

MGNREGA -AUTOMATIC ATTENDENCE SYSTEM

A

Mini Project Report

*Submitted in partial fulfilment of the
Requirements for the award of the Degree of*

BACHELOR OF ENGINEERING

IN

INFORMATION TECHNOLOGY

By

K. HARSHITH KUMAR , 1602-20-737-135

G.NITHIN ,1602-20-737-149

B.SANTHOSH REDDY ,1602-20-737-313



Department of Information Technology

Vasavi College of Engineering (Autonomous)

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

(AFFILIATED OSMANIA UNIVERSITY)

HYDERABAD - 500 030

Department of Information Technology



DECLARATION BY CANDIDATE

We, **K. HARSHITH KUMAR, G. NITHIN , B.SANTHOSH REDDY** , bearing hall ticket number, **1602-20-737-135, 1602-20-737-149, 1602-20-737-313**

hereby declare that the project report entitled **"MGNREGA AUTOMATIC ATTENDANCE"** Department of Information Technology, Vasavi College of Engineering, Hyderabad, is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Engineering in Information Technology**

This is a record of bonafide work carried out by me and the results embodied in this project report has not been submitted to any other university or institute for the award of any other degree or diploma.

K. HARSHITH KUMAR ,1602-20-737-135

G. NITHIN ,1602-20-737-149

B.SANTHOSH REDDY ,1602-20-737-313

VASAVI COLLEGE OF ENGINEERING (AUTONOMOUS)

(AFFILIATED TO OSMANIA UNIVERSITY)

HYDERABAD - 500 030

Department of Information Technology



BONAFIDE CERTIFICATE

This is to certify that the project entitled “**MGNREGA AUTOMATIC ATTENDENCE**” being submitted by **K.HARSHITH , G.NITHIN, B.SANTHOSH** bearing **1602-20-737-135, 1602-20-737-149, 1602-20-737-313**, in partial fulfillment of the requirements for the completion of MINI PROJECT of Bachelor of Engineering in Information Technology is a record of bonafide work carried out by them under my guidance.

Internal Guide
Dr.SK.prashanth

External Examiner
Dr. K Ram Mohan Rao

HOD, IT

ACKNOWLEDGEMENT

We thank the department of INFORMATION TECHNOLOGY, for introducing the subject “Mini Project-2” in BE fifth semester.

We would also like to show our appreciation to our Honorable principal, Dr S V Ramana sir, our HOD K. Ram Mohan Rao for supporting us and our mini project Associate Professor, Dr.SK.prashanth , for letting us properly understand the process of doing a project and for providing valuable insight and expertise that has greatly assisted us in the making of the project.

TABLE OF CONTENTS:

1 INTRODUCTION

1.1 PURPOSE

1.2 INTENDED AUDIENCE

1.3 PRODUCT SCOPE

1.4 PROBLEM DEFINITION

2. RELATED WORK

3. PROPOSED WORK

3.1 USE CASES

3.2 UI PROTOTYPES OR SCREENSHOTS

3.3 TECHNOLOGY USED

3.4 DESIGN

3.5 IMPLEMENTATION

3.5.1 MODULES

3.5.2 ALGORITHMS USED

3.5.3 CODE

3.6 TESTING

4. RESULTS

5. DISCUSSION AND FUTURE WORK

6. REFERENCES

ABSTRACT

Do you know who are MGNREGA workers and how much time it takes for attendance of MGNREGA labourers rural areas? When the attendance is taken manually it actually consuming a lot of time, as it is having very less working hours(approximately 2hrs). Basically our tool when given an image containing group of workers, it will detect, separate and match the workers and mark them for their attendance by using Face Recognition Mechanism. The goal of our project is to reduce the time allotted for marking attendance of the workers under MGNREGA in such a way that the working hours will be utilized efficiently. We are going to develop a web-application which classifies the documents uploaded.

CHAPTER 1

INTRODUCTION

What is Face classifier?

“MGNREGA AUTOMATIC ATTENDENCE” developed by us uses OpenCV classifier which classifies the given images into pickle file. This Face classifier can be integrated with the websites to classify the uploaded images of the user/applicant.

Why does attendance application websites need Face classifier?

- It minimizes the risk of generating faulty attendance
- It makes sure that face detected gets the attendance in the excel sheet generated.

1.1PURPOSE

- ❖ Attendance taking at the rural areas takes a lot of worktime for the people working there. When the attendance is taken manually it is actually consuming a lot of time and confusing for the attendance taker in not knowing whether it is the same person or not sometimes, as it is having very fewer working hours (approximately 2hrs). A system may be developed using automatic analysis of uploaded documents to upload and detect and match the faces for automatic attendance in the portal so that this problem of taking attendance manually which is a time taking process may be completely resolved and that time can be used for working.

1.2 INTENDED AUDIENCE

The intended audience for this project is rural manager applicant takes attendance for workers. In the upload photo there should be all the workers present there and their faces should be seen clearly.

1.3 PRODUCT SCOPE

Now we have designed a website. We are planning to develop an APP by next semester which can be directly integrated with the MGNREGA website. If the APP designed is successful then there will no mistake capturing the attendance. The generated attendances will always have accurate results.

1.4 PROBLEM DEFINITION

- ❖ Attendance taking at the rural areas takes lot of worktimes for the people working there. When the attendance is taken manually it actually consuming a lot of time and confusing for the attendance taker in not knowing whether it is the same person or not sometimes, as it is having very fewer working hours (approximately 2hrs). A system may be developed using automatic analysis of uploaded documents to upload and detect and match the faces for automatic attendance in the portal so that this problem of taking attendance manually which is time taking process may be completely resolved and that time can be used for working.

