# BINUS UNIVERSITY
# BINUS INTERNATIONAL

## Assignment Cover Letter

## (Individual Work )

| **Student Information**: | **Surname** | **Given Names** | **Student ID Number** |
|---|---|---|---|
| 1. | Sanjaya | Hengky | 2201852492 |

**Course Code** : COMP6056

**Course Name** : Program Design Methods

**Class** : L1CC

**Name of Lecturer(s)** : Jude Joseph Lamug Martinez

**Major** : Computer Science

**Title of Assignment** : Face Recognition Attendance

**Type of Assignment** : Final Project

**Submission Pattern**

**Due Date** : 21-11-2018

**Submission Date** :

The assignment should meet the below requirements.

1. Assignment (hard copy) is required to be submitted on clean paper, and (soft copy) as per lecturer's instructions.
2. Soft copy assignment also requires the signed (hardcopy) submission of this form, which automatically validates the softcopy submission.
3. The above information is complete and legible.
4. Compiled pages are firmly stapled.
5. Assignment has been copied (soft copy and hard copy) for each student ahead of the submission.

**Plagiarism/Cheating**

BiNus International seriously regards all forms of plagiarism, cheating and collusion as academic offenses which may result in severe penalties, including loss/drop of marks, course/class discontinuity and other possible penalties executed by the university. Please refer to the related course syllabus for further information.

**Declaration of Originality**

By signing this assignment, I understand, accept and consent to BiNus International terms and policy on plagiarism. Herewith I declare that the work contained in this assignment is my own work and has not been submitted for the use of assessment in another course or class, except where this has been notified and accepted in advance.

Signature of Student: (Name of Student)

Hengky Sanjaya

**“Face Recognition Attendance”**

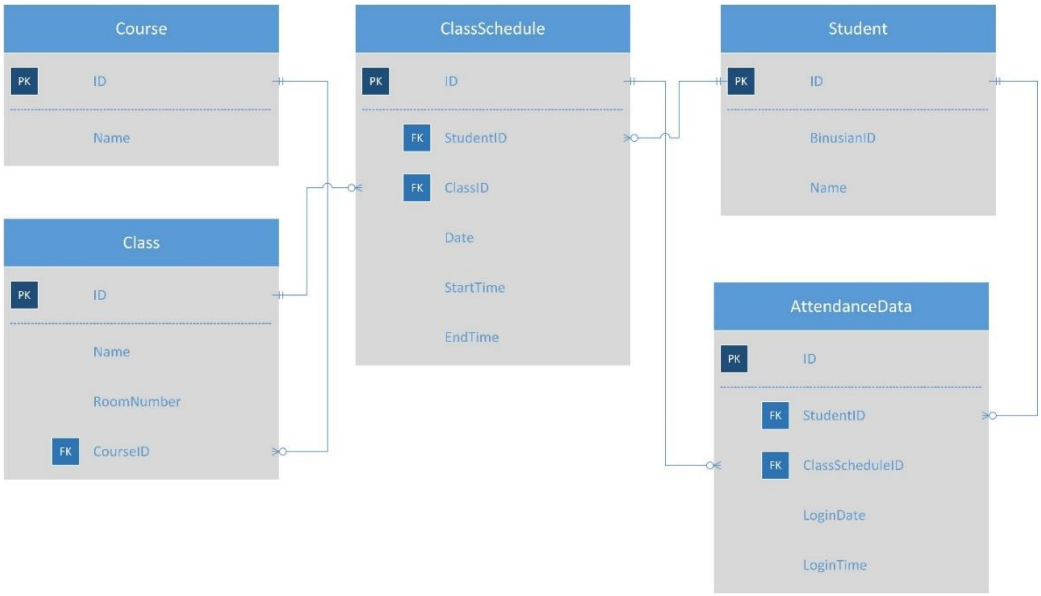
**Name : Hengky Sanjaya**

**ID : 2201852492**

## I. Program Description

This application is a web-based application to help people do their attendance easier using face recognition. This application is built using python version 3.7, Django framework version 2.1.2, SQLite for the database management, bootstrap for the user interface design. This program is meant to help people who always forget to bring their identity card.

## II. ERD & Class Diagram




1. Course

   To store all the courses data

2. Class

   To store the class data with the room number and the course.

3. Student

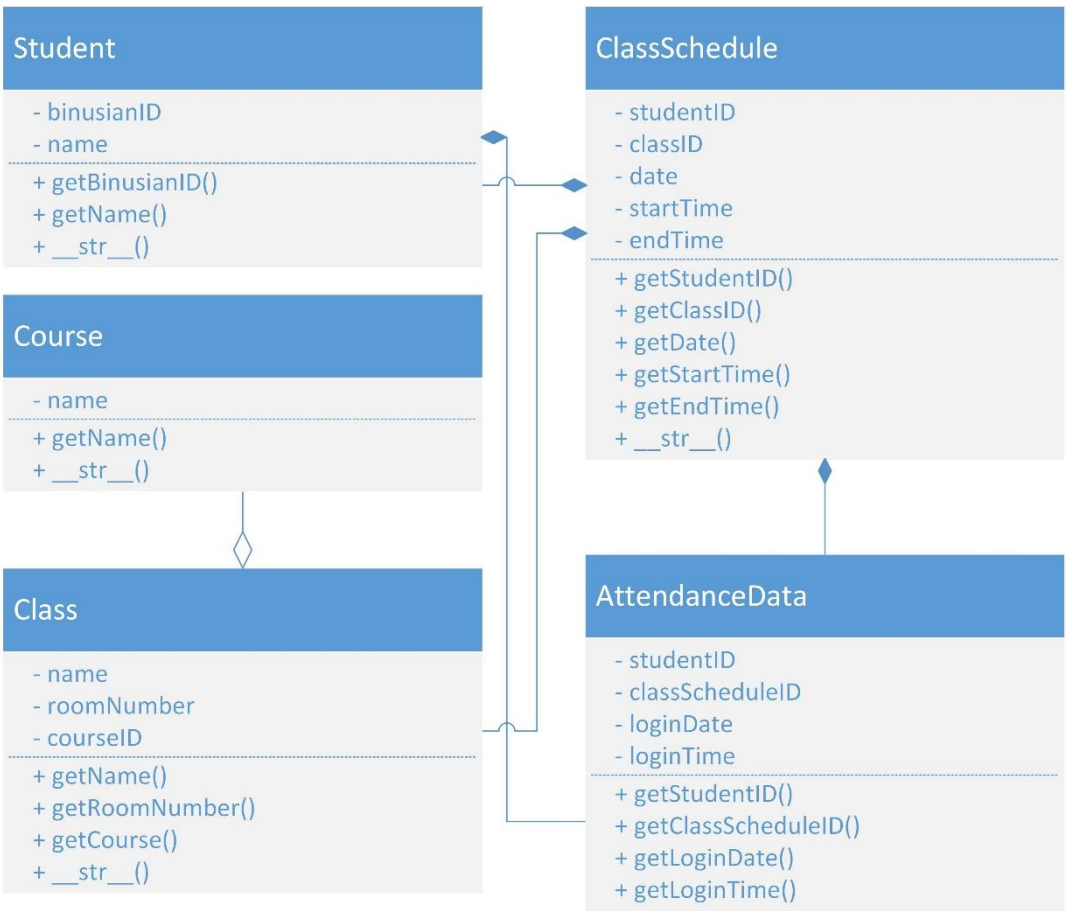
   To store all the student data

4. ClassSchedule

   To save the student’s schedule data with the start date time, end time and what course in it.

