1. Introduction

1.1 Project Description

People frequently use their faces to express their emotions.Facial expression recognition software is a technological advancement that uses algorithms for biometric systems to extract information from human facial expressions, including the emotion they convey. Better described as an emotion analysis system, this technology is capable of identifying a variety of human expressions, including happy, sad, angry, disgust, etc. Nonverbal communication cues are conveyed by facial expressions and other gestures, which are crucial in interpersonal interactions. Face reading can therefore deliver unfiltered, unbiased emotional responses as data because it gathers and analyzes information from an image or video feed. Only 7% of actual information is transmitted verbally, according to psychologist Mehrabian's research, and 38% is passed through language's auxiliary mechanisms, such as body language.

As the current generation is more inclined toward multimedia for their entertainment—which is actually a very key factor for the future aspects of the system—the recommendation system uses facial expression recognition for movies and music.

There is a genre assigned to things like movies and music/songs that aids in connecting them to the various facial expressions produced by a human face. The algorithm will therefore include steps like face identification, facial expression recognition, asking the user whether they want the system to return a movie or music/song, returning a random entity that corresponds to the genre that is associated with the expression, or returning a list of the same and a hyperlink to a webpage where the user can get more information.

Automated emotion detection in multimedia entities like music or movies is expanding quickly thanks to technological advancements in digital signal processing and other efficient feature extraction algorithms, and this recommendation system can play a significant role in many potential applications like human-computer interaction systems, music entertainment, and movie recommendations for theaters.

1.2 Existing Work

A few approaches have been put out and accepted to successfully group human emotions. The majority of techniques placed a strong emphasis on seven fundamental emotions that endure regardless of age, culture, or character.

Explains the benefits of using OpenCV, particularly the Adaboost algorithm, for face recognition. Combining a specific algorithm with the AdaBoost algorithm can be used to detect and recognise faces in complex color images. Additionally, it discusses the drawbacks of using a timer for face detection.

It is suggested that Support Vector Machines (SVM) be used as the main characterization method to rank eight facial expressions. The faces were identified using OpenCV's channels, and the color was changed to grayscale. The project also describes robotized constant coding of external appearances in continuous video streaming, which is practical for applications that accept frontal perspectives using a webcam.

The web camera or the hard circle itself was used to take the picture that needed to be prepared. The image is subjected to improvements, where a few mapping and upgrade procedures are connected to reestablish the image's necessary differentiation. The "one versus all" approach of SVM is maintained for preparation and arrangement in order to support multi-class characterization.

The process demonstrates how to interpret the mental state of the music/movie work using Thayer's model of mind-sets. Through trained neural systems, the edge level of a musical composition is resolved, and the emotion it evokes is felt.

1.3 Objectives

- To design an efficient user interface for multimedia.
- Technology implementation of machine learning.
- Providing movie or music fans with a modern digital platform
- Automating some of the actions a user must take.
- Providing users with entertainment options is crucial.
- Using expression as a source of ideas for amusement.

1.4 Purpose, Scope and Applicability

1.4.1 Purpose

Its purpose is to recommend multimedia to the user using facial expression. When recommending items to a user, such as movies or songs, our system analyzes his facial expression. Therefore, it is important to develop a recommendation system that uses less user data while still functioning well because a user's needs might not be related to his or her past but rather to the present, which is denoted by the user's expressions.

1.4.2 Scope

Future improvements could include adding compatibility for more emotions to the facial expression recognition-based entertainment recommendation system. Even a music player can be integrated into the system, eliminating the need for pre-defined suggested data since the music/movie will only play within the system itself.

1.4.3 Applicability

It can be applied to expand the system's library, it might also be connected to streaming services like Netflix, Amazon Prime, Spotify, and others.For future it can also be applicable as multimedia suggestions to passengers, as we can integrate this system with taxis.

1.5 Overview of the Report

Project Overview is to provide a Recommendation System to the people using their Facial Expression Sentiments.In the world where communication is one of the most important acts, facial expression is the means of non-verbal communication. Through this we can recommend Multimedia to Users. This Project is used in many places like integrating it with a car multimedia player for suggestions of music or movie according to facial expression of passenger or driver.

