BSD2513 ARTIFICIAL INTELLIGENCE

SEMESTER 2 2022/2023

AttendEase

NAME	STUDENT ID
LIM KA QUAN	**SD21033**
TOK CHIA WEN	SD21001
LEE ZHI LIN	SD21022
LIM JOEY	SD21038
CHONG JIA XIN	SD21039

TABLE OF CONTENTS

1.0 EXECUTIVE SUMMARY

1.1 Description of the selected project

Face recognition is a technology that is able to capture a human face and match it with a digital image or a video frame to confirm an individual's identity. The technology is widely used for security and law enforcement. Indeed, many people are now familiar with face recognition technologies such as FaceID that used to unlock phones by Apple Incorporation, which not only allows us to access our phones faster but also limits access from other users to preserve our privacy. Furthermore, facial recognition is critical for automation and efficiency. With the help of face recognition, we could detect criminals or missing persons in real-time or from recording once their picture is taken and uploaded into databases. Thus, face recognition technology will become more likely pervasive.

Without face recognition, identification processes would more heavily rely on manual methods.This could lead to low efficiency and time-consuming, and it will potentially result in reduction of productivity and increasing in operational costs.

This technology would be integrated into an attendance taking system in this project. To maximise the effectiveness of the lesson, most lecturers in higher education institutions prefer students to self-report their attendance. However, this resulted in some absentees' attendances being signed by their friends, resulting in an incorrect attendance percentage at the end of the semester.

To address the problem, our group thinks that this technology will be a great solution. Face recognition allows users to be identified based on their unique facial feature by comparing with the pre-stored image in the database. With the application of face recognition on the attendance system, time arrival of students can be automatically recorded and reduce the chance of fraudulent practices.

Face recognition varies in their ability to determine people under several conditions like low quality image resolution, lack of lighting , weird angle of taking pictures and so on. When it comes to error, it doesn't mean that it is a bad face recognition system. There are 2 situations which are false negative and false positive.

In simple terms, false negative means it fails to match a person with the image inside the database which is bad, while false positive means it fails to match a person with the image that is not inside the database which indicates the face recognition system is working properly.

Given that there is almost always a trade-off, it is critical to pay close attention to the "false positive" and "false negative" rates. Face recognition may grow less accurate as more people are added to the database. The reason for this is the widespread similarity among people globally. As the occurrence of similar facial features becomes more common, the accuracy of facial matching may decrease.

1.2 Problem to be solved

- How to develop a system that accurately recognises and matches faces to ensure reliable attendance taking?
- How to create a system that can handle changes in lighting, facial emotions, and position in order to retain reliable face identification?
- How to develop an user-friendly Graphical User Interface (GUI) for both instructors and students?

1.3 Basic Description of the data selected

Data used in this project will be the image from our coursemates. Some of them from different races and gender will be asked to take a photo and put into our system to diversify the database.

2.0 SUMMARY OF THE PROJECT CONTEXT AND OBJECTIVES

2.1 Summary of the Project Context

The project context is about developing an attendance taking system utilising facial recognition AI technology. A camera will be used to record images of students as they enter the classroom. The images will subsequently be analysed by AI technology in order to identify the students. The system will then record the students' attendances.

The system will provide various advantages. First and foremost, it will be more precise than traditional attendance methods in terms of time arrived, since our system is accurate down to the unit of seconds. Second, it will be more efficient because lecturers would no longer have to manually take attendance which can save the time for his lecture. Last but not least, students will find it more convenient because they will no longer need to sign in or out of class.

There are some challenges in this project. To begin, it must be capable of reliably identifying kids in a variety of lighting circumstances. Second, it must be able to accommodate a significant number of pupils. Third, it must be capable of safeguarding students' privacy.

Overall, the project has the potential to be a useful educational tool. Addressing the problems will be critical to ensuring that the system is accurate, efficient, and easy.

2.2 Objectives

- To develop a system that accurately recognises and matches faces to ensure reliable attendance tracking.
- To create a system that can handle changes in lighting, facial emotions, and position in order to retain reliable face identification.
- To develop an user-friendly Graphical User Interface (GUI) for both lecturers and students.

3.0 METHODOLOGY

3.1 Data collection and preparation

First and foremost, because our project is about facial recognition on an attendance taking system, we will collect photographs from our coursemates. We gather a few photographs and load them into the system ahead of time. Furthermore, we ensure that these photos are clearly captured in order for the analysis and detection to be more accurate.

For the sample photographs we used, kindly refer to Appendix 1.

3.2 Coding

We use Jupyter Notebook to create coding that automatically detects the students. Few libraries should be installed into the Jupyter Notebook terminal before starting to model the coding.

First, make sure these 5 libraries are installed into the Jupyter Notebook terminal.

- Numpy - library of Python to work with arrays
- Opencv-python(cv2) - library of Python bindings designed to solve computer vision problems
- Face recognition - library of Python to detect and recognise or detect face
- Os - operating system-related functions.
- Datetime - library for handling date and time

```
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime
```

Step 2: capture video frames by using the default camera (index with 0)

```
video_capture = cv2.VideoCapture(0)
```

Step 3: Load known face image by using face_recognition.load_image_file by providing the file path in the device and encode them into a list to store the known face encoding.

```
LZL_image = face_recognition.load_image_file("LZL1.jpg")
LZL_encoding = face_recognition.face_encodings(LZL_image)[0]

zy_image = face_recognition.load_image_file("zy.jpg")
zy_encoding = face_recognition.face_encodings(zy_image)[0]

jr_image = face_recognition.load_image_file("jr.jpg")
jr_encoding = face_recognition.face_encodings(jr_image)[0]

kq_image = face_recognition.load_image_file("kq.jpg")
kq_encoding = face_recognition.face_encodings(kq_image)[0]

known_face_encoding = [
LZL_encoding,
zy_encoding,
jr_encoding,
kq_encoding
]
```

Step 4: Store the known face names by using copy() function and initialized 3 empty lists which are face_location, face_encodings, face_names.

```
students = known_faces_names.copy()

face_locations = []
face_encodings = []
face_names = []
```

Step 5: Declare s=True for the purpose of video capture later on

```
s=True
```

Step 6:Declare a now variable that store current date and time, when execute this code, it will open the csv file that name with the current date and store data into it.

```
now = datetime.now()
current_date = now.strftime("%Y-%m-%d")
f = open(current_date+'.csv','w+',newline = '')
lnwriter = csv.writer(f)
```

Step 7: This code will capture frames from the video and resize the captured frame to smaller size for the purpose of increasing the speed of processing by using cv2.resize. After that, it will convert the frame format from BGR to RGB because in opencv its format is BGR but the pre-training model for face recognition library is RGB format.

```
while True:
    _,frame = video_capture.read()
    small_frame = cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
    rgb_small_frame = small_frame[:,:,::-1]
```

Step 8: For this if statement, we declare s=True in step 5 thus it will start running. When s is True face recognition is enabled, it will start to detect the face location and compute the face encodings for face in frame captured.