5. AttendanceData

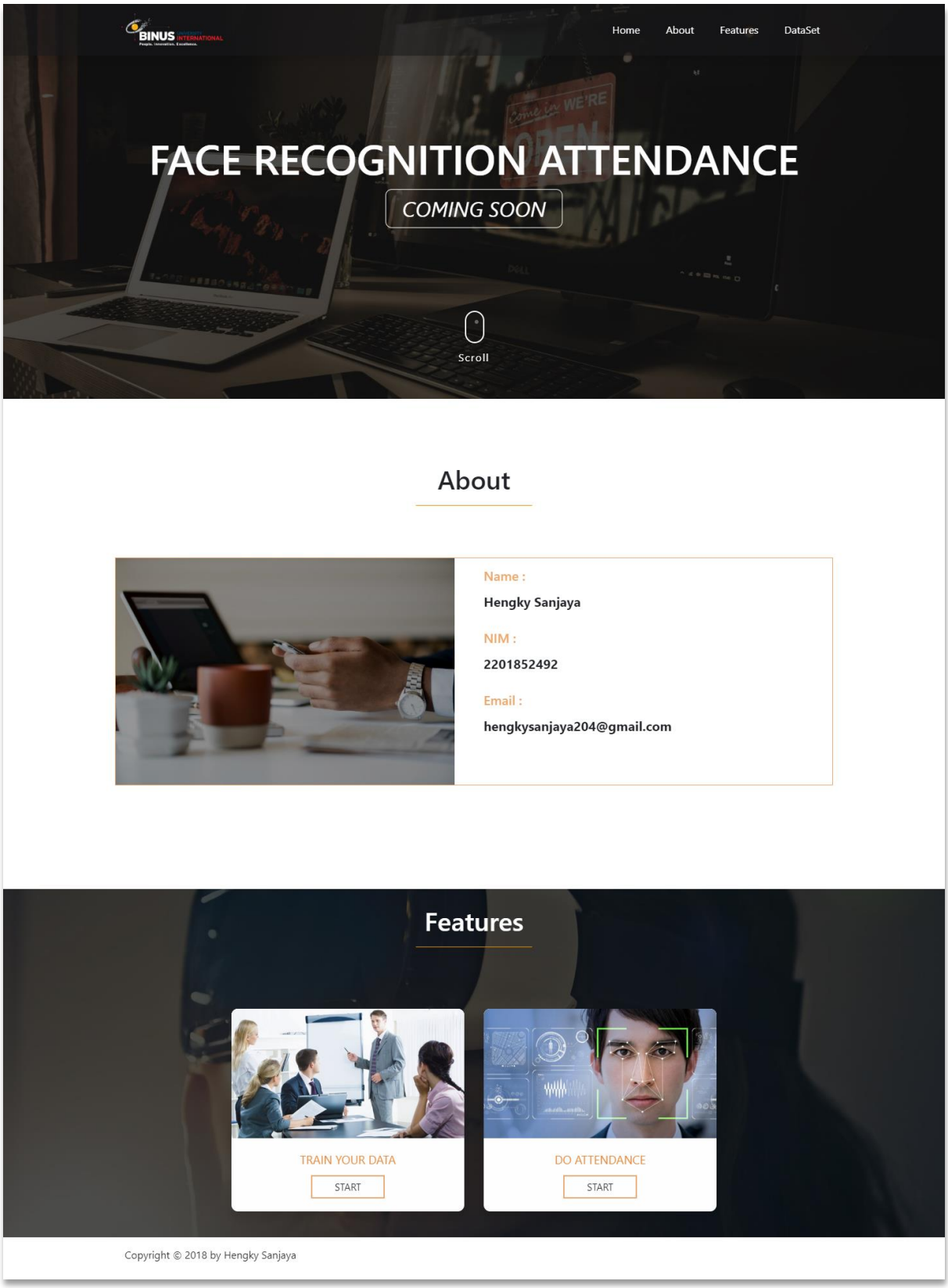
   To record the student attendance data in specific ClassSchedule




This is my class diagram. For each model I create the setter getter for each variable and also built in function called __str__() to return a string with specific format when we call the instance of the model.

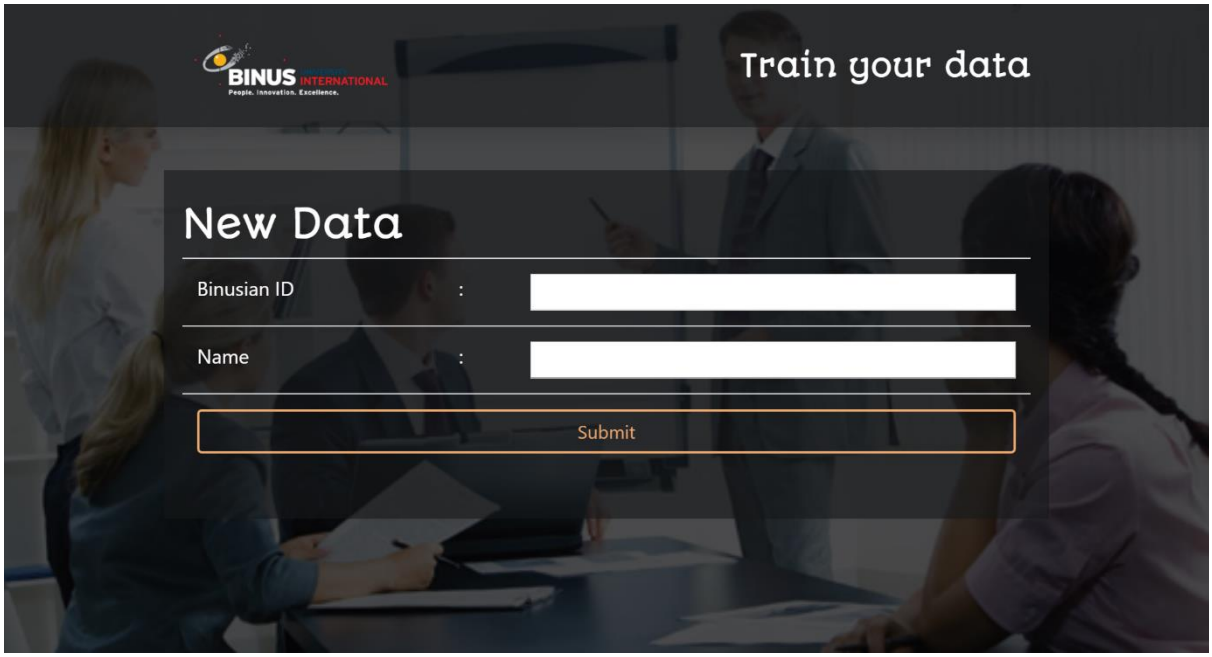
## III. Application Interface

a) Homepage



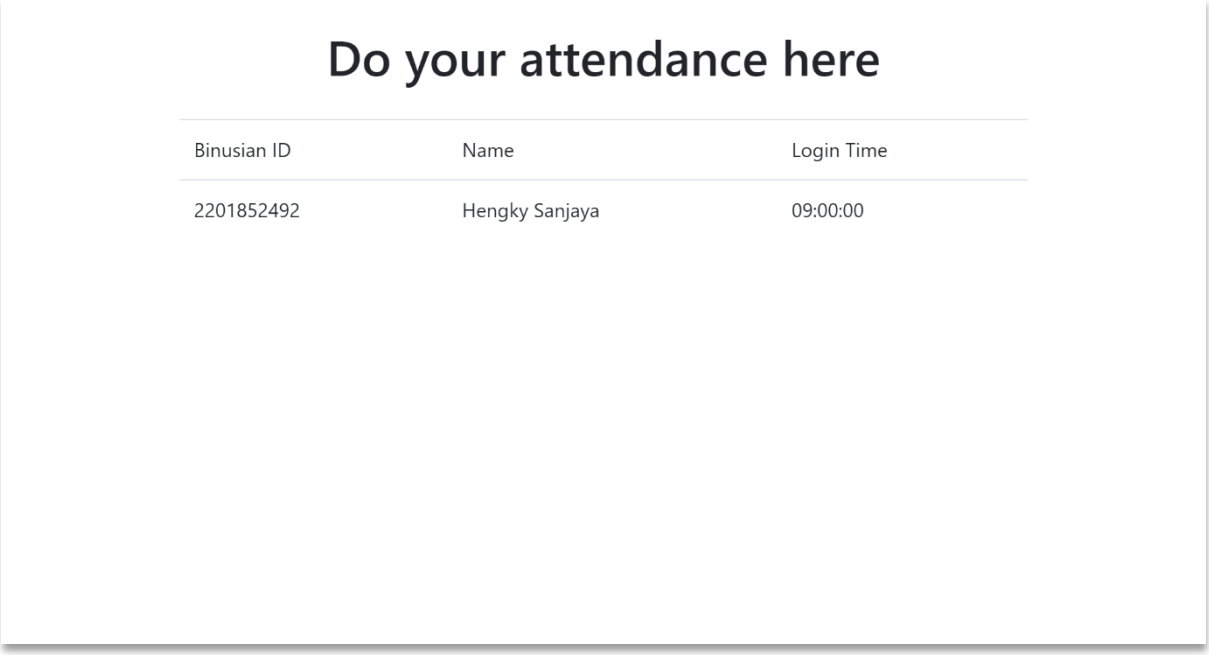
b) Train Data Page

This page is used for train the face recognition dataset, so when the submit button is clicked the system will open a new window to take a picture of the student new save the picture and save the data into database.




c) Do Attendance Page

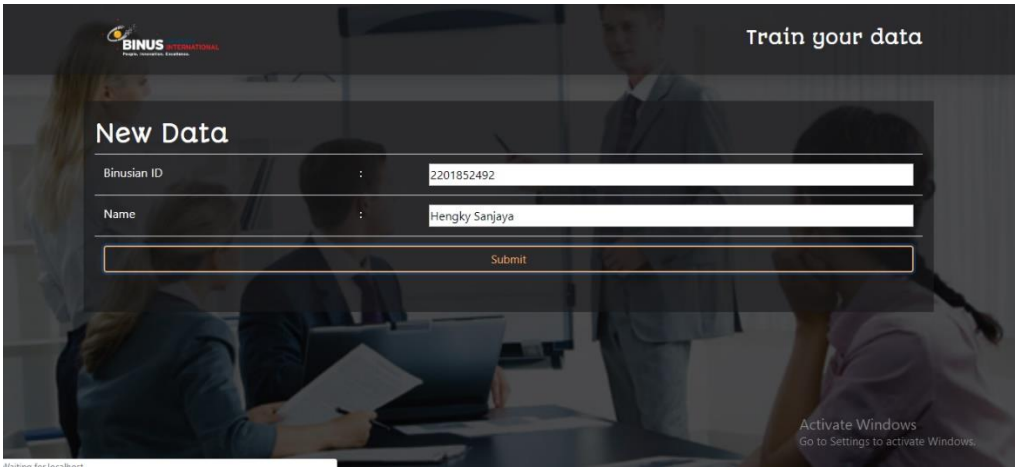
To see the detail attendance data, we go to this page by click the start do attendance button in the homepage.



