info_table_fields)
126             writer.writeheader()
127             writer.writerows(Data.info_table)
128         Data.initialize()
129
130     @staticmethod
131     def organize_csv_file():
132         with open(Data.records_table_path, 'r', newline='') as file:
133             csv_reader = csv.reader(file)
134             data = list(csv_reader)
135
136         data_indices = [row for row in data if row[0] != 'record_id']
137         chosen_header_index = data[0]
138         cleaned_data = [chosen_header_index] + data_indices
139
140         with open(Data.records_table_path, 'w', newline='') as file:
141             csv_writer = csv.writer(file)
142             csv_writer.writerows(cleaned_data)
143
144     @staticmethod
145     def render_data():
146         datas = []
147         for each in Data.records_table:
148             datas.append((
149                 each.get('user_id'),
150                 each.get('name'),
151                 each.get('date'),
152                 each.get('check_in'),
153                 each.get('check_out')
154             ))
```

```
        return datas

    @staticmethod
    def run_detect():
        Data.App = PulseDetect()
        flag = True
        while flag:
            flag = Data.App.main_loop()
```

Listing 2: AttendanceSystemData.py

```
import time

import customtkinter
import customtkinter as cstk
import tkinter as tk
from tkinter import ttk

import cv2

from src.AttendanceSystemControl import Control as ctrl
from src.CustomDialog import ToplevelWindow, SecondTimeCheck, ChangePasswordDialog
from src.AttendanceSystemData import Data


class AttendanceSystemView(customtkinter.CTk):
    def __init__(self, *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.message = None
        self.attend_info = None
        self.initialize_GUI()

        self.toplevel_window = None
        self.image_window = None

    def run(self):
        self.mainloop()

    def initialize_GUI(self):
        cstk.set_appearance_mode('dark')
        cstk.set_default_color_theme('blue')

        # configure the window
        self.title('Facial Recognition Attendance System')
        self.geometry('1100x580')
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure((2, 3), weight=0)
        self.grid_rowconfigure((0, 1, 2), weight=1)

        # create the labels in the UI view
        self.create_sidebar_frame()
        self.create_attendance_frame()
        self.create_time_frame()
        self.create_option_frame()
        self.create_register_frame()

    # create sidebar frame with widgets
    def create_sidebar_frame(self):
        sidebar_frame = cstk.CTkFrame(self, width=140)
        sidebar_frame.grid(row=0, column=0, rowspan=10, sticky="nsew")
```

```
        sidebar_frame.grid_rowconfigure(5, weight=1)
        logo_label = cstk.CTkLabel(sidebar_frame, text="Main Menu", font=cstk.CTkFont(size=24,
            weight="bold"))
        logo_label.grid(row=0, column=0, padx=5, pady=(20, 10))
        sidebar_label = cstk.CTkLabel(sidebar_frame, text="Enter ID", font=cstk.CTkFont(size
            =15, weight="bold"))
        sidebar_label.grid(row=1, column=0, padx=20)
        sidebar_entry_ID = cstk.CTkEntry(sidebar_frame, placeholder_text="001")
        sidebar_entry_ID.grid(row=2, column=0, padx=20, pady=(10, 10))
        sidebar_button_1 = cstk.CTkButton(sidebar_frame,
                                          command=lambda: self.view_face(sidebar_entry_ID.get
                                              ()),
                                          text="View Face")
        sidebar_button_1.grid(row=3, column=0, padx=20, pady=(10, 10))
        sidebar_button_2 = cstk.CTkButton(sidebar_frame,
                                          command=lambda: self.delete_face(sidebar_entry_ID.
                                              get()),
                                          text="Delete Faces")
        sidebar_button_2.grid(row=4, column=0, padx=20, pady=(10, 10))
        sidebar_button_3 = cstk.CTkButton(sidebar_frame,
                                          command=self.change_password,
                                          text="Change PassWord")
        sidebar_button_3.grid(row=5, column=0, padx=20, pady=(160, 10))
        scaling_label = cstk.CTkLabel(sidebar_frame, text="UI Scaling:", anchor="w")
        sidebar_button_4 = cstk.CTkButton(sidebar_frame,
                                          command=self.pulse_detect,
                                          text='Pulse Detect')
        sidebar_button_4.grid(row=6, column=0, padx=20, pady=10)
        scaling_label.grid(row=7, column=0, padx=20, pady=(10, 0))
        scaling_option_menu = cstk.CTkOptionMenu(sidebar_frame, values=["80%", "90%", "100%",
            "110%", "120%"],
                                                 command=ctrl.change_scaling_event)
        scaling_option_menu.grid(row=8, column=0, padx=20, pady=(10, 20))
        scaling_option_menu.set("100%")

