

# **EMPOWERING EMOTIONAL CONNECTIONS THROUGH AN ADVANCED AI-POWERED MUSIC PLAYER**

## **ABSTRACT**

Due to the fact that it contains vital information about human emotional states, facial expression is a powerful means for humans to communicate. It is a vital component of computing systems that are competent at identifying human emotions and responding to them more appropriately. However, the challenge of automatically recognizing various facial expressions makes the automated recognition of facial expressions a significant issue in human-system interactions, human emotion appraisal, and decision making. Facial expression detection has therefore become a hot area for research in the domains of image processing, pattern recognition, machine learning, and human reputation in addition to human-computer interaction. With the help of the HAAR CASCADES set of rules and the Support Vector Machine set of rules, we may use techniques in this mission to robotically identify face features and categories emotions. Using a K-Nearest Neighbor technique, provide a playlist of songs that are suited for his current state of mind. You may include a glance at photo of an expression you want to be recognized while trying out a feature. This look-at image may be compared to face database files to play music based on identified emotions. Finally, a player with an advanced recognition rate that is fully emotion-based is offered.

## **CHAPTER 1**

### **1. INTRODUCTION**

The objective of the proposed project is to develop a facial expression recognition system integrated with a music recommendation system, aimed at enhancing human-computer interaction and emotional response in computing environments. Implement facial expression detection using the HAAR CASCADES set of rules and the Support Vector Machine (SVM) set of rules. The system will be trained to automatically recognize various facial expressions, including happiness, sadness, anger, surprise, disgust, and fear. Utilize a K-Nearest Neighbor (KNN) technique to generate a playlist of songs suited for the user's current emotional state based on the detected facial expression. The system will analyze the user's emotional state and select music tracks that align with their mood, enhancing the overall user experience. Develop a seamless integration between facial expression detection and music recommendation systems. When a user's facial expression is detected, the system will automatically generate a personalized music playlist tailored to their emotional state in real-time. Design and deploy a user-friendly interface for the emotion-based music player, featuring advanced recognition algorithms and real-time feedback. The player will provide an intuitive and immersive experience for users, enhancing their engagement and emotional connection with the system.

## **CHAPTER 2**

### **2. SYSTEM STUDY AND ANALYSIS**

#### **2.1 EXISTING SYSTEM**

There are several regions in human–laptop interaction that would effectively use the functionality to understand emotion. The trouble of face detection may be taken into consideration as a hassle of binary class of photo body as both containing and now not containing a face. In order at the way to analyze any such type model, we first need to give an explanation for an photograph in terms of features, which would be specific signs of face presence or absence on a given image. The present technique is commonly entails tasks: The first is for extracting ASM movement primarily based a pyramid ASM model fitting method and the second for the projected movement type acquired via making use of Adaboost classifiers. After the segmentation of face applicants, 68 function factors in every face are then extracted the usage of ASM fitting method. The machine then line up three extracted feature factors, eyes and nostril element, to the imply form of ASM, and ignore the other part of the ASM towards the imply face shape of ASM to estimate the geometrical dislocation statistics between present day and suggest ASM factors coordinates. Then, facial expressions popularity is the acquired based totally on this geometrical motion the usage of Adaboost classifier. And also extracting features using viola jones. The functions that Viola and Jones used are based totally on wavelets. Wavelets are unmarried wavelength rectangular waves (one high c programming language and one low interval). In dimensions, a rectangular wave is a couple of adjacent rectangles - one light and one darkish

#### **DISADVANTAGES OF EXISTING SYSTEM**

- Provide large number of features points from facial images
- Emotions may be wrongly classified
- Complexity is high
- Difficult to implement in real time environments

## **2.2 PROPOSED SYSTEM**

In this task, a novel emotion regard machine primarily based on the processing of physiological alerts is supplied. This system indicates a reputation ratio tons higher than risk risk, whilst applied to physiological signal databases acquired from tens to masses of subjects. The system consists of feature face detection, characteristic extraction and pattern type ranges. Although the face detection and feature extraction tiers had been designed carefully, there was a large amount of inside-class variation of features and overlap among instructions. In order to discover Emotion from an picture, used frontal view facial images. If computer structures can apprehend more of human emotion, we will make better structures to reduce the space of human laptop interplay .To control the emotion recognition hassle from arbitrary view facial photographs. The facial vicinity and others a part of the frame were segmented from the complex surroundings based on pores and skin color model. Thus, on this task confirmed some variations among special shade fashions that are used to put in force the machine and which coloration version may be used in which. Another feature is to extract facial parts from the face. And for that used HAAR CASCADES to locate the eye and lips region from a face and then by the assist of SVM class detected emotion from those functions. From the positioning of mouth and eyes, tried to locate emotion of a face. The proposed machine attempts to supply an speaking manner for the consumer to perform the task of making a playlist. The operating is based on KNN mechanisms wearing out their function in a pre-defined order to get the preferred output. The labeled expression acts as an enter and is used to pick out the suitable playlist from the first of all generated playlists and the songs from the playlists are done. At this stage, the face symmetry is measured and the life of the precise facial capabilities is validated for each face candidate. And draw the bounding box and additionally calculate distance size from web cameras.

### **ADVANTAGES OF PROPOSED SYSTEM**

- Ease of use
- No trouble of troublesome selection of songs
- Can be used in real time environments
- Reduce number of features are extracted

## **CHAPTER 3**

### **3. SOFTWARE REQUIREMENTS**

- Operating system : Windows OS
- Front End : PYTHON
- IDE : PYCHARM

#### **3.1 HARDWARE ENVIRONMENT**

- Processor : Intel processor
- RAM : 1GB
- Hard disk : 160 GB
- Compact Disk : 650 Mb
- Keyboard : Standard keyboard
- Monitor : 15 inch color monitor

