CONTENTS

SYNOPSIS

The Face Emotions Based Music Player is an artificial intelligence-based music player that uses facial recognition technology to detect human emotions and match the user's current emotional state with an appropriate song. By analyzing the user's facial expressions, the AI-based Music Player can interpret their emotional state and then provide a personalized list of tracks tailored to the individual’s emotion.

This technology can help people to identify and even manage their own emotional states by using music as a therapeutic aid. It has the potential to enhance user experience and could give music streaming services a competitive edge in the digital music industry.

1. INTRODUCTION

Introducing a face emotion-based music player! This innovative music player uses facial recognition technology to detect the user's emotions and plays music that matches their mood.

The concept is simple: the music player uses the camera on the user's device to capture their facial expressions and then analyzes the data to determine their emotional state. Based on this information, the music player selects and plays songs that match the user's current mood.

For example, if the user is feeling sad, the music player might play slow, mellow songs with sad lyrics. If the user is feeling happy, the music player might play upbeat, cheerful songs that match their mood.

This music player is perfect for those who want to enhance their mood with music but don't know what to listen to or don't have the time to create a playlist. It's also great for parties or gatherings, where the music can be customized to match the mood of the crowd.

With the face emotion-based music player, users can enjoy a personalized music experience that is tailored to their emotions.

1. 1 SYSTEM SPECIFICATION

1.1.1 Hardware Specification

CPU type	: 5 Compute cores 2C + 3G
Hard Disk	: 500 GB
Memory	: 4 GB RAM
Speaker	: System Speaker
Camera	: System Camera

1.1.2 Software Specification

Operating System	: Windows 10 pro
Programming Languages	: Python, HTML and JavaScript
Front End	: Python, HTML and CSS
Back End	: Python and JavaScript

2. SYSTEM STUDY

2. 1 EXISTING SYSTEM

2.1.1 DESCRIPTION

Currently, there are many existing music player applications. Some of the interesting applications among them are:

- **Spotify :** These application gives good user accessibility features to play songs and recommends user with other songs of similar genre.
- **Moodfuse :** In this application , user should manually enter mood and genre that wants to be heard and moodfuse recommends the songs-list.
- **YouTube Music :** YouTube Music is a music streaming platform offered by YouTube that allows users to listen to music through a desktop or mobile app. It has a large library of songs, albums, and playlists, and also offers personalized recommendations based on a user's listening history. YouTube Music also offers music videos and live performances in addition to audio-only playback.

2.1.2 DISADVANTAGES

- **Spotify:** although it offers personalized playlists and recommendations, its algorithm may not always accurately capture a user's current mood. Additionally, its algorithm can be biased towards popular or mainstream music, which may not always be the best fit for a user's desired emotional state.

- **Moodfuse:** on the other hand, requires manual input from the user, which can be time-consuming and may not always accurately capture a user's current emotional state. It also has a smaller music library compared to Spotify and YouTube Music, which may limit the variety of music choices available to users.

- **YouTube Music:** offers personalized recommendations based on a user's listening history, but its recommendations may not always accurately capture a user's current emotional state. Additionally, some users may prefer to listen to music without music videos or live performances, which are often prominently featured on the platform.

2. 2 PROPOSED SYSTEM

2.2.1 DESCRIPTION

The primary goal of an emotion-based music player is to provide user-preferred music with emotion identification by automatically playing songs based on the user's feelings. In the current system, the user must manually choose the tunes because songs generated at random may not suit the user's mood. The user must first categorize the songs into several emotion groups before manually choosing a certain emotion. Using an emotion-based music player will help you avoid complications. The music will be played from the predetermined folders in accordance with the emotion detected.

2.2.2 ADVANTAGES

- ✓ **Personalized experience:** By using facial recognition technology to detect the user's emotions and music preferences, the system can create a personalized experience for each user, providing them with a playlist that matches their current mood.

- ✓ **Discovery of new music:** The system can recommend new music that is similar to what the user likes, helping them discover new artists and songs that they may not have found otherwise.

- ✓ **Time-saving**: With the system's ability to create a personalized playlist, users don't have to spend time searching for music that matches their mood or preferences. The system does it for them.

- ✓ **Convenient:** The system is convenient to use as it eliminates the need for manual input of music preferences or searching for music. The user can simply use the facial recognition feature, and the system will create a playlist automatically.

- ✓ **Engagement:** By creating an engaging and personalized experience for each user, the system can keep them engaged and interested in using the app regularly, leading to more frequent use and a higher level of satisfaction.

5. SYSTEM DESIGN AND DEVELOPMENT

3. 1 FILE DESIGN

The music player will use the computer's webcam to capture the user's facial expression in real-time. The OpenCV library will be used to detect the user's facial landmarks and analyze their expression to determine their mood. The program will use a pre-trained machine learning model to classify the user's facial expression into one of several emotions, such as happy, sad, angry, or surprised. The model will be trained using a dataset of labeled facial expressions. Once the user's emotion has been determined, the music player will select and play a song from a playlist that matches their mood. The playlist can be created manually or automatically based on the user's preferences. The program will continuously monitor the user's facial expression and update the music selection as their mood changes.

3. 2 INPUT DESIGN

The interface of emotion based music player consists of Logo and the tagline as **"Let music flow and enrich your soul"**. The interface has all songs aligned according to the emotions designed i.e Happy, Sad, Neutral, and Angry.There are three types of mode described such as Queue mode, Emotion Mode and Random Mode. The emotion detected by the user is Happy and the song is played.

Figure 1

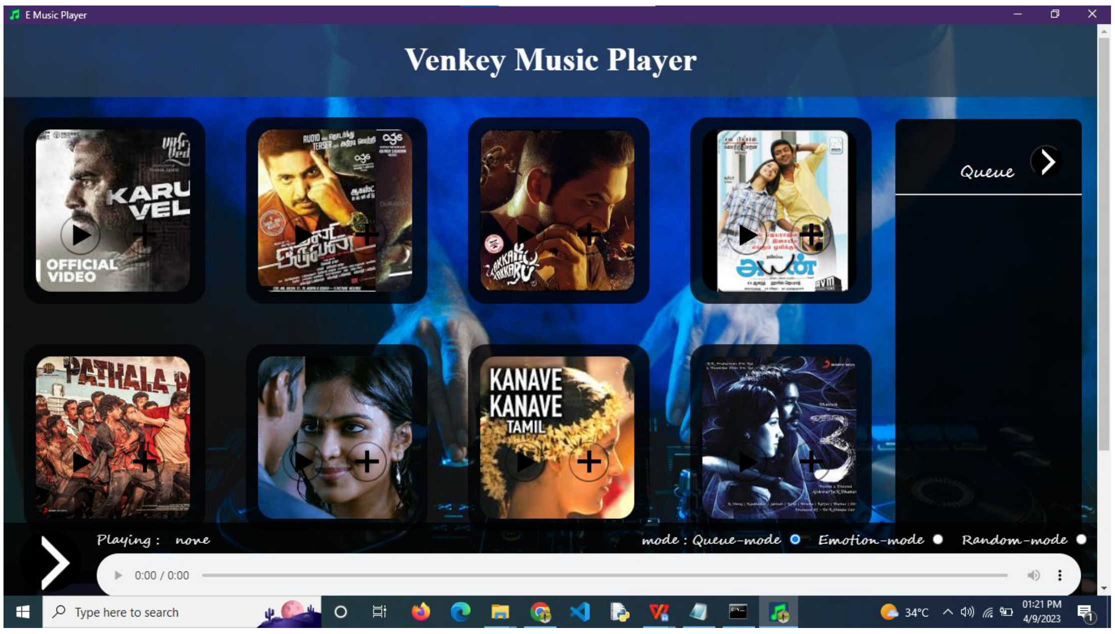
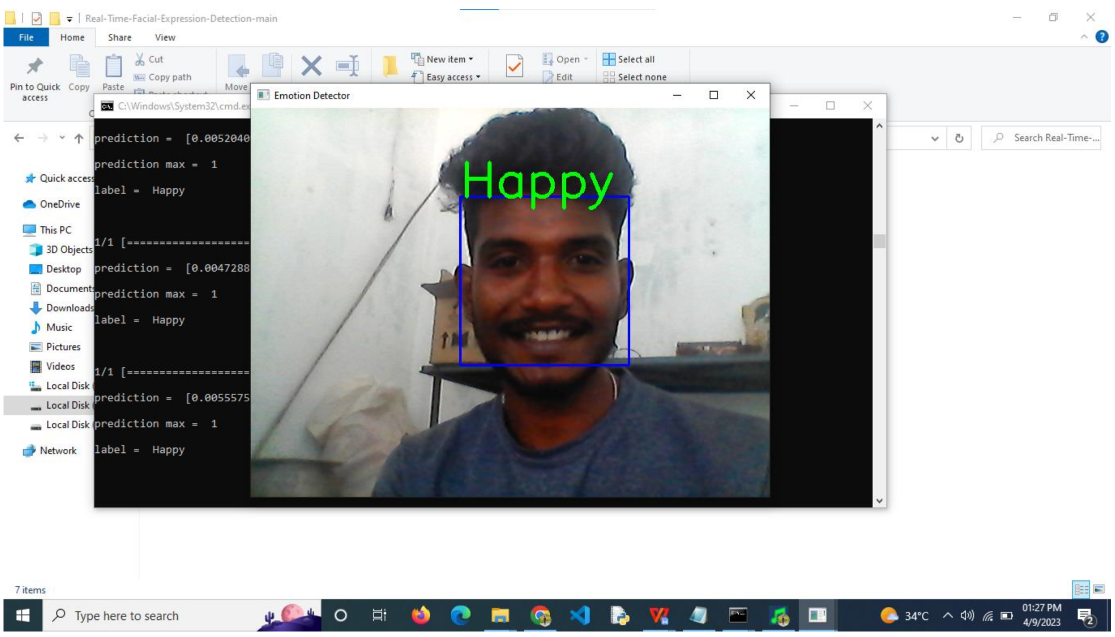