    # create attendance frame with widgets
    def create_attendance_frame(self):
        attendance_frame = cstk.CTkScrollableFrame(self, label_text="ATTENDANCE LIST",
                                                   label_font=cstk.CTkFont(size=24, weight="
                                                       bold"))
        attendance_frame.grid(row=0, column=1, padx=(30, 20), pady=(0, 0), sticky="nsew")
        attendance_frame.grid_columnconfigure(0, weight=1)
        self.attend_info = ttk.Treeview(attendance_frame, show="headings",
                                        columns=('ID', 'name', 'date', 'check_in', 'check_out'
                                            ),
                                        height=10)
        self.attend_info.column('ID', width=100, anchor=tk.CENTER)
        self.attend_info.column('name', width=100, anchor=tk.CENTER)
        self.attend_info.column('date', width=100, anchor=tk.CENTER)
        self.attend_info.column('check_in', width=150, anchor=tk.CENTER)
        self.attend_info.column('check_out', width=150, anchor=tk.CENTER)
        self.attend_info.heading('ID', text='ID')
        self.attend_info.heading('name', text='NAME')
        self.attend_info.heading('date', text='DATE')
        self.attend_info.heading('check_in', text='CHECK_IN')
        self.attend_info.heading('check_out', text='CHECK_OUT')
        ttk.Style().configure("Treeview", rowheight=50, font=('Arial', 18, 'bold'))
        ttk.Style().configure('Treeview.Heading', font=('Arial', 20, 'bold'))
        self.attend_info.grid(row=1, column=0, padx=(10, 10), pady=(5, 5), sticky='nsew')
```

```
102
103     # create register frame with widgets
104     def create_register_frame(self):
105         register_frame = cstk.CTkFrame(self, width=300)
106         register_frame.grid(row=0, column=2, padx=(10, 20), pady=(10, 10), sticky="nsew")
107         register_label_1 = cstk.CTkLabel(register_frame, text="For New Registrations",
108                                          font=cstk.CTkFont(size=20, weight="bold"))
109         register_label_1.grid(row=0, column=0, padx=20, pady=(20, 10))
110         register_label_2 = cstk.CTkLabel(register_frame, text="Enter ID",
111                                          font=cstk.CTkFont(size=16, weight="bold"))
112         register_label_2.grid(row=1, column=0, padx=20, pady=(20, 10))
113         register_entry_ID = cstk.CTkEntry(register_frame, placeholder_text="001")
114         register_entry_ID.grid(row=2, column=0, padx=20, pady=(20, 10))
115         register_label_3 = cstk.CTkLabel(register_frame, text="Enter Name",
116                                          font=cstk.CTkFont(size=16, weight="bold"))
117         register_label_3.grid(row=3, column=0, padx=20, pady=(20, 10))
118         register_entry_name = cstk.CTkEntry(register_frame, placeholder_text="Mike")
119         register_entry_name.grid(row=4, column=0, padx=20, pady=(20, 10))
120         register_button = cstk.CTkButton(register_frame,
121                                          command=lambda: self.register_new_figure(
                                                 register_entry_ID.get(),
122                                                                                  register_entry_name
                                                                                      .get()),
123                                          text="Add Faces",
124                                          font=cstk.CTkFont(size=16, weight="bold"))
125         register_button.grid(row=5, column=0, padx=20, pady=(40, 10))
126         self.message = cstk.CTkLabel(register_frame, font=cstk.CTkFont(size=12, weight="bold")
                )
127         self.message.grid(row=6, column=0, padx=20, pady=(20, 10))
128         self.message.configure(text='Total Registrations till now  : {}'.format(ctrl.
                get_total_register_number()))
129
130     # create option frame with widgets
131     def create_option_frame(self):
132         option_frame = cstk.CTkFrame(self)
133         option_frame.grid(row=1, column=1, padx=(20, 20), pady=(10, 10), sticky="nsew")
134         option_button_1 = cstk.CTkButton(option_frame,
135                                          command=self.make_attendance,
136                                          text="Take  Attendance",
137                                          font=cstk.CTkFont(size=20, weight="bold"), width=300)
138         option_button_1.grid(row=0, column=0, padx=20, pady=40)
139         option_button_2 = cstk.CTkButton(option_frame,
140                                          command=self.clear_history,
141                                          text="Clear  History",
142                                          font=cstk.CTkFont(size=20, weight="bold"), width=300)
143         option_button_2.grid(row=0, column=1, padx=20, pady=40)
144         option_frame.grid_columnconfigure(0, weight=1)
145         option_frame.grid_columnconfigure(1, weight=1)
146         option_frame.grid_columnconfigure(0, weight=1)
147
148     # create time frame with widgets
149     def create_time_frame(self):
150         time_frame = cstk.CTkFrame(self)
151         time_frame.grid(row=1, column=2, padx=(20, 20), pady=(10, 10), sticky="nsew")
152         date_frame = cstk.CTkLabel(time_frame, text=ctrl.get_current_time(),
153                                    font=cstk.CTkFont('times', 22, 'bold'))
154         date_frame.grid(row=0, column=0, padx=45, pady=(20, 10))
155         clock = cstk.CTkLabel(time_frame, font=cstk.CTkFont('times', 22, 'bold'))
156         clock.grid(row=1, column=0, padx=45, pady=(10, 10))
```