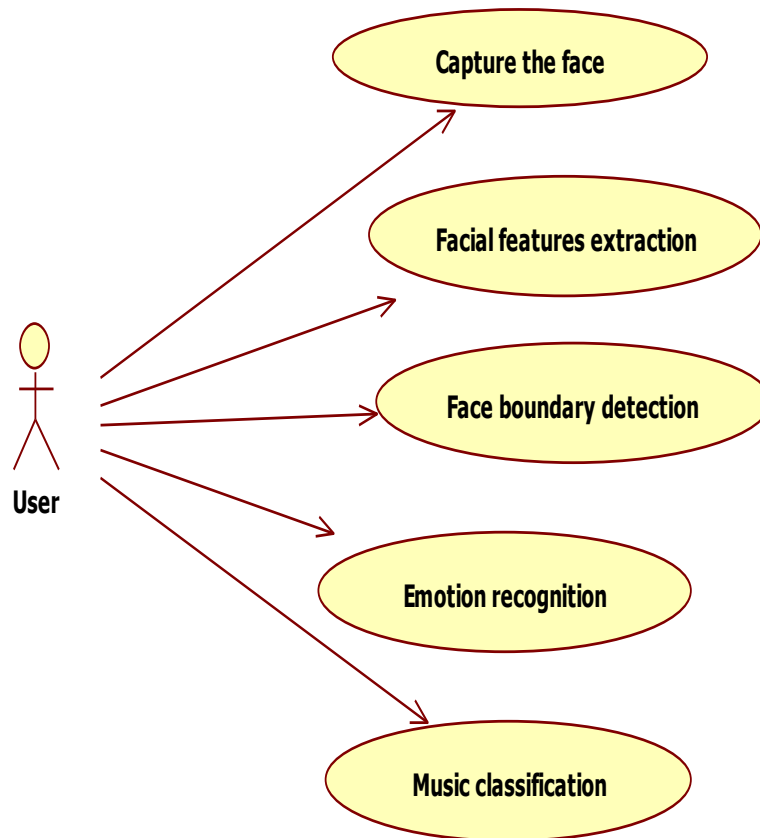
## CHAPTER 4

### 4. DIAGRAM REPRESENTATION

#### 4.1 UML DIAGRAM

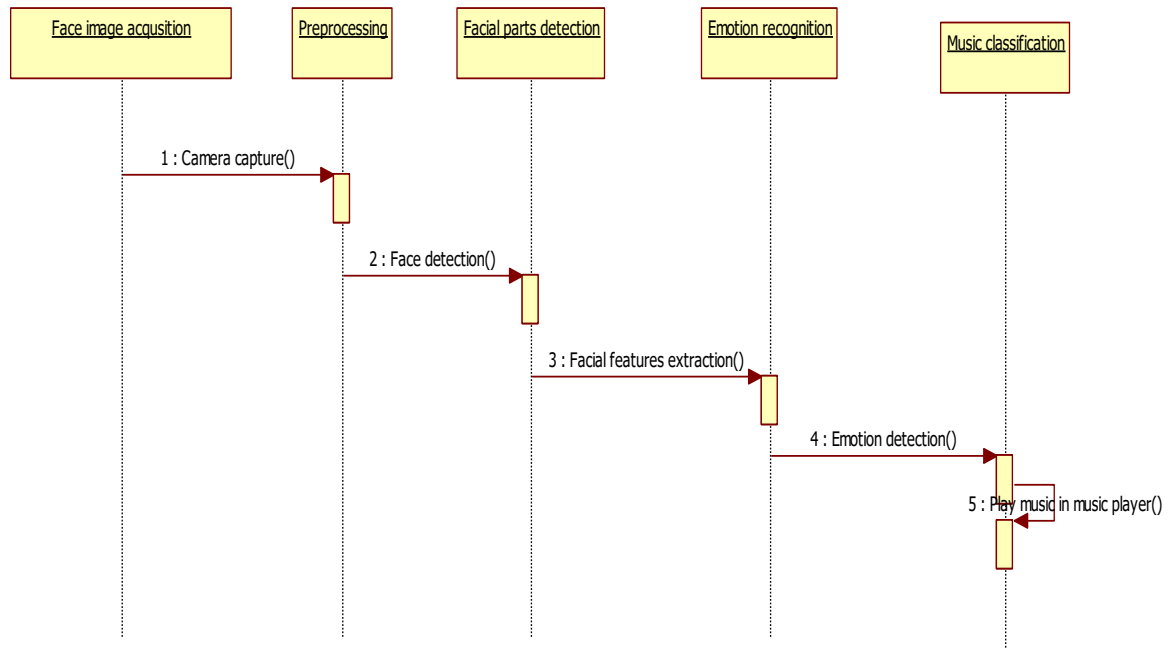
##### Use Case Diagram

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.



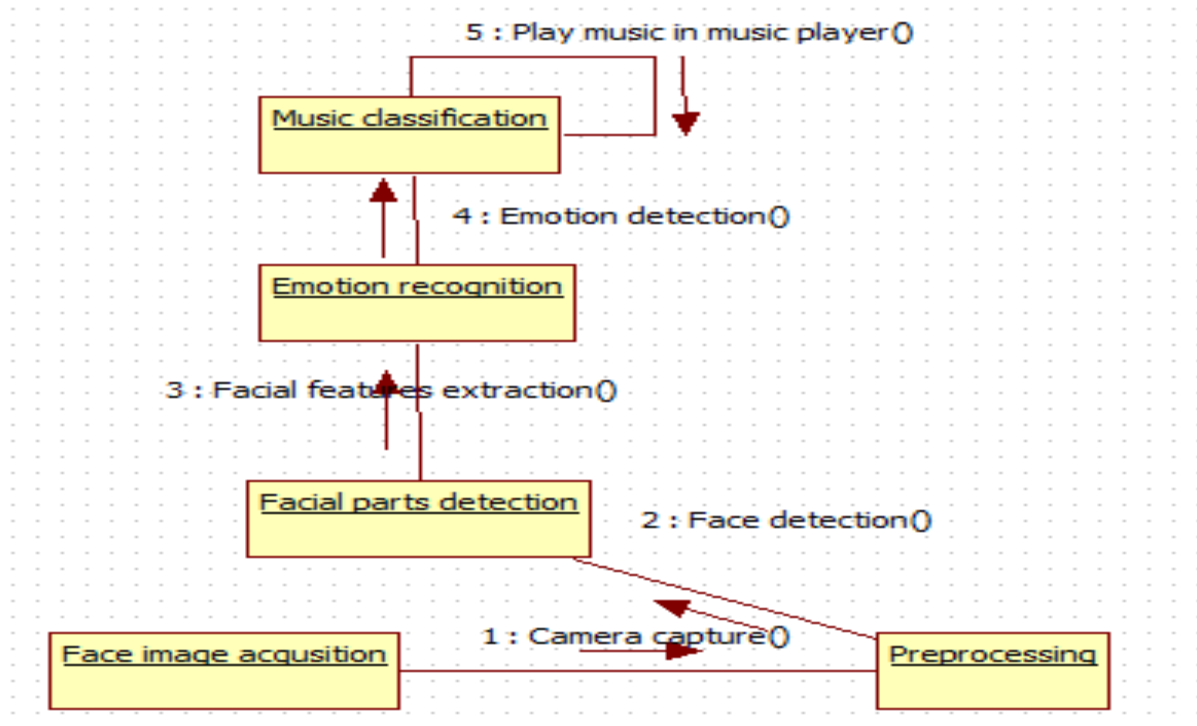
## Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.



## Collaboration Diagram

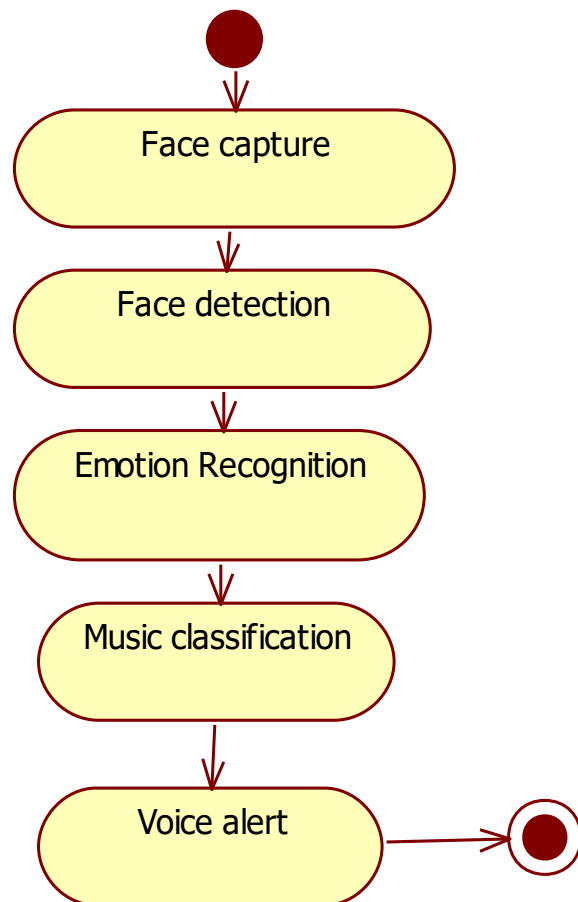
A collaboration diagram, also called a communication diagram or interaction diagram, is an illustration of the relationships and interactions among software objects in the Unified Modeling Language (UML).