Figure 2

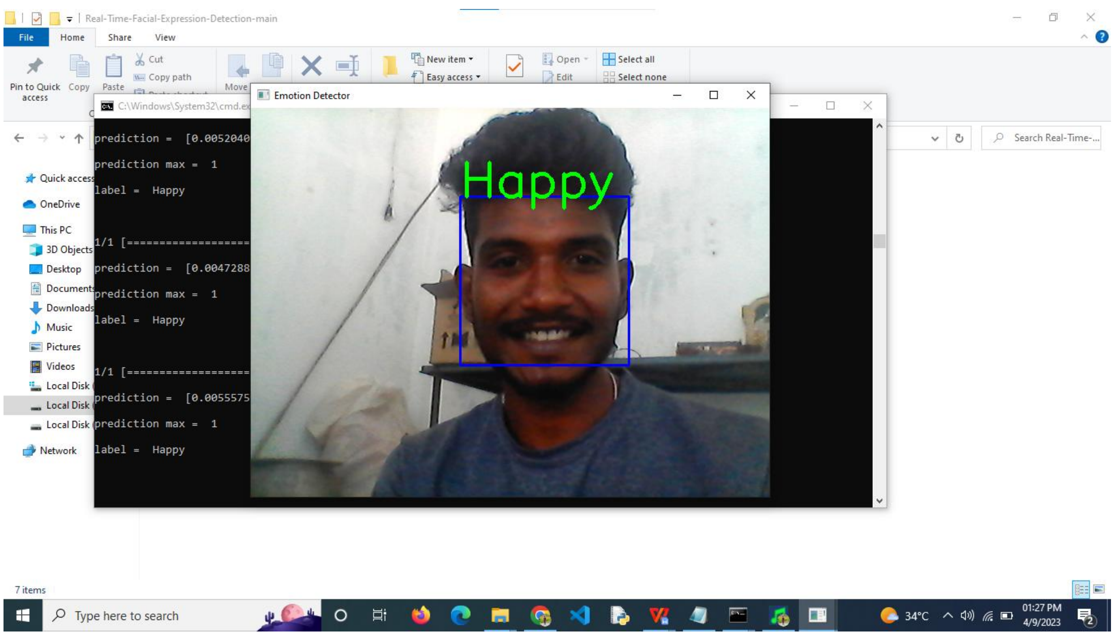

Real-Time-Facial-Expression-Detection-main
C:\Windows\System32\cmd.exe
Emotion Detector
Happy
Type here to search
34°C
01:27 PM
4/9/2023

```
prediction =  [0.0052040
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0047288
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0055575
prediction max =  1
label =  Happy
```

2. 3 OUTPUT DESIGN

After the user clicks the Sequential Mode or Sensory Mode or Random Mode radio button. The function for the button they selected will activate and the song will play. The interface also has four emoji's lined up, when the user places the cursor on the emoji, the song will be played accordingly according to emoji described such as Happy,Sad, Neutral, Angry.

Figure 1 **Emotion Detection**


Real-Time-Facial-Expression-Detection-main
C:\Windows\System32\cmd.exe
Emotion Detector
Happy
Type here to search
34°C
01:27 PM
4/9/2023

```
prediction =  [0.0052040
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0047288
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0055575
prediction max =  1
label =  Happy
```

Figure 2 **Random Mode**

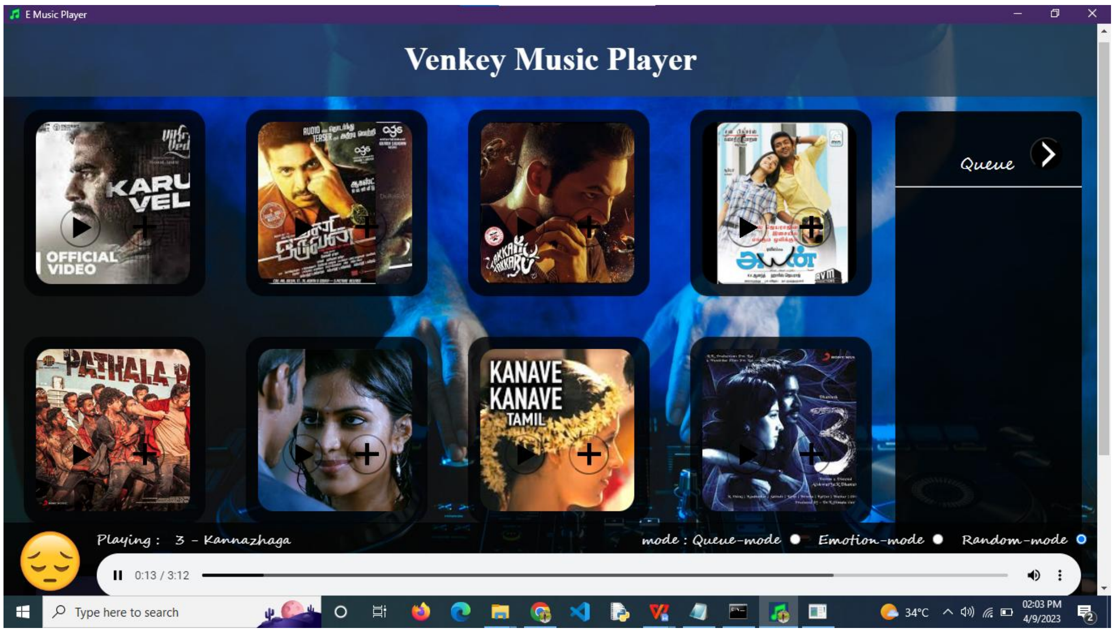
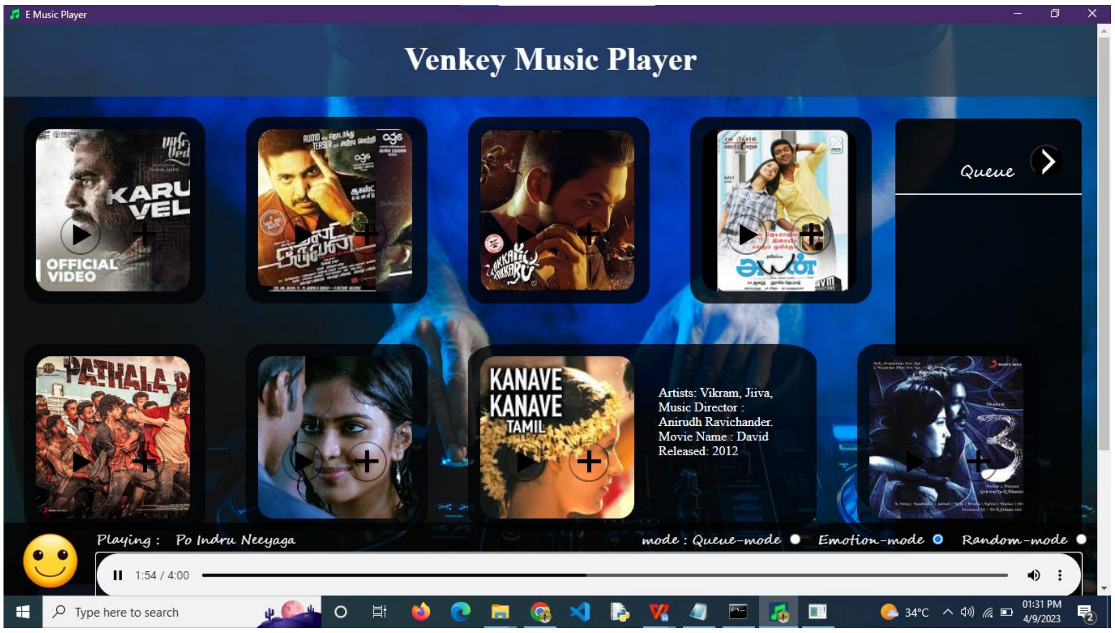
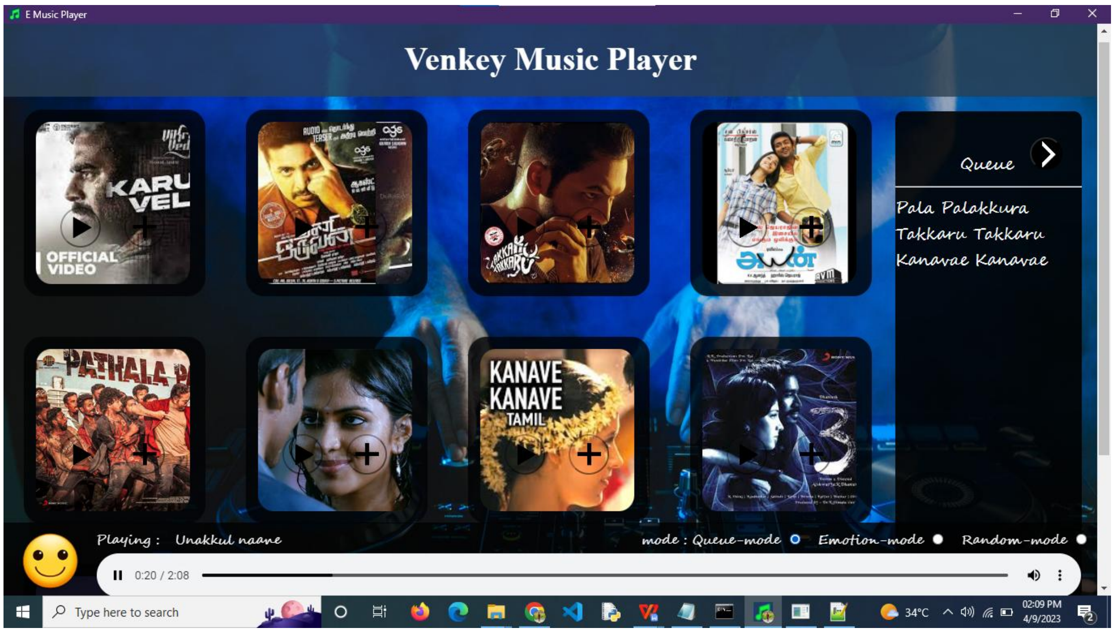