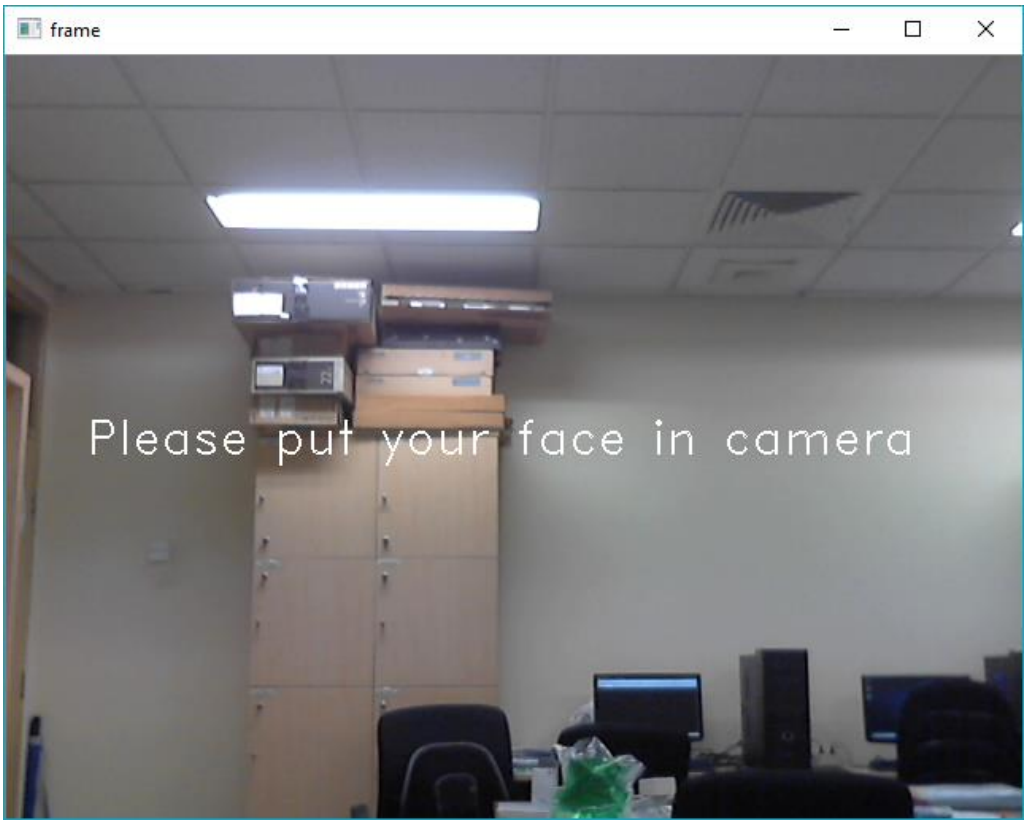
## IV. How the program works

- **To Train the Data**

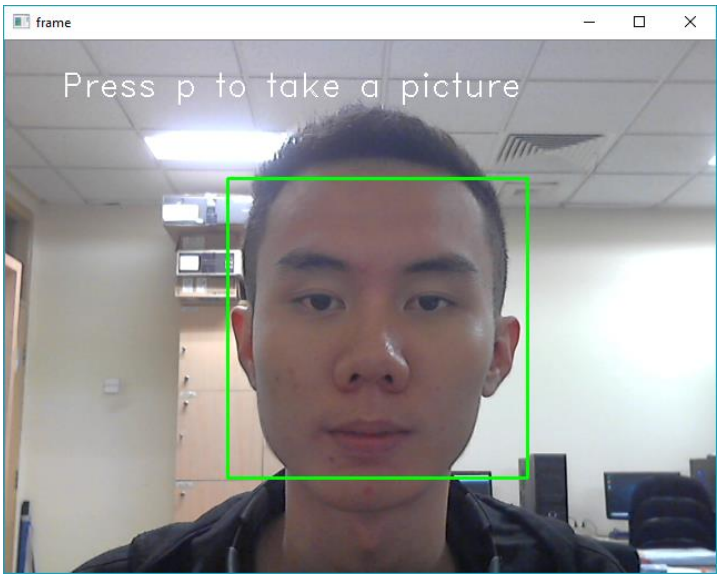
After input the BinusianID and name, then click 'Submit' Button. Then the system will open a window to take your picture.



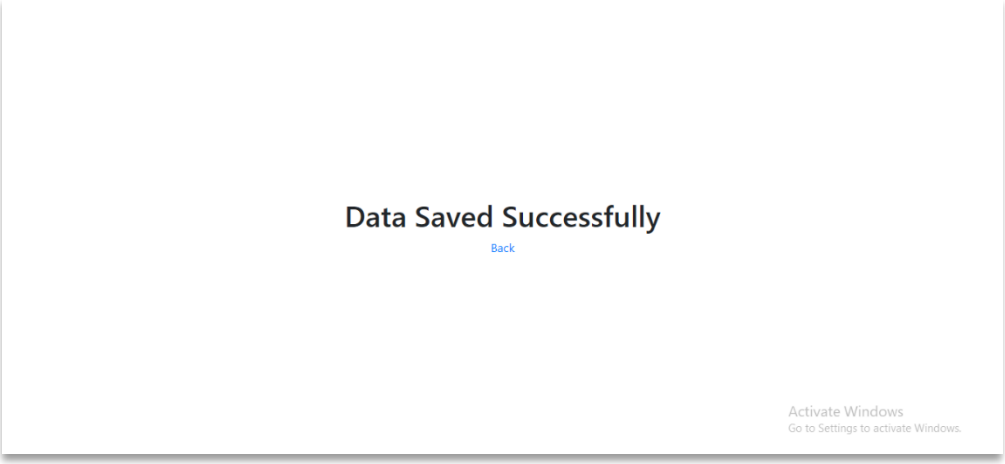

If the system does not find any face then a message will appear in the window to ask the user to put their face in camera.



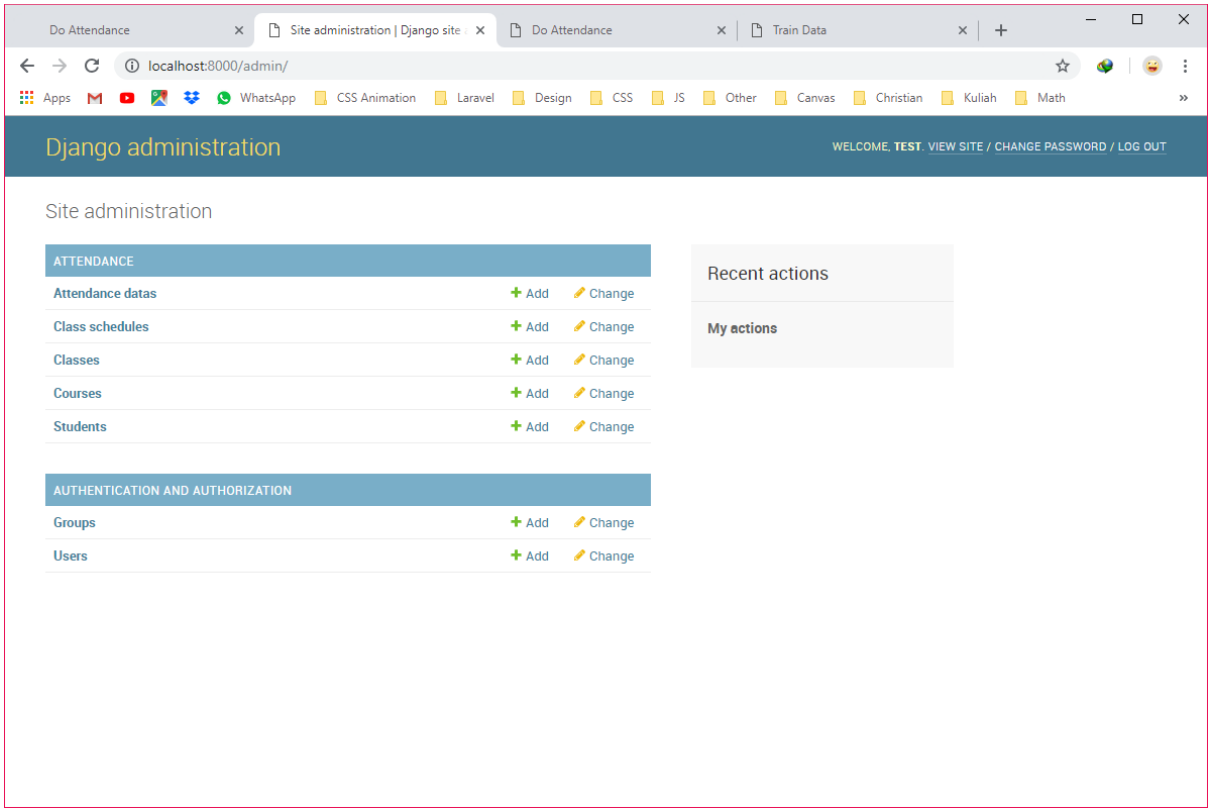
Then if a face founded, you can press ‘p’ in your keyboard to take a picture