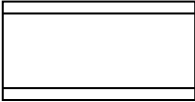
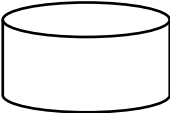
## Activity Diagram

Activity diagram displays a special state diagram, where most of the state are action states and most of the transitions are triggered by completion of the action in the source states. The activity can be described as an operation of the system. So the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. Activity diagrams deals with all type of flow control by using different elements.



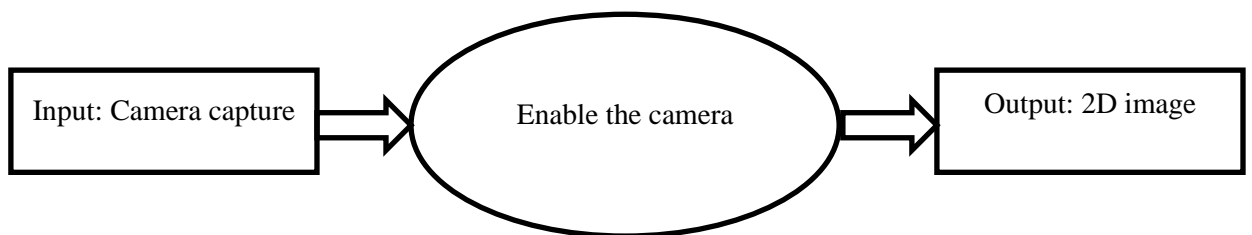
## 4.2 DATA FLOW DIAGRAM

A data flow diagram shows how data is processed within a system based on inputs and outputs. Visual symbols are used to represent the flow of information, data sources and destinations, and where data is stored. Data flow diagrams are often used as a first step toward redesigning a system. They provide a graphical representation of a system at any level of detail, creating an easy-to-understand picture of what the system does. A general overview of a system is represented with a context diagram, also known as a level 0 DFD, which shows a system as a single process. A level 1 diagram provides greater detail, focusing on a system's main functions. Diagrams that are level 2 or higher illustrate a system's functioning with increasing detail. It's rare for a DFD to go beyond level 2 because of the increasing complexity, which makes it less effective as a communication tool.

SYMBOL	NAME	FUNCTION
	External entity	A source of system inputs or sink of system outputs
	Data flow	Used to connect processes to each other to sources or sinks, to arrow head indicates direction of data flow
	Process	Perform some transformation of input data to yield output data.
	Data base	A repository of data; the arrow heads indicate net inputs and net outputs to store

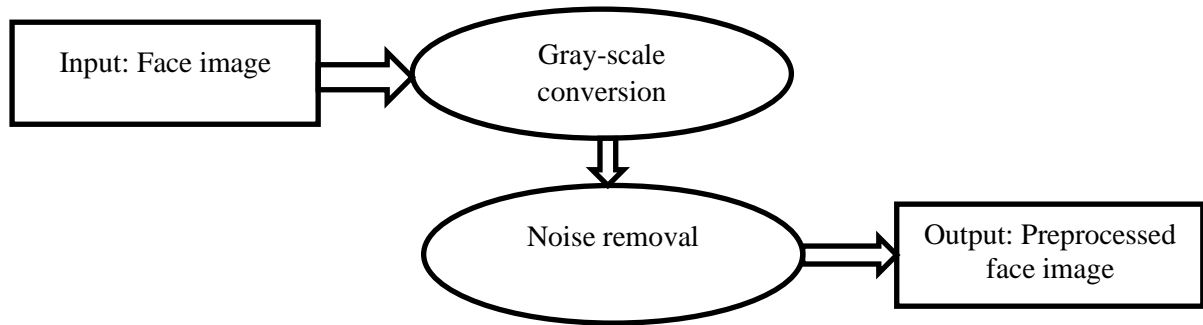
## LEVEL 0

The Level 0 DFD shows how the system is divided into 'sub-systems' (processes), each of which deals with one or more of the data flows to or from an external agent, and which together provide all of the functionality of the system as a whole. It also identifies internal data stores that must be present in order for the system to do its job, and shows the flow of data between the various parts of the system.



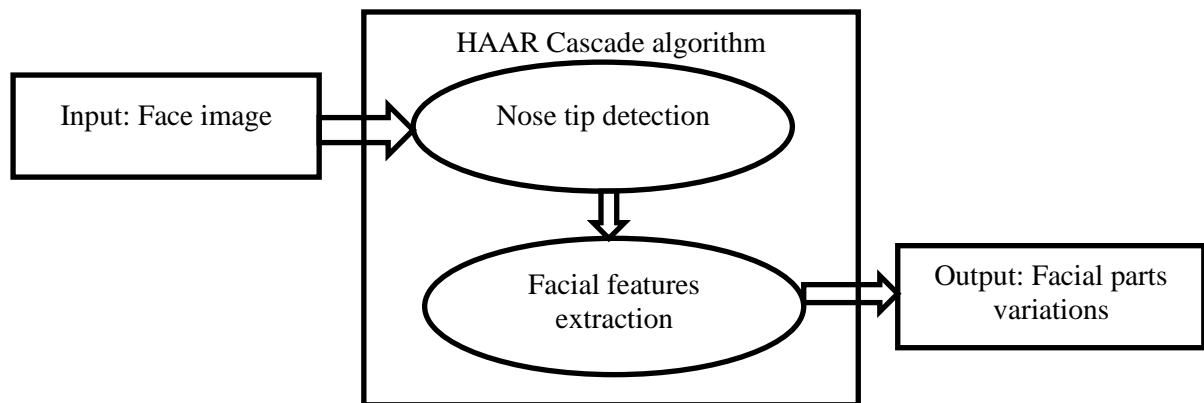
## LEVEL 1

The next stage is to create the Level 1 Data Flow Diagram. This highlights the main functions carried out by the system. As a rule, to describe the system was using between two and seven functions - two being a simple system and seven being a complicated system. This enables us to keep the model manageable on screen or paper.



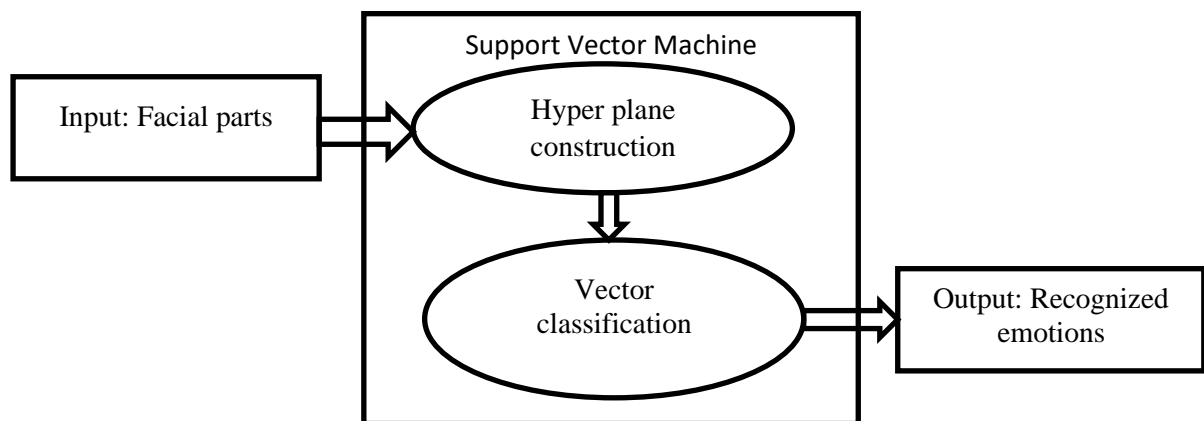
## LEVEL-2

A Data Flow Diagram (DFD) tracks processes and their data paths within the business or system boundary under investigation. A DFD defines each domain boundary and illustrates the logical movement and transformation of data within the defined boundary. The diagram shows 'what' input data enters the domain, 'what' logical processes the domain applies to that data, and 'what' output data leaves the domain. Essentially, a DFD is a tool for process modeling and one of the oldest.



