FaceMelody - Mood-Aware Music Recommendation System with Facial Recognition

A Project Report Submitted in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

PRAVALLIKA (2010030046)

TAHSEEN BEGUM (2010030168)

M. RUSHITHA (2010030362)

K. KAVYA (2010030550)

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
K L DEEMED TO BE UNIVERSITY
AZIZNAGAR, MOINABAD, HYDERABAD-500075

OCTOBER 2023

BONAFIDE CERTIFICATE

This is to certify that the project titled **FaceMelody – Mood Aware Music Recommendation System with Facial Recognition** is a bonafide record ofthe work done by

PRAVALLIKA (2010030046)

TAHSEEN BEGUM (2010030168)

M. RUSHITHA (2010030362)

K.KAVYA (2010030550)

in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** of the **K L DEEMED TO BE UNIVERSITY, AZIZNAGAR, MOINABAD, HYDERABAD-500075**, during the year 2023-2024.

Mr. Chiranjeevi Nuthalapati
Project Guide

Dr. Arpita Gupta
Head of the Department

Project Viva-voce held on _____

Internal Examiner

External Examiner

ABSTRACT

The proposed FaceMelody analyzes the user's facial expressions in real time. Through the user's webcam, the system captures emotional cues, such as joy, sadness, or excitement, and processes this data to understand the user's current mood.

FaceMelody leverages a sophisticated recommendation engine that correlates the detected mood with a vast music database, ensuring that the music selection aligns with the user's emotional context.

The recommendation engine considers a variety of factors, including tempo, genre, lyrics, and historical preferences to provide a personalized and emotionally resonant playlist. The system also adapts in real-time, allowing users to change their music selection as their mood evolves. Whether a user is looking to lift their spirits, relax, or reflect, FaceMelody is designed to cater to these emotional shifts.

ACKNOWLEDGEMENT

We would like to thank the following people for their support and guidance without whom the completion of this project in fruition would not be possible.

Mr. Chiranjeevi Nuthalapati, our project guide, for helping us and guiding us in thecourse of this project.

Dr. Arpita Gupta, the Head of the Department, Department of Computer Science And Engineering.

Our internal reviewers, **Ms. N Anuradha**, **Mr. Faizan Ahmed**, **Dr. Rajib Debnath** for their insight and advice provided during the review sessions.

We would also like to thank our individual parents and friends for their constant support.

TABLE OF CONTENTS

List of Figures

Chapter 1

Introduction

1.1 Background of the Project

Music has been a significant part of human culture and expression for centuries. It holds the power to evoke emotions, create memories, and provide solace. With the digital revolution, music consumption has become more personalized than ever before, thanks to the advent of music streaming platforms and recommendation systems. However, these systems are often limited to algorithms that analyze user history, preferences, and other contextual data, neglecting a crucial element of human experience: emotions.

Emotions play a pivotal role in how music affects us. A song that resonates with one person during a moment of happiness may not have the same impact during a moment of sadness. This interplay between music and emotions has been the subject of numerous studies and has given rise to the field of music psychology. It has long been recognized that music can either enhance or alter one's emotional state.

The FaceMelody transforms the music listening experience, enhancing user engagement and emotional well-being. This project represents a harmonious blend of technology and human emotion to create a more personalized and emotionally resonant music experience.

1.2 Problem Statement

This project is aimed at developing a music recommendation system that is capable of recognizing the user's mood through facial recognition technology. This system will analyze user's facial expressions in real-time to determine their emotional states and preferences. By combining facial data with mood-specific music databases, this system will deliver personalized music recommendations that align with user's emotions, enhancing their listening experience and fostering emotional well-being.

1.3 Objectives

- **Facial Emotion Recognition**: The system uses a pre-trained convolutional neural network (CNN) model to recognize facial emotions. The model takes an input image of a face and predicts the emotion associated with it. The emotions recognized include anger, disgust, fear, happiness, neutrality, sadness, and surprise.
- **Music Recommendation**: The system recommends music based on the user's detected emotion. Each emotion is associated with a specific music genre or playlist. The system retrieves the corresponding music playlist from a CSV file and presents it to the user.
- **Video Streaming**: The system captures video frames from the user's webcam in real-time. It processes each frame to detect facial emotions and recommends music accordingly. The video streaming is performed using OpenCV, a popular computer vision library.

1.4 Scope of the Project

- **Facial Emotion Recognition**: Implementing facial recognition technology to detect and analyze facial expressions in real time. Emotions such as happiness, sadness, anger, and more will be recognized.
- **Emotion - to - Music Mapping**: Creating an emotion-to-music database that associates specific emotional states with music genres, moods, or specific songs.
- **User Interface**: Developing a user-friendly interface, such as a mobile app or web platform, to access and interact with the FaceMelody.
- **Real - time Emotion Analysis**: Ensuring the system can process and interpret facial expressions continuously in real-time to adapt music recommendations on the fly.
- **Music Database Integration**: Connecting FaceMelody with music streaming services to access a vast library of songs and genres.

Chapter 2

Literature Review

Dharmendra Roy, Anjali. CH, G. Kavya Sri, B. Tharun, K. Venu Gopal proposed a system extracts initial or raw data from faces and reduces it to many other classes using methods like principal component analysis (PCA) and Fisher's Linear Discriminant method (LDA). The system uses facial recognition technology to detect the user's emotions, which can be more accurate than relying on self-reported mood.

Mrs. P. P. Kambare, Dr. S. T. Patil, D Y Patil proposed a system that captures facial expressions through a webcam and analyzes them using a convolutional neural network (CNN) to recognize emotions and then maps to a set of songs that match the user's mood, based on sentiment analysis of song lyrics and metadata. The system can adapt to users' listening patterns and preferences over time, providing fresh and relevant recommendations.

Ankita Mahadik, Shambhavi Milgir, Prof. Vaishali Kavathekar proposed a mood-based music player and which creates performs real time mood detection and suggests songs as per detected mood. This becomes an additional feature to the traditional music player apps that come pre-installed in our mobile phones. Neural networks and machine learning have been used for these tasks and have obtained good results.

Magnus, Mortensen, Cathal, Gurrinand, Dag Johansen proposed a novel music recommendation system that incorporates both collaborative filtering and mood-based recommendations. This mood-based recommendation is positively evaluated on a closed set of user listening data, retrospectively gathered with recommendations based on user’s playback history.

2.1 Overview of related works

We will analyze a Flask application that integrates a camera. The application allows users to view a live video feed and interact with a table of music data. We will explore an emotion-based music recommendation system using Python. The system uses facial emotion recognition to detect the user's emotion and recommends music based on their emotional state.