CHAPTER 2

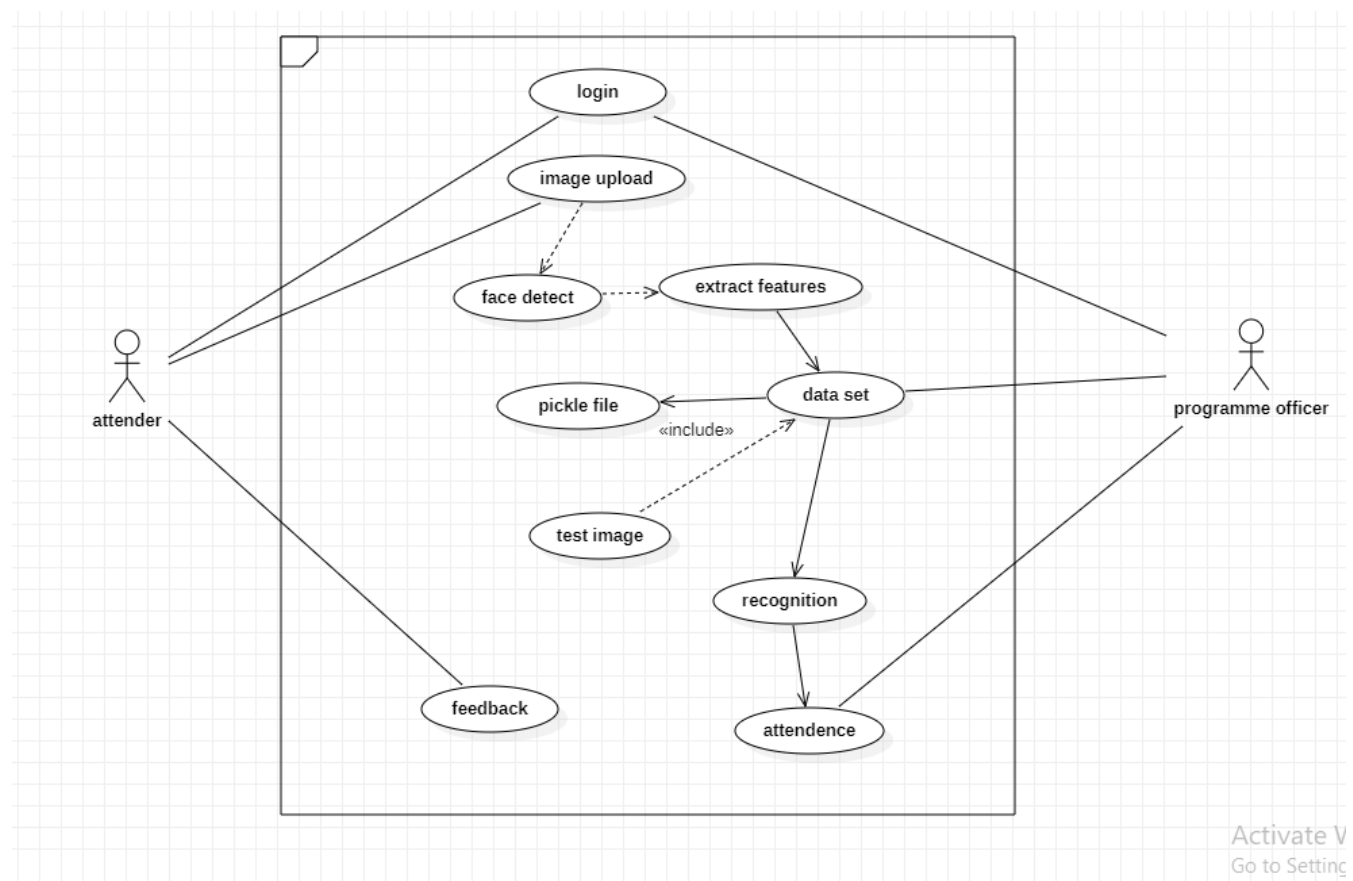
RELATED WORK –

Many of today's new and innovative artificial intelligence (AI) applications use OpenCV- based deep learning technology to capture, interpret, and analyze various kinds of video, audio, and text data. A OpenCV is a type of deep learning algorithm that is most often applied to analyze and learn visual features from large amounts of data. While primarily used for image-related AI applications, OpenCV can be used for other AI tasks, Recognition of objects for counting and object tracking. OpenCV here is used to identify the faces and convert it into a pickle file.