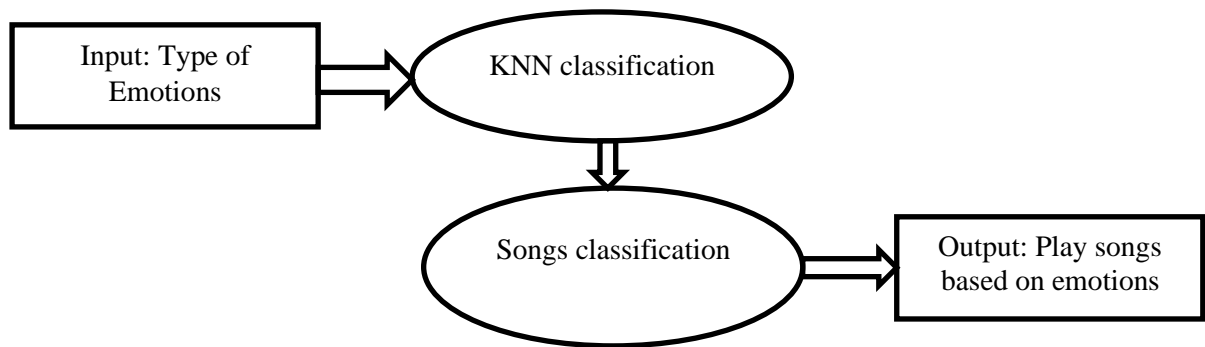
### LEVEL 3

A data flow diagram (DFD) is a graphical representation of the flow of data through an information system. A DFD shows the flow of data from data sources and data stores to processes, and from processes to data stores and data sinks. DFDs are used for modelling and analyzing the flow of data in data processing systems, and are usually accompanied by a data dictionary, an entity-relationship model, and a number of process descriptions.



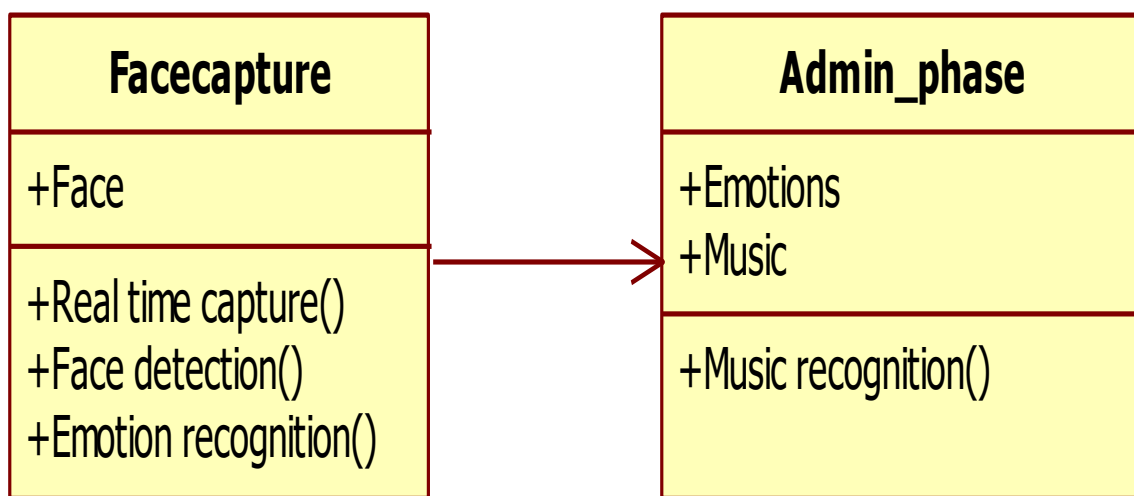
#### LEVEL 4

A DFD may look similar to a flow chart. However, there is a significant difference with the data flow diagram. The arrows in DFDs show that there is a flow of data between the two components and not that the component is sending the data that must be executed in the following component. A component in DFD may not continue execution when sending data and during execution of the component receiving the data. The component sending data can send multiple sets of data along several connections. In fact, a DFD node can be a component that never ends.



### 4.3 CLASS DIAGRAM

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations and the relationships among objects. The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code.





## CHAPTER 5

### 5. MODULES

#### 5.1 MODULES

- Facial Image Acquisition
- Preprocessing
- Facial Features extraction
- Emotion classification
- Music classification

#### 5.2 MODULE DESCRIPTION

##### **Facial Image Acquisition**

In this module, capture the face image or upload the datasets. The uploaded datasets contains 2D face images. In face registration can identify the faces which are captured by web camera. Then web camera images known as 2D images. Admin can be train the face images with multiple emotions. And also train the music player based on languages.

##### **Preprocessing:**

In this module, perform the preprocessing steps such as gray scale conversion, invert, and border analysis, detect edges and region identification. The Grayscale images are also called monochromatic, denoting the presence of only one (mono) color (chrome). The edge detection is used to analyze the connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation.

##### **Facial Features extraction**

In this module implement HAAR cascades which are an algorithm employed the computer technology that determines the locations and sizes of human faces in arbitrary (digital) images. It detects facial features and ignores anything else, such as buildings, trees and bodies. Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one).

## **Emotion classification**

In this module analyze on the expression recognition for testing facial images. For a testing facial image, we first extract the facial features and then perform the questionnaire estimation, where SVM classifier is used for this purpose. After obtaining the question results, we synthesize facial feature vectors based on testing facial feature vector and use them as the model predictors of the positive model. Finally, the model response corresponding to the expression class label vector is calculated and the expression category of the testing facial image can be obtained based on it.

## **Musical classification**

If the emotion is positive means, play happiest songs which are stored in database. Using KNN algorithm to classify the music based on emotions classified by previous modules. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification. The k-NN algorithm is among the simplest of all machine learning algorithms. Both for classification and regression, it can be useful to assign weight to the contributions of the neighbors, so that the nearer neighbors contribute more to the average than the more distant ones. For example, a common weighting scheme consists in giving each neighbor a weight of  $1/d$ , where  $d$  is the distance to the neighbor. The neighbors are taken from a set of objects for which the class (for k-NN classification) or the object property value (for k-NN regression) is known. This can be thought of as the training set for the algorithm, though no explicit training step is required. Based on neighborhood values, music are classified and played in emotional database.