2.2 Advantages and Limitations of existing systems

- **Advantages**

Personalization: Facial expression-based systems can provide highly personalized music recommendations by taking into account the user's current emotional state. This can lead to a more engaging and enjoyable music listening experience.

Emotion Detection: They can detect and respond to a wide range of emotions, including happiness, sadness, excitement, and relaxation, enabling a more comprehensive emotional connection with the user.

Mood Regulation: Such systems can help users regulate their emotions by recommending music that matches their desired mood, which can be beneficial for mental health and well-being.

- **Limitations**

Limited Accuracy: Facial expression analysis is not always accurate in determining a user's emotional state. Users may not always express their emotions through their facial expressions, or there may be misinterpretations.

Privacy Concerns: Analyzing facial expressions for music recommendations can raise privacy concerns. Users may be uncomfortable with their facial data being used for such purposes.

Data Quality: The effectiveness of these systems depends on the quality of the facial expression data and the accuracy of emotion detection algorithms. Inaccurate data can lead to poor recommendations.

Chapter 3

Proposed System

3.1 System Requirements

Software Requirements

The major software requirements of the project are as follows:

Language: Python

Operating system: Windows XP

3.1.1 Hardware Requirements

The hardware requirements that map to the software are as follows:

RAM: 8.00 GB (7.78 GB usable)

Processor: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.10 GHz

System-type: 64-bit operating system, x64-based processor

Version: 22H2

Edition: Windows 10 Pro

3.2 Design of the System

The emotion recognition model is trained on the FER 2013 dataset. It can detect 7 emotions. The project works by getting a live video feed from a webcam, and passing it through the model to get a prediction of emotion. Then according to the emotion predicted, the app will fetch a playlist of songs from Spotify through spotipy wrapper and recommend the songs by displaying them on the screen.

3.2.1 Code Structure

The code consists of several import statements, class and function definitions.

Import Statements: The code imports various libraries and modules required for the emotion recognition system. These include numpy, cv2 (OpenCV), PIL (Python Imaging Library), tensorflow.keras, pandastable, and datetime.

Music and Emotion Dictionaries: The code defines two dictionaries: "emotion_dict" and "music_dist". The "emotion_dict" maps emotion labels to their corresponding names, while the "music_dist" maps emotion labels to the paths of CSV files containing music recommendations for each emotion.

FPS and WebcamVideoStream Classes: The code defines two classes: "FPS" and "WebcamVideoStream". The "FPS" class calculates the frames per second (FPS) of the video stream, while the "WebcamVideoStream" class reads frames from the video stream in a separate thread to boost performance.

VideoCamera Class: The code defines the "VideoCamera" class, which is responsible for capturing video frames, performing emotion recognition, and recommending music. The class uses the face cascade classifier to detect faces, extracts the facial region of interest (ROI), and feeds it to the emotion model for prediction. It also retrieves the corresponding music recommendations from the CSV files based on the detected emotion.

Spotipy: Spotipy is a Python library that allows us to interact with the Spotify Web API. It provides methods to authenticate with Spotify, fetch playlist data, and retrieve information about tracks, albums, and artists.

Spotify Web API: The Spotify Web API is a RESTful API that provides access to various Spotify resources, such as playlists, tracks, albums, and artists. It allows developers to retrieve and manipulate data from the Spotify music catalog.

Emotion Analysis: Emotion analysis, also known as sentiment analysis, is the process of determining the emotional tone of a piece of text or audio. In our case, we will analyze the emotions associated with the tracks in a Spotify playlist.

Emotion Recognition: Emotion recognition involves training a model to recognize and

classify human emotions based on facial expressions. The model is trained on a dataset of labeled facial images, where each image is associated with a specific emotion label (e.g., happy, sad, angry, etc.).

WebcamVideoStream: This class represents a separate thread for video streaming from a web camera. It utilizes the OpenCV library (cv2) to capture frames from the camera.

Thread: The Thread class from the threading module is used to create a separate thread for reading frames from the video stream. This allows the video streaming to run concurrently with other tasks.

VideoCapture: The VideoCapture class from the OpenCV library is used to capture video frames from the web camera. It takes the camera source as an argument (default is 0, which represents the default camera).

The WebcamVideoStream class provides a convenient way to stream video from a web camera using a separate thread in Python. By utilizing a separate thread, we can perform other tasks concurrently while continuously reading frames from the video stream.

Facial Emotion Detection: FaceMelody uses computer vision techniques to analyze facial expressions and detect emotions such as happiness, sadness, anger, and surprise. This is done by capturing frames from the webcam and applying machine learning algorithms to classify the emotions based on facial features.

Song Recommendations: Once the user's emotions are detected, FaceMelody recommends songs that align with their emotional state. These recommendations are based on a pre-defined database of songs and their associated emotional characteristics. The application selects songs that are known to evoke similar emotions in listeners.

Real-time Updates: FaceMelody continuously updates the song recommendations based on the user's changing emotions. It constantly analyzes the facial expressions captured by the webcam and adjusts the recommendations accordingly. This ensures that the suggested songs are always relevant to the user's current emotional state.

3.3 Algorithms and Techniques used

Convolutional Neural Networks (CNNs): CNNs are a type of deep learning model that is particularly effective for image recognition tasks. They consist of multiple layers of convolutional and pooling operations, followed by fully connected layers for classification.

CNN model is used for facial emotion recognition. The model architecture consists of several convolutional and pooling layers, followed by fully connected layers. The model is loaded with pre-trained weights from a file.

A pre-trained convolutional neural network (CNN) model is used to recognize facial emotions. The model takes an input image of a face and predicts the emotion associated with it.

By training a CNN model on a dataset of labeled facial images, we can develop a powerful tool for emotion recognition.

3.3.1 Techniques

ImageDataGenerator: The ImageDataGenerator class in Keras is a powerful tool for data augmentation and preprocessing. It allows us to generate batches of augmented image data on the fly, which helps improve the performance and generalization of our model.

Face Cascade Classifier: It's a technique used for face detection in images and videos. The code initializes a face cascade classifier using the "haarcascade_frontalface_default.xml" file. This classifier is used to detect faces in the video frames. The "VideoCamera" class, is responsible for capturing video frames, performing emotion recognition, and recommending music. The class uses the face cascade classifier to detect faces, extracts the facial region of interest (ROI), and feeds it to the emotion model for prediction. It also retrieves the corresponding music recommendations from the CSV files based on the detected emotion.

