SMART ATTENDENCE SYSTEM USING REALTIME DATA BASES

Note that the software used for this project was PyCharm and a web-based software was used which is firebase. The video links will be provided in the repository.

1. Main.py

import os

import pickle

import numpy as np

import cv2

import face\_recognition

import cvzone

import firebase\_admin

from firebase\_admin import credentials

from firebase\_admin import db

from firebase\_admin import storage

import numpy as np

from datetime import datetime

cred = credentials.Certificate("serviceAccountKey.json")

firebase\_admin.initialize\_app(cred, {

'databaseURL': "",

'storageBucket': ""

})

bucket = storage.bucket()

cap = cv2.VideoCapture(1)

cap.set(3, 640)

cap.set(4, 480)

imgBackground = cv2.imread('Resources/background.png')

# Importing the mode images into a list

folderModePath = 'Resources/Modes'

modePathList = os.listdir(folderModePath)

imgModeList = []

for path in modePathList:

imgModeList.append(cv2.imread(os.path.join(folderModePath, path)))

# print(len(imgModeList))

# Load the encoding file

print("Loading Encode File ...")

file = open('EncodeFile.p', 'rb')

encodeListKnownWithIds = pickle.load(file)

file.close()

encodeListKnown, studentIds = encodeListKnownWithIds

# print(studentIds)

print("Encode File Loaded")

modeType = 0

counter = 0

id = -1

imgStudent = []

while True:

success, img = cap.read()

imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)

imgS = cv2.cvtColor(imgS, cv2.COLOR\_BGR2RGB)

faceCurFrame = face\_recognition.face\_locations(imgS)

encodeCurFrame = face\_recognition.face\_encodings(imgS, faceCurFrame)

imgBackground[162:162 + 480, 55:55 + 640] = img

imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

if faceCurFrame:

for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):

matches = face\_recognition.compare\_faces(encodeListKnown, encodeFace)

faceDis = face\_recognition.face\_distance(encodeListKnown, encodeFace)

# print("matches", matches)

# print("faceDis", faceDis)

matchIndex = np.argmin(faceDis)

# print("Match Index", matchIndex)

if matches[matchIndex]:

# print("Known Face Detected")

# print(studentIds[matchIndex])

y1, x2, y2, x1 = faceLoc

y1, x2, y2, x1 = y1 \* 4, x2 \* 4, y2 \* 4, x1 \* 4

bbox = 55 + x1, 162 + y1, x2 - x1, y2 - y1

imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)

id = studentIds[matchIndex]

if counter == 0:

cvzone.putTextRect(imgBackground, "Loading", (275, 400))

cv2.imshow("Face Attendance", imgBackground)

cv2.waitKey(1)

counter = 1

modeType = 1

if counter != 0:

if counter == 1:

# Get the Data

studentInfo = db.reference(f'Students/{id}').get()

print(studentInfo)

# Get the Image from the storage

blob = bucket.get\_blob(f'Images/{id}.png')

array = np.frombuffer(blob.download\_as\_string(), np.uint8)

imgStudent = cv2.imdecode(array, cv2.COLOR\_BGRA2BGR)

# Update data of attendance

datetimeObject = datetime.strptime(studentInfo['last\_attendance\_time'],

"%Y-%m-%d %H:%M:%S")

secondsElapsed = (datetime.now() - datetimeObject).total\_seconds()

print(secondsElapsed)

if secondsElapsed > 30:

ref = db.reference(f'Students/{id}')

studentInfo['total\_attendance'] += 1

ref.child('total\_attendance').set(studentInfo['total\_attendance'])

ref.child('last\_attendance\_time').set(datetime.now().strftime("%Y-%m-%d %H:%M:%S"))

else:

modeType = 3

counter = 0

imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

if modeType != 3:

if 10 < counter < 20:

modeType = 2

imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

if counter <= 10:

cv2.putText(imgBackground, str(studentInfo['total\_attendance']), (861, 125),

cv2.FONT\_HERSHEY\_COMPLEX, 1, (255, 255, 255), 1)

cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),

cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (255, 255, 255), 1)

cv2.putText(imgBackground, str(id), (1006, 493),

cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (255, 255, 255), 1)

cv2.putText(imgBackground, str(studentInfo['standing']), (910, 625),

cv2.FONT\_HERSHEY\_COMPLEX, 0.6, (100, 100, 100), 1)

cv2.putText(imgBackground, str(studentInfo['year']), (1025, 625),

cv2.FONT\_HERSHEY\_COMPLEX, 0.6, (100, 100, 100), 1)

cv2.putText(imgBackground, str(studentInfo['starting\_year']), (1125, 625),

cv2.FONT\_HERSHEY\_COMPLEX, 0.6, (100, 100, 100), 1)

(w, h), \_ = cv2.getTextSize(studentInfo['name'], cv2.FONT\_HERSHEY\_COMPLEX, 1, 1)

offset = (414 - w) // 2

cv2.putText(imgBackground, str(studentInfo['name']), (808 + offset, 445),

cv2.FONT\_HERSHEY\_COMPLEX, 1, (50, 50, 50), 1)

imgBackground[175:175 + 216, 909:909 + 216] = imgStudent

counter += 1

if counter >= 20:

counter = 0

modeType = 0

studentInfo = []

imgStudent = []

imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]

else:

modeType = 0

counter = 0

# cv2.imshow("Webcam", img)

cv2.imshow("Face Attendance", imgBackground)

cv2.waitKey(1)

1. EncodeGenerator.py

import cv2

import face\_recognition

import pickle

import os

import firebase\_admin

from firebase\_admin import credentials

from firebase\_admin import db

from firebase\_admin import storage

cred = credentials.Certificate("serviceAccountKey.json")

firebase\_admin.initialize\_app(cred, {

'databaseURL': "",

'storageBucket': ""

})

# Importing student images

folderPath = 'Images'

pathList = os.listdir(folderPath)

print(pathList)

imgList = []

studentIds = []

for path in pathList:

imgList.append(cv2.imread(os.path.join(folderPath, path)))

studentIds.append(os.path.splitext(path)[0])

fileName = f'{folderPath}/{path}'

bucket = storage.bucket()

blob = bucket.blob(fileName)

blob.upload\_from\_filename(fileName)

# print(path)

# print(os.path.splitext(path)[0])

print(studentIds)

def findEncodings(imagesList):

encodeList = []

for img in imagesList:

img = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

encode = face\_recognition.face\_encodings(img)[0]

encodeList.append(encode)

return encodeList

print("Encoding Started ...")

encodeListKnown = findEncodings(imgList)

encodeListKnownWithIds = [encodeListKnown, studentIds]

print("Encoding Complete")

file = open("EncodeFile.p", 'wb')

pickle.dump(encodeListKnownWithIds, file)

file.close()

print("File Saved")

1. AddDatatoDatabase.py

import firebase\_admin

from firebase\_admin import credentials

from firebase\_admin import db

cred = credentials.Certificate("serviceAccountKey.json")

firebase\_admin.initialize\_app(cred, {

'databaseURL': ""

})

ref = db.reference('Students')

data = {

"321654":

{

"name": "Murtaza Hassan",

"major": "Robotics",

"starting\_year": 2017,

"total\_attendance": 7,

"standing": "G",

"year": 4,

"last\_attendance\_time": "2022-12-11 00:54:34"

},

"852741":

{

"name": "Emly Blunt",

"major": "Economics",

"starting\_year": 2021,

"total\_attendance": 12,

"standing": "B",

"year": 1,

"last\_attendance\_time": "2022-12-11 00:54:34"

},

"963852":

{

"name": "Elon Musk",

"major": "Physics",

"starting\_year": 2020,

"total\_attendance": 7,

"standing": "G",

"year": 2,

"last\_attendance\_time": "2022-12-11 00:54:34"

}

}

for key, value in data.items():

ref.child(key).set(value)