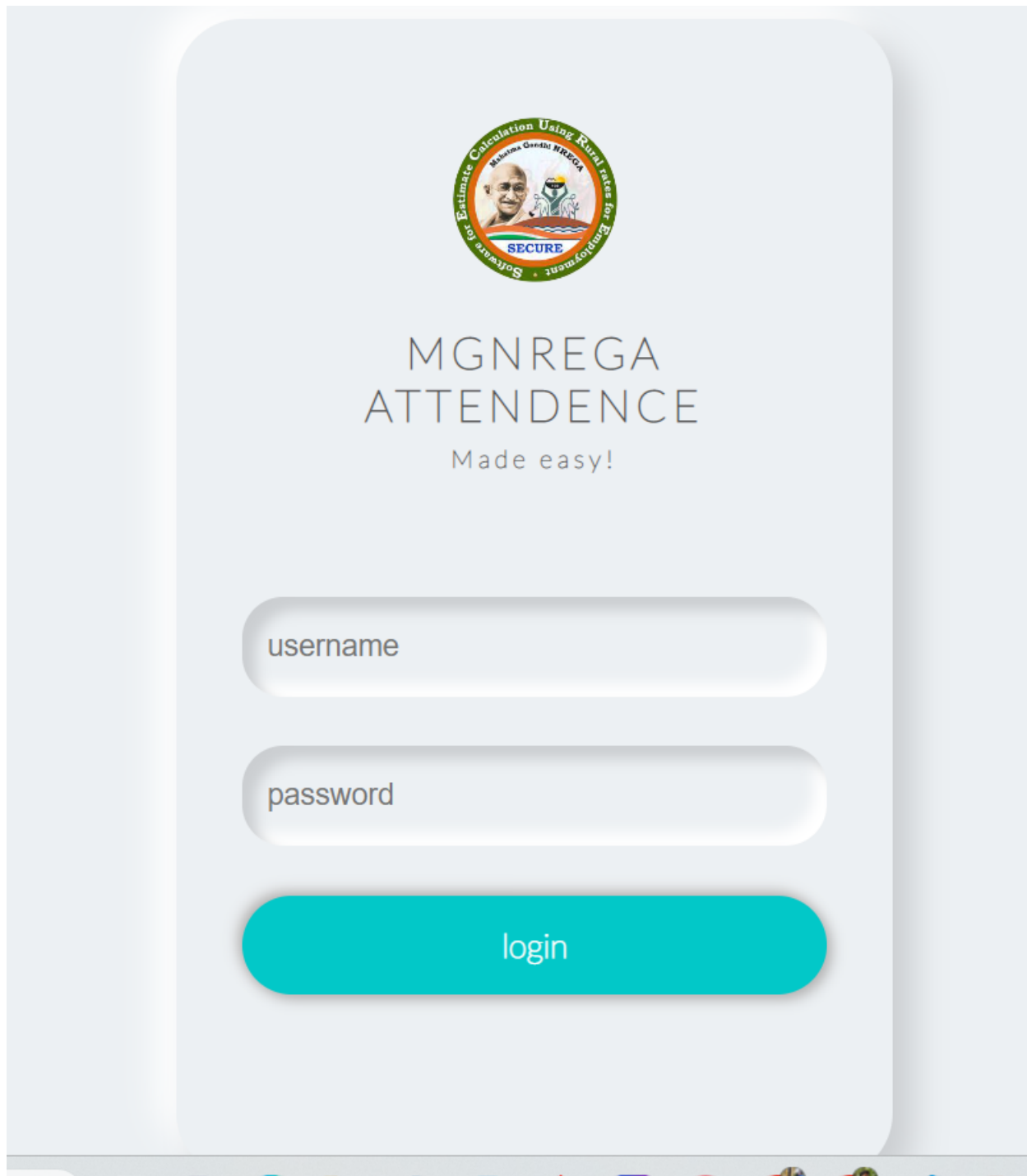
CHAPTER 3

PROPOSED WORK –


3.1 Use cases –



3.2 UI prototypes or screenshots



The image shows a mobile application login screen for MGNREGA ATTENDANCE. At the top center is a circular logo with a portrait of Mahatma Gandhi and the text "Estimate Calculation Using Rural rates for Employment", "Mahatma Gandhi NREGA", and "SECURE". Below the logo, the text "MGNREGA ATTENDANCE" is displayed in a large, sans-serif font, followed by the tagline "Made easy!" in a smaller font. The login form consists of three rounded rectangular fields: a "username" field, a "password" field, and a teal "login" button. The entire interface is set against a light gray background with a subtle shadow effect.



MGNREGA
ATTENDANCE

Made easy!

username

password

login

3.3 Technology used –

Front-end:

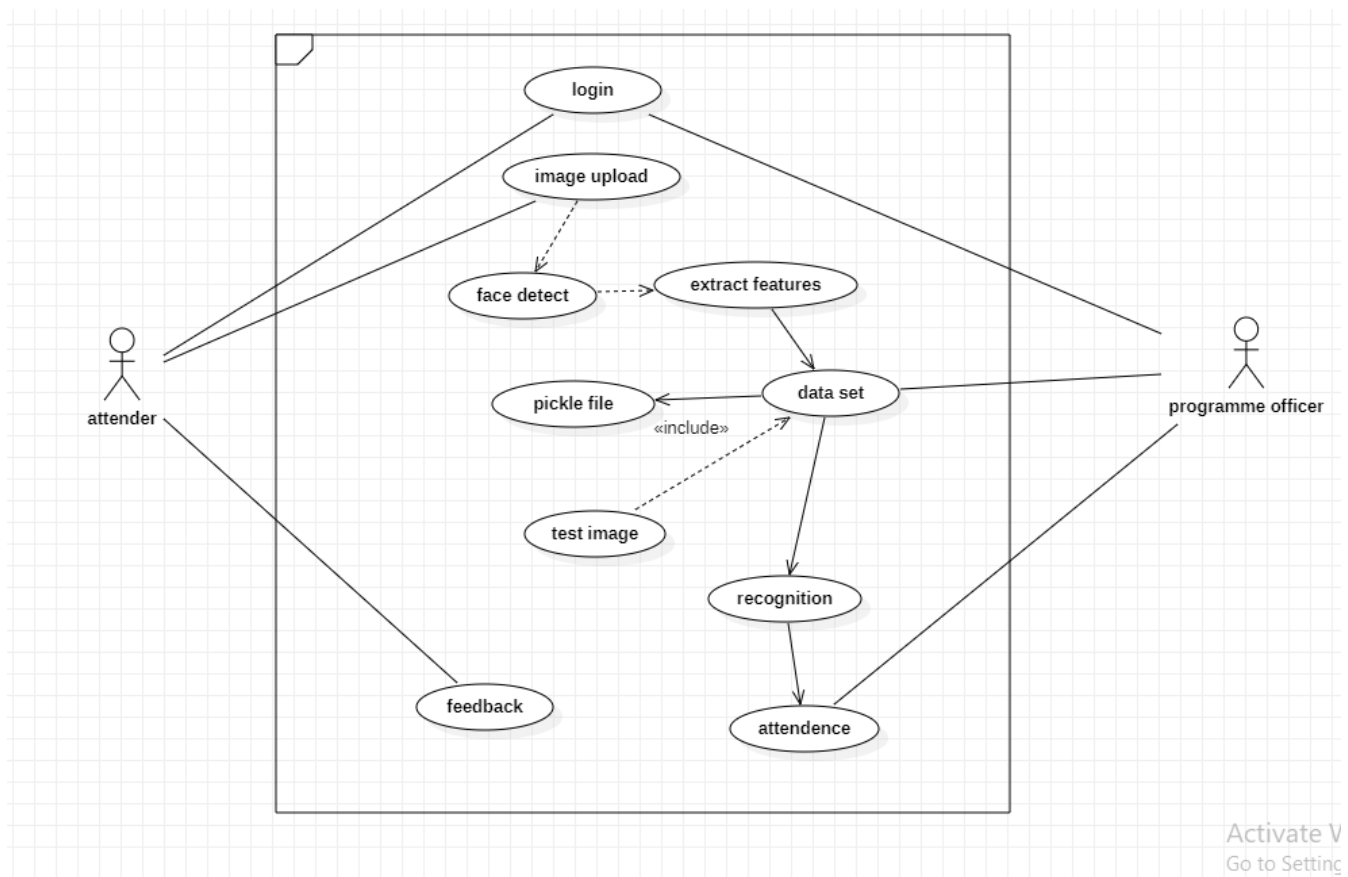
- HTML
- CSS

Back-end:

- OpenCV
- SKlearn
- CNN
- pickle

3.4 Design –

USE CASE DIAGRAM



3.5 Implementation-

3.5.1– Modules :

The various intents the face is trained on are –

3.5.1.1–Upload only images –

If any other format files are uploaded then a message will be flashed on the screen.