Step 9: It will go into a for loop to match whether the detected face is the same as the image we insert into the face_encoding variable. The best match is determined by using face_recognition.face_distance which will determine the face distance between the current face encoding with the known face encoding. If the best match exists, it will assign the name into the name variable.

Step 10: The name will be appended to face_names list and if the name is stored inside known_faces_name, it will display on the frame by using cv2.putText and it will remove the student name from the students list and print the student name as the prove the student attend. The updated list will show the students name that still absent. Afterward, the data about the student name and his/her attend time will be recorded in CSV file.

```
    if s:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,face_locations)
        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(known_face_encoding,face_encoding)
            name=""
            face_distance = face_recognition.face_distance(known_face_encoding,face_encoding)
            best_match_index = np.argmin(face_distance)
            if matches[best_match_index]:
                name = known_faces_names[best_match_index]

            face_names.append(name)
            if name in known_faces_names:
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
                fontScale              = 1.5
                fontColor              = (255,0,0)
                thickness              = 3
                lineType               = 2

                cv2.putText(frame,name+' Present',
                    bottomLeftCornerOfText,
                    font,
                    fontScale,
                    fontColor,
                    thickness,
                    lineType)

                if name in students:
                    students.remove(name)
                    print(students)
                    current_time = now.strftime("%H-%M-%S")
                    lnwriter.writerow([name,current_time])
```

Step 11: The frame will show the attendance system on the frame by using cv2.imshow function and the loop will end when the user presses ‘q’ on the keyboard.

Step 12: The video capture will be released and opencv window and CSV file will be closed using video_capture.release(), cv2.destroyAllWindows(), and f.close().

```
    cv2.imshow("attendance system",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
f.close()
```

Below is the full coding:

```
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime


video_capture = cv2.VideoCapture(0)


LZL_image = face_recognition.load_image_file("LZL1.jpg")
LZL_encoding = face_recognition.face_encodings(LZL_image)[0]


zy_image = face_recognition.load_image_file("zy.jpg")
zy_encoding = face_recognition.face_encodings(zy_image)[0]


jr_image = face_recognition.load_image_file("jr.jpg")
jr_encoding = face_recognition.face_encodings(jr_image)[0]


kq_image = face_recognition.load_image_file("kq.jpg")
```

```
kq_encoding = face_recognition.face_encodings(kq_image)[0]

known_face_encoding = [
LZL_encoding,
zy_encoding,
jr_encoding,
kq_encoding
]

known_faces_names = [
"zhi lin",
"zi ying",
"jing rou",
"ka quan"
]

students = known_faces_names.copy()

face_locations = []
face_encodings = []
face_names = []
s=True


now = datetime.now()
current_date = now.strftime("%Y-%m-%d")
f = open(current_date+'.csv','w+',newline = '')
lnwriter = csv.writer(f)
```

```
        while True:
            _,frame = video_capture.read()
            small_frame = cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
            rgb_small_frame = small_frame[:,:,::-1]
            if s:
                face_locations = face_recognition.face_locations(rgb_small_frame)
                face_encodings = face_recognition.face_encodings(rgb_small_frame,face_locations)
                face_names = []
                for face_encoding in face_encodings:
                                                                    matches =
face_recognition.compare_faces(known_face_encoding,face_encoding)
                    name=""
                                                                face_distance =
face_recognition.face_distance(known_face_encoding,face_encoding)
                    best_match_index = np.argmin(face_distance)
                    if matches[best_match_index]:
                        name = known_faces_names[best_match_index]

                    face_names.append(name)
                    if name in known_faces_names:
                        font = cv2.FONT_HERSHEY_SIMPLEX
                        bottomLeftCornerOfText = (10,100)
                        fontScale              = 1.5
                        fontColor              = (255,0,0)
                        thickness              = 3
                        lineType               = 2

                        cv2.putText(frame,name+' Present',
```

```
                bottomLeftCornerOfText,
                font,
                fontScale,
                fontColor,
                thickness,
                lineType)

            if name in students:
                students.remove(name)
                print(students)
                current_time = now.strftime("%H-%M-%S")
                lnwriter.writerow([name,current_time])
    cv2.imshow("attendance system",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
f.close()
```

3.3 Adding GUI to the coding

We use Jupyter Notebook to create coding that automatically detects the students. Few libraries should be installed into the Jupyter Notebook terminal before starting to model the coding.

First, make sure these 5 libraries are installed into the Jupyter Notebook terminal.

- Numpy - library of Python to work with arrays
- Opencv-python(cv2) - library of Python bindings designed to solve computer vision problems
- Face recognition - library of Python to detect and recognise or detect face

- Pandas - library of Python for data manipulation and analysis
- PIL(Python Imaging Library) - library of Python for image processing tasks

The next step is importing these 10 libraries which are face_recognition, cv2, numpy, csv, os, datetime, tkinter, PIL, threading, pandas into the coding.

```
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime
import tkinter as tk
from PIL import Image, ImageTk
import threading
import pandas as pd
```

This step is to create the main window by using tkinter with window title and a listbox to show the information and attendance recorded.

```
# Create a Tkinter window
window = tk.Tk()
window.title("Attendance System")
canvas = tk.Canvas(window, width=2000, height=1500, bg="Light Yellow")
canvas.pack()

attendance_list = tk.Listbox(window, width=50,height=25)
attendance_list.place(x=1100, y=150)
```

To save the record of attendance in the backend by updating csv with a specific file path.

```
# Specify the path and name of the CSV file
csv_file_path = "attendance.csv"
```