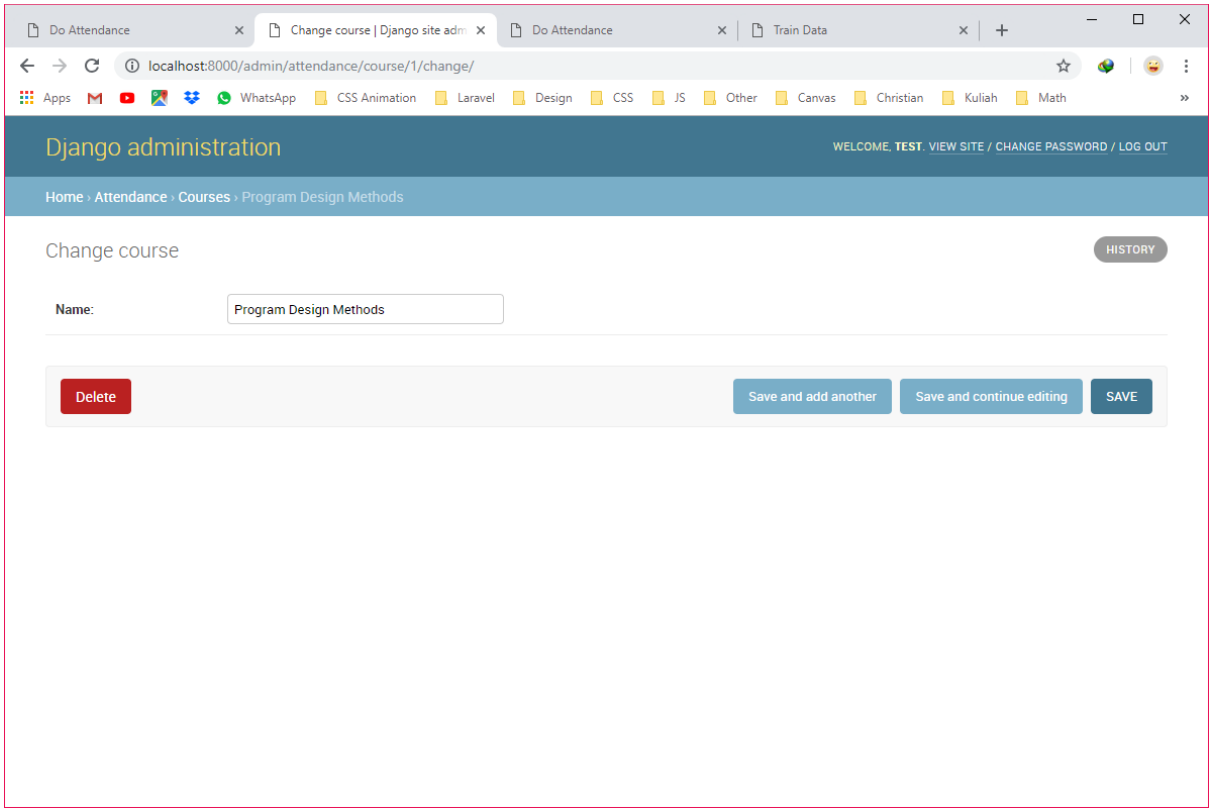

After you press ‘p’ in your keyboard the system will save your picture, and also save your data into database to be used for the attendance then Success message will appear in the page.



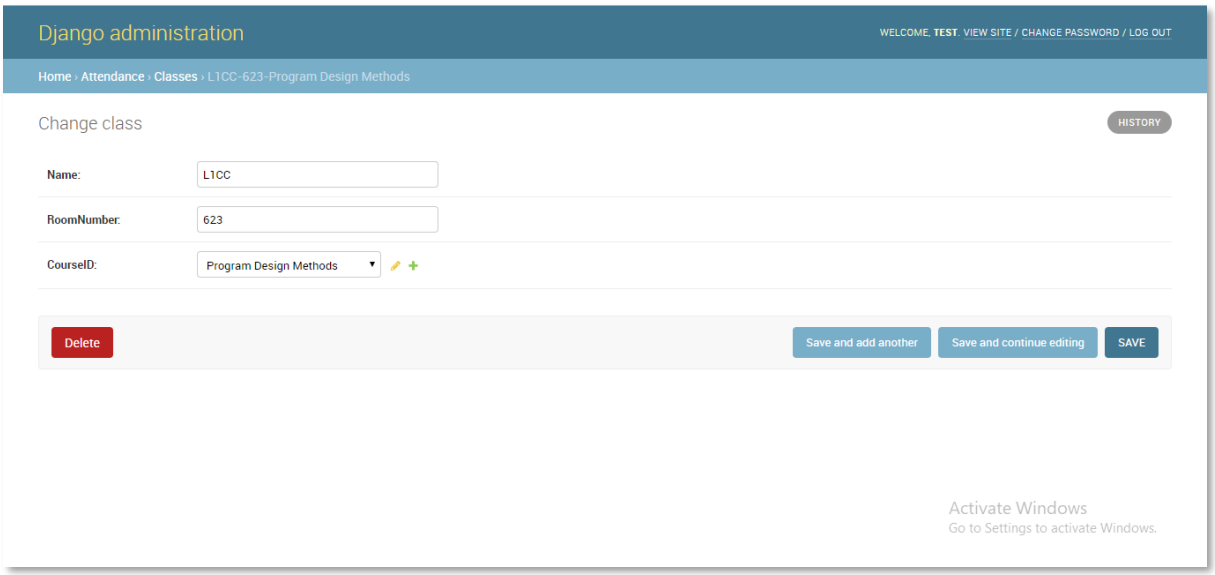
Before you can do your attendance of course you should have a class schedule. So, we need to add some data to several master table.
This is the Django administration homepage.