## CHAPTER 6

### 6. APPENDICES

#### 6.1 SOURCE CODE

```
from flask import Flask, render_template, request, session, flash
import sys
import mysql.connector
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics.pairwise import cosine_similarity
```

```
app = Flask(__name__)
app.config['SECRET_KEY'] = 'aaa'
```

```
@app.route('/')
def home():
    return render_template('index.html')
```

```
@app.route('/AdminLogin')
def AdminLogin():
    return render_template('AdminLogin.html')
```

```
@app.route('/NewFaculty')
def NewFaculty():
    return render_template('NewFaculty.html')
```

```
@app.route('/FacultyLogin')
def FacultyLogin():
    return render_template('FacultyLogin.html')
```

```

@app.route("/adminlogin", methods=['GET', 'POST'])
defadminlogin():
    error = None
    if request.method == 'POST':
        if request.form['uname'] == 'admin' and request.form['password'] == 'admin':

            conn = mysql.connector.connect(user='root', password="", host='localhost',
            database='l1stressdb')
            cur = conn.cursor()
            cur.execute("SELECT * FROM regtb")
            data = cur.fetchall()

            return render_template('AdminHome.html', data=data)

        else:
            return render_template('index.html', error=error)

```

```

@app.route("/AdminHome")
defAdminHome():
    conn = mysql.connector.connect(user='root', password="", host='localhost',
    database='l1stressdb')
    cur = conn.cursor()
    cur.execute("SELECT * FROM regtb")
    data = cur.fetchall()
    return render_template('AdminHome.html', data=data)

```

```

@app.route("/Report")
defReport():
    conn = mysql.connector.connect(user='root', password="", host='localhost',
    database='l1stressdb')

```

```

cur = conn.cursor()
cur.execute("SELECT * FROM reporttb")
data = cur.fetchall()
return render_template('Report.html', data=data)

```

```

@app.route("/newfac", methods=['GET', 'POST'])
defnewfac():
    if request.method == 'POST':
        name = request.form['uname']
        mobile = request.form['mobile']
        email = request.form['email']
        username = request.form['username']
        password = request.form['password']

```

```

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='l1stressdb')
cursor = conn.cursor()
cursor.execute(
    "insert into regtb values('" + name + "','" + mobile + "','" + email + "','" + username + "','" +
    password + "')"
conn.commit()
conn.close()
flash("Record Saved!")
return render_template('FacultyLogin.html')

```

```

@app.route("/facultylogin", methods=['GET', 'POST'])
deffacultylogin():
    if request.method == 'POST':
        username = request.form['uname']
        password = request.form['password']
        session['fname'] = request.form['uname']

```

```

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='lstressdb')
cursor = conn.cursor()
cursor.execute("SELECT * from regtb where username='" + username + "' and password='"
+ password + "'")
data = cursor.fetchone()
if data is None:

flash("UserName Or Password Incorrect!")
return render_template('FacultyLogin.html')
else:
emotion()
conn = mysql.connector.connect(user='root', password="", host='localhost',
database='lstressdb')
cur = conn.cursor()
cur.execute("SELECT * FROM regtb where username='" + username + "' and password='" +
password + "'")
data = cur.fetchall()
return render_template('FacultyHome.html', data=data)

```

```

defemotion():
import csv
import copy
import itertools

import cv2 as cv
import numpyas np
import mediapipeas mp
from model import KeyPointClassifier
import os

os.environ['OPENCV_VIDEOIO_PRIORITY_MSMF'] = '0'

```

```

defcalc_landmark_list(image, landmarks):
    image_width, image_height = image.shape[1], image.shape[0]

    landmark_point = []

    # Keypoint
    for _, landmark in enumerate(landmarks.landmark):
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
        landmark_y = min(int(landmark.y * image_height), image_height - 1)

        landmark_point.append([landmark_x, landmark_y])

    return landmark_point

defpre_process_landmark(landmark_list):
    temp_landmark_list = copy.deepcopy(landmark_list)

    # Convert to relative coordinates
    base_x, base_y = 0, 0
    for index, landmark_point in enumerate(temp_landmark_list):
        if index == 0:
            base_x, base_y = landmark_point[0], landmark_point[1]

    temp_landmark_list[index][0] = temp_landmark_list[index][0] - base_x
    temp_landmark_list[index][1] = temp_landmark_list[index][1] - base_y

    # Convert to a one-dimensional list
    temp_landmark_list = list(
        itertools.chain.from_iterable(temp_landmark_list))

    # Normalization
    max_value = max(list(map(abs, temp_landmark_list)))

    defnormalize_(n):

```

```
return n / max_value
```

```
temp_landmark_list = list(map(normalize_, temp_landmark_list))
```

```
return temp_landmark_list
```

```
def draw_bounding_rect(use_brect, image, brect):
```

```
    if use_brect:
```

```
        # Outer rectangle
```

```
        cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[3]),  
                      (0, 0, 0), 1)
```

```
    return image
```

```
def calc_bounding_rect(image, landmarks):
```

```
    image_width, image_height = image.shape[1], image.shape[0]
```

```
    landmark_array = np.empty((0, 2), int)
```

```
    for _, landmark in enumerate(landmarks.landmark):
```

```
        landmark_x = min(int(landmark.x * image_width), image_width - 1)
```

```
        landmark_y = min(int(landmark.y * image_height), image_height - 1)
```

```
        landmark_point = [np.array((landmark_x, landmark_y))]
```

```
    landmark_array = np.append(landmark_array, landmark_point, axis=0)
```

```
    x, y, w, h = cv.boundingRect(landmark_array)
```

```
    return [x, y, x + w, y + h]
```

```
def draw_info_text(image, brect, facial_text):
```

```
    cv.rectangle(image, (brect[0], brect[1]), (brect[2], brect[1] - 22),  
                  (0, 0, 0), -1)
```



```

if facial_text != "":
    info_text = 'Emotion :' + facial_text
    cv.putText(image, info_text, (brect[0] + 5, brect[1] - 4),
               cv.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 255), 1, cv.LINE_AA)

return image

cap_device = 0
cap_width = 1920
cap_height = 1080

use_brect = True

# Camera preparation
cap = cv.VideoCapture(cap_device)
cap.set(cv.CAP_PROP_FRAME_WIDTH, cap_width)
cap.set(cv.CAP_PROP_FRAME_HEIGHT, cap_height)

# Model load
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(
    max_num_faces=1,
    refine_landmarks=True,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5)

keypoint_classifier = KeyPointClassifier()

# Read labels
with open('model/keypoint_classifier/keypoint_classifier_label.csv',
          encoding='utf-8-sig') as f:
    keypoint_classifier_labels = csv.reader(f)
    keypoint_classifier_labels = [

```

```
row[0] for row in keypoint_classifier_labels  
]
```

```
mode = 0
```

```
flag = 0
```

```
flag1 = 0
```

```
c1 = 0
```

```
c2 = 0
```

```
c3 = 0
```

```
c4 = 0
```

```
c5 = 0
```

```
c6 = 0
```

```
while True:
```

```
# Process Key (ESC: end)
```

```
key = cv.waitKey(10)
```

```
if key == 27: # ESC
```

```
break
```

```
# Camera capture
```

```
ret, image = cap.read()
```

```
if not ret:
```

```
break
```

```
image = cv.flip(image, 1) # Mirror display
```

```
debug_image = copy.deepcopy(image)
```

```
# Detection implementation
```

```
image = cv.cvtColor(image, cv.COLOR_BGR2RGB)
```

```
image.flags.writeable = False
```

```
results = face_mesh.process(image)
```

```
image.flags.writeable = True
```

```

if results.multi_face_landmarks is not None:
    for face_landmarks in results.multi_face_landmarks:
        # Bounding box calculation
        brect = calc_bounding_rect(debug_image, face_landmarks)

        # Landmark calculation
        landmark_list = calc_landmark_list(debug_image, face_landmarks)

        # Conversion to relative coordinates / normalized coordinates
        pre_processed_landmark_list = pre_process_landmark(
            landmark_list)

        # emotion classification
        facial_emotion_id = keypoint_classifier(pre_processed_landmark_list)
        # Drawing part
        debug_image = draw_bounding_rect(use_brect, debug_image, brect)
        debug_image = draw_info_text(
            debug_image,
            brect,
            keypoint_classifier_labels[facial_emotion_id])

        emo = keypoint_classifier_labels[facial_emotion_id]
        print(emo)
        if emo == "Angry":
            c1 = c1 + 1
            if c1 == 200:
                import datetime
                import time
                name = session['fname']
                date = datetime.datetime.now().strftime('%Y-%m-%d')
                ts = time.time()
                date = datetime.datetime.now().strftime('%d-%b-%Y')
                timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

