Logo, company name

Description automatically generated

**GROUP PROJECT**

**BSD2513 ARTIFICIAL INTELLIGENCE**

**SEMESTER II 2021/2022**

**LECTURER’S NAME: DR KU MUHAMMAD NA’IM KU KHALIF**

|  |  |  |
| --- | --- | --- |
| **Matric ID** | **Name** | **Section** |
| SD20003 | MUHAMMAD AMIR BIN AMINUDDIN | 01G |
| SD20025 | EIFFEL SYAFHAMKA NADIMEN A/L SYAMSUL | 01G |
| SD20033 | HAFIZ ASYRAAF BIN GHANI | 01G |
| SD20039 | MUHAMMAD ‘ARIF BIN MOHD ANUWA | 01G |
| SD20041 | QIRYN ADRIANA BINTI KHARUL ZAMAN | 01G |

**TABLE OF CONTENTS**

1. Executive Summary------------------------------------------------------------------------ 3
   1. Project Description---------------------------------------------------------------- 3
   2. Problems to be solved ------------------------------------------------------------ 3
   3. Basic data description------------------------------------------------------------ 3
2. Summary of the Project Context and Objectives ------------------------------------- 4
   1. Summary of the project ---------------------------------------------------------- 4
   2. Objectives ------------------------------------------------------------------------- 4
3. Methodology --------------------------------------------------------------------------------5
   1. Data Acquisition ------------------------------------------------------------------ 5
      1. Image acquisition-----------------------------------------------------5
      2. Dataset creation------------------------------------------------------ 5
      3. Storing-----------------------------------------------------------------6
   2. Face Recognition Process------------------------------------------------------- 6
      1. Face Detection and Extraction------------------------------------ 6
      2. Face Positioning----------------------------------------------------- 7
      3. Face Encoding------------------------------------------------------- 7
      4. Face Matching------------------------------------------------------- 7
      5. Attendance Marking------------------------------------------------ 7

Result And Discussion------------------------------------------------------------------------ 8

Registering New Face------------------------------------------------------------------------- 9

Marking Attendance-------------------------------------------------------------------------- 11

Conclusion------------------------------------------------------------------------------------- 13

Appendices------------------------------------------------------------------------------------- 14

References-------------------------------------------------------------------------------------- 22

**1.0 Executive Summary**

**1.1 Project Description**

The Face Recognition based Attendance System is a technology that makes use of facial recognition to identify a person and mark their attendance automatically using Artificial Intelligence (AI). This project system is contactless, as it doesn’t require physical interaction with the machine. With that, this system uses image processing, namely facial recognition software to record and compare patterns on a person’s face as well as analyses the features, in order to identify and authenticate the correct individual. For this project, it can be divided into a few major step s:

1. Student information acquisition, add new faces, name and their student ID into the dataset to be in the attendance list.
2. Face detection, to locate the student face in real time to be analysed.
3. Data Transformation, compute 128-d face embedding via deep metric learning and compare to dataset.
4. Face recognition matches the data and verifies the student attendance.

**1.2 Problems to be solved**

1. Can the attendance record process be improved using face recognition?
2. Is the implementation of AI and Machine learning to recognise people’s faces better than manual attendance records?
3. Can Face Recognition based Attendance System avoid people from lying about their attendance?

**1.3 Basic data description**

The data used are those of students in this group. The system takes 40 pictures, 30 pictures will be processed by machine learning to recognize and authenticate the correct individuals. The 30 pictures will be placed in the train file while the other 10 will be put in the val file. The data also include student information such as name and student ID to be known as their roll number.

**2.0 Summary of the Project Context and Objectives**

**2.1 Summary of the project**

As the name says this project takes attendance using biometrics (in this case, face) and is one of the most famous projects amongst college students out there. This project is tracking attendance by using machine learning. These methods can easily track or scan the face if you have registered from the register faces recognition coding (from jupyter, file from add\_new\_faces). In this code, we are using advanced packages such as tensorflow, sklearn, joblib and numpy. After registering by running the code, the windows camera will appear. It will snap your face to store it in the file(faces file). The name, faces and roll number will be saved in the specific file that sync with the code.