```
157
158         def tick():
159             time_string = time.strftime('%H:%M:%S')
160             clock.configure(text=time_string)
161             clock.after(200, tick)
162
163         tick()
164
165     def validate_password_window(self):
166         dialog = customtkinter.CTkInputDialog(text="Type in the administrative password:",
                title="validation")
167         pwd = dialog.get_input()
168
169         # the password is not correct
170         if not ctrl.validate_password(pwd):
171             if self.toplevel_window is None or not self.toplevel_window.winfo_exists():
172                 self.toplevel_window = ToplevelWindow(self)  # create window if its None or
                        destroyed
173             else:
174                 self.toplevel_window.focus()  # if window exists focus it
175             return False
176         else:
177             return True
178
179     def register_new_figure(self, id, name):
180         # validate the password
181         if not self.validate_password_window():
182             return
183
184         # add the new tuple(id, name) and figure into the resources
185         ctrl.take_photos(id, name)
186         ctrl.train_images()
187         self.toplevel_window = ToplevelWindow(self)  # create window if its None or destroyed
188         self.toplevel_window.change_text('\n\nSuccessful add \n {0} \ninformation!!! '
189                                          .format(Data.fetch_name_by_id(id)))
190
191         self.message.configure(text='Total Registrations till now  : {}'.format(ctrl.
                get_total_register_number()))
192
193     def view_face(self, index):
194         # validate the password
195         if not self.validate_password_window():
196             return
197         image_path = ctrl.fetch_image_by_id(index)
198         image = cv2.imread(image_path)
199         if image is None:
200             self.toplevel_window = ToplevelWindow(self)  # create window if its None or
                    destroyed
201             self.toplevel_window.change_text('\n\nDon\'t have \nid: {0}\n information!!!'.
                    format(index))
202         else:
203             cv2.imshow('{0}'.format(index), image)
204
205     def delete_face(self, index):
206         # validate the password
207         if not self.validate_password_window():
208             return
209         image_path = ctrl.fetch_image_by_id(index)
210         image = cv2.imread(image_path)
```

```
211         if image is None:
212             self.toplevel_window = ToplevelWindow(self)  # create window if its None or
                    destroyed
213             self.toplevel_window.change_text('\n\nDon\'t have \nid: {0}\n information!!!'.
                    format(index))
214         else:
215             dialog = customtkinter.CTkInputDialog(text="Are you sure delete {0} information?\n
                    "
216                                                   "This operation is irreversible\n"
217                                                   "Input 'yes' to confirm".format(Data.
                                                          fetch_name_by_id(index)),
218                                                   title="confirm",
219                                                   font=cstk.CTkFont('times', 20, 'bold'))
220             response = dialog.get_input()
221             if response == 'yes':
222                 ctrl.delete_image_by_id(index)
223         ctrl.train_images()
224 
225     def make_attendance(self):
226         def handler_data(func):
227             id, name, pwd = func()
228             self.toplevel_window.destroy()
229             if not ctrl.validate_password(pwd):
230                 self.toplevel_window = ToplevelWindow(self)
231             elif Data.fetch_name_by_id(id) != name:
232                 self.toplevel_window = ToplevelWindow(self)
233                 self.toplevel_window.change_text('\n\nDon\'t have \nname: {0}\n information!!!
                        ', name)
234             else:
235                 status, id, name = Data.make_records(Data.fetch_label_by_id(id))
236                 string = '\n\n{0}({1})\n check in \n successfully'.format(name, id)
237                 if not status:
238                     string = '\n\n{0}({1})\n check out \n successfully'.format(name, id)
239                 self.toplevel_window = ToplevelWindow(self)
240                 self.toplevel_window.change_text(string)
241             self.toplevel_window.lift()
242             self.render_list_data()
243 
244         label = ctrl.recognise_face()
245         if label == -1:
246             self.toplevel_window = SecondTimeCheck(handler_data, self)  # create window if its
                     None or destroyed
247             self.toplevel_window.lift()
248         else:
249             status, id, name = Data.make_records(label)
250             string = '\n\n{0}({1})\n check in \n successfully'.format(name, id)
251             if not status:
252                 string = '\n\n{0}({1})\n check out \n successfully'.format(name, id)
253             if self.toplevel_window is not None:
254                 self.toplevel_window.destroy()
255             self.toplevel_window = ToplevelWindow(self)
256             self.toplevel_window.change_text(string)
257         self.toplevel_window.lift()
258         self.render_list_data()
259 
260     def clear_history(self):
261         Data.update_records_table()
262         self.render_list_data()
263 
```