```

```

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='lstressdb')
cursor = conn.cursor()
cursor.execute(
"insert into reporttb values(", " + name + ", " + date + ", " + timeStamp + ", " + emo + ")")
conn.commit()
conn.close()

```

```

cv.waitKey(2)
cap.release()
cv.destroyAllWindows()
for i in range(1, 5):
cv.waitKey(1)
session['emo'] = emo
flash("You Are " + str(emo))
return render_template('FacultyHome.html', emo=session['emo'])

```

```

if emo == "Happy":
c2 = c2 + 1
if c2 == 200:
import datetime
import time
name = session['fname']
date = datetime.datetime.now().strftime('%Y-%m-%d')
ts = time.time()
date = datetime.datetime.now().strftime('%d-%b-%Y')
timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

```

```

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='lstressdb')
cursor = conn.cursor()
cursor.execute(
"insert into reporttb values(", " + name + ", " + date + ", " + timeStamp + ", " + emo + ")")
conn.commit()

```

```

conn.close()
cv.waitKey(2)
cap.release()
cv.destroyAllWindows()
for i in range(1, 5):
cv.waitKey(1)
session['emo'] = emo
flash("You Are " + str(emo))
return render_template('FacultyHome.html', emo=session['emo'])

```

```

if emo == "Neutral":
c3 = c3 + 1
if c3 == 200:
import datetime
import time
name = session['fname']
date = datetime.datetime.now().strftime('%Y-%m-%d')
ts = time.time()
date = datetime.datetime.now().strftime('%d-%b-%Y')
timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

```

```

conn = mysql.connector.connect(user='root', password="", host='localhost',
database='l1stressdb')
cursor = conn.cursor()
cursor.execute(
"insert into reporttb values('" + name + "','" + date + "','" + timeStamp + "','" + emo + "')"
conn.commit()
conn.close()

```

```

cv.waitKey(2)
cap.release()
cv.destroyAllWindows()
for i in range(1, 5):
cv.waitKey(1)

```

```

session['emo'] = emo
flash("You Are " + str(emo))
return render_template('FacultyHome.html', emo=session['emo'])

if emo == "Sad":
    c4 = c4 + 1
    if c4 == 200:
        import datetime
        import time
        name = session['fname']
        date = datetime.datetime.now().strftime('%Y-%m-%d')
        ts = time.time()
        date = datetime.datetime.now().strftime('%d-%b-%Y')
        timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

        conn = mysql.connector.connect(user='root', password='', host='localhost',
        database='l1stressdb')
        cursor = conn.cursor()
        cursor.execute(
            "insert into reporttb values('" + name + "','" + date + "','" + timeStamp + "','" + emo + "')"
        )
        conn.commit()
        conn.close()

    cv.waitKey(2)
    cap.release()
    cv.destroyAllWindows()
    for i in range(1, 5):
        cv.waitKey(1)
        session['emo'] = emo
        flash("You Are " + str(emo))
        return render_template('FacultyHome.html', emo=session['emo'])

if emo == "Surprise":
    c5 = c5 + 1

```

```

if c5 == 200:
import datetime
import time
name = session['fname']
date = datetime.datetime.now().strftime('%Y-%m-%d')
ts = time.time()
date = datetime.datetime.now().strftime('%d-%b-%Y')
timeStamp = datetime.datetime.fromtimestamp(ts).strftime('%H:%M:%S')

conn = mysql.connector.connect(user='root', password='', host='localhost',
database='lstressdb')
cursor = conn.cursor()
cursor.execute(
"insert into reporttb values('" + name + "','" + date + "','" + timeStamp + "','" + emo + "')"
conn.commit()
conn.close()

cv.waitKey(2)
cap.release()
cv.destroyAllWindows()
for i in range(1, 5):
cv.waitKey(1)
session['emo'] = emo
flash("You Are " + str(emo))
return render_template('FacultyHome.html', emo=session['emo'])

# Screen reflection
cv.imshow('Facial Emotion Recognition', debug_image)

cap.release()
cv.destroyAllWindows()

@app.route("/playsong", methods=['GET', 'POST'])