After that, we should run in the live face recognition attendance coding (from jupyter, file from live attendance). We are using the advanced coding that goes to add\_new\_faces code, but we added date and time to track the student’s attendance. After running, the windows camera will appear. It will show the green square around the student face who has registered from the system with the name and matric number. The system will capture and show the statement-” Attendance marked for \_\_\_{name} whose matric number is \_\_\_{matric\_id}”. The statement is approved that the students are present during that time.

**2.2 Objectives**

* To easily mark students attendance without using manual system such as sign the attendance
* To mark the student attendance more efficient.
* To apply Artificial Intelligence (AI) in the student lifestyle.
* To avoid students cheating about their attendance.
* More exact time marks of the students’ attendance.

**3.0 Methodology**

The proposed system is designed for automating the attendance of the students in the class and replaces the traditional way such as filling in attendance sheets manually or punch card attendance system. The data of students are entered by the administrator, and when the time for the corresponding subject arrives, the system automatically begins snapping photos and determining whether human faces appear in the image.

**3.1 Data Acquisition**

**3.1.1 Image acquisition**

Image is acquired when the administrators run the option where they can add new faces. The students need to go through a process where their image is captured by the system’s camera. Those images are given as an input to the system.

**3.1.2 Dataset creation**

Dataset of students is created before the recognition process. In this project, we have created a dataset of 5 students which involves their name, roll number, and images of students in different poses and variations. For better accuracy, 40 images of each student should be captured. 30 images will be saved in the ‘train’ file and another 10 images will be saved in the ‘val’ file. These two folders which are train and val are in the ‘faces’ file. When we register a student’s data and images in our system to create a dataset, deep learning is applied to each face to compute 128-d facial features and save them in the student face data file so that we may recall that face throughout the recognition process. This procedure is followed for each image captured throughout the registration process.

**3.1.3 Storing**

We used an excel file to store the students’ data. Each time we create data for a student, it will automatically save the inputs entered the file.

**3.2 Face recognition process**

A picture containing diagram

Description automatically generated

***Figure 1 :*** *Block Diagram*

**3.2.1 Face Detection and Extraction**

Face detection is significant because when a system receives an image from a camera, a face detection method is used to identify the human faces in that picture. A variety of image processing techniques are used to recognise faces in images and also the position of those discovered faces.

**3.2.2 Face Positioning**

In a human face, there are 68 distinct points. To put it another way, there are 68 facial landmarks. This step's main purpose is to recognise face landmarks and position the image. Python script is utilised to automatically recognise facial landmarks and position the face as much as possible without distorting the image.

**3.2.3 Face Encoding**

Following the detection of faces in a given image, the following step is to extract each image's unique distinguishing facial characteristic. Whenever we acquire a face localization, we extract 128 key facial points for each image input that are very accurate, and we save these 128-d facial points in a data file for face recognition.

**3.2.4 Face Matching**

This is the final stage in the face recognition procedure. Deep metric learning, which is very precise and capable of generating actual value feature vectors, was employed as one of the greatest learning techniques. The faces are ratified by our system, which creates a 128-d embedding (ratification) for each one.

**3.2.5 Attendance Marking**

When the face is recognised by the picture saved in a file, Python produces roll numbers for current students and returns them. When the data is returned, the system generates an attendance table containing the name and roll number. The data is then passed to Python, which automatically stores the table in an Excel file. Each sheet is saved according to the subjects that the administrator has already entered. For example, when the system generates an excel sheet by sending the compiled sheet in an array to python, the python first checks whether there is an existing excel sheet for that date, and if there is, it creates a separate worksheet by subject id, allowing attendance to be differentiated for different subjects.

# **Result And Discussion**

Graphical user interface, application

Description automatically generatedIn this section we will discuss the features of the application for Facial Recognition Base Attendance, as well as the guided user interface (GUI) of the program. We incorporated the knowledge of python tkinter library in the process of creating the GUI.