3.5.3– Code –

Extract embeddings

```
from imutils import paths
import numpy as np
import argparse
import imutils
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
                help="path to input directory of faces + images")
ap.add_argument("-e", "--embeddings", required=True,
                help="path to output serialized db of facial embeddings")
ap.add_argument("-d", "--detector", required=True,
                help="path to OpenCV's deep learning face detector")
ap.add_argument("-m", "--embedding-model", required=True,
                help="path to OpenCV's deep learning face embedding model")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
                              "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)
```

```

# load our serialized face embedding model from disk
print("[INFO] loading face recognizer...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])

# grab the paths to the input images in our dataset
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))

# initialize our lists of extracted facial embeddings and
# corresponding people names
knownEmbeddings = []
knownNames = []

# initialize the total number of faces processed
total = 0

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    print("[INFO] processing image {}/{}".format(i + 1,
        len(imagePaths)))
    name = imagePath.split(os.path.sep)[-2]

    # load the image, resize it to have a width of 600 pixels (while
    # maintaining the aspect ratio), and then grab the image
    # dimensions
    image = cv2.imread(imagePath)
    image = imutils.resize(image, width=600)
    (h, w) = image.shape[:2]

    # construct a blob from the image
    imageBlob = cv2.dnn.blobFromImage(
        cv2.resize(image, (300, 300)), 1.0, (300, 300),
        (104.0, 177.0, 123.0), swapRB=False, crop=False)

    # apply OpenCV's deep learning-based face detector to localize
    # faces in the input image
    detector.setInput(imageBlob)
    detections = detector.forward()

    # ensure at least one face was found
    if len(detections) > 0:
        # we're making the assumption that each image has only ONE

```

```

# face, so find the bounding box with the largest probability
i = np.argmax(detections[0, 0, :, 2])
confidence = detections[0, 0, i, 2]

# ensure that the detection with the largest probability also
# means our minimum probability test (thus helping filter out
# weak detections)
if confidence > args["confidence"]:
    # compute the (x, y)-coordinates of the bounding box for
    # the face
    box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
    (startX, startY, endX, endY) = box.astype("int")

    # extract the face ROI and grab the ROI dimensions
    face = image[startY:endY, startX:endX]
    (fH, fW) = face.shape[:2]

    # ensure the face width and height are sufficiently large
    if fW < 20 or fH < 20:
        continue

    # construct a blob for the face ROI, then pass the blob
    # through our face embedding model to obtain the 128-d
    # quantification of the face
    faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255,
                                      (96, 96), (0, 0, 0), swapRB=True, crop=False)
    embedder.setInput(faceBlob)
    vec = embedder.forward()

    # add the name of the person + corresponding face
    # embedding to their respective lists
    knownNames.append(name)
    knownEmbeddings.append(vec.flatten())
    total += 1

```

```

# dump the facial embeddings + names to disk
print("[INFO] serializing {} encodings...".format(total))
data = {"embeddings": knownEmbeddings, "names": knownNames}
f = open(args["embeddings"], "wb")
f.write(pickle.dumps(data))
f.close()

```

Training

```
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
import argparse
import pickle

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--embeddings", required=True,
                help="path to serialized db of facial embeddings")
ap.add_argument("-r", "--recognizer", required=True,
                help="path to output model trained to recognize faces")
ap.add_argument("-l", "--le", required=True,
                help="path to output label encoder")
args = vars(ap.parse_args())

# load the face embeddings
print("[INFO] loading face embeddings...")
data = pickle.loads(open(args["embeddings"], "rb").read())

# encode the labels
print("[INFO] encoding labels...")
le = LabelEncoder()
labels = le.fit_transform(data["names"])

# train the model used to accept the 128-d embeddings of the face and
# then produce the actual face recognition
print("[INFO] training model...")
recognizer = SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)

# write the actual face recognition model to disk
f = open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer))
f.close()

# write the label encoder to disk
f = open(args["le"], "wb")
f.write(pickle.dumps(le))
f.close()
```

#recognize.py

```
import numpy as np
import argparse
import imutils
import pickle
import cv2
import os

# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--image", required=True,
                help="path to input image")
ap.add_argument("-d", "--detector", required=True,
                help="path to OpenCV's deep learning face detector")
ap.add_argument("-m", "--embedding-model", required=True,
                help="path to OpenCV's deep learning face embedding model")
ap.add_argument("-r", "--recognizer", required=True,
                help="path to model trained to recognize faces")
ap.add_argument("-l", "--le", required=True,
                help="path to label encoder")
ap.add_argument("-c", "--confidence", type=float, default=0.5,
                help="minimum probability to filter weak detections")
args = vars(ap.parse_args())

# load our serialized face detector from disk
print("[INFO] loading face detector...")
protoPath = os.path.sep.join([args["detector"], "deploy.prototxt"])
modelPath = os.path.sep.join([args["detector"],
                              "res10_300x300_ssd_iter_140000.caffemodel"])
detector = cv2.dnn.readNetFromCaffe(protoPath, modelPath)

# load our serialized face embedding model from disk
print("[INFO] loading face recognizer...")
embedder = cv2.dnn.readNetFromTorch(args["embedding_model"])

# load the actual face recognition model along with the label encoder
recognizer = pickle.loads(open(args["recognizer"], "rb").read())
le = pickle.loads(open(args["le"], "rb").read())

# load the image, resize it to have a width of 600 pixels (while
# maintaining the aspect ratio), and then grab the image dimensions
image = cv2.imread(args["image"])
image = imutils.resize(image, width=600)
(h, w) = image.shape[:2]
```



```

# construct a blob from the image
imageBlob = cv2.dnn.blobFromImage(
    cv2.resize(image, (300, 300)), 1.0, (300, 300),
    (104.0, 177.0, 123.0), swapRB=False, crop=False)

# apply OpenCV's deep learning-based face detector to localize
# faces in the input image
detector.setInput(imageBlob)
detections = detector.forward()

# loop over the detections
for i in range(0, detections.shape[2]):
    # extract the confidence (i.e., probability) associated with the
    # prediction
    confidence = detections[0, 0, i, 2]

    # filter out weak detections
    if confidence > args["confidence"]:
        # compute the (x, y)-coordinates of the bounding box for the
        # face
        box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
        (startX, startY, endX, endY) = box.astype("int")

        # extract the face ROI
        face = image[startY:endY, startX:endX]
        (fH, fW) = face.shape[:2]

        # ensure the face width and height are sufficiently large
        if fW < 20 or fH < 20:
            continue

        # construct a blob for the face ROI, then pass the blob
        # through our face embedding model to obtain the 128-d
        # quantification of the face
        faceBlob = cv2.dnn.blobFromImage(face, 1.0 / 255, (96, 96),
            (0, 0, 0), swapRB=True, crop=False)
        embedder.setInput(faceBlob)
        vec = embedder.forward()

        # perform classification to recognize the face
        preds = recognizer.predict_proba(vec)[0]
        j = np.argmax(preds)
        proba = preds[j]

```

```
name = le.classes_[j]
```

```
# draw the bounding box of the face along with the associated  
# probability
```

```
text = "{}: {:.2f}%".format(name, proba * 100)
```

```
y = startY - 10 if startY - 10 > 10 else startY + 10
```

```
cv2.rectangle(image, (startX, startY), (endX, endY),  
              (0, 0, 255), 2)
```

```
cv2.putText(image, text, (startX, y),
```

```
           cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
```

```
# show the output image
```

```
cv2.imshow("Image", image)
```

```
cv2.waitKey(0)
```