Figure 3 **Emotion Mode**



Artists: Vikram, Jiiva,
Music Director :
Anirudh Ravichander.
Movie Name : David
Released: 2012

Figure 4 **Queue Mood**



3. 4 CODE DESIGN

```
import eel
import argparse
from keras.models import load_model
from time import sleep
from keras_preprocessing.image import img_to_array
from keras_preprocessing import image
import cv2
import numpy as np
from gtts import gTTS
import os
from playsound import playsound
eel.init('WD_INNOVATIVE')

count=0
@eel.expose
def getEmotion():
    face_classifier                                                          =
cv2.CascadeClassifier('./keras/haarcascade_frontalface_default.xml')
    classifier =load_model('./keras/Emotion_Detection.h5')
    class_labels = ['angry','happy','neutral','sad','Surprise']
    cap = cv2.VideoCapture(0)
    while True:
        label=""
        # Grab a single frame of video
        ret, frame = cap.read()
        labels = []
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray,1.3,5)

        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray = gray[y:y+h,x:x+w]
            roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
```

3. 5 SYSTEM DEVELOPMENT

3.5.1 MODULES

A module is a software component or part of a program that contains one or more routines. One or more independently developed modules make up a program. An enterprise-level software application may contain several different modules, and each module serves unique and separate business operations.

Modules make a programmer's job easy by allowing the programmer to focus on only one area of the functionality of the software application. Modules are typically incorporated into the program (software) through interfaces.

➢ This project mainly consists five modules:

- ✓ Face Detection Module
- ✓ Emotion Recognition Module
- ✓ Music recommendation module
- ✓ Music playback module
- ✓ User feedback module

3.5.2 MODULES DESCRIPTION

1. Face detection module

This module would be responsible for detecting the user's face and locating the key facial features necessary for emotion recognition.

2. Emotion recognition module

This module would use machine learning algorithms to analyze the user's facial expressions and determine their emotional state. This could be done using facial landmarks, such as the position of the eyebrows, mouth, and eyes.

3. Music recommendation module

Once the user's emotional state has been determined, this module would suggest music that is likely to fit their mood. This could be done by analyzing the tempo, rhythm, and tone of different songs and selecting the most appropriate ones.

4. Music playback module

This module would be responsible for playing the selected music, adjusting the volume and other playback settings as needed.

5. User feedback module

Finally, this module would allow the user to provide feedback on the music selection and adjust the system's recommendations over time based on their preferences.

4. TESTING

A. UNIT TESTING

Unit Testing is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit tests are typically automated tests written and run by software developers to ensure that a section of an application (known as the "unit") meets its design and behaves as intended. Unit Testing is defined as a type of software testing where individual components of a software are tested.

Unit Testing of the software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit Testing is typically performed by the developer. In SDLC or V Model, Unit testing is the first level of testing done before integration testing. Unit testing is such a type of testing technique that is usually performed by developers. Although due to the reluctance of developers to test, quality assurance engineers also do unit testing.

```
def getEmotion():
    face_classifier=cv2.CascadeClassifier('./keras/haarcascade_frontalface_default.xml')
    classifier =load_model('./keras/Emotion_Detection.h5')

    class_labels = ['angry','happy','neutral','sad','Surprise']

    cap = cv2.VideoCapture(0)
    while True:
        label=""
        # Grab a single frame of video
        ret, frame = cap.read()
        labels = []
        gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        faces = face_classifier.detectMultiScale(gray,1.3,5)
        for (x,y,w,h) in faces:
            cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
            roi_gray = gray[y:y+h,x:x+w]
            roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
            if np.sum([roi_gray])!=0:
                roi = roi_gray.astype('float')/255.0
                roi = img_to_array(roi)
                roi = np.expand_dims(roi,axis=0)

            # make a prediction on the ROI, then lookup the class

                preds = classifier.predict(roi)[0]
                print("\nprediction = ",preds)
                label=class_labels[preds.argmax()]
                print("\nprediction max = ",preds.argmax())
                print("\nlabel = ",label)
                label_position = (x,y)

cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0)
,3)
            else:
                cv2.putText(frame,'NoFace
Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
            print("\n\n")
            if label != "":
                break

        cv2.imshow('Emotion Detector',frame)
        if label != " ":
            break
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
        count=getEmotion()
```

```
    print(label)
    '''t="You Seem To Be"+label
    print(t)
    myobj = gTTS(text=t, lang='en', slow=False)
    myobj.save("s.mp3")
    playsound("s.mp3")
    os.remove("s.mp3")
    return label
```

B. VALIDATION TESTING

Validation testing is the process of ensuring that the tested and developed software satisfies the client /user's needs. The business requirement logic or scenarios have to be tested in detail. All the critical functionalities of an application must be tested here. As a tester, it is always important to know how to verify the business logic or scenarios that are given to you. One such method that helps in detailed evaluation of the functionalities is the Validation Process.

Whenever you are asked to perform a validation test, it takes great responsibility as you need to test all the critical business requirements based on the user's needs. There should not be even a single miss on the requirements asked by the user. Hence a keen knowledge of validation testing is much important.

```
function addq(elem){
        console.log(elem.id);
        var x=elem.id.charAt(1);
        if(!songrun){
                var z=songs[x][0];
                document.getElementById("sname").innerHTML=sname[x];
                document.getElementById("sel").src= z;
                document.getElementById("main_slider").load();
                document.getElementById("main_slider").play();
        document.getElementById("emoji").style.backgroundImage="url('"+songs[x][
3]+"')";
                songrun=true;
                return;
        }
        if(bool[x]==true)
                return;

        bool[x]=true;
        var l=document.createElement("label");
        l.id="e"+eqc;
        l.name=x;
        l.innerHTML=sname[x]+"<br>";
        //var text=document.createTextNode(sname[x]+"<br>");
        //l.appendChild(text);
        document.getElementById("queue").appendChild(l);
        eqc=eqc+1;
}
```

C. INTEGRATION TESTING

Integration testing is a type of software testing in which different components of a software system are tested together as a group to evaluate their interactions and ensure that they work properly when integrated.

Integration testing is performed after unit testing and before system testing. It aims to discover defects in the interfaces and interactions between different modules or subsystems of a software system before the system is tested as a whole.