2. SYSTEM ANALYSIS & REQUIREMENTS

2.1 Problem Definition

The problem is introduced using Facial Expression to recommend a Multimedia to the User.Our System uses the users facial expression to recommend him entities like movies or songs. Hence creating a recommendation system that will require less user data and should still be able to work nicely as user requirements might not be related to his past but with the present that is signified by his/her expressions.

2.2 Requirement Specification

2.2.1. Functional Requirements

These are the specifications that the system must meet in order to satisfy the end user's basic needs. As a requirement of the contract, all of these functionalities must be built into the system. These are shown or stated as the input to be provided to the system, the operation carried out, and the expected output. They are essentially the user's requirements, which are evident in the finished product, directly.

Following are the functional requirements for our project :

2.2.1.1 Capturing Button

It Helps to keep an image of an user and detect its emotions from different kinds of moods our facial expressions can show.

2.2.1.2 Real Time Detection

A real-time video system that can identify activities in real-time should be able to connect with the system.

2.2.1.3 Emotion recognition

The system should detect emotion from the user's real time expression.

2.2.1.4 Recommending Movies and Songs

The System will recommend movies and songs to the user based on emotion detection from facial expressions.

2.2.2 Non-functional Requirements

In essence, these are the requirements for quality that the system must meet in accordance with the project contract. Depending on the project, these factors may be prioritized differently or applied to a different degree. Additionally known as non-behavioral requirements.

2.2.2.1 Usability

2.2.2.1.1 Graphical User Interface

The appearance and user experience of every web page must be consistent. A digital image of that time period must be provided for each activity the system detects.

The user interface would look into a web-cam showing facial expression in real-time and detecting faces.

2.2.2.2 Performance

The application must be web-based and run from a web server. Initial loading of the product will take some time, depending on the speed of the internet connection and the media source.
The user-provided data stream will determine how well the system performs.

2.2.2.3 Security

The possibility of application functionality becoming insecure will be decreased because it will only be permitted in secure networks. On the other hand, interoperability with cameras, colors, and platforms is crucial because the system works in a real-time application environment. In situations of command transfer using particular linked devices or wireless connections, the proper port assignment would also be an important factor to consider.

2.2.2.6 Legal, Copyright, and Other Notices

Project must display the disclaimers, copyright, wordmark, trademark ,and product warranties.

2.2.2.7 Applicable Standards

It shall be as per the industry standard.

2.3 System Requirements

2.3.1 Hardware Requirements

1. A decent camera for capturing facial expressions.
2. A system with a minimum of 8 GB of RAM and a processor i5 or above processor.

2.3.2 Software Requirements

1. Python 3.8 or above.
2. Tensorflow (Latest version or older version if required).
3. Knowledge of HTML, CSS, and Javascript.
4. Knowledge of Flask connection of back-end and front-end.
5. Knowledge of Neural Networks.

3. SYSTEM DESIGN

The process of defining a system's components, including modules, architecture, components, their interfaces, and data, based on the given requirements is known as systems design. A business or organization's unique goals and requirements are satisfied through the process of identifying, developing, and creating systems.

A system needs a systematic approach to be cohesive and functional. It is necessary to use a bottom-up or top-down method to account for all the system's connected factors. In order to express information and knowledge in a system structure that is governed by a standardized set of rules and definitions, a designer employs modeling languages. Both graphical and textual modeling languages can be used to specify the designs.

