APPENDIX

CODING LOGIN WEBSITE

```
import streamlit as web
import streamlit_authenticator as loginauth
import mysql.connector
def login():
    connection = mysql.connector.connect(
    host="localhost",
    user="root",
    password="",
    database="dbproject" )
     login = connection.cursor()
     login.execute("CALL spGetLogin()")
     logindata = login.fetchall()
     for x,y,z in logindata:
         user.append(x)
         psw.append(y)
         nama.append(z)
     login.close()
     connection.close()

login()
hashed = loginauth.Hasher(psw).generate()
authenticator=
loginauth.Authenticate(nama,user,hashed,'cookie_name', 88.
                       'signature_key',cookie_expiry_days=1)
nama,status,usr= authenticator.login('Login','sidebar')
```

CODING FOR INPUT IMAGES

```
detector = cv2.CascadeClassifier(haarcascade_frontalface_default.xml")
def preprocess_image(picture):
   img= Image.open(picture)
   img = np.array(img)
   try:
        detected_face = detector.detectMultiScale(img, 1.3, 5)
        x, y, w, h = detected_face[0]
        image = img[int(y):int(y+h),
                  int(x):int(x+w)]
       image=cv2.resize(image,(96,96))
       image=cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
       image = np.expand_dims(image, axis=0)
       image = preprocess_input(image)
   except:
       exit('Face not detected')
   return image
```

CODING OPENFACE FEATURE EXTRACTION

```
    # Inception3a
     inc3a_3,3=Conv2D(96,(1,1),name='inc3a_3,3_conv1')(x)
     inc3a_3,3 = BatchNormalization(epsilon=0.00001, axis=3,
                         name='inc3a_3,3_BatchNorm1')(inc3a_3,3)
     inc3a_3,3 = Activation('relu')(inc3a_3,3)
     inc3a_3,3 = ZeroPadding2D(padding=(1, 1))(inc3a_3,3)
     inc3a_3,3 = Conv2D(128, (3, 3),
                       name='inc3a_3,3_conv2')(inc3a_3,3)
     inc3a_3,3 = BatchNormalization(epsilon=0.00001, axis=3,
                          name='inc3a_3,3_BatchNorm2')(inc3a_3,3)
     inc3a_3,3 = Activation('relu')(inc3a_3,3)

     inc3a_5,5 = Conv2D(16, (1, 1),
            name='inc3a_5,5_conv1')(x)
     inc3a_5,5 = BatchNormalization(epsilon=0.00001, axis=3,
            name='inc3a_5,5_BatchNorm1')(inc3a_5,5)
     inc3a_5,5 = Activation('relu')(inc3a_5,5)
     inc3a_5,5 = ZeroPadding2D(padding=(2, 2))(inc3a_5,5)
     inc3a_5,5 = Conv2D(32, (5, 5),
            name='inc3a_5,5_conv2')(inc3a_5,5)
     inc3a_5,5 = BatchNormalization(epsilon=0.00001, axis=3,
            name='inc3a_5,5_BatchNorm2')(inc3a_5,5)
     inc3a_5,5 = Activation('relu')(inc3a_5,5)

     inc3a_pool = MaxPooling2D(pool_size=3, strides=2)(x)
     inc3a_pool = Conv2D(32, (1, 1),
            name='inc3a_pool_conv')(inc3a_pool)
     inc3a_pool = BatchNormalization(epsilon=0.00001, axis=3,
            name='inc3a_pool_BatchNorm')(inc3a_pool)
     inc3a_pool = Activation('relu')(inc3a_pool)
     inc3a_pool = ZeroPadding2D(padding=((3, 4),
           (3,4)))(inc3a_pool)

     inc3a_1,1 = Conv2D(64, (1, 1),
            name='inc3a_1,1_conv')(x)
     inc3a_1,1 = BatchNormalization(epsilon=0.00001, axis=3,
            name='inc3a_1,1_BatchNorm')(inc3a_1,1)
     inc3a_1,1 = Activation('relu')(inc3a_1,1)

     inc3a = concatenate([inc3a_3,3, inc3a_5,5,
            inc3a_pool, inc3a_1,1], axis=3)
   #Inception 3b …
   #Inception 3c …
   #Inception 4a …
   #Inception 4e …
   #Inception 5a …
   #Inception 5b …

 model = builtModel()
 model.load_weights('openface_weights.h5')
```

CODING FIND SIMILARITY SCORE BETWEEN 2 FACES

```
def counteuclidean(testface, trainingface):
    subs = trainingface - testface
    skor = np.sum(np.multiply(subs, subs))
    skor = np.sqrt(score)
    return skor

def countcosine(testface, trainingface):
    transposeimg=np.transpose(trainingface)
    x = np.sum(np.multiply(trainingface, trainingface))
    y = np.sum(np.multiply(testface, testface))
    z = np.matmul(transposeimg, testface)
    skor = 1 - (z / (np.sqrt(x) * np.sqrt(y)))
    return skor
```

CODING TAKING ATTENDANCE SYSTEM

```
datatraining=[]
dir_training="./database/%s" %(usr)

    for file in listdir(dir_training):
        data, extension = file.split(".")
        training = './database/%s/%s.jpg' %(usr,data)
        datatraining.append(model.predict(preprocess_image(training))
         [0,:])
    method = web.radio("Using Method :",('Cosine', 'Euclidean'))
    label="0"
    if method=='Cosine':
        picture = web.camera_input(label,label_visibility="hidden")
        if picture:
            datatest = model.predict(preprocess_image(picture))[0,:]
            score=0
            for x in datatraining:
                score=score+countcosine(datatest,x)
            score=score/len(datatraining)
            if score<0.014:
                attendance()
                web.warning('Face Verified')
            else:
                web.warning('Face Not Verified')
     elif method=='Euclidean':
        picture = web.camera_input(label,label_visibility="hidden")
        if picture:
            score=0
            datatest = model.predict(preprocess_image(picture))[0,:]
            for x in datatraining:
                score=score+counteuclidean(datatest,x)
            score=score/len(datatraining)
            if score<0.17:
                attendance()
                web.warning('Face Verified')
            else:
                web.warning('Face Not Verified')
```

PAPER NAME

TA-19.K1.0026.docx

WORD COUNT

3429 Words

CHARACTER COUNT

17156 Characters

PAGE COUNT

17 Pages

FILE SIZE

62.7KB

SUBMISSION DATE

Dec 16, 2022 3:46 PM GMT+7

REPORT DATE

Dec 16, 2022 3:47 PM GMT+7

11% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

- 8% Internet database
- Crossref database
- 9% Submitted Works database
- 6% Publications database
- Crossref Posted Content database