**Figure 4.1** Main window of Live Attendance

## Registering New face

A picture containing timeline

Description automatically generatedFor the live attendance system to function properly, it is required for each student to register their face **once**. Only then can the recognition process begin. When a user clicks the "Register New Face" button, the program will pop a second window, prompting user to enter their NAME and MATRIC ID.When user click the "Add Face" button, the webcam will open, and after 5 seconds, the burst mode will be triggered to snap 40 pictures of the user's face in an instant.

**Figure 4.2** Input Student Details

A screen shot of a video game

Description automatically generated with low confidence

**Figure 4.3** First 30 image is saved in Test Folder

A picture containing text, wall, screenshot, posing

Description automatically generated

**Figure 4.4** Last 10 image saved in Validation folder

## Marking Attendance

Marking students’ attendance is the main motive of this program. Clicking the ‘Live Attendance’ button in the main menu will turn the webcam on for face recognition process. The user must clearly position their face in front of the webcam until a pink square border and user information appears around their face. If a face is identified, this indicates that the recognition procedure has been successful, and the user's name, ID, and timestamp will be automatically recorded in an excel sheet as attendance. To close the camera, simply press the esc key.

A person wearing headphones

Description automatically generated with medium confidence

**Figure 4.5** User's Face Recognizing

Graphical user interface, application, table, Excel

Description automatically generated

**Figure 4.7** Generated Attendance sheet

**Conclusion**

The conclusion we can conclude from this project is that attendance face detection has many contributions that can be given to society, the environment and the system where with the presence of attendance face detection can facilitate many people in filling their presence in various sectors from the government sector or private sector.

In addition, it can also save the environment where the use of pencils or pens is no longer used to fill the presence because the use of pencils requires cutting down trees to produce the pencil, then for the pen it is made of plastic where he plastics used in the manufacture of the exterior of the pen can contain not only dioxins but also other chemicals that are equally harmful to human health as well as the environment. To overcome the problem is only need attendance face detection to overcome the problem, with the presence of attendance face detection then the environment can be saved

In addition, the use of attendance face detection can improve and advance technology systems in today's world where most of our daily lives use a lot of attendance face detection such as on smartphones, laptops, and also in the security of banks and others.

We improve the attendance process by using face recognition clearly and well. This will provide a great benefit where the time taken to take attendance can be shortened and speeded up with the presence of face recognition. Therefore, the attendance record process can be improved by using face recognition.

The implementation of AI and machine learning can recognize people's faces better than manual attendance records where facial image data is taken first before performing a face recognition scan to ensure the scanned face is accurate and correct and also not exchanged with people. others.Therefore, the implementation of AI and Machine learning to recognize people's faces better than manual attendance records.

**APPENDICES**

1. **MAIN GUI CODE**

**from** tkinter **import** \*

**import** os

**import** shared

**from** time **import** strftime

*# 2nd window: REGISTER NEW FACE WINDOW*

**def** register():*# window to input student name and matricID and add face*

**global** register\_screen

    register\_screen =Toplevel(main\_screen)

    register\_screen.title("Register New Face")

    register\_screen.geometry("300x250")

    register\_screen.config(bg="#a6dbbc")

**global** name\_info

**global** studID\_info

**global** name\_entry

**global** studID\_entry

    name= StringVar()

    studID= StringVar()

    Label(register\_screen, text="Please enter details below",font=('Arial', 12,'bold'), fg='#69473E',bg='#8fb6ab',width="200").pack()

    Label(register\_screen, text="",bg="#a6dbbc").pack()

    name\_lable = Label(register\_screen, text="Student Name \* ",font=('Arial', 12,'bold'),bg="#a6dbbc",fg="#98450b")

    name\_lable.pack()

    name\_entry = Entry(register\_screen, textvariable=name) *# name input box*

    name\_entry.pack()

    studID\_lable = Label(register\_screen, text="Matric ID \* ",font=('Arial', 12,'bold'),bg="#a6dbbc",fg="#98450b")

    studID\_lable.pack()

    studID\_entry = Entry(register\_screen, textvariable=studID) *#matric ID input box*

    studID\_entry.pack()

