```python
import cv2

import os

from flask import Flask,request,render_template

from datetime import date

from datetime import datetime

import numpy as np

from sklearn.neighbors import KNeighborsClassifier

import pandas as pd

import joblib


#### Defining Flask App

app = Flask(__name__)



#### Saving Date today in 2 different formats

datetoday = date.today().strftime("%m_%d_%y")

datetoday2 = date.today().strftime("%d-%B-%Y")



#### Initializing VideoCapture object to access WebCam

face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')



#### If these directories don't exist, create them

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

    os.makedirs('Attendance')

if not os.path.isdir('static'):

    os.makedirs('static')

if not os.path.isdir('static/faces'):

    os.makedirs('static/faces')

if f'Attendance-{datetoday}.csv' not in os.listdir('Attendance'):

    with open(f'Attendance/Attendance-{datetoday}.csv','w') as f:

        f.write('Name,Roll,Time')



#### get a number of total registered users

def totalreg():

    return len(os.listdir('static/faces'))
```

```python
#### extract the face from an image
def extract_faces(img):
    try:
        if img.shape!=(0,0,0):
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            face_points = face_detector.detectMultiScale(gray, 1.3, 5)
            return face_points
        else:
            return []
    except:
        return []

#### Identify face using ML model
def identify_face(facearray):
    model = joblib.load('static/face_recognition_model.pkl')
    return model.predict(facearray)


#### A function which trains the model on all the faces available in faces folder
def train_model():
    faces = []
    labels = []
    userlist = os.listdir('static/faces')
    for user in userlist:
        for imgname in os.listdir(f'static/faces/{user}'):
            img = cv2.imread(f'static/faces/{user}/{imgname}')
            resized_face = cv2.resize(img, (50, 50))
            faces.append(resized_face.ravel())
            labels.append(user)
    faces = np.array(faces)
    knn = KNeighborsClassifier(n_neighbors=5)
    knn.fit(faces,labels)
    joblib.dump(knn,'static/face_recognition_model.pkl')


#### Extract info from today's attendance file in attendance folder
def extract_attendance():
    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')
    names = df['Name']
    rolls = df['Roll']

    times = df['Time']

    l = len(df)

    return names,rolls,times,l



#### Add Attendance of a specific user

def add_attendance(name):

    username = name.split('_')[0]

    userid = name.split('_')[1]

    current_time = datetime.now().strftime("%H:%M:%S")


    df = pd.read_csv(f'Attendance/Attendance-{datetoday}.csv')

    if int(userid) not in list(df['Roll']):

        with open(f'Attendance/Attendance-{datetoday}.csv','a') as f:

            f.write(f'\n{username},{userid},{current_time}')



def getallusers():

    userlist = os.listdir('static/faces')

    names = []

    rolls = []

    l = len(userlist)


    for i in userlist:

        name,roll = i.split('_')

        names.append(name)

        rolls.append(roll)


    return userlist,names,rolls,l


def deletefolder(duser):

    pics = os.listdir(duser)


    for i in pics:

        os.remove(duser+'/'+i)


    os.rmdir(duser)
```

```python
################## ROUTING FUNCTIONS #########################

#### Our main page
@app.route('/')
def home():
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=l,totalreg=totalreg(),datetoday2=dat
etoday2)


#### This function will run when we click on Take Attendance Button
@app.route('/start',methods=['GET'])
def start():
    if 'face_recognition_model.pkl' not in os.listdir('static'):
        return render_template('home.html',totalreg=totalreg(),datetoday2=datetoday2,mess='There is no
trained model in the static folder. Please add a new face to continue.')

    ret = True
    cap = cv2.VideoCapture(0)
    while ret:
        ret,frame = cap.read()
        if len(extract_faces(frame))>0:
            (x,y,w,h) = extract_faces(frame)[0]
            cv2.rectangle(frame,(x, y), (x+w, y+h), (255, 0, 20), 2)
            face = cv2.resize(frame[y:y+h,x:x+w], (50, 50))
            identified_person = identify_face(face.reshape(1,-1))[0]
            add_attendance(identified_person)
            cv2.putText(frame,f'{identified_person}',(30,30),cv2.FONT_HERSHEY_COMPLEX_SMALL,1,(255,
255, 255),2,cv2.LINE_AA)
        cv2.imshow('Attendance',frame)
        if cv2.waitKey(1)==27:
            break
    cap.release()
    cv2.destroyAllWindows()
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=l,totalreg=totalreg(),datetoday2=dat
etoday2)
```

```
#### This function will run when we add a new user
@app.route('/add',methods=['GET','POST'])
def add():
    newusername = request.form['newusername']
    newuserid = request.form['newuserid']
    userimagefolder = 'static/faces/'+newusername+'_'+str(newuserid)
    if not os.path.isdir(userimagefolder):
        os.makedirs(userimagefolder)
    i,j = 0,0
    cap = cv2.VideoCapture(0)
    while 1:
        _,frame = cap.read()
        faces = extract_faces(frame)
        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x, y), (x+w, y+h), (255, 0, 20), 2)
            cv2.putText(frame,f'Images Captured:
{i}/50',(30,30),cv2.FONT_HERSHEY_COMPLEX_SMALL,1,(255, 255, 255),2,cv2.LINE_AA)
            if j%10==0:
                name = newusername+'_'+str(i)+'.jpg'
                cv2.imwrite(userimagefolder+'/'+name,frame[y:y+h,x:x+w])
                i+=1
            j+=1
        if j==500:
            break
        cv2.imshow('Adding new User',frame)
        if cv2.waitKey(1)==27:
            break
    cap.release()
    cv2.destroyAllWindows()
    print('Training Model')
    train_model()
    names,rolls,times,l = extract_attendance()
    return
render_template('home.html',names=names,rolls=rolls,times=times,l=l,totalreg=totalreg(),datetoday2=dat
etoday2)
```

```python
#### Our main function which runs the Flask App

if __name__ == '__main__':

    app.run(debug=True)
```