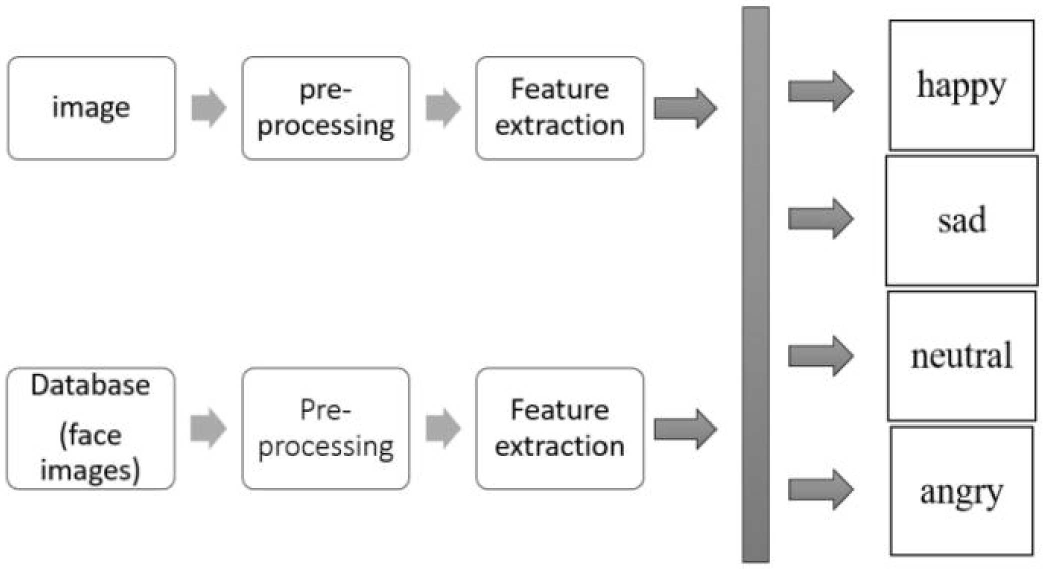
```
@eel.expose
def getEmotion():
```

```
face_classifier=cv2.CascadeClassifier('./keras/haarcascade_frontalface_default.xml')
            classifier =load_model('./keras/Emotion_Detection.h5')

            class_labels = ['angry','happy','neutral','sad','Surprise']

            cap = cv2.VideoCapture(0)
            while True:
               label=""
               # Grab a single frame of video
               ret, frame = cap.read()
               labels = []
               gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
               faces = face_classifier.detectMultiScale(gray,1.3,5)

               for (x,y,w,h) in faces:
                  cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
                  roi_gray = gray[y:y+h,x:x+w]
                  roi_gray =
cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)
                  if np.sum([roi_gray])!=0:
                     roi = roi_gray.astype('float')/255.0
                     roi = img_to_array(roi)
                     roi = np.expand_dims(roi,axis=0)

                  # make a prediction on the ROI, then lookup the class

                     preds = classifier.predict(roi)[0]
                     print("\nprediction = ",preds)
                     label=class_labels[preds.argmax()]
                     print("\nprediction max = ",preds.argmax())
                     print("\nlabel = ",label)
                     label_position = (x,y)
```

```
cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0)
,3)
                    else:
                        cv2.putText(frame,'No Face
Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
                    print("\n\n")
                    if label != "":
                        break

                cv2.imshow('Emotion Detector',frame)
                if label != " ":
                    break
                if cv2.waitKey(1) & 0xFF == ord('q'):
                    break
                count=getEmotion()
            print(label)
        return label
```

5. IMPLEMENTATION

A. ARCHITECTURE DIAGRAM



B. SAMPLE CODE

Main Page:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>E Music Player </title>
<script type="text/javascript" src="/eel.js"></script>
<link rel="stylesheet" type="text/css" href="header.css">
<link rel="stylesheet" type="text/css" href="background.css">
<link rel="stylesheet" type="text/css" href="player.css">
<style>
body{
margin: 0;
}
</style>
<link rel="icon" type="image/png" href="./bg/i.png">
</head>



<body>
<br><br><br><br><br>
<div id="first">
<h1 font-family="verdana" align="center" style="font-size: 40px;color: white;margin:
0;">Venkey Music Player<h1>
</div>
<!--<div id="second" class="second">

<button onclick="myFunction()">
</div>
-->
```

```
<script src="code.js"></script>

<div
id="queue"><br>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;Queue<input type="button" id="next" onclick="nextsong()"><hr></div>

<div id="third">
<div id="emoji"></div>
<div id="xyz">
&nbsp;&nbsp;&nbsp;Playing :&nbsp;&nbsp;
<label id="sname" align="center">none</label></div>
<div id="mod">mode : Queue-mode <input type="radio" name="mode"
checked="checked" onclick="setmod(this)" value="1"> &nbsp;&nbsp;Emotion-mode
<input type="radio" name="mode" onclick="setmod(this)" value="2">
&nbsp;&nbsp;Random-mode
<input type="radio" name="mode" onclick="setmod(this)" value="3"></div>
<audio controls id="main_slider">
<source id="sel"  type="audio/mpeg">
Your browser does not support the audio element.
</audio>

<script>
document.getElementById("main_slider").onended=function(){
if(mod==1)
next_in_Q();
else if(mod==2){
getTime();
}
else
rand_play();

};
```

```
</script>
</div>
</body>
</html>
```

Python File Page:

```
import eel
import argparse
from keras.models import load_model
from time import sleep
from keras_preprocessing.image import img_to_array
from keras_preprocessing import image
import cv2
import numpy as np
from gtts import gTTS
import os
from playsound import playsound
eel.init('WD_INNOVATIVE')

count=0
@eel.expose
def getEmotion():


    face_classifier=cv2.CascadeClassifier('./keras/haarcascade_frontalface_default.xml')
    classifier =load_model('./keras/Emotion_Detection.h5')

    class_labels = ['angry','happy','neutral','sad','Surprise']

    cap = cv2.VideoCapture(0)
```

```
while True:
    label=""
    # Grab a single frame of video
    ret, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray,1.3,5)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(255,0,0),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)



        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi,axis=0)

        # make a prediction on the ROI, then lookup the class

            preds = classifier.predict(roi)[0]
            print("\nprediction = ",preds)
            label=class_labels[preds.argmax()]
            print("\nprediction max = ",preds.argmax())
            print("\nlabel = ",label)
            label_position = (x,y)

cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0)
,3)
        else:
            cv2.putText(frame,'No
Face Found',(20,60),cv2.FONT_HERSHEY_SIMPLEX,2,(0,255,0),3)
        print("\n\n")
        if label != "":
            break

    cv2.imshow('Emotion Detector',frame)
    if label != " ":
        break
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    count=getEmotion()

print(label)
'''t="You Seem To Be"+label
print(t)
myobj = gTTS(text=t, lang='en', slow=False)
myobj.save("s.mp3")
playsound("s.mp3")
os.remove("s.mp3")'''

return label
#print(getEmotion())

eel.start('m.html')#//WD_INNOVATIVE//main.html')
```

Javascript Page:\

```
var songrun=false;
var count=1;
var mod=1;
var path=["songs\\Pala Palakkura.mp3"
,"songs\\Po Indru Neeyaga.mp3"
,"songs\\Takkaru Takkaru.mp3"
,"songs\\Unakkul naane.mp3"
,"songs\\VIKRAM – Pathala Pathala Lyric.mp3"
,"songs\\Vaa Vaathi Lyrical Song.mp3"
,"songs\\Theemai Dhaan Vellum.mp3"
,"songs\\musafir.mp3"
,"songs\\Vikram Vedha Songs _ Karuppu Vellai.mp3"
,"songs\\phir bhi.mp3",
"songs\\Po Indru Neeyaga.mp3"];

var path=["songs\\Vikram Vedha Songs _ Karuppu Vellai.mp3"
,"songs\\Theemai Dhaan Vellum.mp3"
,"songs\\Takkaru Takkaru.mp3"
,"songs\\Pala Palakkura.mp3"
,"songs\\VIKRAM – Pathala Pathala Lyric.mp3"
,"songs\\sad\\yuvan voice songs.mp3"
,"songs\\sad\\Kanavae Kanavae.mp3"
,"songs\\sad\\3 - Kannazhaga.mp3"
,"songs\\Unakkul naane.mp3"
,"songs\\Vaa Vaathi Lyrical Song.mp3",
"songs\\Po Indru Neeyaga.mp3"];

var sname=["Vikram Vedha Songs _ Karuppu Vellai",
"Theemai Dhaan Vellum",
"Takkaru Takkaru",
"Pala Palakkura",
"VIKRAM – Pathala Pathala Lyric",
"yuvan voice songs",
"Kanavae Kanavae",
"3 - Kannazhaga",
"Unakkul naane",
"Vaa Vaathi Lyrical Song",
"Po Indru Neeyaga "
];