```

```
def playsong():
    if request.method == 'POST':
        emo = session['emo']
        print(emo)

    if emo == "Angry":
        import AngryMusic

    if emo == "Happy":
        import HappyMusic

    if emo == "Neutral":
        import NeutralMusic

    if emo == "Sad":
        import SadMusic

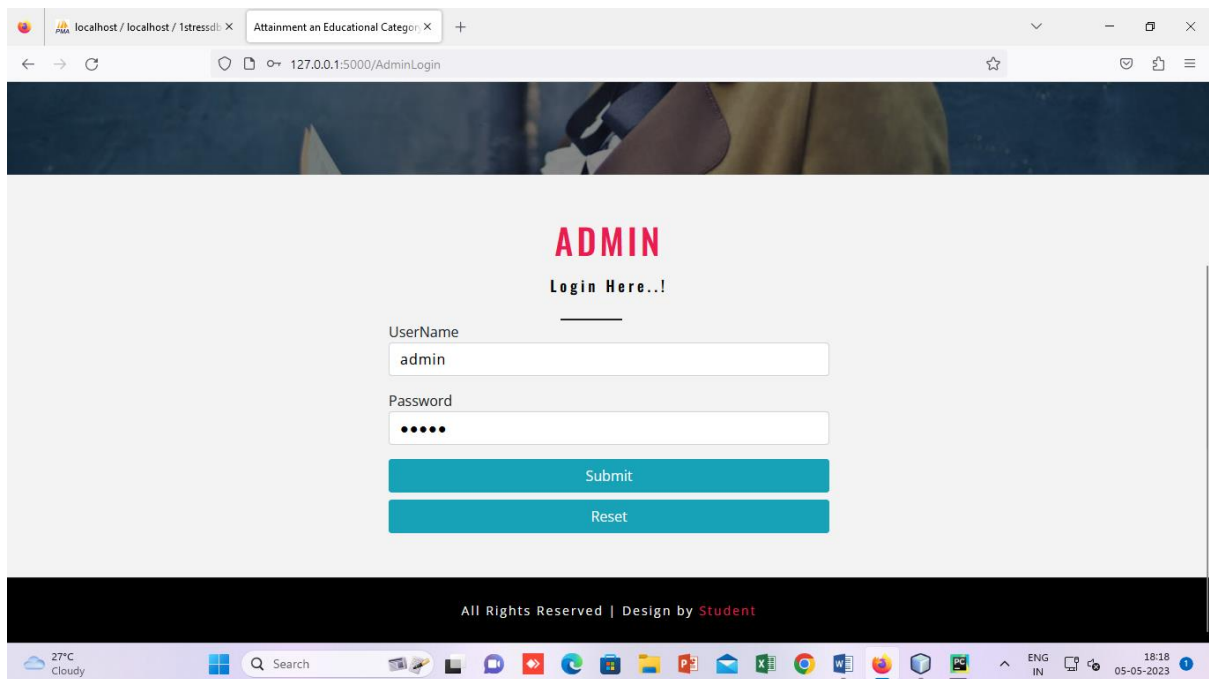
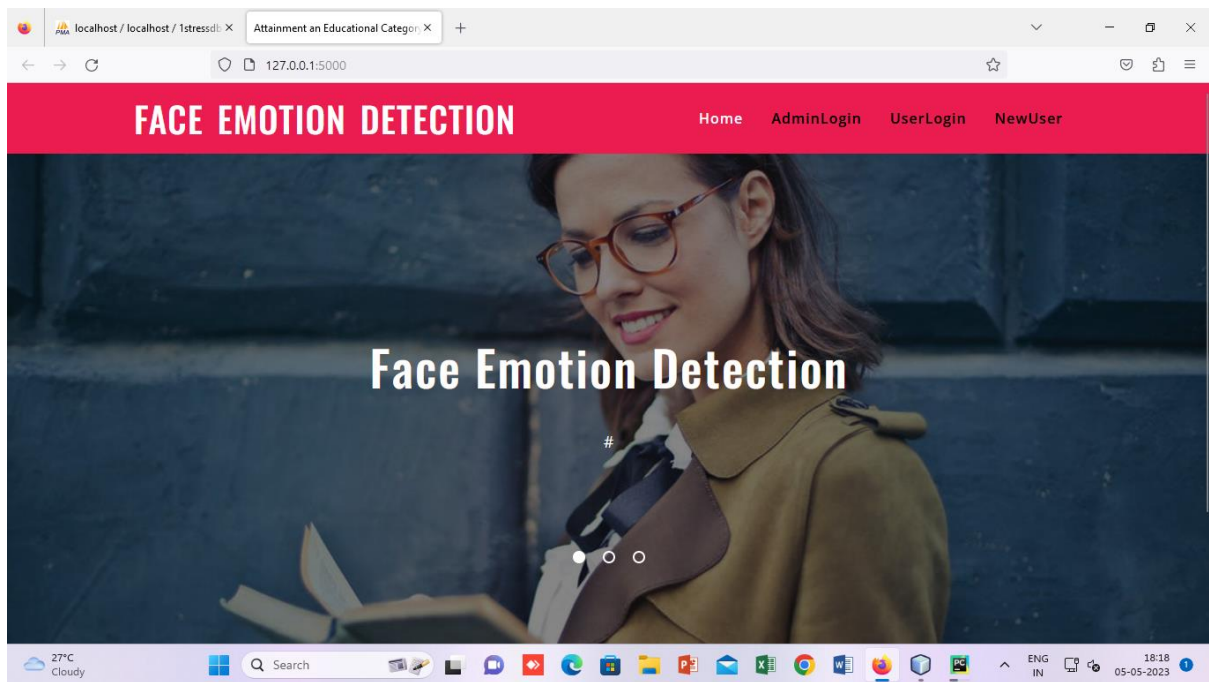
    if emo == "Surprise":
        import SurpriseMusic

    return render_template('FacultyHome.html', emo=session['emo'])

if __name__ == '__main__':
    app.run(debug=True, use_reloader=True)
```




## 6.2 SCREENSHOTS



localhost / localhost / 1stresdli: X Attainment an Educational Categor X +

127.0.0.1:5000/adminlogin



## USER


Information

Name	Mobile	Emailid	UserName
sangeeth Kumar	9486365535	sangeeth5535@gmail.com	san
akash	9486365535	sangeeth5535@gmail.com	akash
ahamad	9486365535	sangeeth5535@gmail.com	ahamad
narasimha	9486365535	narasimha@gmail.com	narasimha

27°C Cloudy Search 18:19 05-05-2023

localhost / localhost / 1stresdli: X Attainment an Educational Categor X +

127.0.0.1:5000/Report



## EMOTION

Information

Name	Date	Time	Emotion
narasimha	02-Apr-2023	14:06:48	Neutral
narasimha	02-Apr-2023	14:08:08	Angry
san	02-Apr-2023	14:15:08	Neutral

All Rights Reserved | Design by Student

27°C Cloudy Search 18:19 05-05-2023

localhost / localhost / 1stresdli: X Attainment an Educational Categor: X

127.0.0.1:5000/NewFaculty

## NEW FACULTY

Register Here..!

Name  
siddiq

Mobile  
7395889223

Email  
asarsiddiq08@gmail.com

Username  
siddiq

Password  
...

Submit

Reset

27°C Cloudy Search 18:20 05-05-2023

localhost / localhost / 1stresdli: X Attainment an Educational Categor: X

127.0.0.1:5000/newfac

## USER

Login Here..!