3.1 SYSTEM ARCHITECTURE

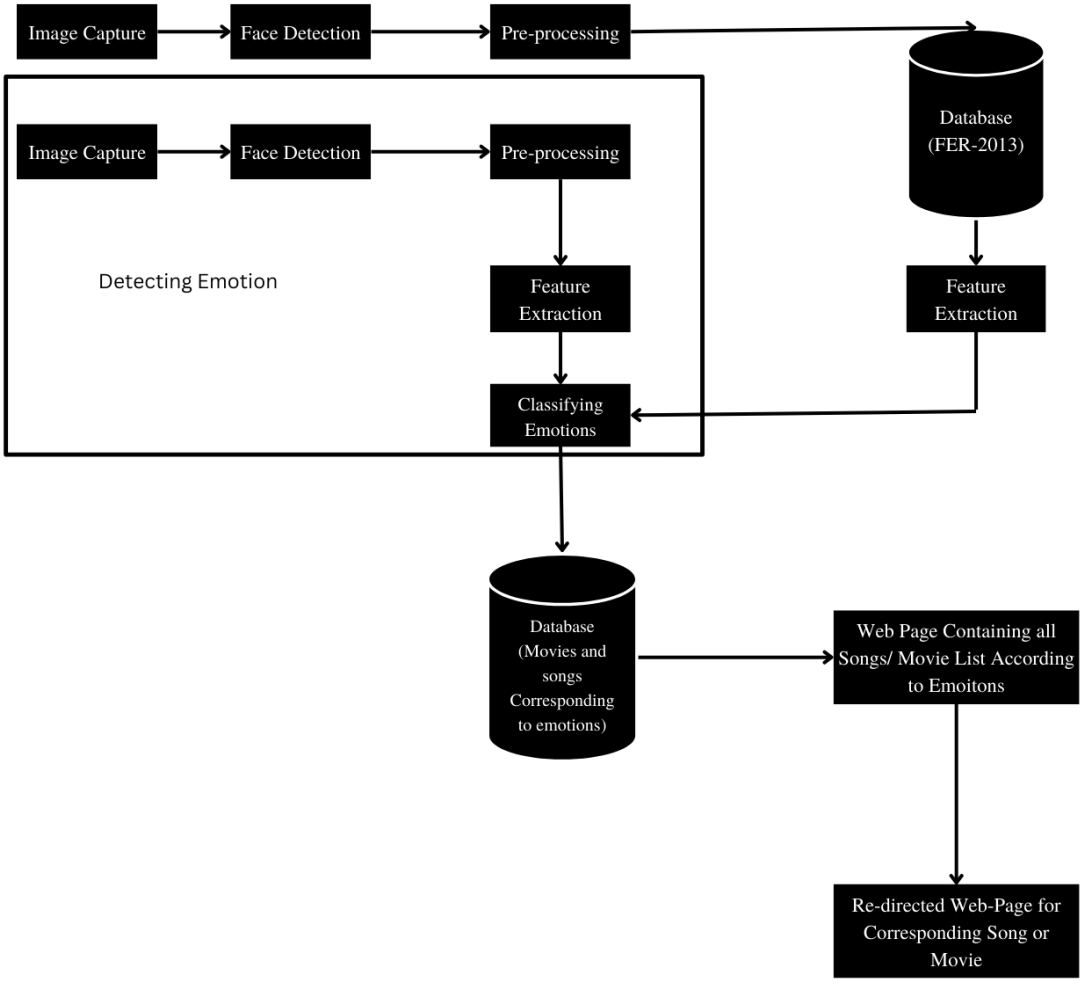

Fig 3.1 System Architecture for System

Here, Initially we take an image, detect a face from the image then feature selection is performed.Once an image is processed it's Emotions are classified. Once Emotion is classified the Movies/Songs associated with corresponding emotions are listed on the website then redirected to desired selection from the list.