#Program for connectivity.

```
#app.py
```

```
from flask import Flask, flash, request, redirect, url_for, render_template
```

```
import urllib.request
```

```
from module import check
```

```
#from module import imageQuality
```

```
import os
```

```
from werkzeug.utils import secure_filename
```

```
app = Flask(__name__)
```

```
UPLOAD_FOLDER = 'static/uploads/'
```

```
app.secret_key = "secret key"
```

```
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
```

```
app.config['MAX_CONTENT_LENGTH'] = 16 * 1024 * 1024
```

```
file_arr = []
```

```
ALLOWED_EXTENSIONS = set(['png', 'jpg', 'jpeg'])
```

```
def allowed_file(filename):
```

```
    return '.' in filename and filename.rsplit('.', 1)[1].lower() in ALLOWED_EXTENSIONS
```

```
@app.route('/')
```

```
def home():
```

```
    return render_template('index.html')
```

```
@app.route('/', methods=['POST'])
```

```
def upload_image():
```

```
    if 'file1' not in request.files:
```

```
        flash('No file part')
```

```

    return redirect(request.url)
file1 = request.files['file1']
file2 = request.files['file2']
if file1.filename == '':
    flash('No image selected for uploading')
    return redirect(request.url)
if file1 and allowed_file(file1.filename):
    filename1 = secure_filename(file1.filename)
    filename2 = secure_filename(file2.filename)
    file1.save(os.path.join(app.config['UPLOAD_FOLDER'], filename1))
    file2.save(os.path.join(app.config['UPLOAD_FOLDER'], filename2))

    # print('upload_image filename: ' + filename)
    flash('Images successfully uploaded and displayed below')
    # flash('\n FileName = '+filename)
    f1 = UPLOAD_FOLDER+filename1
    f2 = UPLOAD_FOLDER+filename2
    '''bscore = imageQuality(f1)
    if(bscore>30):
        flash(filename1+' Quality is low please reupload it')
    bscore = imageQuality(f2)
    if (bscore > 30):
        flash(filename2 + ' Quality is low please reupload it')'''

    print('FileName1 = '+filename1)
    #Processing the images
    a, b = check((f1), (f2))
    #check((f1),(f2))
    if(b=='photos'):
        flash('Upload only one photo')
    else:
        s = 'static/uploads/'
        if s in a:
            a = a.replace(s,"")
            print('a = '+a)
        if s in b:
            b = b.replace(s,"")
        #file_arr.clear()
        file_arr.append(a)
        file_arr.append(b)

    print(file1, file2)
    #print(file_arr)
    #print(a, b)
    return render_template('index.html', filename1=filename1, filename2 = filename2)

else:

```

```

        flash('Allowed image types are - png, jpg, jpeg')
        return redirect(request.url)
@app.route('/first')
def first():
    print('array = '+file_arr[0]+' '+file_arr[1])

    return render_template("first.html",filename1=file_arr[0], filename2 = file_arr[1])
@app.route('/display/<filename>')
def display_image(filename):
    print('display_image filename: ' + filename)
    return redirect(url_for('static', filename='uploads/' + filename), code=301)

if __name__ == "__main__":
    app.debug = True
    app.run()

```

FRONTEND

```

<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJISAWiGgFAW/dAiS6JXm"
    crossorigin="anonymous">
    <link
    href="https://fonts.googleapis.com/css2?family=Lato:wght@100&display=swap"
    rel="stylesheet">
    <link rel="stylesheet" href="style.css">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/font-awesome/6.0.0-beta2/css/all.min.css" integrity="sha512-
YWzhKL2whUzgiheMoBFwW8CKV4qpHQA Euvilg9FAn5VJUDwKZZxkJNuGM4XkWuk94W
Crrwslk8yWNGmY1EduTA==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
    <div class="login-div">
    <div class="logo">
        
    </div>
    <div class="title"> MGNREGA ATTENDENCE </div>

```

```

<div class="sub-title">Made easy!</div>
<div id="errmsg"></div>
<div class="fields">
  <div class="username">
    <form1 class="login" name="username " onsubmit="return validateloginform()"
method="POST">
      <input type="username" class="user-input" placeholder="username"/>
    </form1>
  </div>
  <div class="password">
    <form1 class="login" name="password" onsubmit="return validateloginform()"
method="POST">
      <input type="password" class="pass-input" placeholder="password"/>
    </form1>
  </div>
  <script src ="sripy.js"></script>
  <a href="2ndwebpage.html">
    <button class="signin-button">login</button>
  </a>

  </div>
</div>
</div>
</body>
</html>

```

Upload.html

```

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
integrity="sha384-
Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm"
crossorigin="anonymous">
  <link
href="https://fonts.googleapis.com/css2?family=Lato:wght@100&display=swap"
rel="stylesheet">
  <link rel="stylesheet" href="style.css">

```

```

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.0.0-beta2/css/all.min.css" integrity="sha512-
YWzhKL2whUzgiheMoBFwW8CKV4qpHQA Euvilg9FAn5VJUDwKZZxkJNuGM4XkWuk94W
Crrwslk8yWNGmY1EduTA==" crossorigin="anonymous" referrerpolicy="no-referrer" />
</head>
<body>
  <div class="login-div">
    <div class="fields">
      <div class="title1">PLEASE UPLOAD PHOTO
      </div>
      <form action="/action_page.php">
        <input type="file" id="myFile" name="filename"/>
      </form>
      <a >
        <button class="upload-button">upload</button>
      <a href
    </a>
  </div>
</div>
</body>

```

CSS:

```

{
  box-sizing: border-box;
}
body{
  margin: 0;
  height: 105vh;
  width:105vw;
  overflow: hidden;
  font-family: "Lato",sans-serif;
  font-weight: 700;
  display: flex;
  align-items:center;
  justify-content: center;
  color:#555;
  background-color: #ecf0f3;
}
.login-div{
  width:435px;
  height: 705px;
  padding: 60px 35px 35px 35px;
  border-radius:40px ;