Define a function to update the attendance list by recording the name and time in the csv file.

```
def update_attendance_list(name, time):
    attendance_list.insert(tk.END, f"{name} - {time}")
    with open(csv_file_path, 'a', newline='') as f:
        lnwriter = csv.writer(f)
        lnwriter.writerow([name, time])
```

Define a function to add information with entries bar to insert information like name, class section, date and also class name. At the bottom, include the “save” button to record the information entered.

```
def user_information():
    info_window = tk.Toplevel(window)
    info_window.title("User Information")
    info_window.configure(bg="Light Yellow")

    name_label = tk.Label(info_window, text="Name:",bg="Light Yellow")
    name_label.pack()

    name_entry = tk.Entry(info_window)
    name_entry.pack()

    class_section_label = tk.Label(info_window, text="Class Section:",bg="Light Yellow")
    class_section_label.pack()

    class_section_entry = tk.Entry(info_window)
    class_section_entry.pack()

    date_label = tk.Label(info_window, text="Date:",bg="Light Yellow")
    date_label.pack()

    date_entry = tk.Entry(info_window)
    date_entry.pack()

    class_name_label = tk.Label(info_window, text="Class Name:",bg="Light Yellow")
    class_name_label.pack()

    class_name_entry = tk.Entry(info_window)
    class_name_entry.pack()

    save_button = tk.Button(info_window, text="Save", command=lambda: save_information(name_entry.get(),
                        class_section_entry.get(), date_entry.get(), class_name_entry.get()),bg="orange")
    save_button.pack()
```

Define a function of save information where the inserted information is then saved and updated in the csv file.

```
def save_information(name, class_section, date, class_name):
    now = datetime.now()
    current_time = now.strftime("%H-%M-%S")

    with open(csv_file_path, 'a', newline='') as f:
        lnwriter = csv.writer(f)
        lnwriter.writerow([name, class_section, date, class_name, current_time])

    update_attendance_list(name, current_time)
```

The first two parts of label is just the naming of label, the third part is to initialize an empty list to store related information, followed by the part where current date and time is retrieved and formatted as file name and updated information to the csv with csv writer.

```
label = tk.Label(window, text="Attendance List")
label.config(font=("Arial", 20, "bold"), fg="white", bg="orange")
label.place(x=1150, y=100)

video_label = tk.Label(window)
video_label.place(x=100, y=120)

known_face_encodings = []
known_faces_names = []

students = []

face_locations = []
face_encodings = []
face_names = []

now = datetime.now()
current_date = now.strftime("%Y-%m-%d")

f = open(current_date+'.csv', 'w+', newline='')
lnwriter = csv.writer(f)
```

Define a function of update frame, capture the users face by frame to frame to increase the accuracy of detection. This also includes the part while, for and if to detect whether users face similar with the images stored. If yes then it will show the name of users is present and if no then it will show nothing.

```
def update_frame():
    global face_locations, face_encodings, face_names

    video_capture = cv2.VideoCapture(0)

    while s:
        # Capture frame-by-frame
        ret, frame = video_capture.read()

        if not ret:
            break

        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_small_frame = small_frame[:, :, ::-1]

        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
        face_names = []

        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
            name = ""
            face_distance = face_recognition.face_distance(known_face_encodings, face_encoding)
            best_match_index = np.argmin(face_distance)

            if matches[best_match_index]:
                name = known_faces_names[best_match_index]

            face_names.append(name)

            if name in known_faces_names:
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottom_left_corner_of_text = (10, 100)
                font_scale = 1.5
                font_color = (255, 0, 0)
                thickness = 3
                line_type = 2

                cv2.putText(frame, name + ' Present',
                            bottom_left_corner_of_text,
                            font,
                            font_scale,
                            font_color,
                            thickness,
                            line_type)

                if name in students:
                    students.remove(name)
                    print(students)
                    current_time = now.strftime("%H-%M-%S")
                    lnwriter.writerow([name, current_time])
                    update_attendance_list(name, current_time)
```

To convert the captured frame from BGR to RGB colour format and then create an ImageTk.

```
# Convert the frame to ImageTk format
img = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
imgtk = ImageTk.PhotoImage(image=img)
```

To update the image displayed with the newly captured frame.

```
# Update the label widget with the new frame
video_label.imgtk = imgtk
video_label.configure(image=imgtk)
```

To release camera resources and close the csv file properly with data saved.

```
# Release the video capture and close the CSV file
video_capture.release()
f.close()
```

Define a function of start attendance to connect the button of start attendance with reading the library to enable the process of face recognition to run simultaneously with the main application.

```
s = False

def start_attendance():
    global s

    if not s:
        s = True
        toggle_btn.config(text='Stop Attendance')
        threading.Thread(target=update_frame).start()
```

Define a function of stop attendance

```
def stop_attendance():
    global s

    if s:
        s = False
        toggle_btn.config(text='Start Attendance')
        video_label.configure(image='')
        video_label.image = None
```

Define a function of export attendance where the attendance will be exported in xlsx file in the specific file path for users to record the attendance.

```
def export_attendance():
    data = pd.read_csv(csv_file_path)
    export_file_path = os.path.join(os.path.expanduser("~"), "attendance.xlsx")
    data.to_excel(export_file_path, index=False)
    export_message.config(text=f"Attendance exported to: {export_file_path}")
```

Define a function of export of the current date file of csv and xlsx where both the attendance will be updated and saved in two file formats in the specific file path.

```
def export_current_date_file():
    current_file_path = os.path.join(os.path.expanduser("~"), current_date + '.csv')
    export_file_path = os.path.join(os.path.expanduser("~"), current_date + '_attendance.xlsx')
    data = pd.read_csv(current_file_path)
    data.to_excel(export_file_path, index=False)
    export_message.config(text=f"Current date file exported to: {export_file_path}")
```

Define the function on window close to close or quit the window.

```
def on_window_close():
    stop_attendance()
    window.destroy()
```