3.2 System Flow-Chart

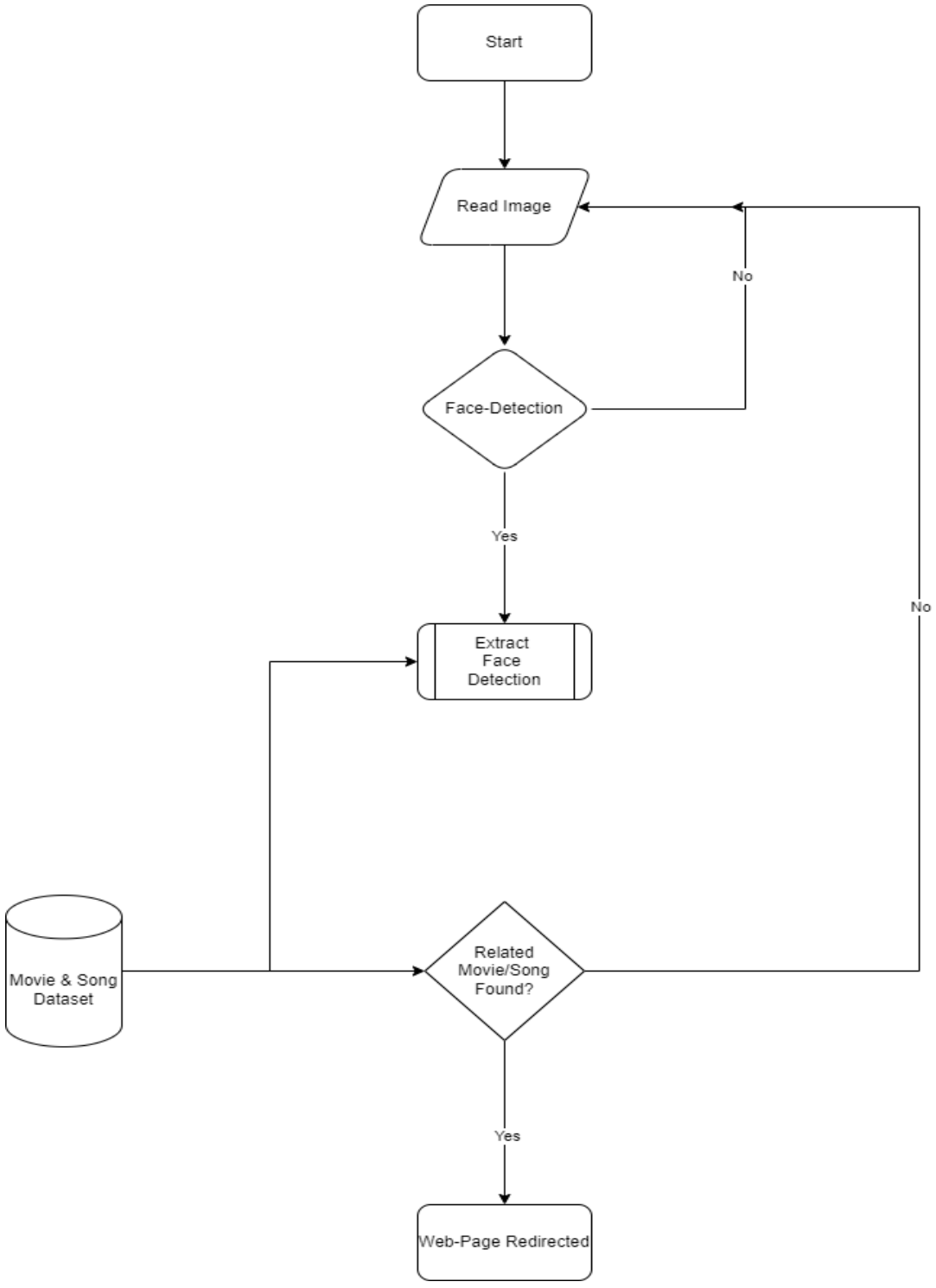

Fig 3.2 Flowchart for System

3.3 USE CASE DIAGRAM

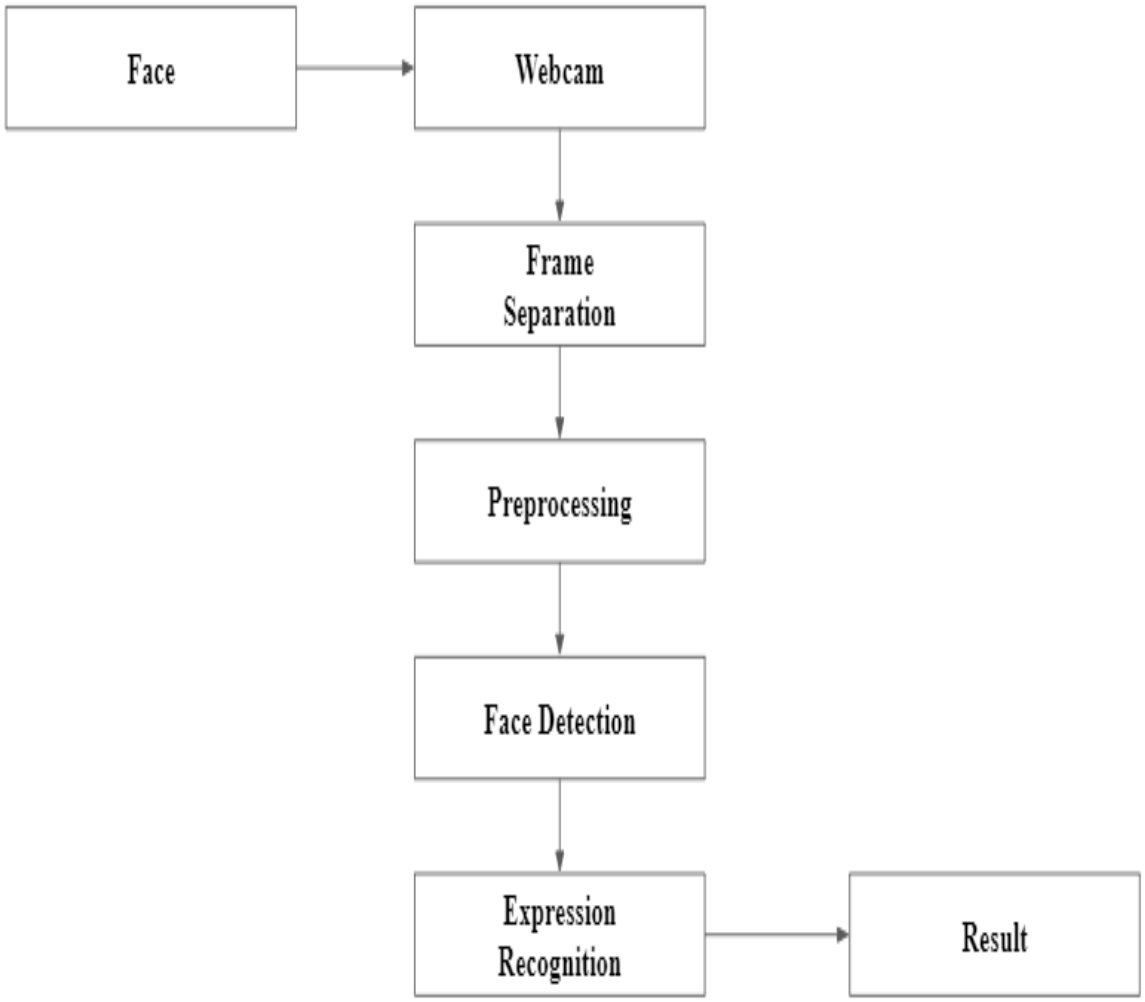

Fig 3.3 Use Case Diagram

The diagram above illustrates how the system and its main features function for both the system and the user. Webcam access is granted by users. The next step is to extract the characteristics and behaviors, which results in the recognition of emotions. After that, a selection of music and movies related to the identified emotions will be displayed.