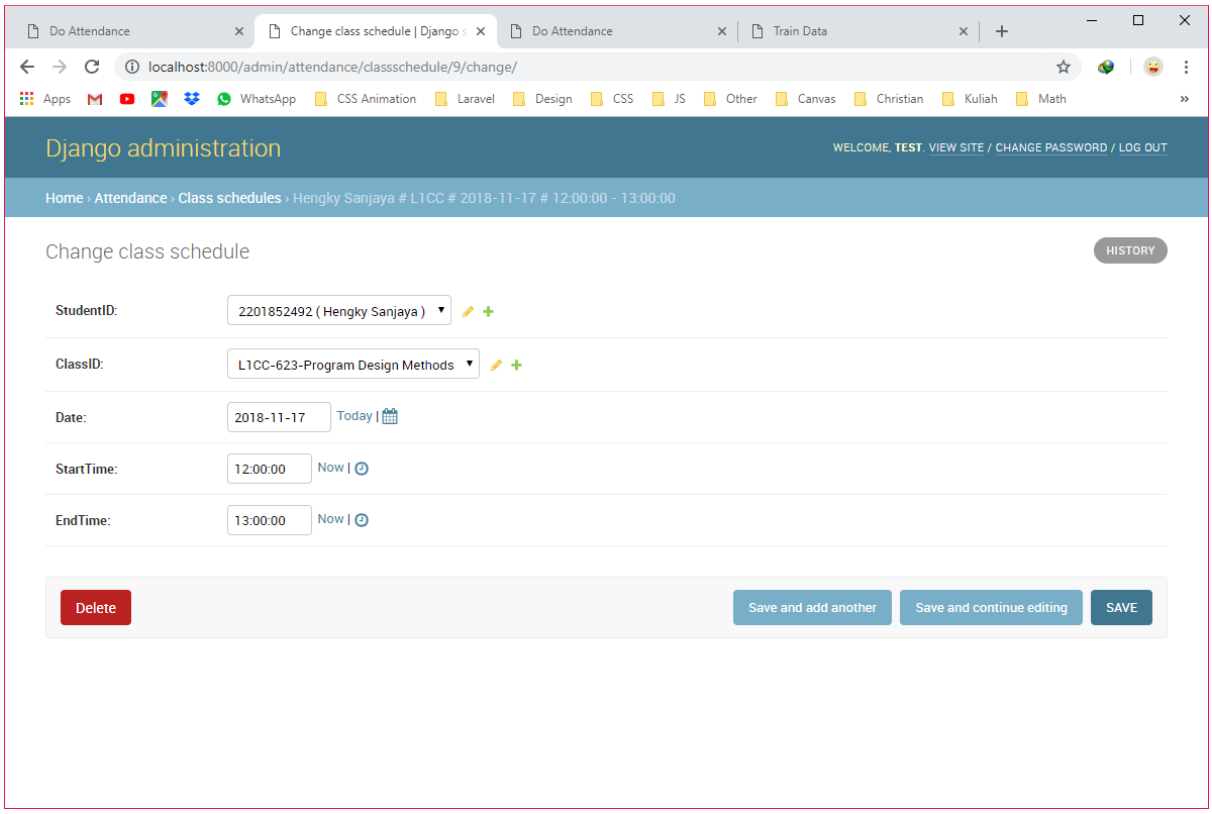

First, add course data for example “Program Design Methods”

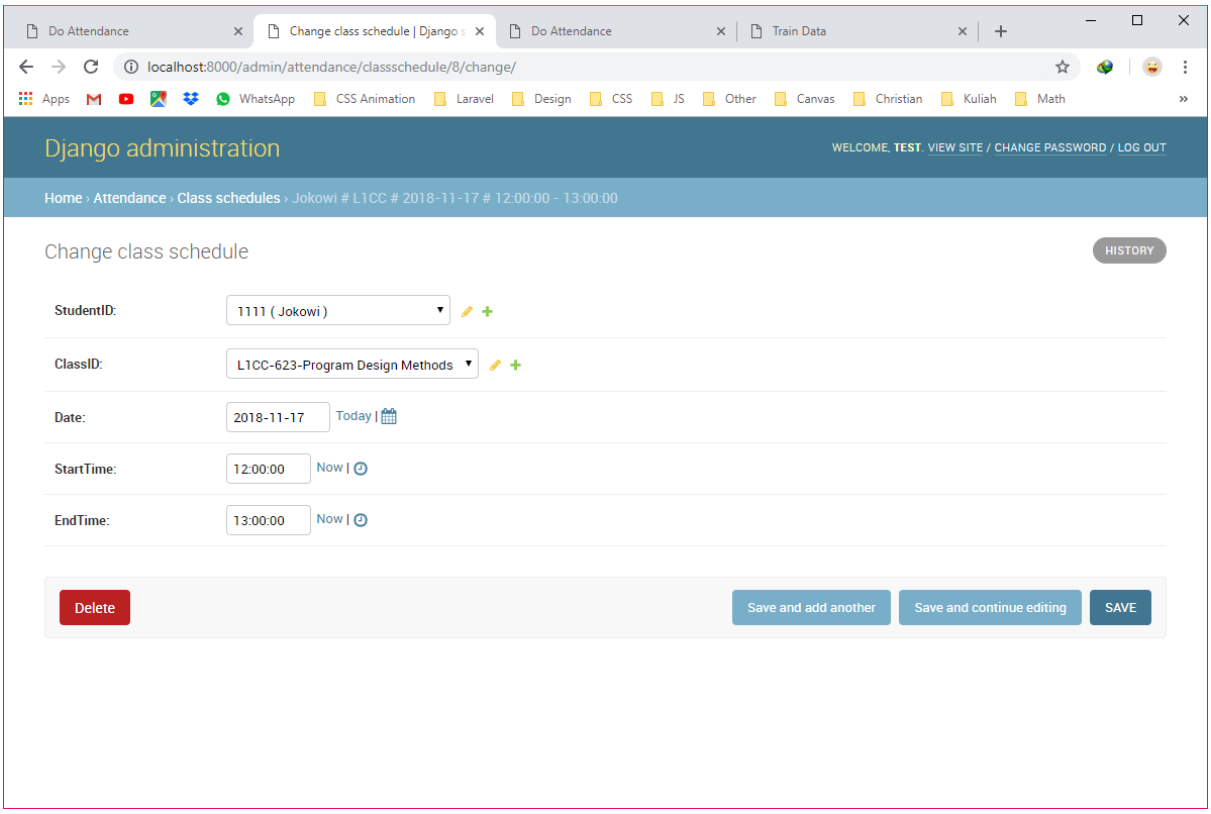


Add class data such as Name, Room Number, and choose Course that has been added before.

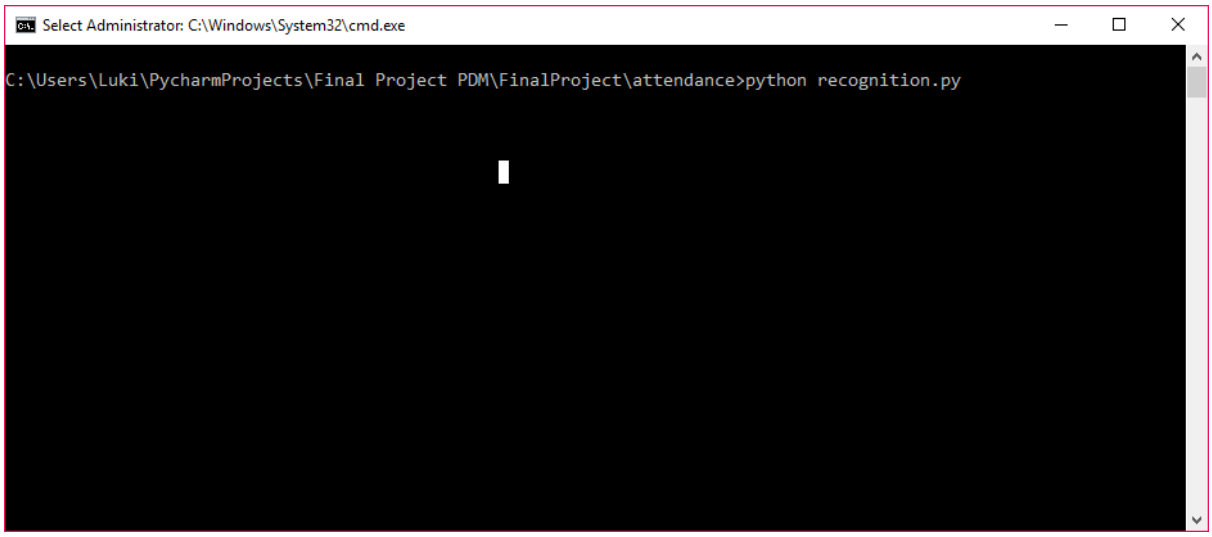



Then we add Class Schedule data. The student can only do the attendance within their class schedule time range