The formation of buttons with specific commands, so that when users click on it, it will respond respectively.

```
toggle_btn = tk.Button(window, text="Start Attendance", width=15, command=start_attendance,bg="orange")
toggle_btn.place(x=30, y=70)

stop_btn = tk.Button(window, text="Stop Attendance", width=15, command=stop_attendance,bg="orange")
stop_btn.place(x=180, y=70)

export_btn = tk.Button(window, text="Export Attendance", width=15, command=export_attendance,bg="orange")
export_btn.place(x=330, y=70)

user_info_btn = tk.Button(window, text="User Information", width=15, command=user_information,bg="orange")
user_info_btn.place(x=480, y=70)

export_curr_file_btn = tk.Button(window, text="Export Current Date File", width=20, command=export_current_date_file,bg="orange")
export_curr_file_btn.place(x=630, y=70)
```

An empty label to display static messages referring to different update status.

```
export_message = tk.Label(window, text="")
export_message.place(x=330, y=110)
```

To close the window when the on window close button is pressed.

```
# Bind the window close event to release resources
window.protocol("WM_DELETE_WINDOW", on_window_close)
```

Load the images of known faces and name them, so that the images are stored and users can be recognized once he/she is similar to the images.

```
# Load the known face encodings and names
jy_image = face_recognition.load_image_file("jy.jpg")
jy_encoding = face_recognition.face_encodings(jy_image)[0]
known_face_encodings.append(jy_encoding)
known_faces_names.append("Lim Joey")
students.append("Lim Joey")

kq_image = face_recognition.load_image_file("kq.jpg")
kq_encoding = face_recognition.face_encodings(kq_image)[0]
known_face_encodings.append(kq_encoding)
known_faces_names.append("Lim Ka Quan")
students.append("Lim Ka Quan")

abra_image = face_recognition.load_image_file("abra.jpg")
abra_encoding = face_recognition.face_encodings(abra_image)[0]
known_face_encodings.append(abra_encoding)
known_faces_names.append("Abraham Lim Bing Sern")
students.append("Abraham Lim Bing Sern")

zy_image = face_recognition.load_image_file("zy.jpg")
zy_encoding = face_recognition.face_encodings(zy_image)[0]
known_face_encodings.append(zy_encoding)
known_faces_names.append("Ooi Zi Ying")
students.append("Ooi Zi Ying")

jr_image = face_recognition.load_image_file("jr.jpg")
jr_encoding = face_recognition.face_encodings(jr_image)[0]
known_face_encodings.append(jr_encoding)
known_faces_names.append("Lim Jing Rou")
students.append("Lim Jing Rou")

zl_image = face_recognition.load_image_file("zl.jpg")
zl_encoding = face_recognition.face_encodings(zl_image)[0]
known_face_encodings.append(zl_encoding)
known_faces_names.append("Lee Zhi Lin")
students.append("Lee Zhi Lin")
```