UserName  
siddiq

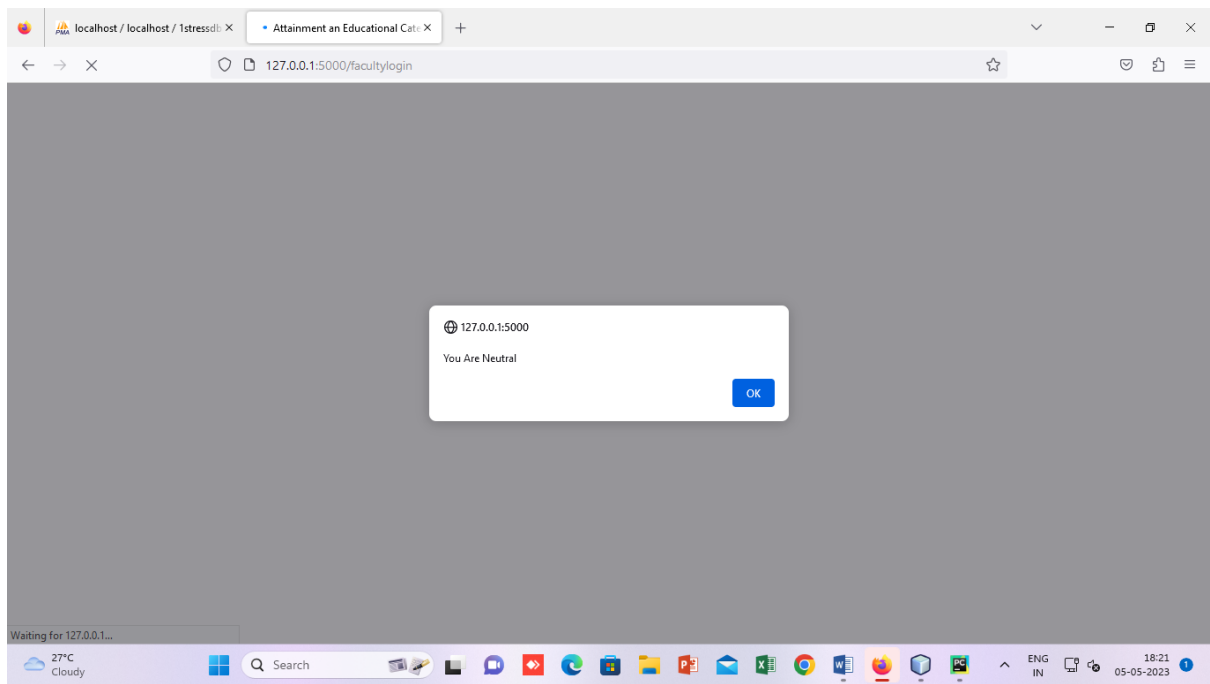
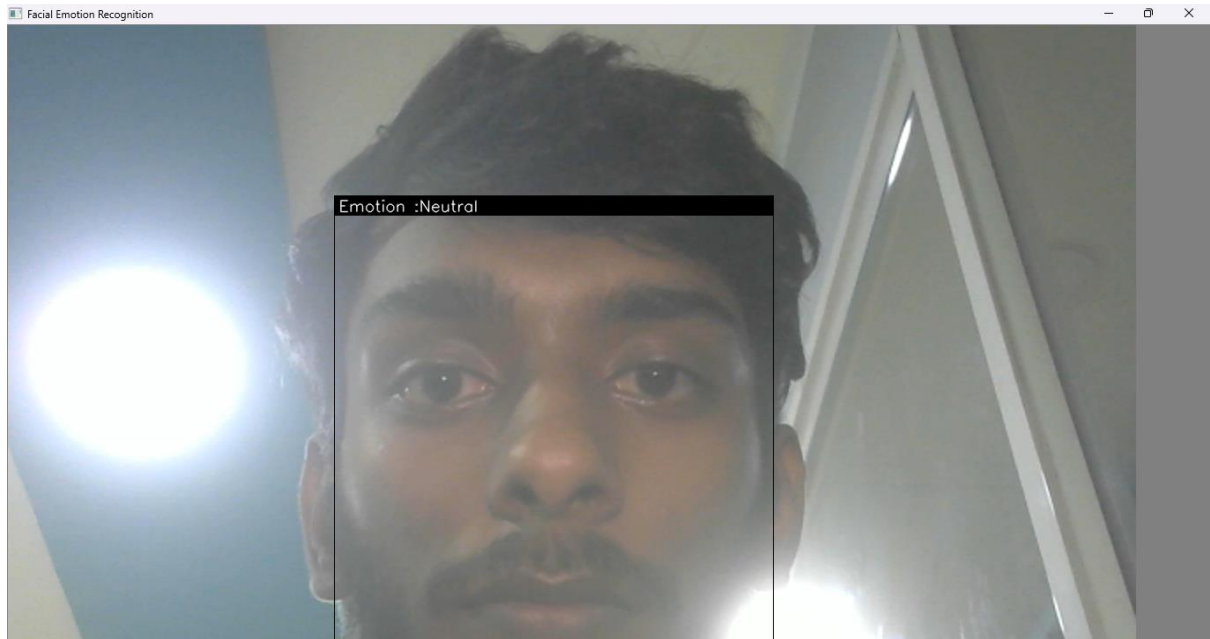
Password  
...

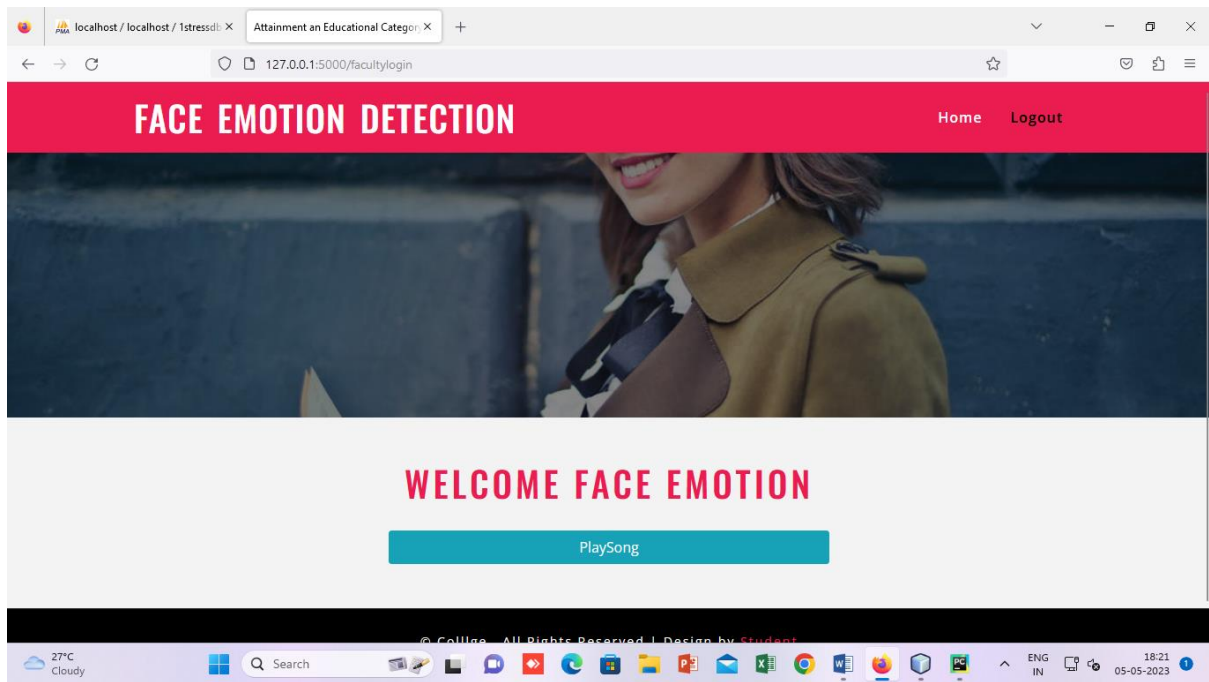
Submit

Reset

©All Rights Reserved | Design by Student

27°C Cloudy Search 18:20 05-05-2023





## **CHAPTER 7**

### **7. CONCLUSION AND REFERENCES**

#### **7.1 CONCLUSION**

In this project proposed support vector machine which contains the set of rules for emotion popularity. Considering an sensitive face as a superposition of a unbiased face with expression thing, we proposed an set of rules to decompose an expressive test face into its constructing additives. For this motive, we first generate grids for captured face the usage of HAAR Cascade algorithm. Knowing that the face component of the take a look at face has sparse representation inside the face database and the expression element can be carefully represented using the expression database; we decompose the test face into those characteristic vectors. The factors of the check face at the side of the vectors are then used for face and expression reputation. For this resolution, the detached components are carefully decomposed the use of vectors even as the grouping systems of the vectors are enforced into the sparse decomposition. The experimental results on both databases confirmed that the proposed technique achieves aggressive recognition overall performance compared with the state of the art methods below same experimental settings and equal facial function.

## **7.2 REFERENCES**

1. Heinold, Brian. "A practical introduction to Python programming." (2021).
2. Kneusel, Ronald T. Practical deep learning: A Python-based introduction. No Starch Press, 2021.
3. Dhruv, Akshit J., Reema Patel, and NishantDoshi. "Python: the most advanced programming language for computer science applications." Science and Technology Publications, Lda (2021): 292-299.
4. Sundnes, Joakim. Introduction to scientific programming with Python. Springer Nature, 2020.
5. Hill, Christian. Learning scientific programming with Python. Cambridge University Press, 2020.

## **WEBSITE REFERENCES**

1. <https://medium.com/javarevisited/10-free-python-tutorials-and-courses-from-google-microsoft-and-coursera-for-beginners-96b9ad20b4e6>
2. <https://www.bestcolleges.com/bootcamps/guides/learn-python-free/>
3. <https://www.programiz.com/python-programming>
4. <https://realpython.com/>
5. <https://www.codecademy.com/learn/learn-python>