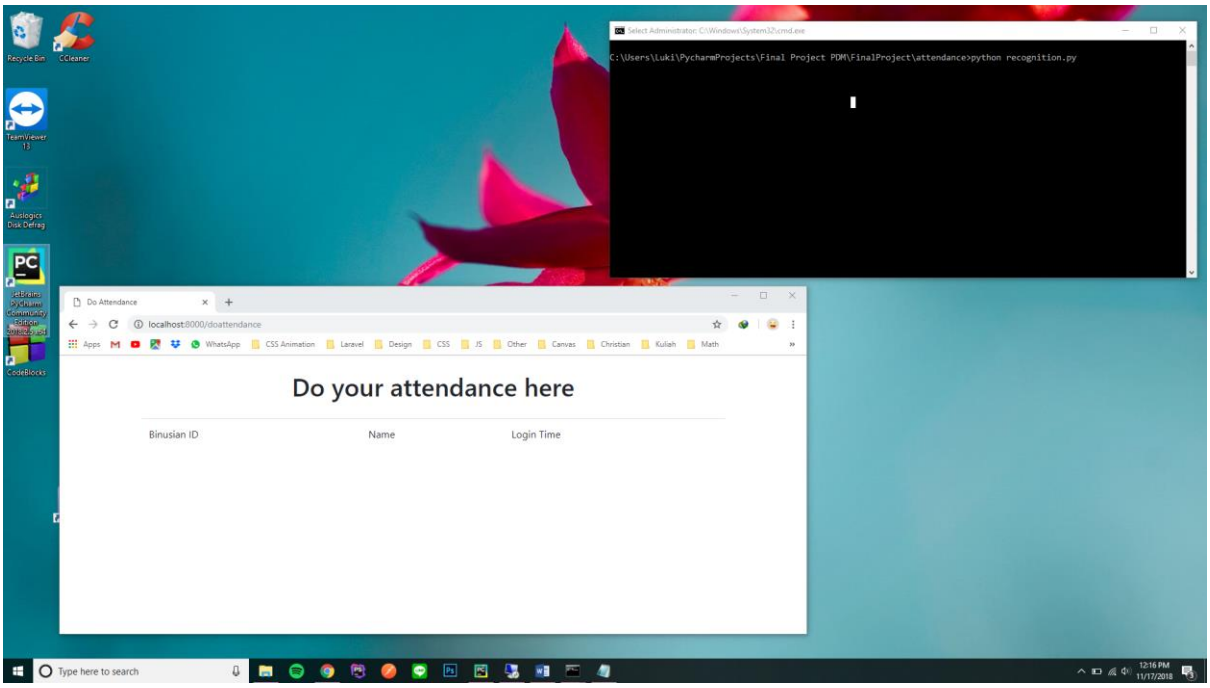




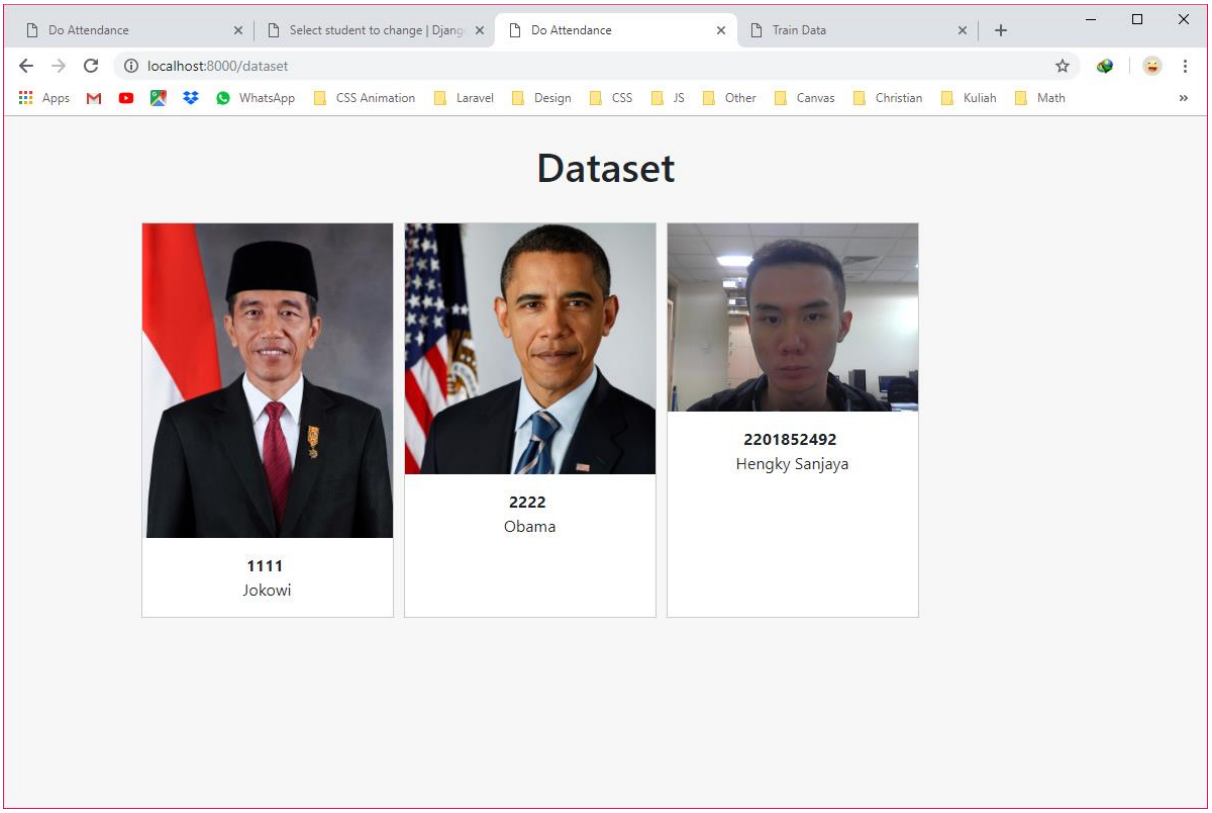

Then to do the attendance I run the face recognition to run in background



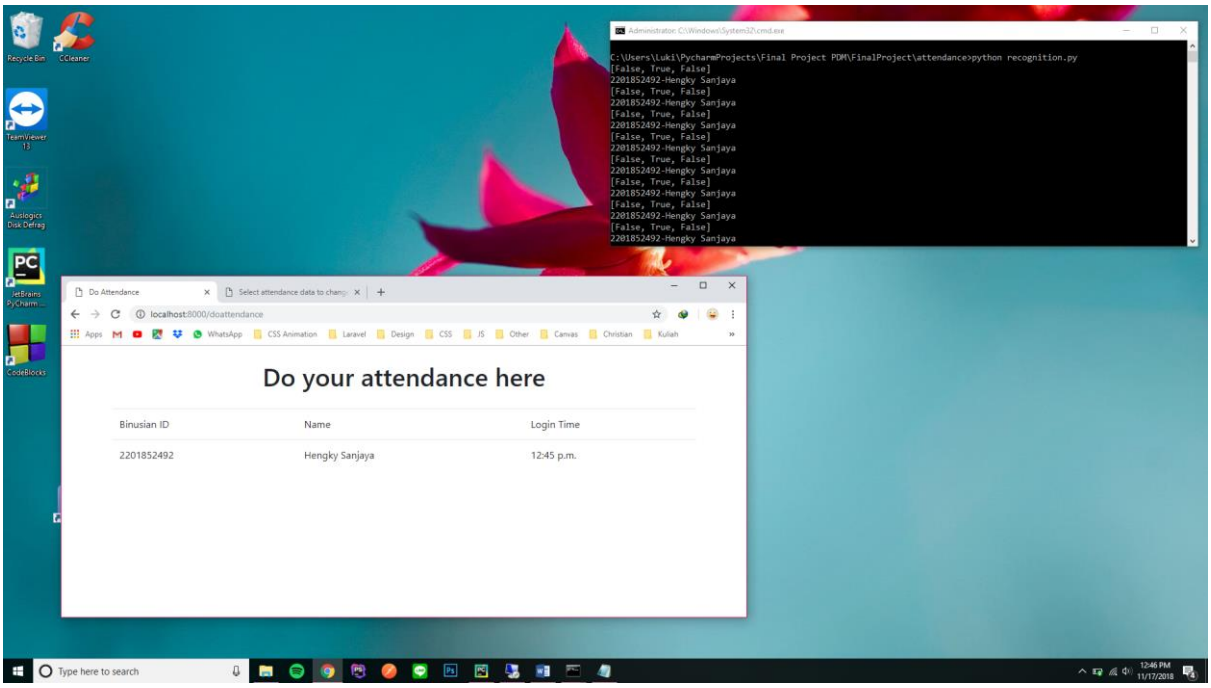
Then I also open the Do attendance page to see the result.




Here I have several datasets



after I run the face recognition server then I put my face in the camera. Then the doattendance page will show the attendance data. But the attendance data will be recorded once only in one class schedule for each student.