To start the main loop event of the GUI application.

```
window.mainloop()
```

Below is the full coding of GUI:

```
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime
import tkinter as tk
from PIL import Image, ImageTk
import threading
import pandas as pd

# Create a Tkinter window
window = tk.Tk()
window.title("Attendance System")
canvas = tk.Canvas(window, width=2000, height=1500, bg="Light Yellow")
canvas.pack()

attendance_list = tk.Listbox(window, width=50,height=25)
attendance_list.place(x=1100, y=150)

# Specify the path and name of the CSV file
csv_file_path = "attendance.csv"

def update_attendance_list(name, time):
    attendance_list.insert(tk.END, f"{name} - {time}")
    with open(csv_file_path, 'a', newline='') as f:
        lnwriter = csv.writer(f)
```

```
        lnwriter.writerow([name, time])

def user_information():
    info_window = tk.Toplevel(window)
    info_window.title("User Information")
    info_window.configure(bg="Light Yellow")

    name_label = tk.Label(info_window, text="Name:",bg="Light Yellow")
    name_label.pack()

    name_entry = tk.Entry(info_window)
    name_entry.pack()

    class_section_label = tk.Label(info_window, text="Class Section:",bg="Light Yellow")
    class_section_label.pack()

    class_section_entry = tk.Entry(info_window)
    class_section_entry.pack()

    date_label = tk.Label(info_window, text="Date:",bg="Light Yellow")
    date_label.pack()

    date_entry = tk.Entry(info_window)
    date_entry.pack()

    class_name_label = tk.Label(info_window, text="Class Name:",bg="Light Yellow")
    class_name_label.pack()
```

```
    class_name_entry = tk.Entry(info_window)
    class_name_entry.pack()

    save_button = tk.Button(info_window, text="Save", command=lambda:
save_information(name_entry.get(),
                    class_section_entry.get(), date_entry.get(), class_name_entry.get()),bg="orange")
    save_button.pack()



def save_information(name, class_section, date, class_name):
    now = datetime.now()
    current_time = now.strftime("%H-%M-%S")

    with open(csv_file_path, 'a', newline='') as f:
        lnwriter = csv.writer(f)
        lnwriter.writerow([name, class_section, date, class_name, current_time])

    update_attendance_list(name, current_time)

label = tk.Label(window, text="Attendance List")
label.config(font=("Arial", 20, "bold"), fg="white", bg="orange")
label.place(x=1150, y=100)

video_label = tk.Label(window)
video_label.place(x=100, y=120)

known_face_encodings = []
known_faces_names = []
```

```
students = []

face_locations = []
face_encodings = []
face_names = []

now = datetime.now()
current_date = now.strftime("%Y-%m-%d")

f = open(current_date+'.csv', 'w+', newline='')
lnwriter = csv.writer(f)

def update_frame():
    global face_locations, face_encodings, face_names

    video_capture = cv2.VideoCapture(0)

    while s:
        # Capture frame-by-frame
        ret, frame = video_capture.read()

        if not ret:
            break

        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_small_frame = small_frame[:, :, ::-1]

        face_locations = face_recognition.face_locations(rgb_small_frame)
```

```
face_encodings = face_recognition.face_encodings(rgb_small_frame, face_locations)
face_names = []

for face_encoding in face_encodings:
    matches = face_recognition.compare_faces(known_face_encodings, face_encoding)
    name = ""
    face_distance = face_recognition.face_distance(known_face_encodings, face_encoding)
    best_match_index = np.argmin(face_distance)

    if matches[best_match_index]:
        name = known_faces_names[best_match_index]

    face_names.append(name)

    if name in known_faces_names:
        font = cv2.FONT_HERSHEY_SIMPLEX
        bottom_left_corner_of_text = (10, 100)
        font_scale = 1.5
        font_color = (255, 0, 0)
        thickness = 2
        line_type = 2

        cv2.putText(frame, name + ' Present',
                    bottom_left_corner_of_text,
                    font,
                    font_scale,
                    font_color,
                    thickness,
```

```
                    line_type)

            if name in students:
                students.remove(name)
                print(students)
                current_time = now.strftime("%H-%M-%S")
                lnwriter.writerow([name, current_time])
                update_attendance_list(name, current_time)

        # Convert the frame to ImageTk format
        img = Image.fromarray(cv2.cvtColor(frame, cv2.COLOR_BGR2RGB))
        imgtk = ImageTk.PhotoImage(image=img)

        # Update the label widget with the new frame
        video_label.imgtk = imgtk
        video_label.configure(image=imgtk)

    # Release the video capture and close the CSV file
    video_capture.release()
    f.close()

s = False

def start_attendance():
    global s

    if not s:
        s = True
```

```
        toggle_btn.config(text='Stop Attendance')
        threading.Thread(target=update_frame).start()

def stop_attendance():
    global s

    if s:
        s = False
        toggle_btn.config(text='Start Attendance')
        video_label.configure(image='')
        video_label.image = None

def export_attendance():
    data = pd.read_csv(csv_file_path)
    export_file_path = os.path.join(os.path.expanduser("~"), "test.xlsx")
    data.to_excel(export_file_path, index=False)
    export_message.config(text=f"Attendance exported to: {export_file_path}")

def export_current_date_file():
    current_file_path = os.path.join(os.path.expanduser("~"), current_date + '.csv')
    export_file_path = os.path.join(os.path.expanduser("~"), current_date + '_test.xlsx')
    data = pd.read_csv(current_file_path)
    data.to_excel(export_file_path, index=False)
    export_message.config(text=f"Current date file exported to: {export_file_path}")

def on_window_close():
    stop_attendance()
    window.destroy()
```

```
#button

toggle_btn = tk.Button(window, text="Start Attendance", width=15,
command=start_attendance,bg="orange")

toggle_btn.place(x=30, y=70)


stop_btn = tk.Button(window, text="Stop Attendance", width=15,
command=stop_attendance,bg="orange")

stop_btn.place(x=180, y=70)


export_btn = tk.Button(window, text="Export Attendance", width=15,
command=export_attendance,bg="orange")

export_btn.place(x=330, y=70)


user_info_btn = tk.Button(window, text="User Information", width=15,
command=user_information,bg="orange")

user_info_btn.place(x=480, y=70)


export_curr_file_btn = tk.Button(window, text="Export Current Date File", width=20,
command=export_current_date_file,bg="orange")

export_curr_file_btn.place(x=630, y=70)


export_message = tk.Label(window, text="")

export_message.place(x=330, y=110)


# Bind the window close event to release resources

window.protocol("WM_DELETE_WINDOW", on_window_close)


# Load the known face encodings and names

jy_image = face_recognition.load_image_file("jy.jpg")
```

```
jy_encoding = face_recognition.face_encodings(jy_image)[0]
known_face_encodings.append(jy_encoding)
known_faces_names.append("Lim Joey")
students.append("Lim Joey")

kq_image = face_recognition.load_image_file("kq.jpg")
kq_encoding = face_recognition.face_encodings(kq_image)[0]
known_face_encodings.append(kq_encoding)
known_faces_names.append("Lim Ka Quan")
students.append("Lim Ka Quan")

abra_image = face_recognition.load_image_file("abra.jpg")
abra_encoding = face_recognition.face_encodings(abra_image)[0]
known_face_encodings.append(abra_encoding)
known_faces_names.append("Abraham Lim Bing Sern")
students.append("Abraham Lim Bing Sern")

zy_image = face_recognition.load_image_file("zy.jpg")
zy_encoding = face_recognition.face_encodings(zy_image)[0]
known_face_encodings.append(zy_encoding)
known_faces_names.append("Ooi Zi Ying")
students.append("Ooi Zi Ying")

jr_image = face_recognition.load_image_file("jr.jpg")
jr_encoding = face_recognition.face_encodings(jr_image)[0]
known_face_encodings.append(jr_encoding)
known_faces_names.append("Lim Jing Rou")
students.append("Lim Jing Rou")
```

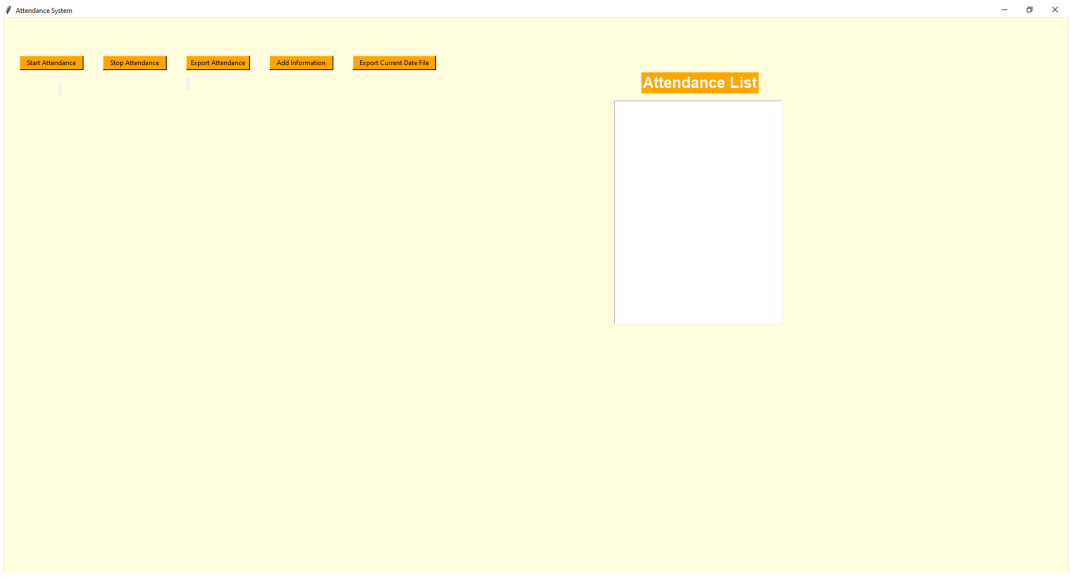
```
zl_image = face_recognition.load_image_file("zl.jpg")
zl_encoding = face_recognition.face_encodings(zl_image)[0]
known_face_encodings.append(zl_encoding)
known_faces_names.append("Lee Zhi Lin")
students.append("Lee Zhi Lin")

window.mainloop()
```