    Label(register\_screen, text="",bg="#a6dbbc").pack()

    Button(register\_screen, text="Add Face",font=('Arial', 11,'bold'),bg="#E6EDEC",fg="#007782" ,height="1", width="18", command = register\_user).pack()

**def** register\_user ():

    shared.name=name\_entry.get() *#store user input in shared.py module to be access by other script*

    shared.studID=studID\_entry.get()

    %run Add\_new\_face.py  *#run Add\_new\_face.py*

    name\_entry.delete(0, END)

    studID\_entry.delete(0, END)

    Label(register\_screen, text="Registration Complete!",bg="#a6dbbc", fg="brown", font=("calibri", 11,'bold')).pack()

**def** liveAttendance():

*#run live\_attendance.py*

    %run Live\_attendance.py

    Label(main\_screen, text="Attendance Taken !",bg="#a6dbbc", fg="purple", font=("calibri", 11,'bold')).pack()

*# 1st window: MAIN WINDOW*

**def** main\_account\_screen():

**global** main\_screen

    main\_screen = Tk()

    main\_screen.geometry("300x250")

    main\_screen.config(bg="#a6dbbc")

    main\_screen.title("LiveAttendance")

    Label(text="Welcome! Select your choice",font=('Arial', 12,'bold'), fg='#69473E',bg='#8fb6ab', width="300", height="2").pack()

*#clock widget*

**def** time():

        string = strftime('%H:%M:%S %p')

        mark.config(text = string)

        mark.after(1000, time)

    mark = Label( main\_screen ,font = ('calibri', 22, 'bold'),

                pady=10,foreground = '#8A4888',bg="#a6dbbc")

    mark.pack(anchor = 'center')

    time()

    Button(text="Register New Face",font=('Arial', 11,'bold'),bg="#E6EDEC",fg="#007782",height="2", width="30", command=register).pack() *#button 1*

    Label(text="",bg="#a6dbbc").pack()

    Button(text="Live Attendance",font=('Arial', 11,'bold'),bg="#E6EDEC",fg="#007782" ,height="2", width="30", command = liveAttendance).pack() *#button 2*

    main\_screen.mainloop()

main\_account\_screen()

1. **ADD NEW FACE CODE**

**import** cv2

**import** os

**import** joblib

**import** numpy **as** np

**import** time

**import** sys

**from** PIL **import** Image

**from** mtcnn.mtcnn **import** MTCNN

**from** tensorflow.keras.models **import** load\_model

**from** sklearn.preprocessing **import** Normalizer, LabelEncoder

**from** sklearn.ensemble **import** RandomForestClassifier

**import** shared *#importing using input stored in shared.py*

*######### if these directories doesn't exist, make them #######*

**try**:os.makedirs('faces')

**except**:**pass**

**try**:os.makedirs('faces/train')

**except**:**pass**

**try**:os.makedirs('faces/val')

**except**:**pass**

*############### take name and roll no from user input in main\_GUI.py script #################*

stud\_name = str(shared.name)

roll = str(shared.studID)

name = stud\_name+'-'+roll

*######### make face folders in train and val ##################*

**if** name **in** os.listdir('faces/train'):

**print**('User already exist')

**else**:

    os.makedirs('faces/train/'+name)

    os.makedirs('faces/val/'+name)

    cap = cv2.VideoCapture(0)  *#open cam to add face*

    i = 0

**print**()

**for** i **in** range(5):

**print**(f'Capturing starts in {5-i} seconds...')

        time.sleep(1)

**print**('Taking photos...')

**while** i<=200:

        ret,frame = cap.read()

        cv2.imshow('taking your pictures',frame)

**if** i%5==0 **and** i<=150 **and** i!=0:

            cv2.imwrite('faces/train/'+name+'/'+str(i)+'.png',frame)

**elif** i%5==0 **and** i>150:

            cv2.imwrite('faces/val/'+name+'/'+str(i)+'.png',frame)

        i+=1

**if** cv2.waitKey(1)==27:  *#Esc Key to close cam window*

**break**

    cv2.destroyAllWindows()

    cap.release()