Chapter 4

Implementation

4.1 Tools and Technologies used

OpenCV (cv2): An open-source computer vision library that provides various tools for image and video analysis. The video streaming is performed using OpenCV(a popular computer vision library). The WebcamVideoStream class utilizes the OpenCV library (cv2) to capture frames from the camera. The VideoCapture class from the OpenCV library is used to capture video frames from the web camera. It takes the camera source as an argument (default is 0, which represents the default camera).

PIL (Python Imaging Library): A library for opening, manipulating, and saving many different image file formats in Python.

Pandastable: A tool for creating graphical data tables for pandas data frames.

4.1.1 Technologies

Keras: Convolution Neural Network along with keras library is used for emotion recognition.

TensorFlow.keras: Part of the TensorFlow framework, it's a high-level neural networks API that is based on the Keras library.

Spotipy: Spotipy is a Python library that allows us to interact with the Spotify Web API. It provides methods to authenticate with Spotify, fetch playlist data, and retrieve information about tracks, albums, and artists. The Spotify Web API is a RESTful API that provides access to various Spotify resources, such as playlists, tracks, albums, and artists. It allows developers to retrieve and manipulate data from the Spotify music catalog. Emotion analysis, also known as sentiment analysis, is the process of

determining the emotional tone of a piece of text or audio. In our case, we will analyze the emotions associated with the tracks in a Spotify playlist.

Flask: The application is built using Flask which integrates a camera. It allows users to view a live video feed and interact with a table of music data.

4.2 Modules and their Descriptions

Modules(Python Libraries)

Numpy

DateTime

Spotipy

4.2.1 Description of Modules

NumPy: A fundamental package for scientific computing with Python, used for numerical operations and data manipulation.

DateTime: A module used to work with dates and times in Python.

Spotipy: Spotipy is a Python library that allows us to interact with the Spotify Web API. It provides methods to authenticate with Spotify, fetch playlist data, and retrieve information about tracks, albums, and artists. The Spotify Web API is a RESTful API that provides access to various Spotify resources, such as playlists, tracks, albums, and artists. It allows developers to retrieve and manipulate data from the Spotify music catalog. Emotion analysis, also known as sentiment analysis, is the process of determining the emotional tone of a piece of text or audio. In our case, we will analyze the emotions associated with the tracks in a Spotify playlist.

4.3 Flow of the System

FaceMelody is an innovative web application that combines facial emotion detection with song recommendations. By analyzing the user's facial expressions in real time, it provides personalized song suggestions that match their emotional state. The provided code defines the structure and styling of the web application, as well as the JavaScript code for updating the song recommendations dynamically. With FaceMelody, users can discover new music that resonates with their emotions and enhances their listening experience.

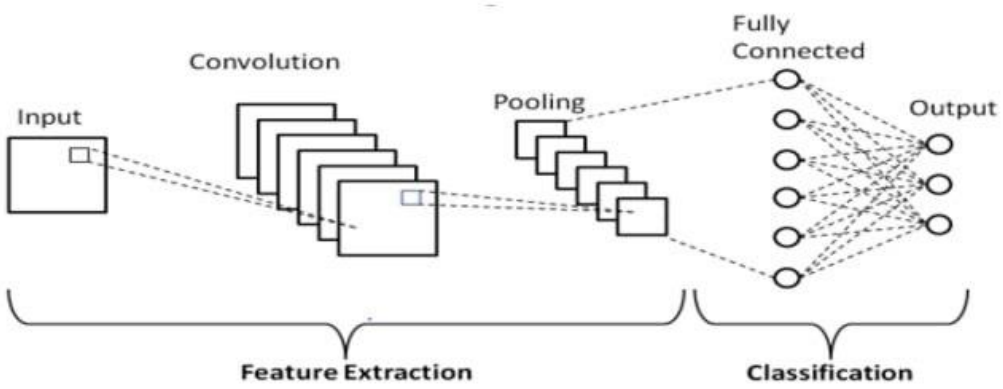

Fig 4.3 Flow of the System

We will input an user image and then it will pass through the convolution and pooling layers for feature extraction and for classification it goes through fully connected layer and provides an output.

Chapter 5

Results and Analysis

5.1 Performance Evaluation

Performance evaluation of a music recommendation system based on face emotion recognition involves assessing the system's effectiveness in recommending music that matches the emotional state of the user.

5.1.1 Model Architecture

The model architecture is a sequential model consisting of Conv2d, Maxpool2d, Dropout and Dense layers:

Conv2D layers throughout the model have different filter size from 32 to 128, all with activation 'relu'

Pooling layers have pool size (2,2)

Dropout is set to 0.25 as anything above results in poor performance

Final Dense layer has 'softmax' activation for classifying 7 emotions

Used 'categorical_crossentropy' for loss with 'Adam' optimizer with 'accuracy' metric

Note:- This model architecture gives good enough accuracy. A bit more tinkering with hyperparameters might lead to better accuracy.

Image Processing and Training: The images were normalised, resized to (48,48) and converted to grayscale in batches of 64 with the help of 'ImageDataGenerator' in Keras API.

Training took around 13 hours locally for 75 epochs with an accuracy of ~88 %

5.2 Comparison with existing systems

The FaceMelody is better than other available facial emotion recognition-based music recommendation system because of the following things:

Emotion Recognition Accuracy

Recommendation Quality

Real-time Processing

User Experience

5.3 Limitations and future scope

Privacy Concerns: Face emotion recognition raises privacy concerns, as it involves processing and storing facial images. Users might be uncomfortable with their facial data being used for recommendations.

Real-Time Processing: The real-time processing requirement for live recommendations can be resource-intensive. This could limit the system's performance on devices with limited processing power.

Cultural Differences: Emotion expressions can vary across cultures. A system trained on one cultural group may not work as effectively for others.

5.3.1 Future Scope

Multimodal Emotion Recognition: Combine facial emotion recognition with other modalities such as text sentiment analysis to improve accuracy.

Context Awareness: Incorporate contextual information, such as location, time of day, or recent activities, to enhance recommendations based on emotional states.

Personalization: Implement user profiles and historical data to provide more personalized recommendations tailored to an individual's preferences.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

In conclusion, the development of music recommendation systems based on face emotion recognition, leveraging Convolutional Neural Networks (CNN) and OpenCV, represents a significant stride in the field of personalized and emotionally intelligent music streaming. This innovative system harnesses the power of computer vision to enhance user experiences by tailoring music recommendations to individual emotional states. Throughout this journey, we've explored the fundamental components and techniques involved in this system. It combines the capabilities of CNNs to accurately recognize facial emotions and OpenCV for real-time image processing. By categorizing and analyzing the emotions detected, it provides users with music selections that align with their current emotional context.