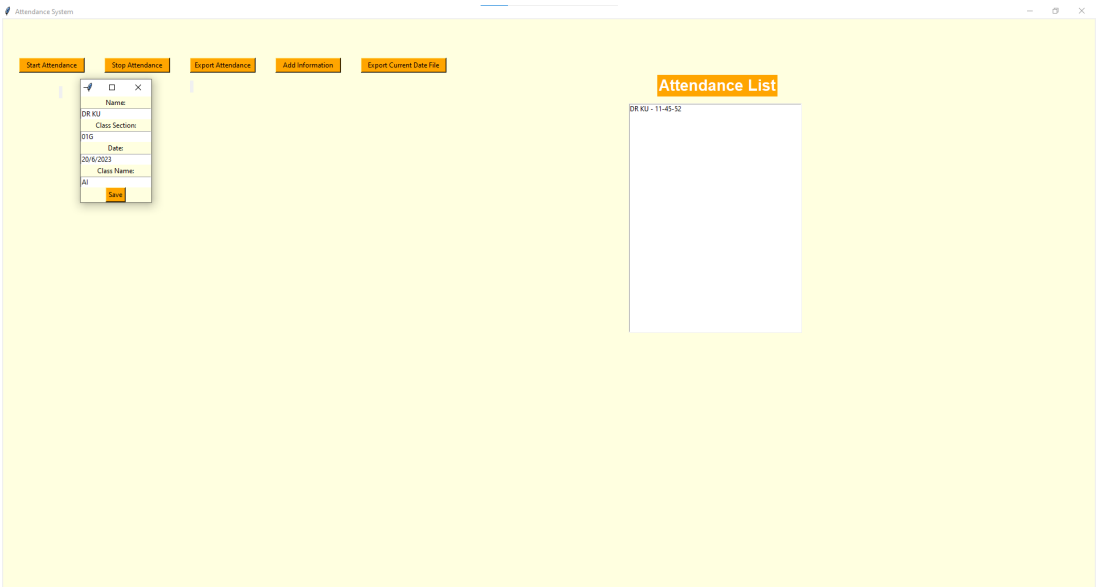
4.0 RESULTS AND DISCUSSION

4.1 Results

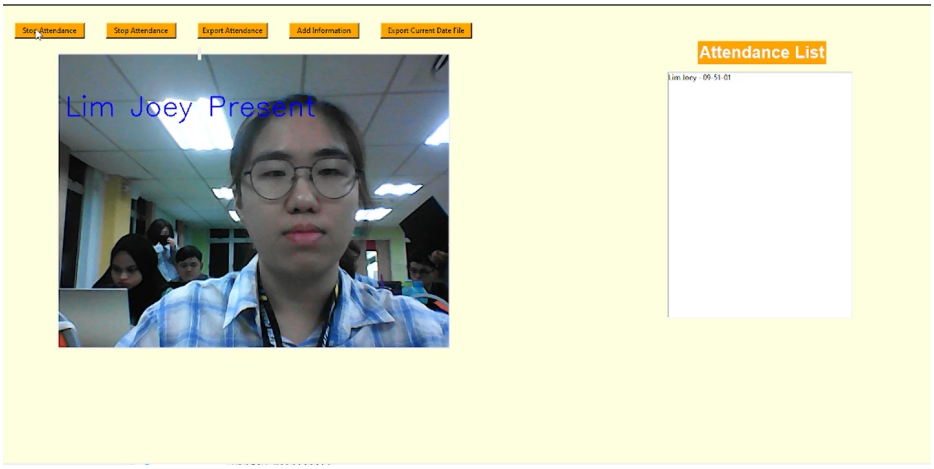
Firstly, we run the coding where the users would see the graphical user interface (GUI) with a few buttons of different functions and an empty boxes of attendance list at the right side of the panel.

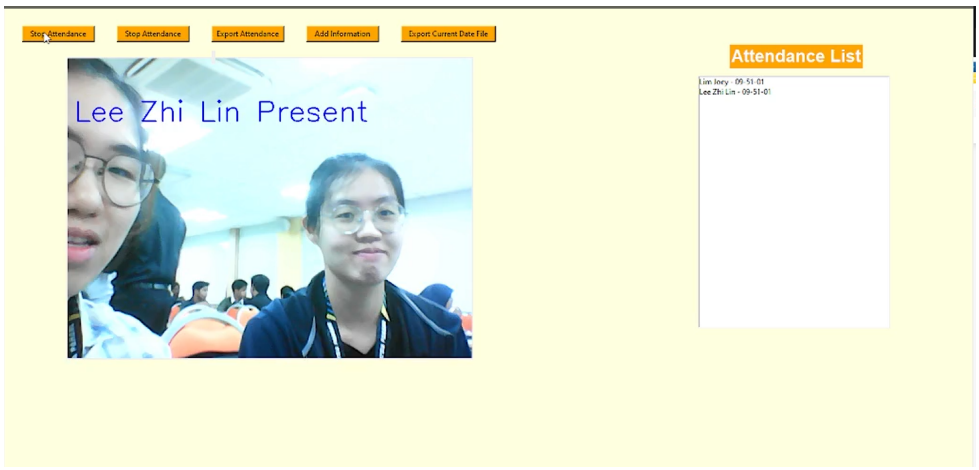


Then, users can click the button “add information”, it will jump to another window which will let users to key in the basic information such as name, class section, date, and class name. Users need to press save so that the information keyed in will display in the attendance list.