```

```
background-color: #ecf0f3;
box-shadow: 13px 13px 20px #cbced1,
-13px -13px 20px #fff;

}
.logo{

    width: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
}
.logo img{
    width: 100px;
    border-radius: 50%;

}
.title{
    text-align: center;
    font-size: 27px;
    padding-top: 27px;
    letter-spacing: 3px;

}
.title1{
    text-align: bottom;
    font-size: 25px;
    padding-bottom: 45px;
    letter-spacing: 3px;
}
.sub-title{
    text-align: center;
    font-size: 15px;
    padding-top: 7px;
    letter-spacing: 3px;
}
.fields{
    width: 100%;
    padding: 75px 5px 5px 5px;

}
```

```
.fields input{
  border:none;
  outline:none;
  background:none ;
  font-size:18px;
  color:rgba(170, 39, 39, 0.856);
  padding:20px 10px 20px 5px;
}
```

```
.username,
.password{
  margin-bottom: 30px;
  border-radius:25px;
  box-shadow: inset 8px 8px 8px #cbced1,inset
-8px -8px 8px #fff;
  padding-left: 10px;
}
```

```
.signin-button{
  outline:none;
  border:none;
  cursor: pointer;
  width:100%;
  height: 60px;
  border-radius: 30px;
  font-size: 20px;
  font-weight: 700;
  font-family: "Lato",sans-serif;
  color:#fff;
  text-align:center;
  background-color: #02c8c8;
  box-shadow: 3px 3px 8px #b1b1b1,
-3px -3px 8px #b1b1b1, -3px -3px 8px #ffff;
  transition: all 0.5s;
}
```

```
.upload-button{
  outline:none;
  border: none;
  cursor: pointer;
  position: relative;
  bottom: -50px;
  padding:bottom;
```



```
padding-bottom: 30px;
width:40%;
height: 30px;
border-radius: 25px;
font-size: 20px;
font-weight: 700;
font-family: "Lato",sans-serif;
color:#fff;
text-align:center;
background-color: #02c8c8;
box-shadow: 3px 3px 8px #b1b1b1,
-3px -3px 8px #b1b1b1, -3px -3px 8px #fff;
transition: all 0.5s;

}
```

```
.signin-button:hover{
    background-color: #50aee5;
```

```
}
```

```
.upload-button:hover{
    background-color: #50aee5;
}
```

```
.signin-button:active{
    background-color: #50e5b9;
}
```

```
.upload-button:active{
    background-color: #50aee5;
}
```

```
.link{
    padding-top:20px;
    text-align:center;
}
```

```
.link a{
    text-decoration: none;
    color:#aaa;
    font-size:15px;
```

```
}
```

```
#errorMsg{
    color:red;
    text-align: center;
    font-size: 12px;
```

```
padding-bottom: 20px;
```

```
}
```

```
.form1 {
```

```
position: relative;
```

```
display: inline-flex;
```

```
width: 100%;
```

```
height: 100%;
```

```
    flex-shrink: 0;
```

```
    flex-direction: column;
```

```
    transition: right 0.5s;
```

```
}
```

```
Script.js
```

```
function validateloginform(){
```

```
    var username=document.getElementById("username").value;
```

```
    var password=document.getElementById("password").value;
```

```
    if(username=="" || password==""){
```

```
        document.getElementById("errmsg").innerHTML="please fill the requires fields"
```

```
        return false;
```

```
    }
```

```
    else if(password.lenght<6){
```

```
        document.getElementById("errmsg").innerHTML="your password must include
```

```
atleast 6 characters "
```

```
        return false;
```

```
    }
```

```
    else{
```

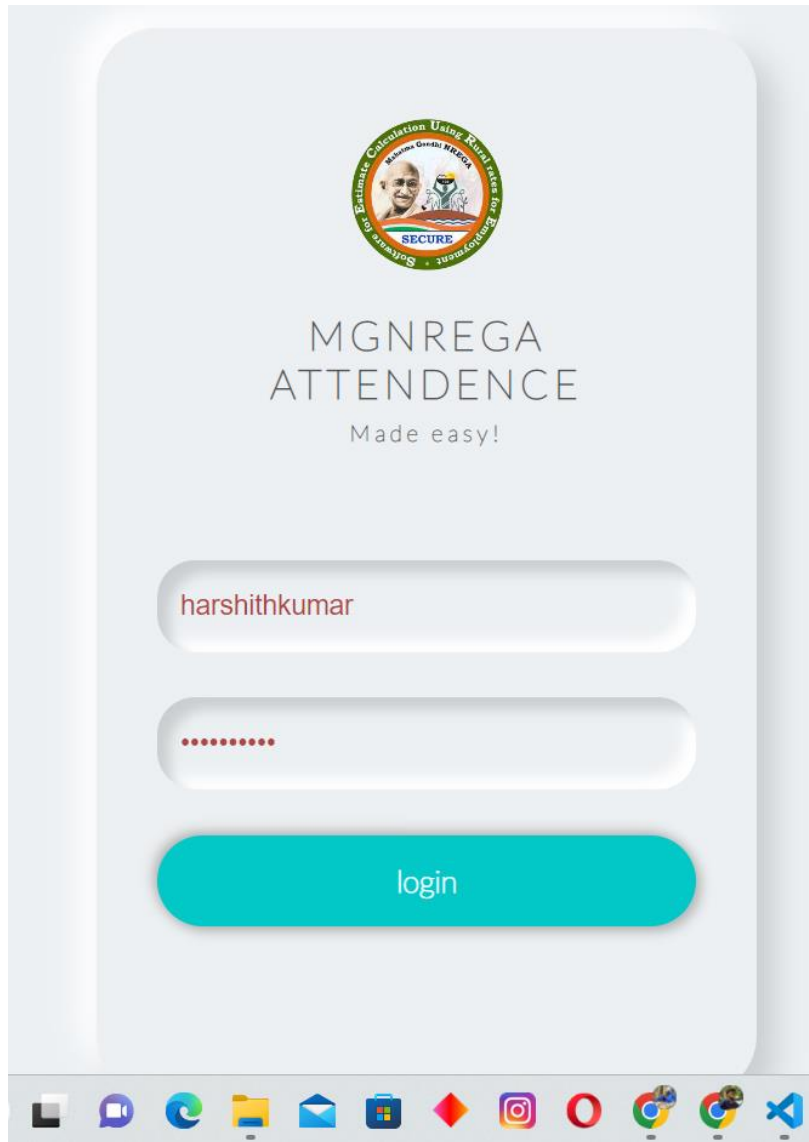
```
        return true;
```

```
    }
```

```
}
```