6.2 Contributions and achievements

2010030168_Tahseen Begum - worked on the application.

2010030046_Eppali Pravallika - worked on voice assistant.

2010030362_Rushitha Sri – worked on facial recognition.

2010030550_K.Kavya - worked on music recognition.

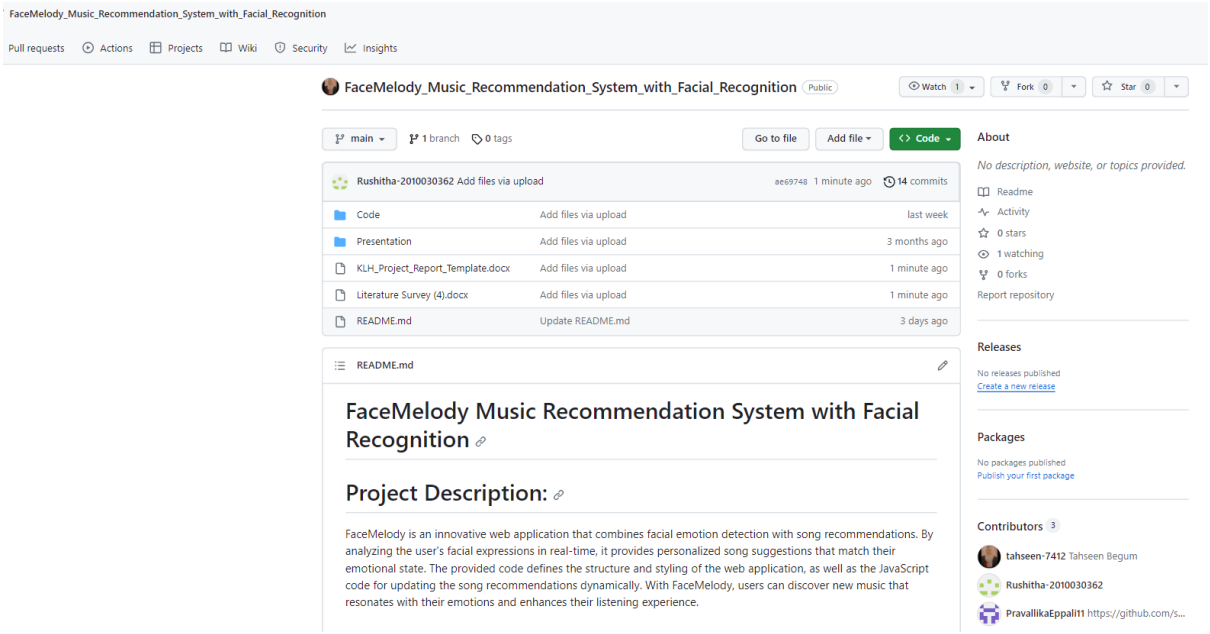

Fig 6.2 Github

6.3 Recommendations for future work

- Instead of CSVs, create a database and connect it to the application. The DB will fetch songs for recommendations and new songs can be updated directly onto the database.
- Add a feature that will update specified playlists for better and more recent recommendations, a specific day over a fixed duration say every sunday, and append it to the database.
- Directly play the song or redirect to the song on Spotify when the user clicks on it.
- Rewrite code such that Video Streaming is done on the client side instead of server side so as it make the app deployable.

Bibliography

[1] Roy, Dharmendra, Anjali CH, G. Kavya Sri, B. Tharun, and K. Venu Gopal. "Music Recommendation Based on Current Mood Using Ai & Ml." (2023).

[2] Kambare, Mrs PP, and S. T. Patil. "FACIAL IMAGE BASED EMOTION DETECTION AND MUSIC RECOMMENDATION SYSTEM.".

[3] Mahadik, Ankita, Shambhavi Milgir, Janvi Patel, Vijaya Bharathi Jagan, and Vaishali Kavathekar. "Mood based music recommendation system." INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 10 (2021).

[4] Mortensen, Magnus, Cathal Gurrin, and Dag Johansen. "Real-world mood-based music recommendation." In Information Retrieval Technology: 4th Asia Infomation Retrieval Symposium, AIRS 2008, Harbin, China, January 15-18, 2008 Revised Selected Papers 4, pp. 514-519. Springer Berlin Heidelberg, 2008.

[5] Joshi, Saurav, Tanuj Jain, and Nidhi Nair. "Emotion based music recommendation system using LSTM-CNN architecture." In 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), pp. 01-06. IEEE, 2021.

[6] Mahadik, Ankita, Shambhavi Milgir, Janvi Patel, Vijaya Bharathi Jagan, and Vaishali Kavathekar. "Mood based music recommendation system." INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 10 (2021).

[7] Sharma, Vijay Prakash, Azeem Saleem Gaded, Deevesh Chaudhary, Sunil Kumar, and Shikha Sharma. "Emotion-based music recommendation system." In 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO), pp. 1-5. IEEE, 2021.

[8] James, H. Immanuel, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan, and R. Saranya. "Emotion based music recommendation system." Emotion 6, no. 3 (2019).

Appendices

Appendix A

Source code

App.py

```
from flask import Flask, render_template, Response, jsonify
import gunicorn
from camera import *

app = Flask(__name__)

headings = ("Name","Album","Artist")
df1 = music_rec()
df1 = df1.head(15)
@app.route('/')
def index():
    print(df1.to_json(orient='records'))
    return render_template('index.html', headings=headings, data=df1)

def gen(camera):
    while True:
        global df1
        frame, df1 = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/t')
def gen_table():
    return df1.to_json(orient='records')

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

camera.py

```
import numpy as np
import cv2
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from pandastable import Table, TableModel
from tensorflow.keras.preprocessing import image
import datetime
from threading import Thread
from Spotipy import *
import time
import pandas as pd
face_cascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
ds_factor=0.6

emotion_model = Sequential()
emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
emotion_model.load_weights('model.h5')

cv2.ocl.setUseOpenCL(False)

emotion_dict =
{0:"Angry",1:"Disgusted",2:"Fearful",3:"Happy",4:"Neutral",5:"Sad",6:"Surprise
d"}
music_dist={0:"songs/angry.csv",1:"songs/disgusted.csv
",2:"songs/fearful.csv",3:"songs/happy.csv",4:"songs/neutral.csv",5:"songs/sad
.csv",6:"songs/surprised.csv"}
global last_frame1
last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
global cap1
show_text=[0]
```