var sd=["Artists: Sam C. S., Sivam<br>Movie: Vikram Vedha<br>Released: 2017",
"Artists: Arvind Swamy, Hiphop Tamizha<br>Movie: Thani Oruvan<br>Released:
2015"
,"Artist: Hiphop Tamizha<br>Album: Takkaru Takkaru<br>Released: 2016"
,"Artist: Pala Palakura<br>Movie: Ayan<br>Released: 2009"
,"Artist: VIKRAM – Pathala Pathala Lyric<br>Movie: Vikram<br>Released: 2022"
,"Artist: Yuvan voice songs<br>Movie: Multiple movies collection<br>Year: 90s to
20s"
,"Artists: Vikram, Jiiva, Music Director : Anirudh Ravichander.<br>Movie Name :
David<br>Released: 2012"
,"Artist: Anirudh Ravichander<br>Movie: 3<br>Released: 2011"
,"Artist:    Jyothika    and    R.    Sarath    Kumar<br>Movie:    Pachaikili
Muthucharam<br>Released: 2002"
,"Artists: G. V. Prakash Kumar, Shweta Mohan<br>Movie: Vaathi<br>Released:
2022"
,"Vikram Vedha Songs _ Karuppu Vellai "];

var bool=[];
for(var i=0; i<sd.length; i++)
        bool[i]=false;

var icon=["images\\\\1.jfif",
"images\\\\2.jfif",
"images\\\\3.jpg",
"images\\\\4.jpg",
"images\\\\5.jpg",
"images\\\\6.jpg",
"images\\\\7.jfif",
"images\\\\8.jfif",
"images\\\\9.jfif",
"images\\\\10.jpg",
"images\\\\11.jfif"];

var mood=[["1","2","3"],["4","5"],["6","7","8"],["9","10","11"]];
var
mmm=["1.png","1.png","1.png","2.png","2.png","3.png","3.png","3.png","4.png","4.
png","4.png"];

var songs=new Array(icon.length);
for (var i = 0; i<icon.length; i++) {
        songs[i]=new Array(4);
        songs[i][0]=path[i];
        songs[i][1]=sd[i];
        songs[i][2]=icon[i];
        songs[i][3]=mmm[i];
        console.log(songs[i][0]);
        console.log(songs[i][1]);
        console.log(songs[i][2]);
        var ins=document.createElement("div");
        ins.id='b'+i;
        //ins.onclick=function(){
        //next(this);
        //};
        ins.setAttribute("class", "song");
        document.body.appendChild(ins);
        document.getElementById('b'+i).innerHTML='<div id="pic"
style=\'background-image: url(\"'+songs[i][2]+'\");\'> <input type="button"
id="'+"a"+i+'" class="play" > <input type="button" id="'+"c"+i+'" class="add">
</div><div id="data"><br><br>'+songs[i][1]+'</div>';
        document.getElementById('a'+i).onclick=function(){
                play(this);
        };
        document.getElementById('c'+i).onclick=function(){
                addq(this);
        };
```

```
}




function setmod(elem){
        mod=elem.value;
        if(elem!=""){
                if(mod==2)
                        getTime();
                if(mod==3)
                        rand_play();
        }
}

function play(elem){
        console.log(elem.id);
        var x=elem.id.charAt(1);
        var z=songs[x][0];
        document.getElementById("sname").innerHTML=sname[x];
        document.getElementById("sel").src= z;
        document.getElementById("main_slider").load();
        document.getElementById("main_slider").play();
        document.getElementById("emoji").style.backgroundImage="url('"+songs[x][
3]+"')";
        songrun=true;

}

var eqc=1;
var sqc=1;

function addq(elem){
        console.log(elem.id);
        var x=elem.id.charAt(1);
        if(!songrun){
                var z=songs[x][0];
                document.getElementById("sname").innerHTML=sname[x];
                document.getElementById("sel").src= z;
                document.getElementById("main_slider").load();
                document.getElementById("main_slider").play();

        document.getElementById("emoji").style.backgroundImage="url('"+songs[x][
3]+"')";
                songrun=true;
                return;
        }
        if(bool[x]==true)
                return;

        bool[x]=true;
        var l=document.createElement("label");
        l.id="e"+eqc;
        l.name=x;
        l.innerHTML=sname[x]+"<br>";
        //var text=document.createTextNode(sname[x]+"<br>");
        //l.appendChild(text);
        document.getElementById("queue").appendChild(l);
        eqc=eqc+1;
}


function nextsong(){
        if(sqc==eqc){
                        alert("Queue is empty.");
                        return;
                }
                var elem=document.getElementById("e"+sqc);
```

```
                        var xa=elem.name;
                        var pa=songs[xa][0];
                        bool[xa]=false;
                        document.getElementById("sname").innerHTML=sname[xa];
                        document.getElementById("sel").src= pa;
                        document.getElementById("main_slider").load();
                        document.getElementById("main_slider").play();

        document.getElementById("emoji").style.backgroundImage="url('"+songs[xa]
[3]+"')";

                        songrun=true;
                        document.getElementById("queue").removeChild(elem);
                        sqc=sqc+1;

}

function next_in_Q(){
                        songrun=false;
                        if(sqc==eqc){
                                alert("Queue is empty.");
                                return;
                        }
                        var elem=document.getElementById("e"+sqc);
                        var xa=elem.name;
                        var pa=songs[xa][0];
                        document.getElementById("sname").innerHTML=sname[xa];
                        document.getElementById("sel").src= pa;
                        document.getElementById("main_slider").load();
                        document.getElementById("main_slider").play();

        document.getElementById("emoji").style.backgroundImage="url('"+songs[xa]
[3]+"')";
                        songrun=true;
                        document.getElementById("queue").removeChild(elem);
                        sqc=sqc+1;
                        }

function rand_play(){
        var index=Math.random()*path.length;
        index=parseInt(index);
        var pa=songs[index][0];
        document.getElementById("sname").innerHTML=sname[index];
        document.getElementById("sel").src= pa;
        document.getElementById("main_slider").load();
        document.getElementById("main_slider").play();
        document.getElementById("emoji").style.backgroundImage="url('"+songs[ind
ex][3]+"')";
        songrun=true;

}
function moody(val){
        var index=Math.random()*mood[val].length;
        index=parseInt(index);
        var pa=songs[mood[val][index]-1][0];
        document.getElementById("sname").innerHTML=sname[mood[val][index]-1];
        document.getElementById("sel").src= pa;
        document.getElementById("main_slider").load();
        document.getElementById("main_slider").play();

        document.getElementById("emoji").style.backgroundImage="url('"+songs[mo
od[val][index]-1][3]+"')";

        songrun=true;
}

async function getTime() {
           let value = await eel.getEmotion()();
```

```
                                //let value='angry';
        if(value=="angry")
              moody(0);
        else if(value=="happy")
              moody(1);
        else if(value=="sad")
              moody(2);
        else
              moody(3);
                                console.log(value);
    }
```

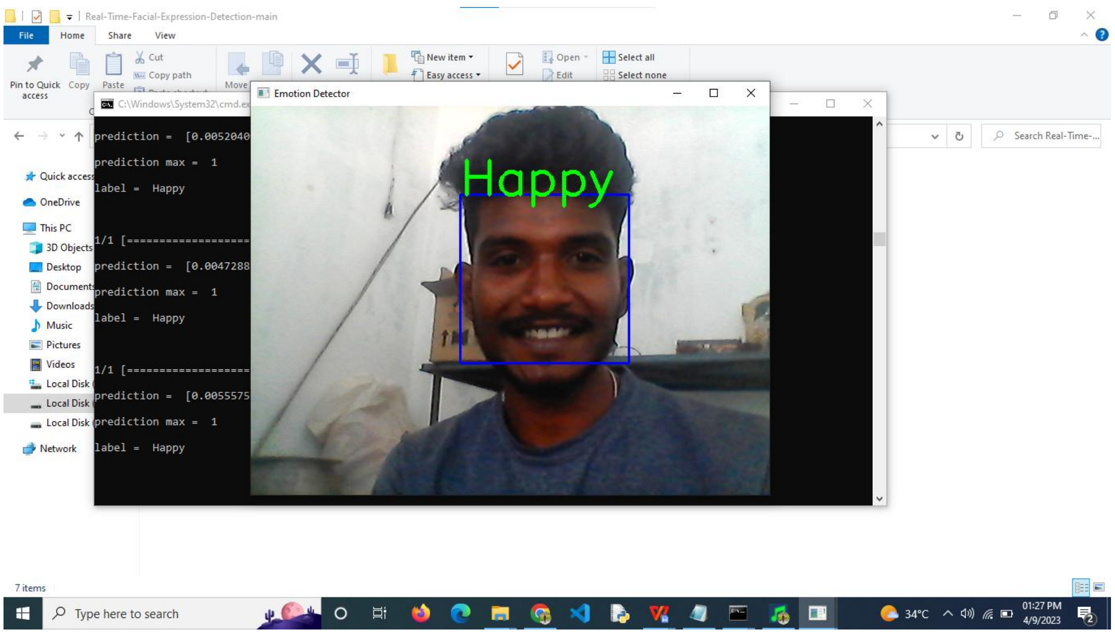
CSS File Page:

```
#first{
        width: 97.1%;
        height: 50px;
        background-color: rgba(44, 62, 80,.8);
        position: fixed;
        padding: 20px;
        top: 0;
        margin-left: 0;
        }
#queue{
        color: white;
        font-size: 20px;
        font-family: "Segoe Script";
        margin-right: 0px;
        margin-top: 27px;
        margin-left: 1100px;
        width: 230px;
        height: 600px;
        background-color: rgba(1,1,1,0.9);
        border-bottom-right-radius: 10px;
        border-top-left-radius: 10px;
        border-bottom-left-radius: 10px;
        border-top-right-radius: 10px;
        position: absolute;
        z-index: -1;
        overflow: hidden;
}

#next{
                border-color: rgba(1,1,1,0.5);
                background:transparent url("next.png");
                /*mix-blend-mode: multiply;*/
                background-size: cover;
                border-bottom-left-radius: 50%;
                border-top-left-radius: 50%;
                border-bottom-right-radius: 50%;
                border-top-right-radius: 50%;
                height: 40px;
                width: 40px;
                margin-left: 20px;
                margin-top: 0px;
                z-index: 2;
        }
        #next:focus{
                outline: none;
        }
        #next:hover{
                box-shadow: 0px 0px 5px 5px white;
                border-color: black;
                background-color: rgba(1,1,1);
```

```
            border-bottom-left-radius: 50%;
            border-top-left-radius: 50%;
            border-bottom-right-radius: 50%;
            border-top-right-radius: 50%;
            z-index: 2;

        }
```

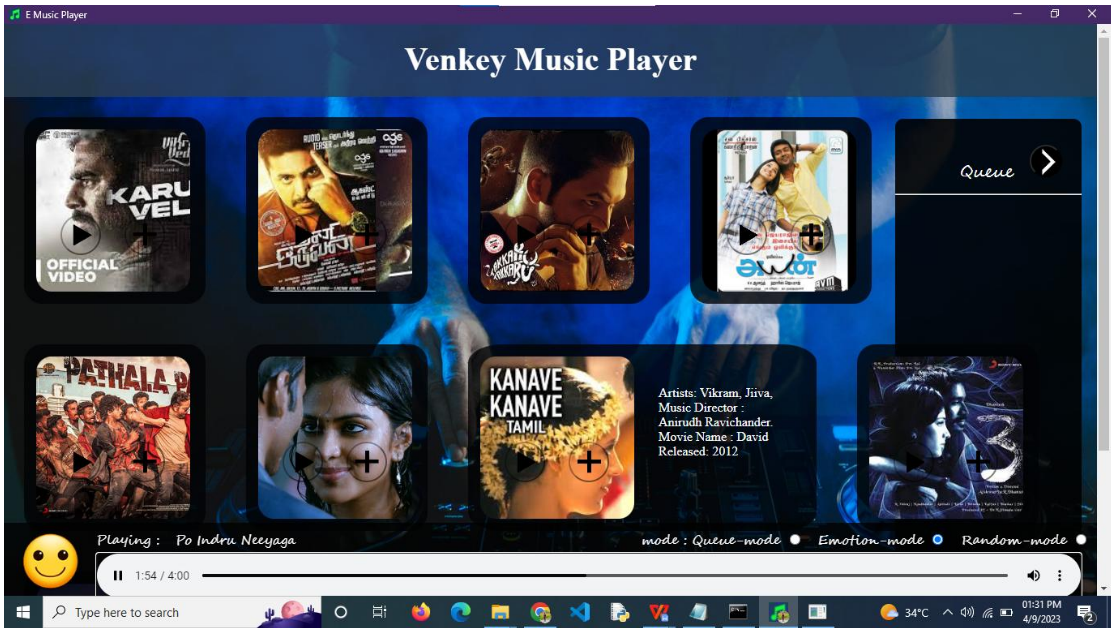
C. SAMPLE INPUT

Figure 1


Real-Time-Facial-Expression-Detection-main
C:\Windows\System32\cmd.exe
Emotion Detector
Happy

```
prediction =  [0.0052040
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0047288
prediction max =  1
label =  Happy