## V. Library used

There are some library that I used to build my “Face Recognition Attendance” project:

- Face-recognition version 1.2.3
  To do the face detection and recognition.
- OpenCV-python 3.4.3.18
  This library is used to open camera / webcam.
- OS
  To use some built-in function from python like getting the list of directories from specific path
- Datetime
  To get current datetime value and also parsing function

## VI. Lessons that Have Been Learned

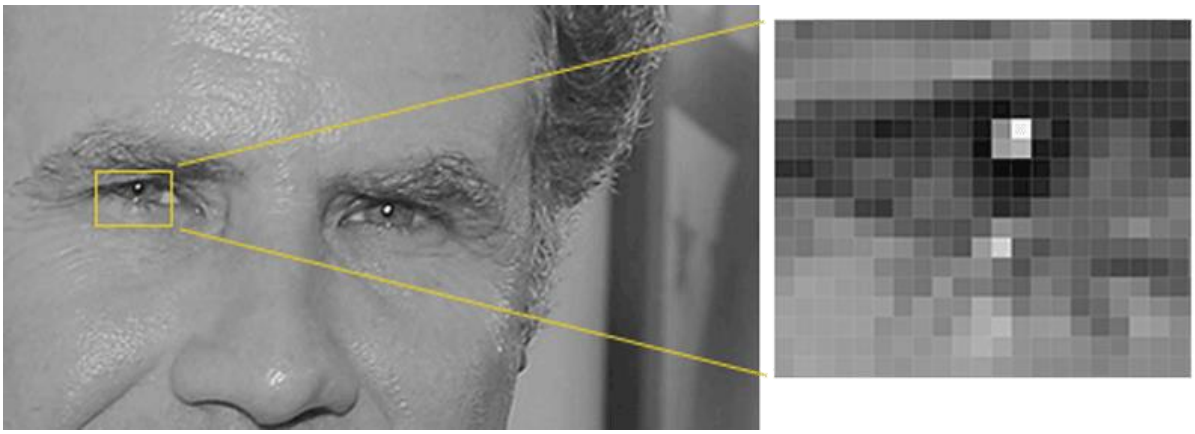
Since this is the first time I am learning python language so this final project makes me learn and get so many new things and it’s really fun, and I also learn new framework called Django to build my final project in web based.

## VII. Project Technical Description

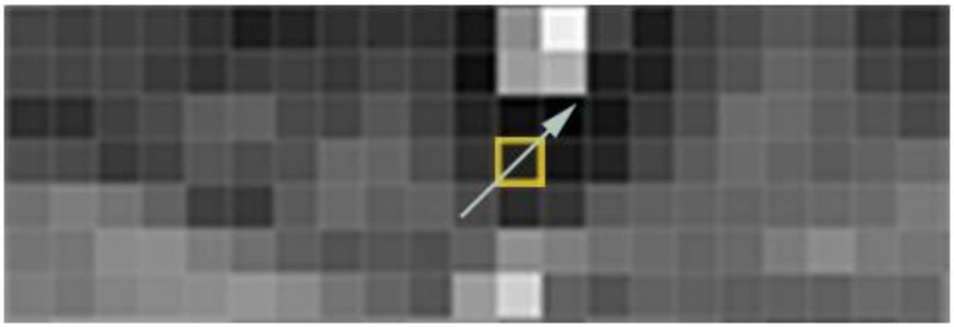
- Finding faces using OpenCV
  There are basically two primary ways to find faces using OpenCV:
  - Haar Classifier
  - LBP Cascade Classifier

  And I am using Haar Classifier because it is more accurate than LBP but it is also much slower, that’s why I have this file “haarcascade_frontalface_default.xml” to do the face detection.

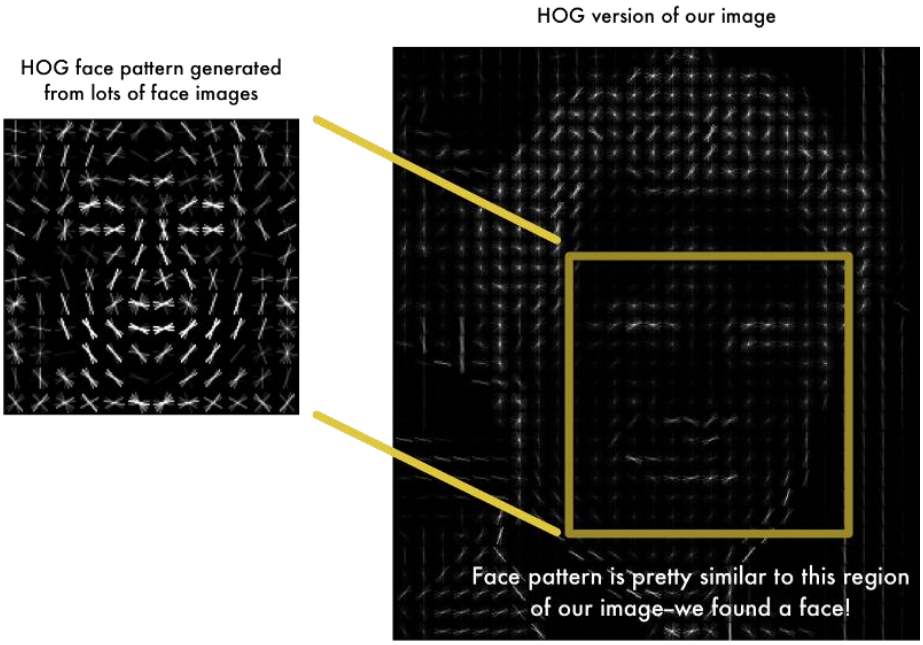
- Finding faces using face-recognition library
  To find faces in an image, I start to make the image in black and white because we don’t need color data to find faces.



Then look at every single pixel in the image one at a time. Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then draw an arrow showing in which direction the image is getting darker.



These arrows are called *gradients* and they show the flow from light to dark across the entire image.
HOG = Histogram Oriented Gradient




- Face recognition
  The explanation more about face recognition I will discuss in the code explanation below.

## VIII. Code Explanation

"Recognition.py" file

```
def main():
    video_capture = cv2.VideoCapture(0)
    # Create arrays of known face encodings and their names
    dictKnownFace = {}

    listKnownFace = []
    listNameKnownFace = []

    listOfFiles = os.listdir("static\\attendance\\dataset")
    for l in listOfFiles:
        listImage = os.listdir("static\\attendance\\dataset\\" + str(l))

        dictKnownFace.setdefault(l,[])
        listNameKnownFace.append(l)

        for j in listImage:
            person_image = face_recognition.load_image_file("static\\attendance\\dataset\\" + str(l) + "\\"+j)
            face_locations = face_recognition.face_locations(person_image)
            person_face_encoding = face_recognition.face_encodings(person_image, face_locations)[0]

            dictKnownFace[l].append(person_face_encoding)
            listKnownFace.append(person_face_encoding)

    # Initialize some variables
    process_this_frame = True


    while True:
        # Grab a single frame of video
        ret, frame = video_capture.read()

        # Resize frame of video to 1/4 size for faster face recognition processing
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)

        # Convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses)
        rgb_small_frame = small_frame[:, :, ::-1]


        # Only process every other frame of video to save time
        if process_this_frame:
            # Find all the faces and face encodings in the current frame of video
            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:
                # See if the face is a match for the known face(s)
                matches = face_recognition.compare_faces(listKnownFace, face_encoding, tolerance=0.50)
```

First, I open the camera using cv2 library then I get all the available dataset from the file and for each image I encode the image then store the encoded data into my list. After that, I read per frame of video and resize the frame into ¼ size for faster face recognition processing, and also I convert the image from BGR color (which OpenCV uses) to RGB color (which face_recognition uses).

Then I find the face location and from the face location I encode it and compare with the list encoded known faces.

```
            face_names = []
            for face_encoding in face_encodings:
                # See if the face is a match for the known face(s)
                matches = face_recognition.compare_faces(listKnownFace, face_encoding, tolerance=0.50)

                name = "Unknown"

                print(matches)

                # If a match was found in known_face_encodings, just use the first one.
                if True in matches:
                    first_match_index = matches.index(True)

                    name = listNameKnownFace[first_match_index]
                    writeData(name)

                    print(name)
                    face_names.append(name)
                    break

        process_this_frame = not process_this_frame


        # Hit 'q' on the keyboard to quit!
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release handle to the webcam
    video_capture.release()
    cv2.destroyAllWindows()

main()
```

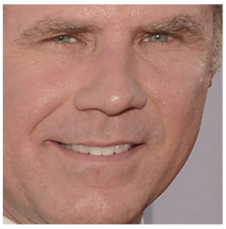
After that if a match was found I list known faces, I use the first one, and call the writeData function to write the founded face data to the text file. Then release the webcam.

```
def writeData(data):
    data = data.split('-')

    f = open('temporarydata.txt', 'a')
    now = datetime.datetime.now()
    f.write("\n"+str(data[0]) +"#" + data[1] + "#" + str(now.date()) + "#" + str(now.strftime("%H:%M:%S")))
    f.close()
```

This is the writeData function to write the founded face into the text file with format binusianID#name#logindate#logintime. So, I can read from this file for attendance data.