3.4 DATA FLOW DIAGRAM

- Level 0

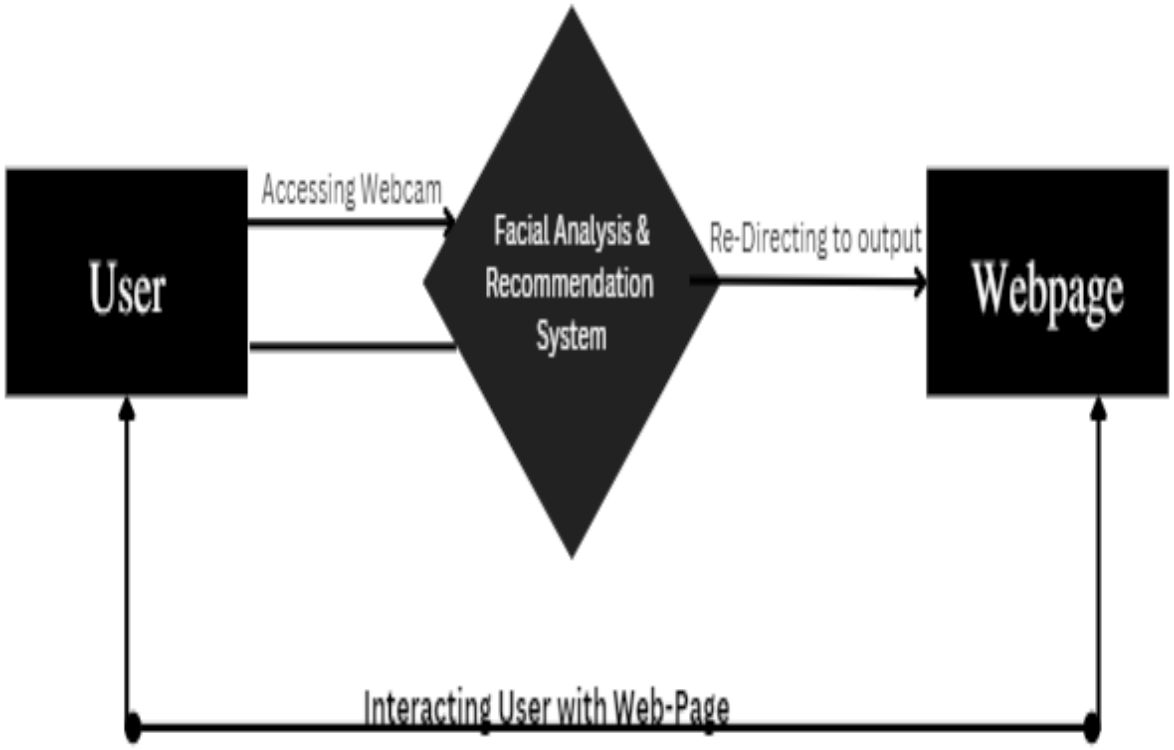

Fig 3.4 Data Flow Diagram Level-0

- Level 1

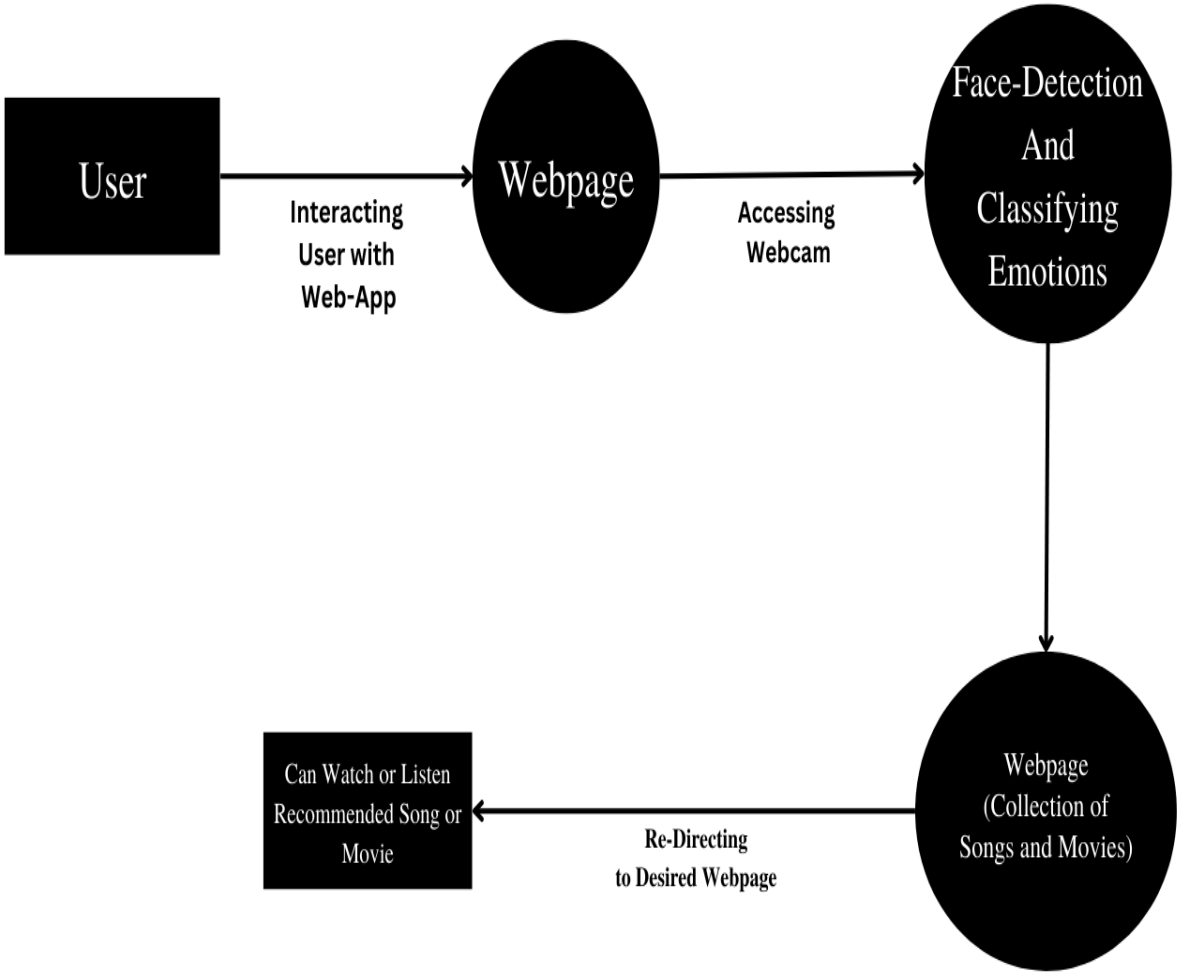