1/1 [==================
prediction =  [0.0055575
prediction max =  1
label =  Happy
```

Figure 2



Artists: Vikram, Jiiva,
Music Director :
Anirudh Ravichander.
Movie Name : David
Released: 2012

Playing : Po Indru Neeyaga

mode : Queue-mode [✓ Emotion-mode] Random-mode

1:54 / 4:00

D. SAMPLE OUTPUT

Figure 1 **Random Mode**

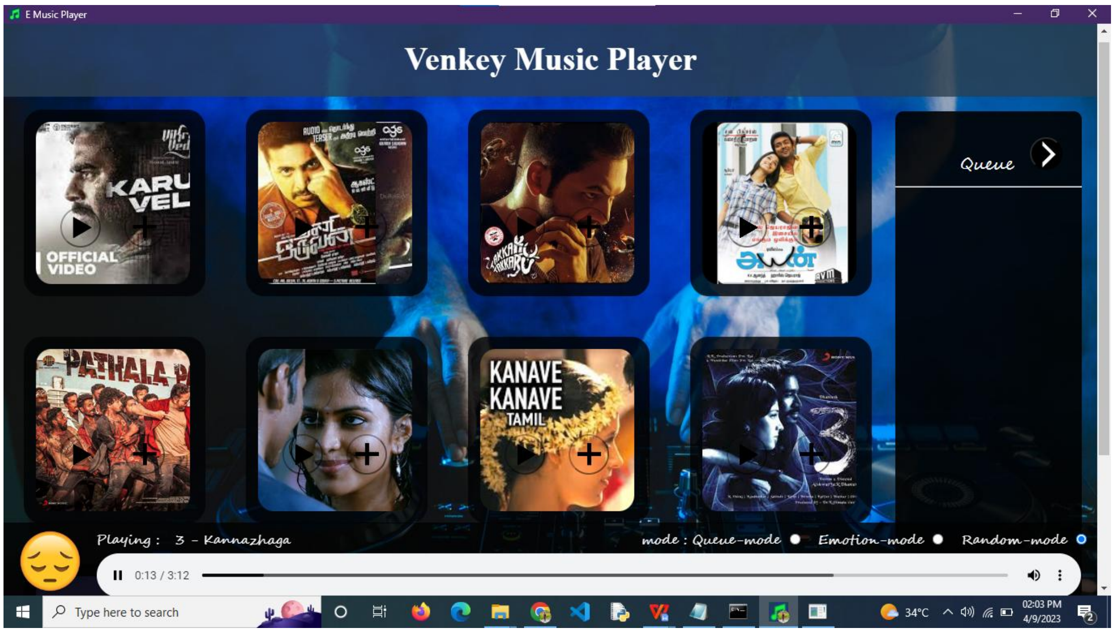
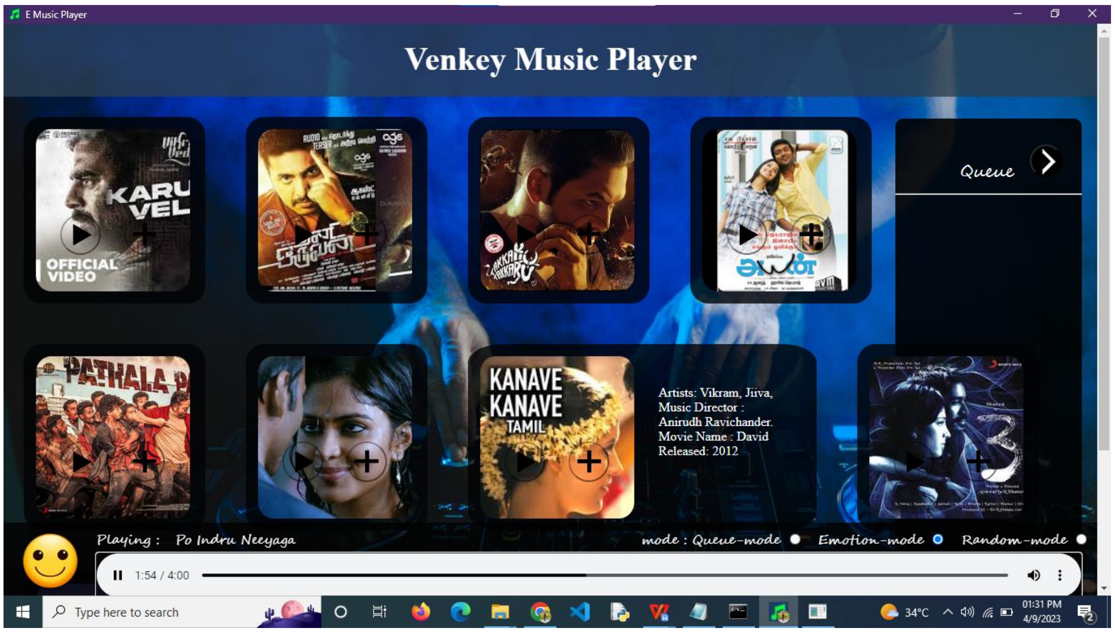
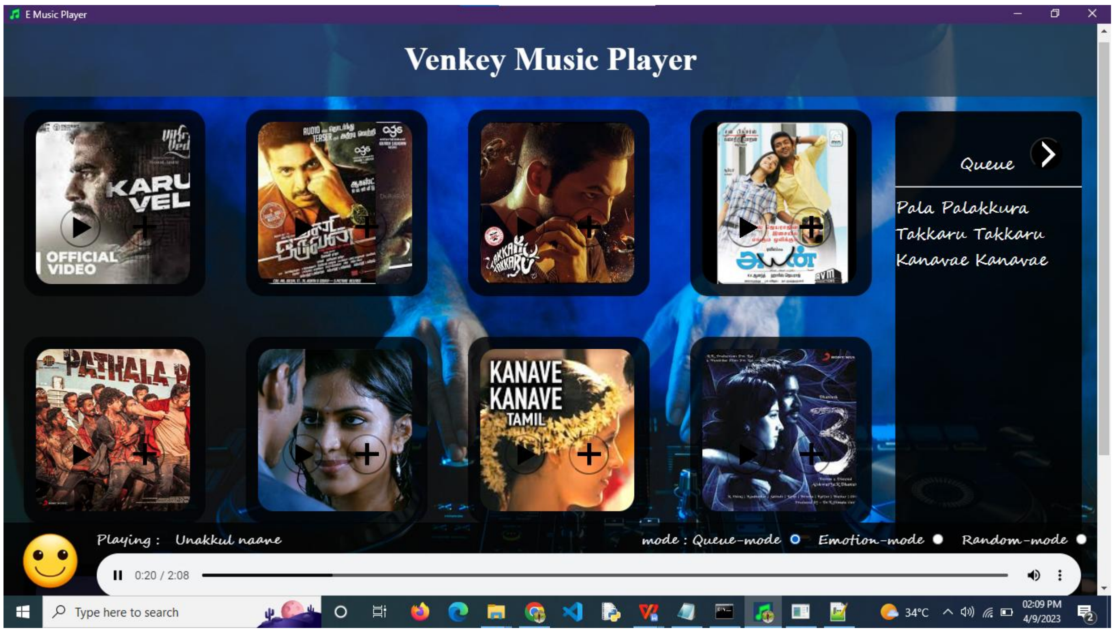


Figure 2 **Emotion Mode**


E Music Player
Venkey Music Player
Queue
Playing : Po Indru Neeyaga
mode : Queue-mode Emotion-mode Random-mode
1:54 / 4:00
Type here to search

Artists: Vikram, Jiiva,
Music Director :
Anirudh Ravichander.
Movie Name : David
Released: 2012

Figure 3 **Queue Mood**



E Music Player
Venkey Music Player
Queue
Pala Palakkura
Takkaru Takkaru
Kanavae Kanavae
Playing : Unakkul naane
mode : Queue-mode
Emotion-mode
Random-mode
0:20 / 2:08

SOFTWARE DESCRIPTION

Python

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.

History of python

Python was conceived in the late 1980s by Guido van Rossum at Centrum Wickenden & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL, capable of exception handling and interfacing with the Amoeba operating system. Its implementation began in December 1989. Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker. In January 2019, active Python core developers elected a five-member Steering Council to lead the project. Python 2.0 was released on 16 October 2000, with many major new features. Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.

Design philosophy and features

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including metaprogramming and metaobjects [magic methods]). Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.[70] It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has filter, map and reduce functions; list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (iter tools and func tools) that implement functional tools borrowed from Haskell and Standard ML.

Its core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- ❖ Beautiful is better than ugly.
- ❖ Explicit is better than implicit.
- ❖ Simple is better than complex.
- ❖ Complex is better than complicated.
- ❖ Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it"

philosophy. Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is not considered a compliment in the Python culture."

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. Python 2.7's end-of-life was initially set for 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3. No further security patches or other improvements will be released for it. With Python 2's end-of-life, only Python 3.6.x and later were supported. Later, support for 3.6 was also discontinued. In 2021, Python 3.9.2 and 3.8.8 were expedited as all versions of Python (including 2.7) had security issues leading to possible remote code execution and web cache poisoning.

Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. Thus, the program's visual structure accurately represents its semantic structure. This feature is sometimes termed the off-side rule. Some other languages use indentation this way; but in most, indentation has no semantic meaning. The recommended indent size is four spaces.

Statements and control flow

Python's statements include:

- ❖ The assignment statement, using a single equals sign (=).
- ❖ The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- ❖ The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- ❖ The while statement, which executes a block of code as long as its condition is true.
- ❖ The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses (or new syntax except* in Python 3.11 for exception groups); it also ensures that clean-up code in a finally block is always run regardless of how the block exits.
- ❖ The raise statement, used to raise a specified exception or re-raise a caught exception

- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement, which encloses a code block within a context manager (for example, acquiring a lock before it is run, then releasing the lock; or opening and closing a file), allowing resource-acquisition-is-initialization (RAII)-like behavior and replacing a common try/finally idiom.
- The break statement, which exits a loop.
- The continue statement, which skips the current iteration and continues with the next
- The del statement, which removes a variable—deleting the reference from the name to the value, and producing an error if the variable is referred to before it is redefined
- The pass statement, serving as a NOP, syntactically needed to create an empty code block
- The assert statement, used in debugging to check for conditions that should apply
- The yield statement, which returns a value from a generator function (and also an operator); used to implement coroutines.
- The return statement, used to return a value from a function.
- The import statement, used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) binds a name as a reference to a separate, dynamically-allocated object. Variables may subsequently be rebound at any time to any object. In Python, a variable name is a generic reference holder without a fixed data type; however, it always refers to some object with a type. This is called dynamic typing—in contrast to statically-typed languages, where each variable may contain only a value of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to van Rossum, it never will. However, better support for coroutine-like functionality is provided by extending Python's generators. Before 2.5, generators were lazy iterators; data was passed unidirectionally out of the generator. From Python 2.5 on, it is possible to pass data back into a generator function; and from version 3.3, it can be passed through multiple stack levels.

Expressions

Some Python expressions are similar to those in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python: floor division (or integer division) // and floating-point/division. Python also uses the ** operator for exponentiation.
- The @ infix operator. It is intended to be used by libraries such as NumPy for matrix multiplication.
- The syntax :=, called the "walrus operator", was introduced in Python 3.8. It assigns values to variables as part of a larger expression.
- In Python, == compares by value, versus Java, which compares numerics by value and objects by reference.[98] Python's is operator may be used to compare object identities (comparison by reference), and comparisons may be chained—for example, a <= b <= c.
- Python uses and, or, and not as boolean operators rather than the symbolic &&, ||, ! in Java and C.
- Python has a type of expression called a list comprehension, as well as a more general expression called a generator expression.
- Anonymous functions are implemented using lambda expressions; however, there may be only one expression in each body.
- Conditional expressions are written as x if c else (different in order of operands from the c ? x : y operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples, written as (1, 2, 3), are immutable and thus can be used as keys of dictionaries, provided all of the tuple's elements are immutable. The + operator can be used to concatenate two tuples, which does not directly modify their contents, but produces a new tuple containing the elements of both. Thus, given the variable t initially equal to (1, 2, 3), executing t = t + (4, 5) first evaluates t + (4, 5), which yields (1, 2, 3, 4, 5), which is then assigned back to t—thereby effectively "modifying the contents"

of t while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts

- ❖ Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned (to a variable, writable property, etc.) are associated in an identical manner to that forming tuple literals—and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions; when iterated through them, it assigns each of the produced values to the corresponding expression on the left
- ❖ Python has a "string format" operator % that functions analogously to printf format strings in C—e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 2.6+ and 3+, this was supplemented by the format() method of the str class, e.g. "spam={0} eggs={1}".format("blah", 2). Python 3.6 added "f-strings": spam = "blah"; eggs = 2; f'spam={spam} eggs={eggs}'.
- ❖ Strings in Python can be concatenated by "adding" them (with the same operator as for adding integers and floats), e.g. "spam" + "eggs" returns "spameggs". If strings contain numbers, they are added as strings rather than integers, e.g. "2" + "2" returns "22".

Python has various string literals:

Delimited by single or double quote marks. Unlike in Unix shells, Perl, and Perl-influenced languages, single and double quote marks work the same. Both use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals". Triple-quoted (beginning and ending with three single or double quote marks), which may span multiple lines and function like here documents in shells, Perl, and Ruby.

Raw string varieties, denoted by prefixing the string literal with r. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. (Compare "@-quoting" in C#.)

Python has array index and array slicing expressions in lists, denoted as a[key], a[start:stop] or a[start:stop:step]. Indexes are zero-based, and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted—for example, a[:] returns a copy of the entire list. Each element of a slice is a shallow copy.

Python – Modules and Packages

This article explores Python modules and Python packages, two mechanisms that facilitate modular programming. Modular programming refers to the process of breaking a large, unwieldy programming task into separate, smaller, more manageable subtasks or modules. Individual modules can then be cobbled together like building blocks to create a larger application.

There are several advantages to modularizing code in a large application:

- ❖ Simplicity: Rather than focusing on the entire problem at hand, a module typically focuses on one relatively small portion of the problem. If you're working on a single module, you'll have a smaller problem domain to wrap your head around. This makes development easier and less error-prone.
- ❖ Maintainability: Modules are typically designed so that they enforce logical boundaries between different problem domains. If modules are written in a way that minimizes interdependency, there is decreased likelihood that modifications to a single module will have an impact on other parts of the program. (You may even be able to make changes to a module without having any knowledge of the application outside that module.) This makes it more viable for a team of many programmers to work collaboratively on a large application.
- ❖ Reusability: Functionality defined in a single module can be easily reused (through an appropriately defined interface) by other parts of the application. This eliminates the need to duplicate code.
- ❖ Scoping: Modules typically define a separate namespace, which helps avoid collisions between identifiers in different areas of a program. (One of the tenets in the Zen of Python is Namespaces are one honking great idea—let's do more of those!)

Functions, modules and packages are all constructs in Python that promote code modularization.

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

The import Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The import has the following syntax :