```
''' Class for calculating FPS while streaming. Used this to check performance
of using another thread for video streaming '''
class FPS:
    def __init__(self):
        # store the start time, end time, and total number of frames
        # that were examined between the start and end intervals
        self._start = None
        self._end = None
        self._numFrames = 0
    def start(self):
        # start the timer
        self._start = datetime.datetime.now()
        return self
    def stop(self):
        # stop the timer
        self._end = datetime.datetime.now()
    def update(self):
        # increment the total number of frames examined during the
        # start and end intervals
        self._numFrames += 1
    def elapsed(self):
        # return the total number of seconds between the start and
        # end interval
        return (self._end - self._start).total_seconds()
    def fps(self):
        # compute the (approximate) frames per second
        return self._numFrames / self.elapsed()


''' Class for using another thread for video streaming to boost performance
'''
class WebcamVideoStream:

        def __init__(self, src=0):
            self.stream = cv2.VideoCapture(src,cv2.CAP_DSHOW)
            (self.grabbed, self.frame) = self.stream.read()
            self.stopped = False

        def start(self):
                # start the thread to read frames from the video stream
            Thread(target=self.update, args=()).start()
            return self

        def update(self):
            # keep looping infinitely until the thread is stopped
            while True:
                # if the thread indicator variable is set, stop the thread
                if self.stopped:
```

```
                    return
                # otherwise, read the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()

        def read(self):
            # return the frame most recently read
            return self.frame
        def stop(self):
            # indicate that the thread should be stopped
            self.stopped = True

''' Class for reading video stream, generating prediction and recommendations
'''
class VideoCamera(object):

    def get_frame(self):
        global cap1
        global df1
        cap1 = WebcamVideoStream(src=0).start()
        image = cap1.read()
        image=cv2.resize(image,(600,500))
        gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
        face_rects=face_cascade.detectMultiScale(gray,1.3,5)
        df1 = pd.read_csv(music_dist[show_text[0]])
        df1 = df1[['Name','Album','Artist']]
        df1 = df1.head(15)
        for (x,y,w,h) in face_rects:
            cv2.rectangle(image,(x,y-50),(x+w,y+h+10),(0,0,0),2)
            roi_gray_frame = gray[y:y + h, x:x + w]
            cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0)
            prediction = emotion_model.predict(cropped_img)

            maxindex = int(np.argmax(prediction))
            show_text[0] = maxindex
            #print("============================================",music_dist[sh
ow_text[0]],"============================================")
            #print(df1)
            cv2.putText(image, emotion_dict[maxindex], (x+20, y-60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
            df1 = music_rec()

        global last_frame1
        last_frame1 = image.copy()
        pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
        img = Image.fromarray(last_frame1)
        img = np.array(img)
        ret, jpeg = cv2.imencode('.jpg', img)
        return jpeg.tobytes(), df1
```

```
def music_rec():
    # print('---------------- Value ------------', music_dist[show_text[0]])
    df = pd.read_csv(music_dist[show_text[0]])
    df = df[['Name','Album','Artist']]
    df = df.head(15)
    return df
```

Spotipy.py

```
import spotipy
import spotipy.oauth2 as oauth2
from spotipy.oauth2 import SpotifyOAuth
from spotipy.oauth2 import SpotifyClientCredentials
import pandas as pd
import time

auth_manager =
SpotifyClientCredentials('b8b938d01d0a4995b8f1f361bf576540','6194d75d995f4ed59
32e8dc6f7ce8fc1')
sp = spotipy.Spotify(auth_manager=auth_manager)

def getTrackIDs(user, playlist_id):
    track_ids = []
    playlist = sp.user_playlist(user, playlist_id)
    for item in playlist['tracks']['items']:
        track = item['track']
        track_ids.append(track['id'])
    return track_ids

def getTrackFeatures(id):
    track_info = sp.track(id)

    name = track_info['name']
    album = track_info['album']['name']
    artist = track_info['album']['artists'][0]['name']
    # release_date = track_info['album']['release_date']
    # length = track_info['duration_ms']
    # popularity = track_info['popularity']

    track_data = [name, album, artist] #, release_date, length, popularity
    return track_data

# Code for creating dataframe of feteched playlist

emotion_dict =
{0:"Angry",1:"Disgusted",2:"Fearful",3:"Happy",4:"Neutral",5:"Sad",6:"Surprise
d"}
music_dist={0:"0l9dAmBrUJLylii66JOsHB?si=e1d97b8404e34343",1:"1n6cpWo9ant4WguE
o91KZh?si=617ea1c66ab6446b
",2:"4cllEPvFdoX6NIVWPKai9I?si=dfa422af2e8448ef",3:"0deORnapZgrxFY4nsKr9JA?si=
7a5aba992ea14c93",4:"4kvSlabrnfRCQWfN0MgtgA?si=b36add73b4a74b3a",5:"1n6cpWo9an
```

```
t4WguEo91KZh?si=617ea1c66ab6446b",6:"37i9dQZEVXbMDoHDwVN2tF?si=c09391805b6c4651"}

'''
Code can def be modularised into a function but i tried to write it when i was extremely sleepy so thought screw it and repeated code block

Uncomment for fetching updated playlists
'''


# track_ids = getTrackIDs('spotify',music_dist[0])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #,'Release_date','Length','Popularity'
#     df.to_csv('songs/angry.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[1])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #,'Release_date','Length','Popularity'
#     df.to_csv('songs/disgusted.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[2])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #,'Release_date','Length','Popularity'
#     df.to_csv('songs/fearful.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[3])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
```

```
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #
,'Release_date','Length','Popularity'
#     df.to_csv('songs/happy.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[4])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #
,'Release_date','Length','Popularity'
#     df.to_csv('songs/neutral.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[5])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #
,'Release_date','Length','Popularity'
#     df.to_csv('songs/sad.csv')
# print("CSV Generated")

# track_ids = getTrackIDs('spotify',music_dist[6])
# track_list = []
# for i in range(len(track_ids)):
#     time.sleep(.3)
#     track_data = getTrackFeatures(track_ids[i])
#     track_list.append(track_data)
#     df = pd.DataFrame(track_list, columns = ['Name','Album','Artist']) #
,'Release_date','Length','Popularity'
#     df.to_csv('songs/surprised.csv')
# print("CSV Generated")
```

Train.py

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator

train_dir = 'data/train'
val_dir = 'data/test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)
```

```
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = (48,48),
    batch_size = 64,
    color_mode = "grayscale",
    class_mode = 'categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size = (48,48),
    batch_size = 64,
    color_mode = "grayscale",
    class_mode = 'categorical'
)

emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape
= (48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))

emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.0001
, decay=1e-6),metrics=['accuracy'])

emotion_model_info = emotion_model.fit_generator(
    train_generator,
    steps_per_epoch = 28709 // 64,
    epochs=75,
    validation_data = val_generator,
    validation_steps = 7178 // 64
)

emotion_model.save_weights('model.h5')
```