**print**('Successfully taken your photos!')

*#### Below part is just training the model with the newly added face. Here we are creating model.*

embedding\_model = load\_model('models/facenet\_keras.h5')

**print**('Embedding Model Loaded')

*# making a a mtcnn instance for detecting faces*

detector = MTCNN()

**def** find\_face(img,img\_size=(160,160)):

    img = cv2.imread(img)

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

    img = np.asarray(img) *# converting our image obj to numpy array*

    faces = detector.detect\_faces(img)

**if** faces:

        x,y,w,h = faces[0]['box']

        x,y=abs(x),abs(y)

        face = img[y:y+h,x:x+w]

        face = Image.fromarray(face) *# converting it to image object to resize it*

        face = face.resize(img\_size) *# resizing it*

        face = np.asarray(face)      *# converting it back to array*

**return** face

**return** None

**def** embed(face):

    face = face.astype('float32')

    fm,fs = face.mean(),face.std()

    face = (face-fm)/fs *# standardizing the data*

    face = np.expand\_dims(face,axis=0) *# flattening it*

    embs = embedding\_model.predict(face) *# embedding model converts our160\*160\*3 vector to 128 features*

**return** embs[0]

**def** load\_dataset(path):

    X = []

    y = []

**for** people **in** os.listdir(path):

**for** people\_images **in** os.listdir(path+people):

            face = find\_face(path+people+'/'+people\_images)

**if** face **is** None:**continue**

            emb = embed(face)

            X.append(emb)

            y.append(people)

**print**('Loaded {} images of {}'.format(len(os.listdir(path+'/'+people)),people))

**return** np.asarray(X),np.asarray(y)

*########### Loading training and testing data using functions defined above #############*

**print**('Loading train data...')

X\_train, y\_train = load\_dataset('faces/train/')

**print**()

**print**('Loading test data...')

X\_test, y\_test = load\_dataset('faces/val/')

*# l2 normalizing the data*

l2\_normalizer = Normalizer('l2')

X\_train = l2\_normalizer.transform(X\_train)

X\_test  = l2\_normalizer.transform(X\_test)

*#label encoding the y data*

label\_enc = LabelEncoder()

y\_train = label\_enc.fit\_transform(y\_train)

y\_test = label\_enc.transform(y\_test)

*############ Training SVC (Support Vector Classifier) for predicting faces ########*

rfc = RandomForestClassifier()

rfc.fit(X\_train,y\_train)

joblib.dump(rfc,'models/face\_prediction\_model.sav')

**print**()

**print**('Random Forest Model saved successfully!!')

1. **LIVE ATTENDANCE CODE**

**import** numpy **as** np

**import** cv2

**from** mtcnn.mtcnn **import** MTCNN

**import** os

**import** joblib

**import** pandas **as** pd

**from** PIL **import** Image

**import** datetime

**import** time

**from** mtcnn.mtcnn **import** MTCNN

**from** tensorflow.keras.models **import** load\_model

embedding\_model = load\_model('models/facenet\_keras.h5')

**print**('Embedding Model Loaded')

ML\_model = joblib.load('models/face\_prediction\_model.sav')

**print**('Loaded ML Model')

detector = MTCNN()

**def** find\_face(img,img\_size=(160,160)):

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

    img = np.asarray(img) *# converting our image obj to numpy array*

    faces = detector.detect\_faces(img)

**if** faces:

        x,y,w,h = faces[0]['box']

        x,y=abs(x),abs(y)

        face = img[y:y+h,x:x+w]

        face = Image.fromarray(face) *# converting it to image object to resize it*

        face = face.resize(img\_size) *# resizing it*

        face = np.asarray(face)      *# converting it back to array*

**return** face,x,y,w,h

**return** None,None,None,None,None

**def** embed(face):

    face = face.astype('float32')

    fm,fs = face.mean(),face.std()

    face = (face-fm)/fs

    face = np.expand\_dims(face,axis=0)

    embs = embedding\_model.predict(face)