```
    def render_list_data(self):
        data = Data.render_data()
        for row in self.attend_info.get_children():
            self.attend_info.delete(row)
        for row_data in data:
            self.attend_info.insert("", "end", values=row_data, tags=("Arial", 16))

    def change_password(self):
        def handle_data(func):
            ori, new = func()
            if self.toplevel_window is not None:
                self.toplevel_window.destroy()
                self.toplevel_window = None
            status = ctrl.change_password(ori, new)
            if status:
                self.toplevel_window = ToplevelWindow(self)
                self.toplevel_window.change_text('\n\nChange Password\n Successfully!!!\n')
            else:
                self.toplevel_window = ToplevelWindow(self)
                self.toplevel_window.change_text('\n\nOriginal Password\n is Wrong!!!\n')
            self.toplevel_window.lift()
        self.toplevel_window = ChangePasswordDialog(handle_data, self)

    def pulse_detect(self):
        Data.run_detect()
```

Listing 3: AttendanceSystemView.py

```
from src.lib.processors_noopenmdao import findFaceGetPulse
from src.lib.interface import plotXY, imshow, waitKey, destroyWindow
from cv2 import moveWindow
import argparse
import numpy as np
import datetime
import cv2, sys


class PulseDetect:
    def __init__(self):
        self.cam = cv2.VideoCapture(0)
        self.w, self.h = 0, 0
        self.pressed = 0
        self.processor = findFaceGetPulse(bpm_limits=[50, 160],
                                          data_spike_limit=2500.,
                                          face_detector_smoothness=10.)
        self.bpm_plot = False
        self.plot_title = "Data display - raw signal (top) and PSD (bottom)"

        self.key_controls = {"s": self.toggle_search,
                             "d": self.toggle_display_plot,
                             "f": self.write_csv}

    def toggle_search(self):
        # state = self.processor.find_faces.toggle()
        state = self.processor.find_faces_toggle()
        print("face detection lock =", not state)

    def toggle_display_plot(self):
        if self.bpm_plot:
            print("bpm plot disabled")
```

```
            self.bpm_plot = False
            destroyWindow(self.plot_title)
        else:
            print("bpm plot enabled")
            if self.processor.find_faces:
                self.toggle_search()
            self.bpm_plot = True
            self.make_bpm_plot()
            moveWindow(self.plot_title, self.w, 0)

    def make_bpm_plot(self):
        """
        Creates and/or updates the data display
        """
        plotXY([[self.processor.times,
                 self.processor.samples],
                [self.processor.freqs,
                 self.processor.fft]],
               labels=[False, True],
               showmax=[False, "bpm"],
               label_ndigits=[0, 0],
               showmax_digits=[0, 1],
               skip=[3, 3],
               name=self.plot_title,
               bg=self.processor.slices[0])

    def write_csv(self):
        """
        Writes current data to a csv file
        """
        fn = './data/pulse/' + str(datetime.datetime.now())
        fn = fn.replace(":", "_").replace(".", "_")
        data = np.vstack((self.processor.times, self.processor.samples)).T
        np.savetxt(fn + ".csv", data, delimiter=',')
        print("Writing csv")

    def key_handler(self):
        self.pressed = waitKey(10) & 255  # wait for keypress for 10 ms
        if self.pressed == 27:  # exit program on 'esc'
            print("Exiting")
            self.cam.release()
            cv2.destroyAllWindows()
            return False

        for key in self.key_controls.keys():
            if chr(self.pressed) == key:
                self.key_controls[key]()
        return True

    def main_loop(self):

        _, frame = self.cam.read()
        self.h, self.w, _c = frame.shape

        # set current image frame to the processor's input
        self.processor.frame_in = frame
        # process the image frame to perform all needed analysis5
        self.processor.run(self.cam)
        # collect the output frame for display
```

```
92          output_frame = self.processor.frame_out
93
94          # show the processed/annotated output frame
95          imshow("Processed", output_frame)
96
97          # create and/or update the raw data display if needed
98          if self.bpm_plot:
99              self.make_bpm_plot()
100
101         # handle any key presses
102         return self.key_handler()
```

Listing 4: PulseDetect.py