Utils.py

```
''' Class for using separate thread for video streaming through web camera'''
import cv2
from threading import Thread
class WebcamVideoStream:

        def __init__(self, src=0):
            self.stream = cv2.VideoCapture(src,cv2.CAP_DSHOW)
            (self.grabbed, self.frame) = self.stream.read()
            self.stopped = False

        def start(self):
                # start the thread to read frames from the video stream
            Thread(target=self.update, args=()).start()
            return self

        def update(self):
            # keep looping infinitely until the thread is stopped
            while True:
                # if the thread indicator variable is set, stop the thread
                if self.stopped:
                    return
                # otherwise, read the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()

        def read(self):
            # return the frame most recently read
            return self.frame
        def stop(self):
            # indicate that the thread should be stopped
            self.stopped = True
```

Appendix B

Screenshots

Figure B.1:Dashboard

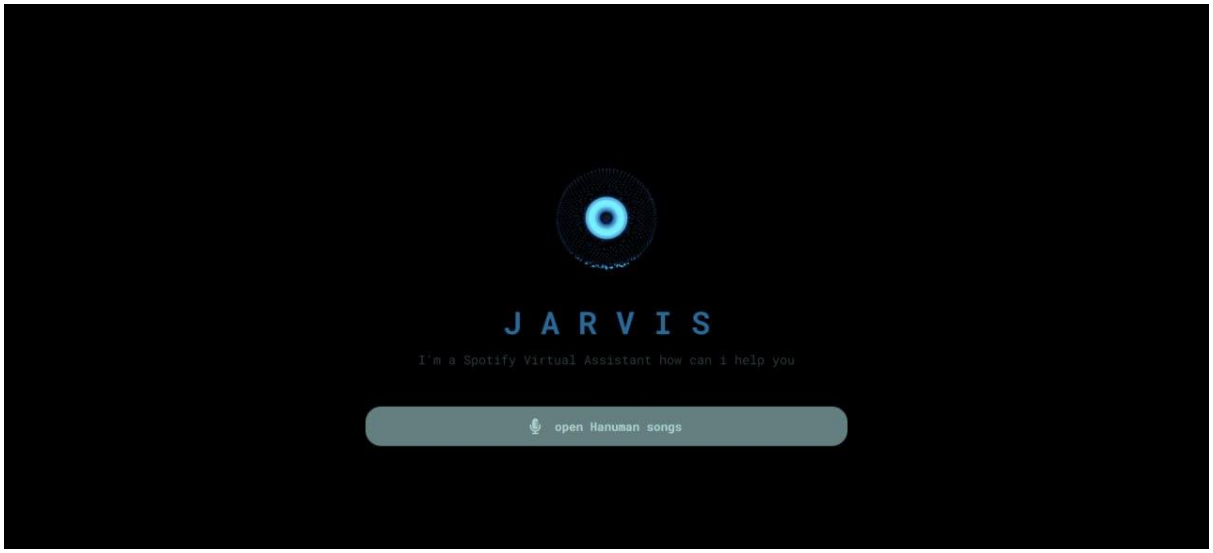

Figure B.2: Voice Assistant

Voice assistant: The Spotify Web API is a RESTful API that provides access to various Spotify resources, such as playlists, tracks, albums, and artists. It allows developers to retrieve and manipulate data from the Spotify music catalog.

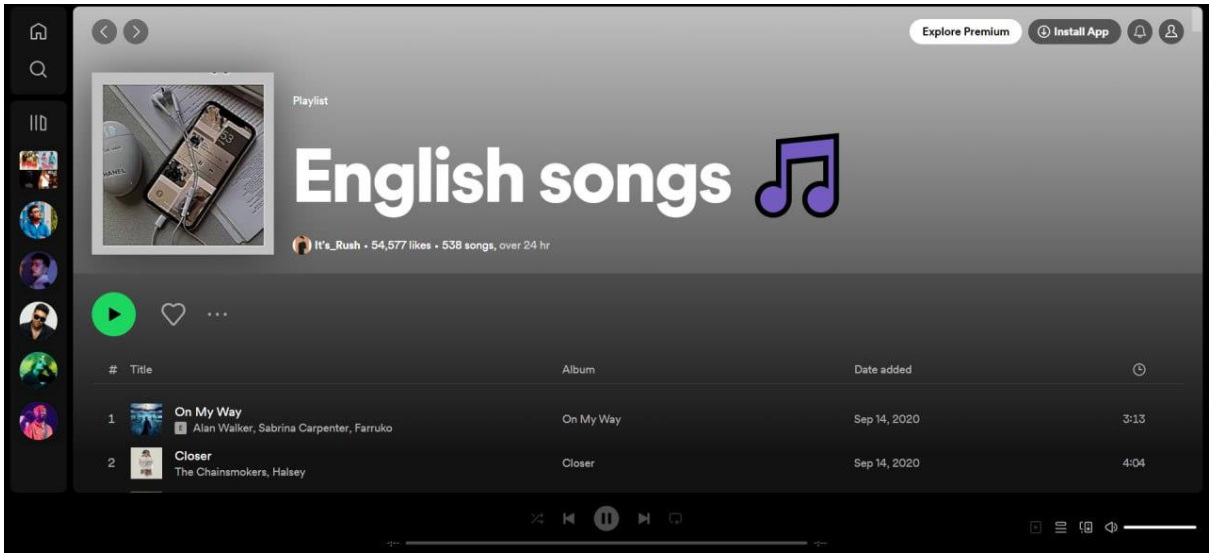

Figure B.3: Songs

We will explore how to analyze emotions in Spotify playlists using Python. We will use the Spotipy library, which is a Python wrapper for the Spotify Web API, to fetch playlist data and extract information about the tracks in the playlist.

Spotipy: Spotipy is a Python library that allows us to interact with the Spotify Web API. It provides methods to authenticate with Spotify, fetch playlist data and extract information about the tracks in the playlist.