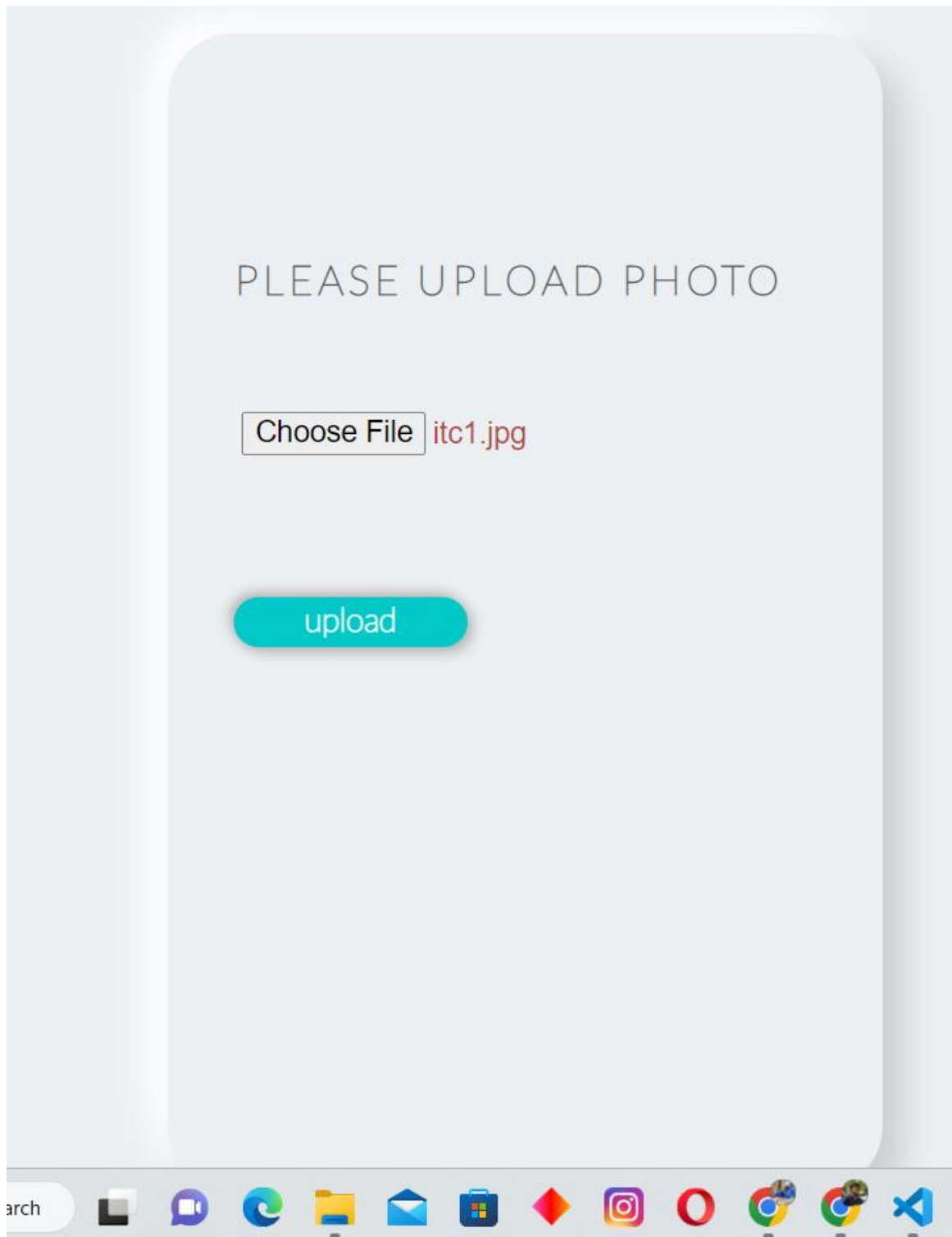
TESTING

3 3.6.1- login photo:



4

3.6.2- upload photo:



3.6.3- photo which is uploaded and photo after recognition:

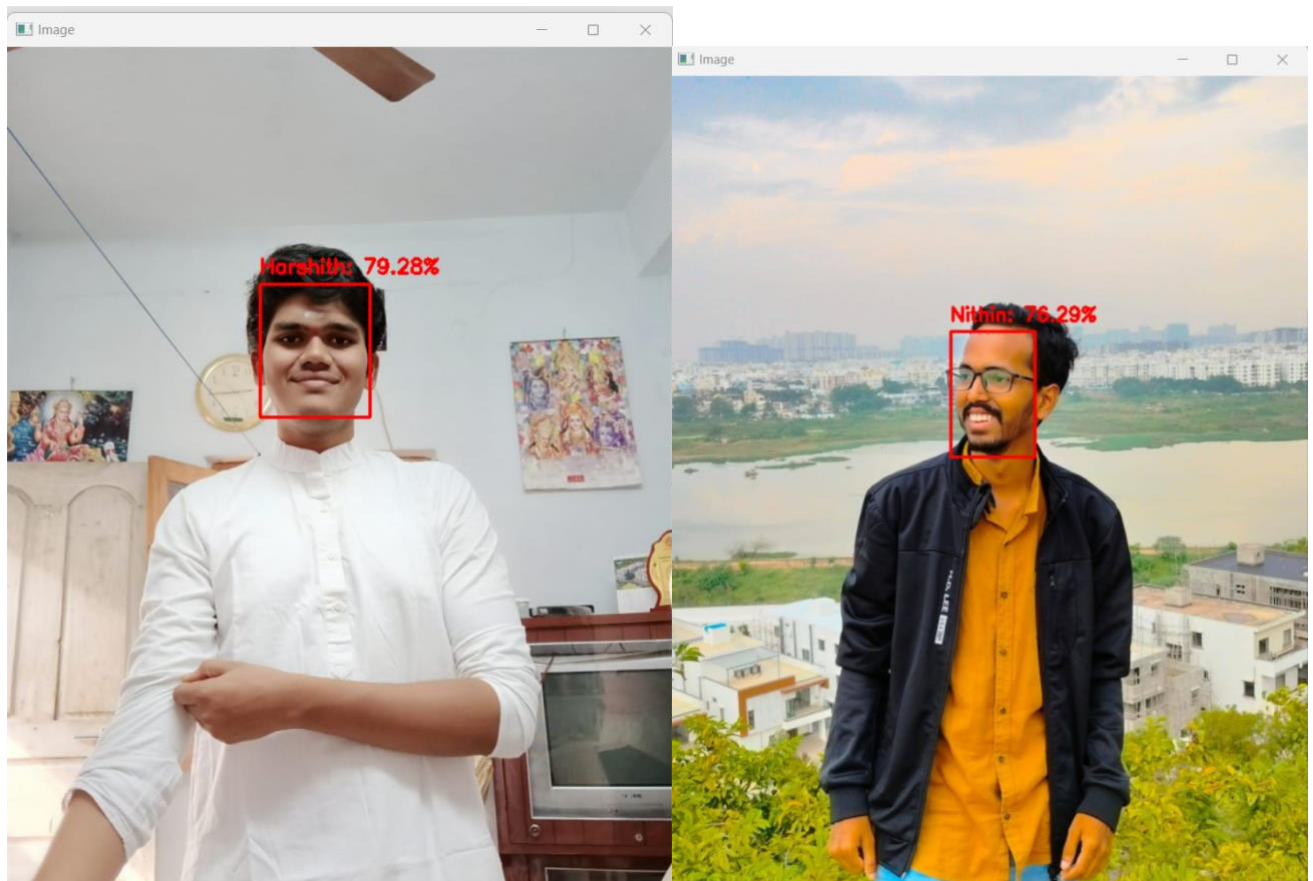


CHAPTER 4 –

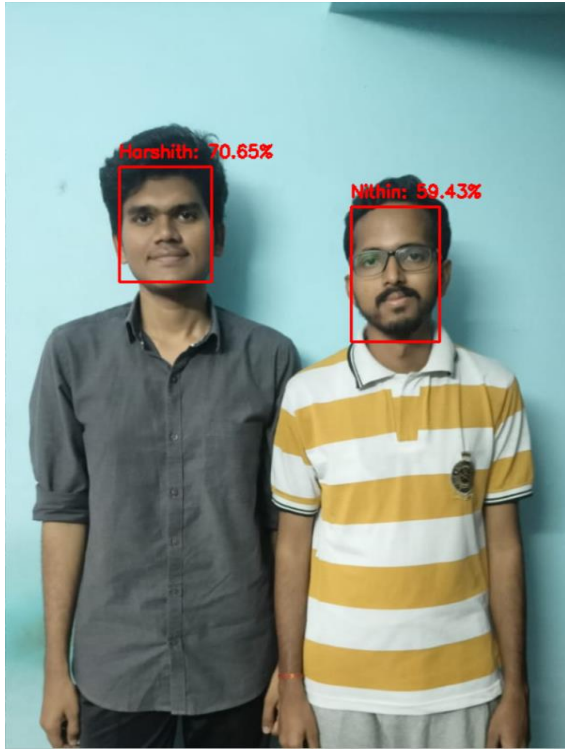
RESULTS:

4.1:

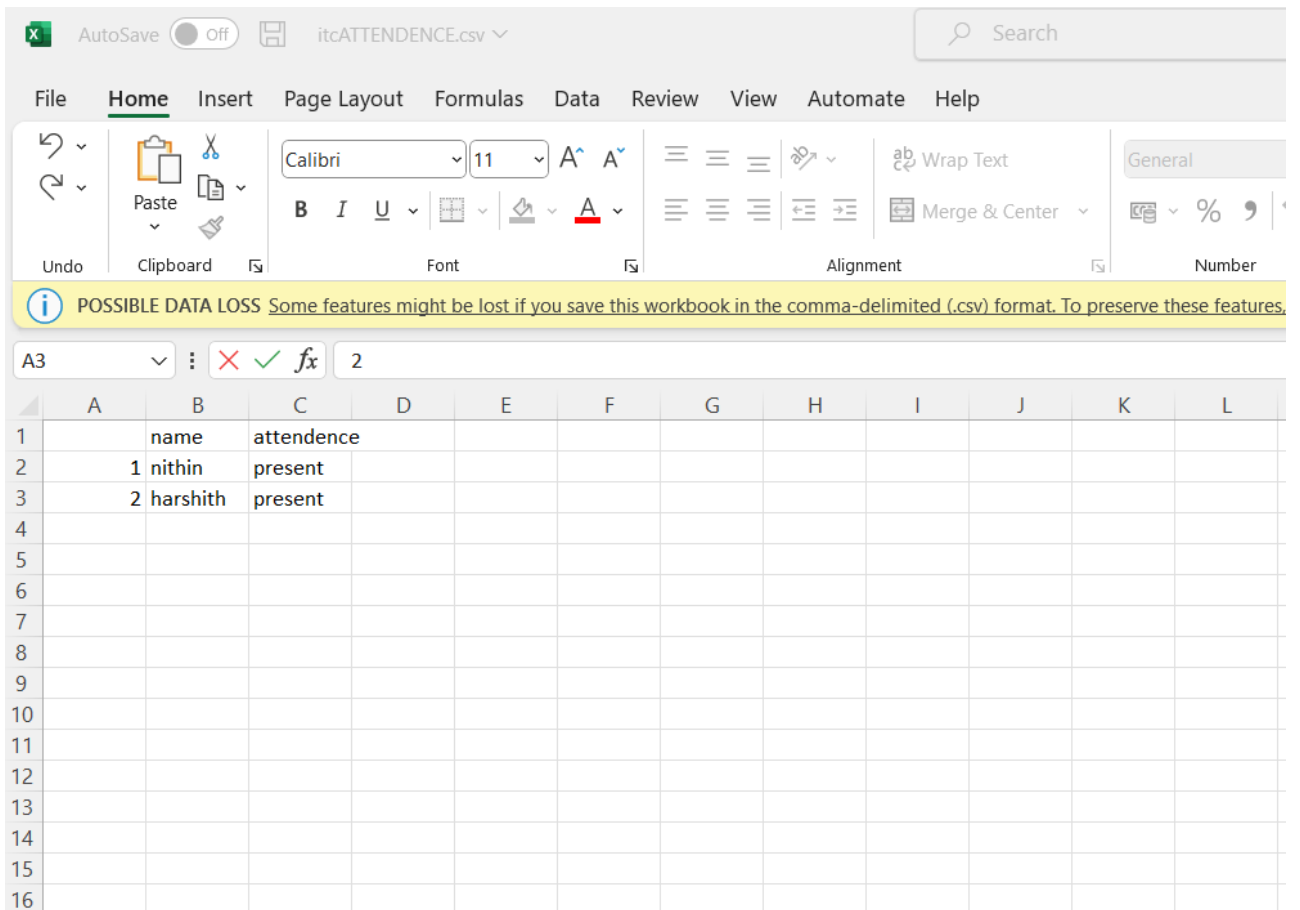
Recognizing one face:



Recognizing two face:



Generating excel sheet:



CHAPTER 5

DISCUSSION AND FUTURE WORK –

- The project can be further extended by developing an APP.
- The developed APP can be integrated with many other portals such as MGNREGA,..etc.

CHAPTER 6

REFERENCES –

- https://nrega.nic.in/Nregahome/MGNREGA_new/Nrega_home.aspx
- <https://www.planindia.org/support-my-village/.aspx>
- <https://pyimagesearch.com/2018/02/26/face-detection-with-opencv-and-deep-learning/>
- <https://rural.nic.in>

Data sets:

Human Faces:

<https://www.kaggle.com/datasets/ashwingupta3012/human-faces>
<https://www.kaggle.com/datasets/harshith25/human-faces>