Input Image



128 Measurements Generated from Image

| | | | |
|---|---|---|---|
| 0.097496084868908 | 0.045223236083984 | -0.12814667820 93 | 0.032084941864014 |
| 0.12529824674129 | 0.060309179127216 | 0.17521631717682 | 0.020976085215807 |
| 0.030809439718723 | -0.01981477253139 | 0.10801389068365 | -0.000521632784511 89 |
| 0.036050599068403 | 0.065554238855839 | 0.07313060015 44 | -0.13189851100111 |
| -0.097486883401871 | 0.122626897253 | -0.029626874253154 | -0.005955751053988 9 |
| -0.006640171166509 4 | 0.036750309169292 | -0.15958009660244 | 0.043374512344599 |
| -0.14131525158882 | 0.14114324748516 | -0.031351584941149 | -0.053343612700701 |
| -0.048540540039539 | -0.061901587992907 | -0.15042643249035 | 0.078198105096817 |
| -0.12567175924778 | -0.10568545013666 | -0.12728653848171 | -0.076289616525173 |
| -0.061418771743774 | -0.074287034571171 | -0.065365232527256 | 0.123694673718058 |
| 0.046741496771574 | 0.006176188122481 1 | 0.14746543765068 | 0.056418422609568 |
| -0.121386501431 47 | -0.210559919147651 | 0.004109122790396 2 | 0.089727647602558 |
| 0.061806746166945 | 0.113457657396 79 | 0.021352224051952 | -0.008584329858422 3 |
| 0.061989940702915 | 0.193722039461 14 | -0.086726233363152 | -0.022388197481632 |
| 0.109041959047 32 | 0.084853030741215 | 0.09463594853878 | 0.020696049556136 |
| -0.019414527341723 | 0.006481129676103 6 | 0.21180312335491 | -0.050584398210049 |
| 0.152459457516 67 | -0.165823280811 31 | -0.035577941685915 | -0.072376452386379 |
| -0.122166685760 02 | -0.007277775555849 1 | -0.036901291459799 | -0.034365277737379 |
| 0.083934605121613 | -0.059730969369411 | -0.070026844739914 | -0.045013956725597 |
| 0.087945111095905 | 0.114784322679 04 | -0.089621491730213 | -0.013955107890069 |
| -0.021407851949334 | 0.148411959409 71 | 0.078333757817745 | -0.178980857133 87 |
| -0.018298890441656 | 0.049525424838066 | 0.13227833807468 | -0.072600327432156 |
| -0.011014151386917 | -0.051016297191381 | -0.141329213976 86 | 0.005051192827522 8 |
| 0.009367933496832 8 | -0.062812767922878 | -0.134074985980 99 | -0.014829395338893 |
| 0.058139257133007 | 0.004863874055445 2 | -0.039491076022387 | -0.043765489012003 |
| -0.024210374802351 | -0.114437922835 35 | 0.071997955441475 | -0.012062266469002 |
| -0.057223934680223 | 0.014683869667351 | 0.052281547337 77 | 0.012774495407939 |
| 0.023535015061498 | -0.081752359867096 | -0.031709920614958 | 0.069833360612392 |
| -0.009803973138332 4 | 0.037022035568953 | 0.11009479314089 | 0.116387888789 18 |
| 0.020220354199409 | 0.127881318330 76 | 0.18632389605045 | -0.015336792916059 |
| 0.004033768083900 2 | -0.094398014247417 | -0.117682486772 54 | 0.102814577519 89 |
| 0.051597066223621 | -0.100343115627 77 | -0.040977258235216 | -0.082041338086128 |

The face_encodings function will return the 128 measurements generated from image (like the picture above)

Then the compare_faces function will find the network generates nearly the same numbers. (has the closest measurements to our known face.)

“detectface.py” file

```python
def detect(id,name):
    # initialize
    cap = cv2.VideoCapture(0)


    cv2.startWindowThread()

    # Create the haar cascade
    faceCascade = cv2.CascadeClassifier("attendance/haarcascade_frontalface_default.xml")

    # declare value to determine the time and number of pictures
    n = 1

    takePicture = True
    while takePicture:
        # Capture frame-by-frame
        ret, frame = cap.read()

        # convert the frame to gray color
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        # Detect faces in the image
        faces = faceCascade.detectMultiScale(
            gray,
            scaleFactor=1.1,
            minNeighbors=5,
            minSize=(100, 100), flags=cv2.CASCADE_SCALE_IMAGE
        )

        font = cv2.FONT_HERSHEY_DUPLEX

        # check if face not found then put message text
        if(len(faces) == 0):
            cv2.putText(frame, "Please put your face in camera", (50,250), font, 1.0, (255,255,255), 1)
        # if face founded
        else:
            cv2.putText(frame, "Press p to take a picture", (50, 50), font, 1.0, (255, 255, 255), 1)
            # Draw a rectangle around the faces
            for (x, y, w, h) in faces:
                cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)

            if cv2.waitKey(1) & 0xFF == ord('p'):
                # save the frame to image file
                newPath = 'attendance/static/attendance/dataset/'+ id + '-' + name
                if not os.path.exists(newPath):
                    os.makedirs(newPath)

                ret, frame = cap.read()
                cv2.imwrite(newPath + '/' + str(n) + ".jpg", frame)

                cv2.putText(frame, "Picture saved", (50, 250), font, 1.0, (255, 255, 255), 1)

                takePicture = False

        #Display the resulting frame
        cv2.imshow('frame', frame)

        #waiting for the 'q' keys
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # When everything done, release the capture
    cap.release()
    cv2.destroyWindow('frame')
    cv2.destroyAllWindows()

    return "Success"
```

First, I open the camera using cv2 library, then I convert the frame into gray color and detect faces in the frame using cascade method. If the system doesn’t find any face, then I display a message to ask the user to put their face in camera, otherwise I will display a message to ask the user to press p to take a picture. After that I draw a rectangle around the faces and wait

for the ‘p’ is pressed and save the frame into image file. Then release the capture when everything has done.

“urls.py” file

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('attendance.urls'))
]
```

When the URL contains ‘admin’ then I will redirect the page into admin page, else if empty then I include the url file in my attendance apps

“urls.py” file in attendance apps

```
from django.urls import path, include
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('traindata', views.traindata, name='traindata'),
    path('doattendance', views.doattendance, name='doattendance'),
    path('submit', views.submit),
    path('response', views.response),
    path('dataset', views.dataset, name='dataset')
]
```

If the url is empty then I will redirect to the homepage, else I will redirect depends on the inputted url

“views.py” file

```python
from django.shortcuts import render
from .models import *
import datetime as dt


from .detectface import detect

# Create your views here.

# function to redirect to homepage
def home(request):
    return render(request, 'attendance/homepage.html')

# function to redirect to traindata page
def traindata(request):
    return render(request, 'attendance/traindata.html')

# function to input and display the attendance data
def doattendance(request):

    #read recognition file
    url = 'attendance/temporarydata.txt'

    f = open(url, 'r')

    now = dt.datetime.now()

    listData = []

    #loop per row
    for i in f:
        try:
            data = i.split('#')
            id = data[0]
            date = data[2]
            time = data[3].replace('\n','')

                        if(check.count() == 0):
                            obj = AttendanceData.objects.create(studentID = student, classScheduleID = i, loginDate = date, loginTime = time)
                            listData.append(obj)
                        else:
                            if(check[0] not in listData):
                                listData.append(check[0])
            else:
                print("cannot do attendant")
        except:
            print("empty line founded")
            continue



    f.close()

    context = {
        'title': 'hello',
        'data': listData
    }

    print(context)

    return render(request, 'attendance/doattendance.html', context)

# function to submit the train data and save to database
def submit(request):
    binusianId = request.POST['binusianid']
    name = request.POST['name']

    response = detect(binusianId, name)


    # insert new student data to database
    Student.objects.create(name=name, binusianID = binusianId)

    print("Response : " + str(response))

    return render(request, 'attendance/response.html')

# function to redirect page to response page
def response(request):
    return render(request, 'attendance/response.html')
# function to get the dataset data and display in dataset page
def dataset(request):

    obj = Student.objects.all()

    context = {
        'data': obj
    }

    return render(request, 'attendance/dataset.html', context)
```

In my “views.py” file I have several functions such as:

- Home():
  To redirect to homepage if the url contains empty string
- Traindata():
  To redirect to “traindata.html” page
- Doattendance():
  This function is used to read the face recognition result from the text file and validate the data with the available class schedule if the login date time within the class time then I will insert new attendance data into database. After that, I will send the attendance data to “doattendance.html” page to display it to the user
- Submit():
  This function will run after the user input the data and clicks submit button in “traindata.html” page. This function will call another function in “detectface.py” file to open a camera window to take the user’s picture then save the data into database.
- Response():
  To redirect to “response.html” page
- Dataset():
  This function is used to open “dataset.html” page, but before that the function will retrieve the all the student’s data from the database and send it to that page.

## IX. Project Link

*https://github.com/hengkysanjaya123/Final-Project-PDM*

## X. References


- *https://www.blog.pythonlibrary.org/2018/08/15/face-detection-using-python-and-opencv/*
- *https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78*
- *https://github.com/ageitgey/face_recognition*