**return** embs[0]

*#for converting ids to names*

**def** id2name(id):

    x = os.listdir('faces/train/')

**return** x[id]

**def** mark\_attendance(name,roll):

    roll\_list = []

**if** **not** os.path.isdir('Attendance'):

        os.makedirs('Attendance')

    date=time.asctime()[8:10]

    month=time.asctime()[4:7]

    year=time.asctime()[-4:]

    tim=time.asctime()[11:16]

*# if csv of current date doesn't exist, make it*

**if** (date+'-'+month+'-'+year+'.csv')  **not** **in** os.listdir('Attendance/'):

        att = pd.DataFrame(columns=['Roll','Name','Time'])

        att.to\_csv('Attendance/'+date+'-'+month+'-'+year+'.csv') *#make sheet for today attendance*

*# here we are just selecting these 3 columns everytime and ignoring the index column*

    att = pd.DataFrame(pd.read\_csv('Attendance/'+date+'-'+month+'-'+year+'.csv'))

    att = att[['Roll','Name','Time']]

**for** i **in** range(len(att)):

        roll\_list.append(str(att.loc[i]['Roll']))

**if** roll **not** **in** roll\_list:

        att1 = pd.DataFrame({'Name':[name], 'Roll':[roll], 'Time':[datetime.datetime.now().strftime("%H:%M:%S")]})

        att = att.append(att1,ignore\_index=False)

**print**(f'Attendance marked for {name} whose roll no is {roll}.')

**else**:

        prev\_time = att[att['Roll']==str(roll)]['Time'].iloc[-1]

        curr\_time = datetime.datetime.now().time().strftime("%H:%M:%S")

*#here we are just checking the time difference between previous timestamp and current time*

**if** datetime.datetime.strptime(curr\_time, '%H:%M:%S') - datetime.datetime.strptime(prev\_time, '%H:%M:%S') > datetime.timedelta(minutes=5):

            att1 = pd.DataFrame({'Name':[name], 'Roll':[roll], 'Time':[datetime.datetime.now().strftime("%H:%M:%S")]})

            att = att.append(att1,ignore\_index=False)

**print**(f'Attendance marked for {name} whose Matric ID  is {roll}.')

    att.to\_csv('Attendance/'+date+'-'+month+'-'+year+'.csv')

cap = cv2.VideoCapture(0)

i = 0

**while** 1:

    i+=1

    ret,frame = cap.read()

**if** i>10:

        face,x,y,w,h = find\_face(frame)

**if** face **is** **not** None:

            face\_emb = embed(face)

            pred = ML\_model.predict(face\_emb.reshape(1,-1))

            name = str(id2name(pred[0]))

**if** name:

                mark\_attendance(name.split('-')[0],name.split('-')[1])

                cv2.rectangle(frame,(x,y),(x+w,y+h),(178,88,239),1)

                cv2.putText(frame,name,(x,y-10),cv2.FONT\_HERSHEY\_TRIPLEX,0.8,(178,88,239),1,cv2.LINE\_AA)

        cv2.imshow('live',frame)

**if** cv2.waitKey(1)==27:   *#press esc to exit*

**break**

cap.release()

cv2.destroyAllWindows()

**References**

*8 advantages of face recognition attendance system | benefits of facial ...* (n.d.). Retrieved June 1, 2022, from <https://truein.com/advantages-of-face-recognition-attendance-system/>

*Face recognition based attendance management system*. (n.d.). Retrieved June 9, 2022, from <https://www.researchgate.net/publication/341876647_Face_Recognition_based_Attendance_Management_System>

Sharma, A. (2022, February 23). *Face recognition based attendance system - easiest way - best project - with source code - 2022*. Machine Learning Projects. Retrieved May 31, 2022, from https://machinelearningprojects.net/face-recognition-based-attendance-system/

*Student attendance system using face recognition*. IEEE Xplore. (n.d.). Retrieved June 9, 2022, from https://ieeexplore.ieee.org/document/9215441