Once the user clicks the button “start attendance”, the GUI will pop out the camera to detect and recognise the users’ face. If the users contain an image stored in the system, then the system will recognise his/her face and show the message of users’ name. For example, Lim Joey present. Then, the name of users and the time (HH-MM-SS) where the attendance recorded will be shown in the attendance list also. Below are some of the successful examples from our coursemates:

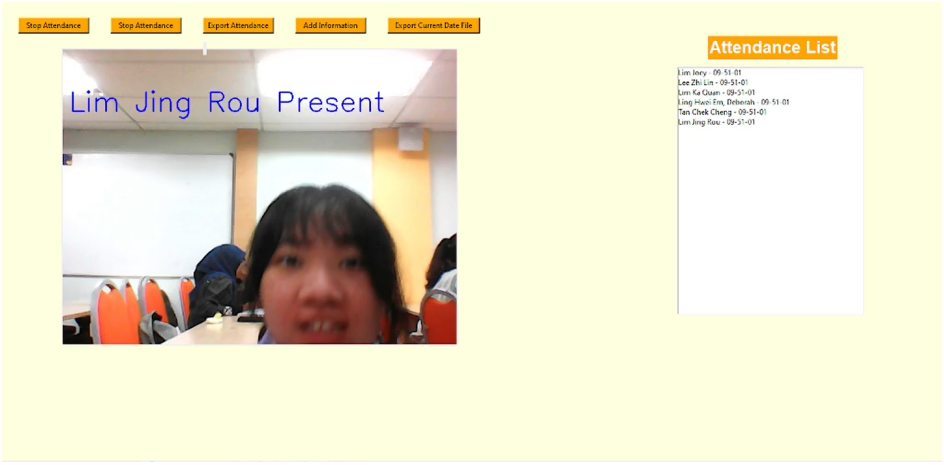


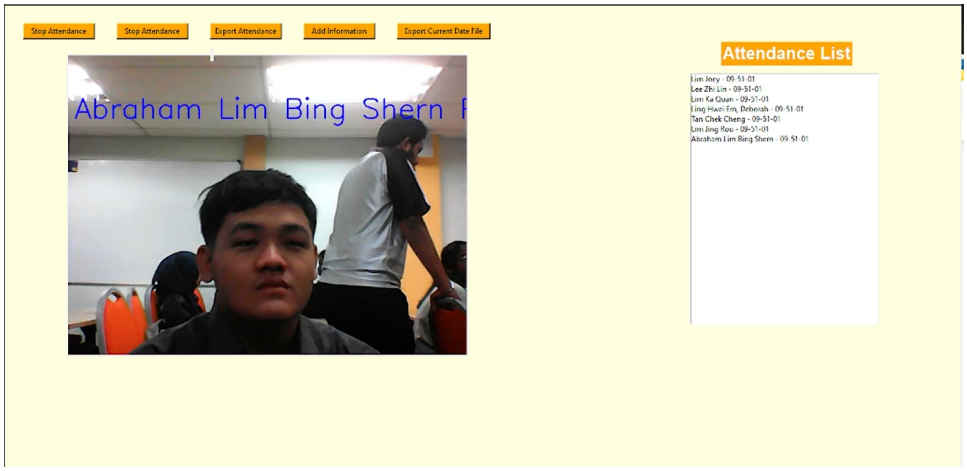


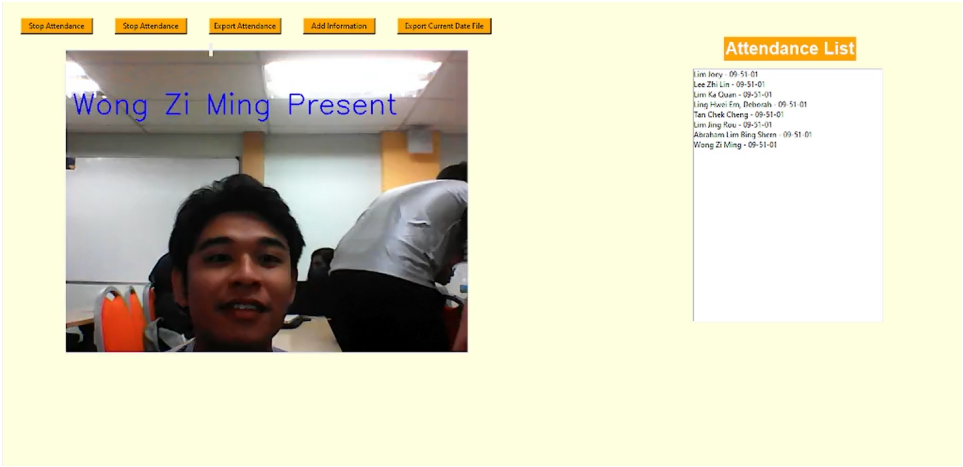

Stop Attendance
Stop Attendance
Export Attendance
Add Information
Export Current Date File
Lim Ka Quan Present
Attendance List
Lim Joey - 09-51-01
Lee Zhi Lim - 09-51-01
Lim Ka Quan - 09-51-01

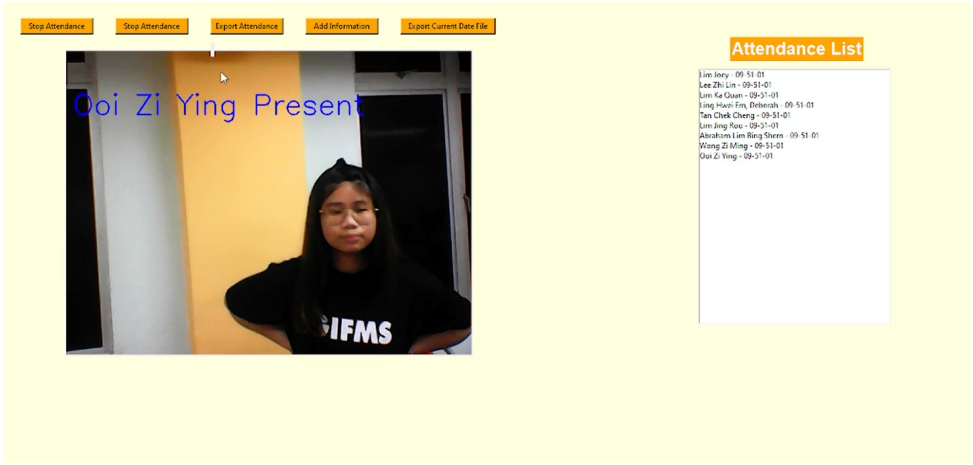

Stop Attendance
Stop Attendance
Export Attendance
Add Information
Export Current Date File
Ling Hwei Ern, Deborah Pr
Attendance List
Lim Joey - 09-51-01
Lee Zhi Lim - 09-51-01
Lim Ka Quan - 09-51-01
Ling Hwei Ern, Deborah - 09-51-01