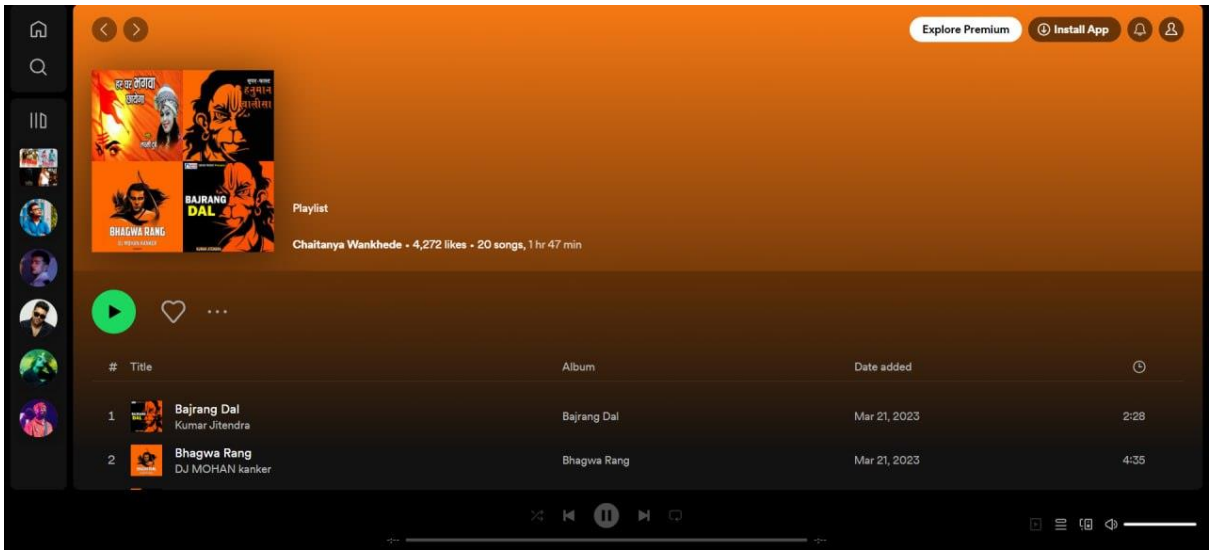

Figure B.4: Songs

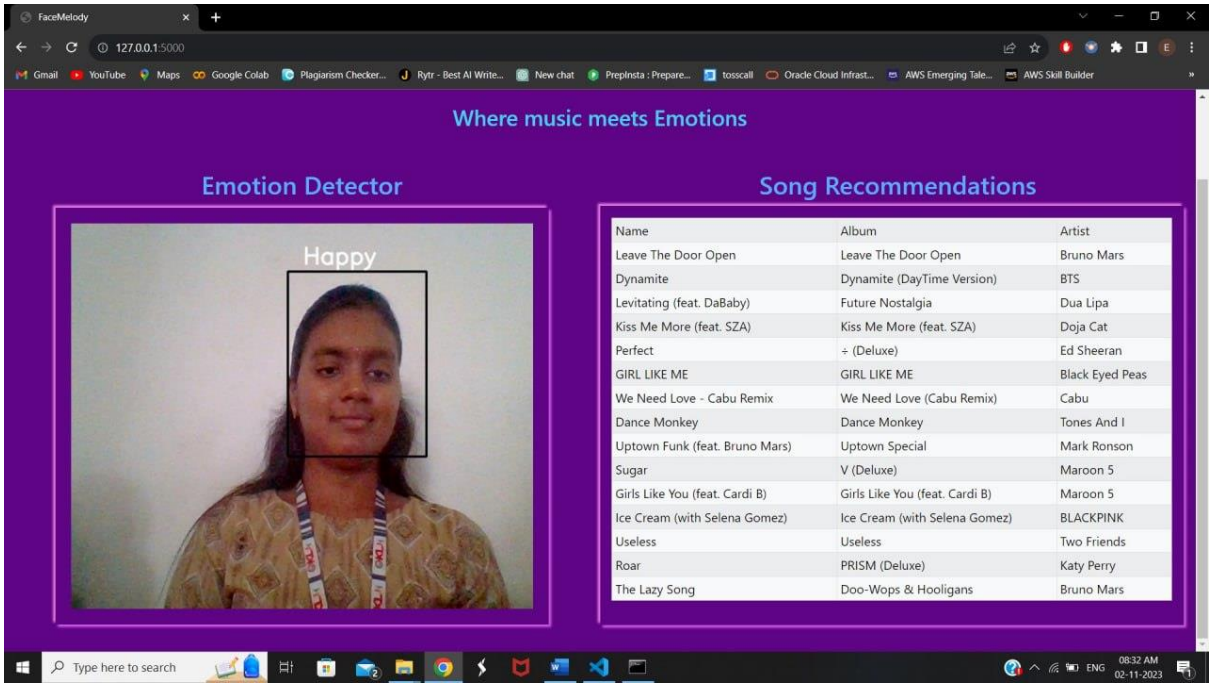

Figure B.5: Happy Face

The WebcamVideoStream class provides a convenient way to stream video from a web camera using a separate thread in Python. By utilizing a separate thread, we can perform other tasks concurrently while continuously reading frames from the video stream. This class can be useful in various applications that require real-time video processing or analysis.