```
import module1[, module2[,... module N]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module support.py, you need to put the following command at the top of the script :

```
#!/usr/bin/python
# Import module support
import support
# Now you can call defined function that module as follows
```

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

The from...import Statement

Python's from statement lets you import specific attributes from a module into the current namespace. The from...import has the following syntax :

```
from mod name import name1[, name2[, ... nameN]]
```

For example, to import the function fibonacci from the module fib, use the following statement :

```
from fib import fibonacci
```

This statement does not import the entire module fib into the current namespace; it just introduces the item fibonacci from the module fib into the global symbol table of the importing module.

The from...import * Statement

It is also possible to import all names from a module into the current namespace by using the following import statement :

from mod name import *

This provides an easy way to import all the items from a module into the current namespace; however, this statement should be used sparingly.

Locating Modules

When you import a module, the Python interpreter searches for the module in the following sequences :

- ❖ The current directory.
- ❖ If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH.
- ❖ If all else fails, Python checks the default path. On UNIX, this default path is normally /usr/local/lib/python/.

If the module isn't found, Python then searches each directory in the shell variable PYTHONPATH. The module search path is stored in the system module sys a variable. The sys path variable contains the current directory, PYTHONPATH, and the installation-dependent default.

Cross-compilers to other languages

There are several compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language:

- ❖ Brython, Transcrypt[149][150] and Pyjs (latest release in 2012) compile Python to JavaScript.
- ❖ Cython compiles (a superset of) Python 2.7 to C (while the resulting code is also usable with Python 3 and also e.g. C++).
- ❖ Nuitka compiles Python into C.
- ❖ Numba uses LLVM to compile a subset of Python to machine code.
- ❖ Pythran compiles a subset of Python 3 to C++ (C++11).
- ❖ RPython can be compiled to C, and is used to build the PyPy interpreter of Python.

- ❖ The Python → 11l → C++ transpiler compiles a subset of Python 3 to C++ (C++17).

Specialized:

- ❖ MyHDL is a Python-based hardware description language (HDL), that converts MyHDL code to Verilog or VHDL code.
- ❖ Older projects (or not to be used with Python 3.x and latest syntax):
- ❖ Google's Grumpy (latest release in 2017) transpiles Python 2 to Go.
- ❖ IronPython allows running Python 2.7 programs (and an alpha, released in 2021, is also available for "Python 3.4, although features and behaviors from later versions may be included") on the .NET Common Language Runtime.
- ❖ Jython compiles Python 2.7 to Java bytecode, allowing the use of the Java libraries from a Python program.
- ❖ Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively.

Uses of Python

Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache webserver. With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle, and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as a data mapper to a relational database. Twisted is framework to program communications between computers, and is used (for example) by Dropbox.

Libraries such as NumPy, SciPy, and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. OpenCV has Python bindings with a rich set of features for computer vision and image processing.

Python is commonly used in artificial intelligence projects and machine learning projects with the help of libraries like TensorFlow, Keras, Pytorch, and Scikit-learn. As a scripting language with a modular architecture, simple syntax, and rich text processing tools, Python is often used for natural language processing. Python can also be used to create games, with libraries such as Pygame, which can make 2D games.

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP, Inkscape, Scribus and Paint Shop Pro, and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4 (using Python 2.7), FreeBSD (as a package), NetBSD, and OpenBSD (as a package) and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora Linux use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

Python is used extensively in the information security industry, including in exploit development. Most of the Sugar software for the One Laptop per Child XO, now[when?] developed at Sugar Labs, is written in Python. The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

6. CONCLUSION

The Emotion based music player has been developed. The music player provides custom playlist of music according to users emotion by real-time analysis of the Facial emotion ex- pressed and according to the emotion the playlist is generated. EBMP have far superior results than other music players that only analyse facial emotions and maintain a fixed song dataset since both Facial And Song Emotion are computed. The music player also has an additional feature which is not in existing system, which is selecting emoji's and the song is played accordingly. There are four moods deployed that is Happy, Sad, Neutral, Angry. When the user clicks on the emoji, the song will be played.

Futures Scope

Songs can be exported to a cloud database so that customers can download whatever song they desire because mobile applications have a finite amount of memory. When we can only use cloud storage for additional memory needs, it is not practical to store a tonne of music inside an app. This also broadens the spectrum of songs we are working with and hence will improve the results. Identification of Complex and Mixed Emotions, Currently only four prominent emotions are being examined and there may be certain other emotions that a facial expression may convey. The user can add mixed emotions to the classes in order to maximize efficiency and prevent incorrect classification of emotions. Analyzing the song lyrics and estimate the emotion of the song and combine the emotion with the emotion of the song. Including voice input for faster emotion recognition.

7. BIBLIOGRAPHY

1) Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, pp. 511-518 IEEE, 2001 (2001)

2) H. Immanuel James, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan, R. Saranya" EMOTION BASED MUSIC RECOMMENDATION SYSTEM": pISSN: 2395-0072 , IRJET 2019

3) Hafeez Kabani, Sharik Khan , Omar Khan , Shabana Tadvi"Emotion Based Music Player" International Journal of Engineering Research and General Science
Volume 3, Issue 1, January-February , 2015

4) Shlok Gikla, Husain Zafar, Chuntan Soni, Kshitija Waghurdekar"SMART MUSIC INTEGRATING AND MUSIC MOOD RECOMMENDATION"2017 International Conference on Wireless Communications, Signal Processing and Networking(WiSppNET)

5) T.-H. Wang and J.-J.J. Lien, "Facial Expression Recognition System Based on Rigid and Non-Rigid Motion Separation and 3D Pose Estimation" J. Pattern Recognition , vol. 42, no. 5,pp. 962-977, 2009

6) Srushti Sawant, Shraddha Patil, Shubhangi Biradar,"EMOTION BASED MUSIC SYSTEM", International Journal of Innovations & Advancement in Coputer Science,IJIACS ISSN 2347-8616 volue 7, Issue 3 March 2018

7) Sudha Veluswamy, Hariprasad Kanna, Balasubramanian Anand, Anshul Sharma "METHOD AND APPARATUS FOR RECOGNIZING AN EMOTION OF AN INDIVIDUAL BASED ON FACIAL ACTION UNITS"US2012/0101735A1

8) Markus Mans Folke Andreasson"GENERATING MUSIC PLAYLIST BASED ON FACIAL EXPRESSION"US8094891B2

9) Mutasem K. Alsmadi " FACIAL EXPRESSION RECOGNITION " US2018/0211102A1

10) https://www.raspberrypi.org/educatio

11) https://www.pyimagesearch.com/2018/09/10 /keras-tutorial-how-to-get- started with - keras-deep-learning-and-python/

12) https://towardsdatascience.com/face-detection-recognition-and-emotion detection-in-8-lines-of- code-b2ce32d4d5de

13)https://medium.com/@hinasharma19se/facial-expressions-recognitionb022318d8a

14) K. S. Nathan,et al "EMOSIC — An emotion based music player for Android,"2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT), Bilbao, 2017, pp. 371-276.