Stop Attendance
Stop Attendance
Export Attendance
Add Information
Export Current Date File
Tan Chek Cheng Present
Attendance List
Lim Joey - 09-51-01
Lee Zhi Lim - 09-51-01
Lim Ka Quan - 09-51-01
Ling Hwei Ern, Deborah - 09-51-01
Tan Chek Cheng - 09-51-01


Stop Attendance
Stop Attendance
Export Attendance
Add Information
Export Current Date File
Lim Jing Rou Present
Attendance List
Lim Joey - 09-51-01
Lee Zhi Lim - 09-51-01
Lim Ka Quan - 09-51-01
Ling Hwei Fm, Deborah - 09-51-01
Tan Chek Cheng - 09-51-01
Lim Jing Rou - 09-51-01


Stop Attendance
Stop Attendance
Export Attendance
Add Information
Export Current Date File
Abraham Lim Bing Shern
Attendance List
Lim Joey - 09-51-01
Lee Zhi Lim - 09-51-01
Lim Ka Quan - 09-51-01
Ling Hwei Fm, Deborah - 09-51-01
Tan Chek Cheng - 09-51-01
Lim Jing Rou - 09-51-01
Abraham Lim Bing Shern - 09-51-01

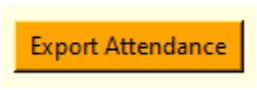


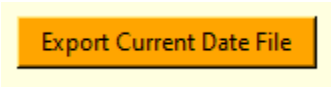


This is the attendance list which has recorded the name of each user and the current time they first detected by the system. This is also to help the lecturers to detect which students have been late for class.

Based on various test runs of results, the detection of face recognition for the attendance system is quite accurate under the condition where the users already have images saved in the system.

Apart from that, there are also buttons for “export attendance” and “export current date file”. These two buttons help to export the attendance list into the csv file and xlsx file as record and reference. The “export attendance” button is to export only the csv file that has all records of detection from the beginning of the system. It is more to the back-end where it records the whole process of the detection flow. However, the “export current date file” can be exported in csv and xlsx which enable users to download the attendance list for reference.





4.2 Limitations

The accuracy of face recognition can be affected by various factors such as size of image in storage, resolution, quality of input images or lighting conditions. This is because as the size of the image in storage increases, the algorithm needs to compare the face detected with the image in the storage. Thus, the time required for matching will vary and slow down the face recognition process indirectly due to different hardware used.

Additionally, resizing the image to a smaller size will speed up the face recognition process but after resizing the image, there will come the cost of reduced face recognition accuracy due to loss of facial details and features because of the reduction of image pixels. So it is often a trade off between speed of processing and the accuracy of the face recognition.

Besides the resolution quality of the input images also have the potential to lead to false positives. As the resolution quality of the input images are low, there will exist some faces with similar facial features, thus it can result in higher chances on machine encounter if a situation matches the detected face with a wrong person from the database.

Lighting conditions can affect the colour of skin tones. Lighting with warm or cool colour may impact the colour based feature used by face recognition systems and lead to recognition errors. To resolve this problem, we may convert the image to grayscale instead of BGR colour to simplify the processing and reduce the computing complexity since it requires less memory and computing resources compared to colourful images.

5.0 CONCLUSION

In this project, we have developed a system that automatically records attendance using face recognition. It can correctly identify the majority of the numerous user photos it saves. However, occasionally it is unable to identify people whose photos are not saved. There were instances where it incorrectly identified distinct names for people who had comparable physical characteristics.

Despite these limitations, our project shows that automatic face recognition is effective for making attendance easier. Lecturers and instructors do not need to record the attendance of students manually with paper and pen but to get the attendance list with the help of the system. The system can also help to detect the person who is late to class. Our solution provides a convenient alternative to human tracking methods by identifying people by their faces. It was a convenient alternative to traditional manual tracking methods, such as sign-in sheets or ID cards.

To improve the system, we can work on making it more accurate and reliable, especially for people whose images are not already stored. We also need to address the challenge of similar-looking individuals to reduce mistakes and make the system perform better. Overall, our project lays the foundation for future developments in the field of attendance systems facial recognition technology.

REFERENCES

Kaspersky. (2021, January 13). *What is Facial Recognition – Definition and Explanation*. Kaspersky. https://www.kaspersky.com/resource-center/definitions/what-is-facial-recognition

EFF. (2019, March 7). *Face Recognition*. Electronic Frontier Foundation. https://www.eff.org/pages/face-recognition

IJRASET, Y. K. (2022, June 13). *Automated Attendance System Using Face Recognition*. Www.ijraset.com. https://www.ijraset.com/research-paper/automated-attendance-system-using-face-recognition

Joshan Athanesious, J., Vanitha, Adithya, S., Bhardwaj, C. A., Lamba, J. S., & Vaidehi, A. V. (2019). Deep Learning Based Automated Attendance System. *Procedia Computer Science*, *165*, 307–313. https://doi.org/10.1016/j.procs.2020.01.045

Luxand, R. (2019, August 8). *Detect and Recognize Faces and Facial Features with Luxand FaceSDK*. Www.luxand.com. https://www.luxand.com/facesdk/?utm_source=google&utm_medium=cpc&utm_campaign=face-recognition-system&gclid=Cj0KCQjwnMWkBhDLARIsAHBOftqVm7-x4E2y9ly2_mdPBqtA1ac6GgU3k9biRCcZpK1OrMZBpCFKDK4aAkOEEALw_wcB

Nguyen, D. D., Nguyen, X. H., Than, T. T., & Nguyen, M. S. (2021, December 1). *Automated Attendance System in the Classroom Using Artificial Intelligence and Internet of Things Technology*. IEEE Xplore. https://doi.org/10.1109/NICS54270.2021.9700991

APPENDICES

Appendix 1

https://drive.google.com/drive/folders/1w0ZaKtDTV6ZyUC8pFtyAe07shPdSz2Ls?usp=sharing