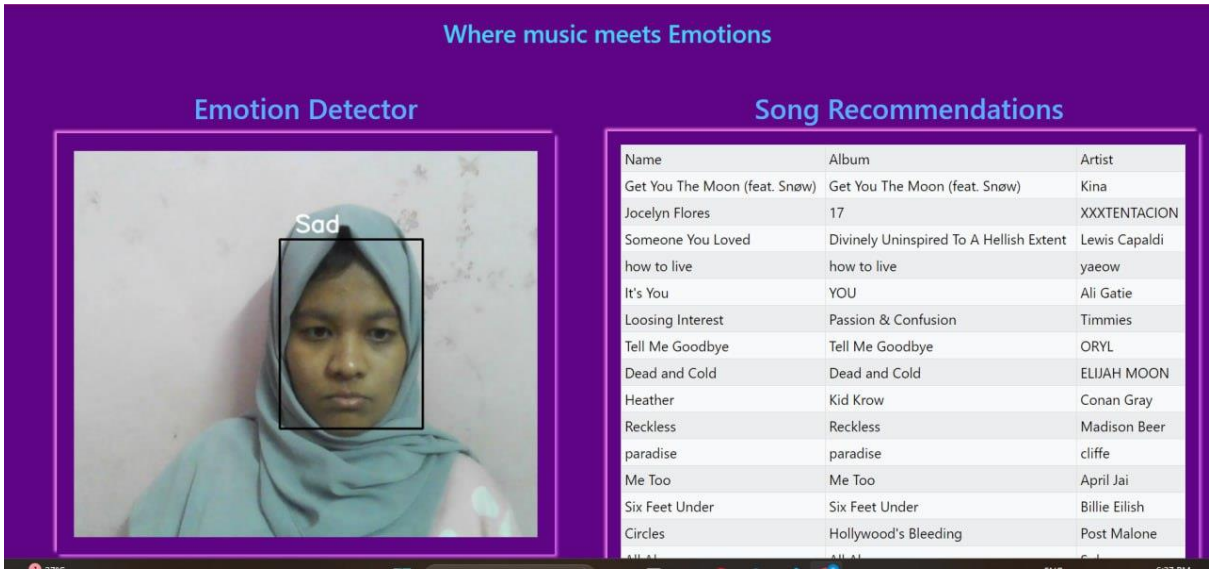

Figure B.6: Sad face

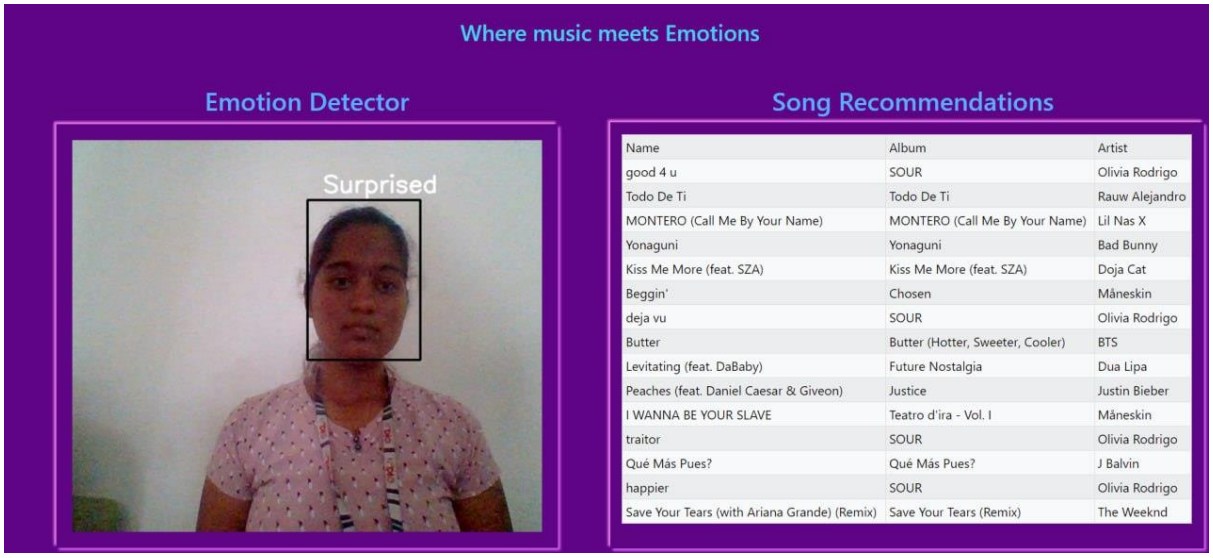

Figure B.7: Surprised Face

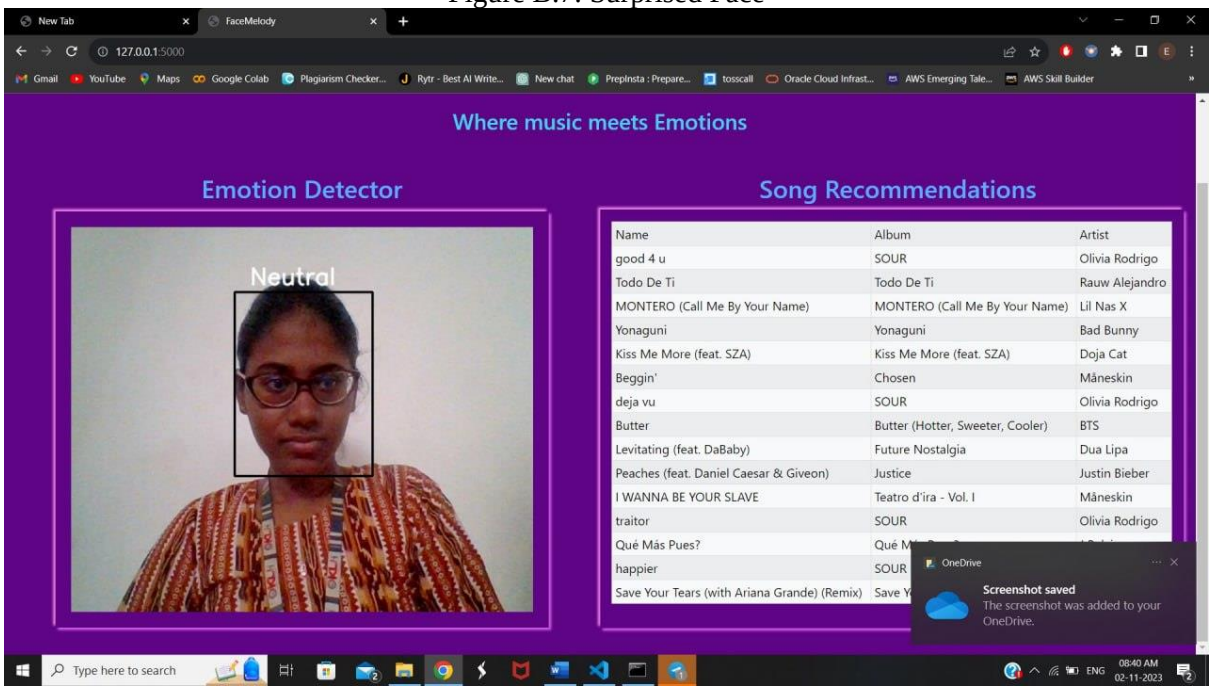

Figure B.8: Neutral Face

The code defines the "VideoCamera" class, which is responsible for capturing video frames, performing emotion recognition, and recommending music. The class uses the face cascade classifier to detect faces, extracts the facial region of interest (ROI), and feeds it to the emotion model for prediction. It also retrieves the corresponding music recommendations from the CSV files based on the detected emotion.

Appendix C

Data sets used in the project

We have used Facial Emotion Recognition (FER-2013) Dataset. The dataset is split into testing and training sets. 80% of the data is used in the training set to train the model. 20% of the data is available in the testing set. Both training and testing sets are again classified into 7 different folders based on emotions.

Figure C.1: Dataset

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples and the public test set consists of 3,589 examples.