Fig 3.5 Data Flow Diagram Level-1

- Level 2

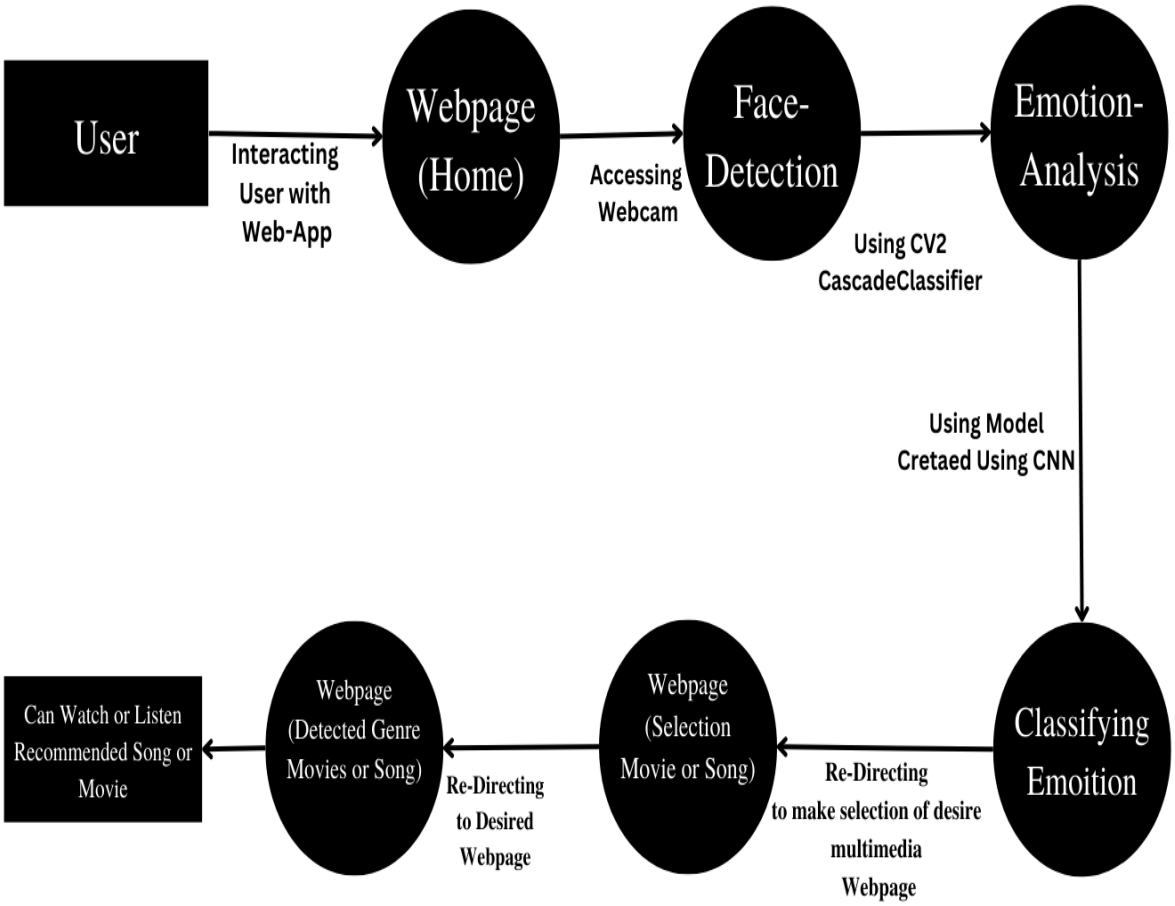

Fig 3.6 Data Flow Diagram Level-2

3.5 INTERFACE DESIGN

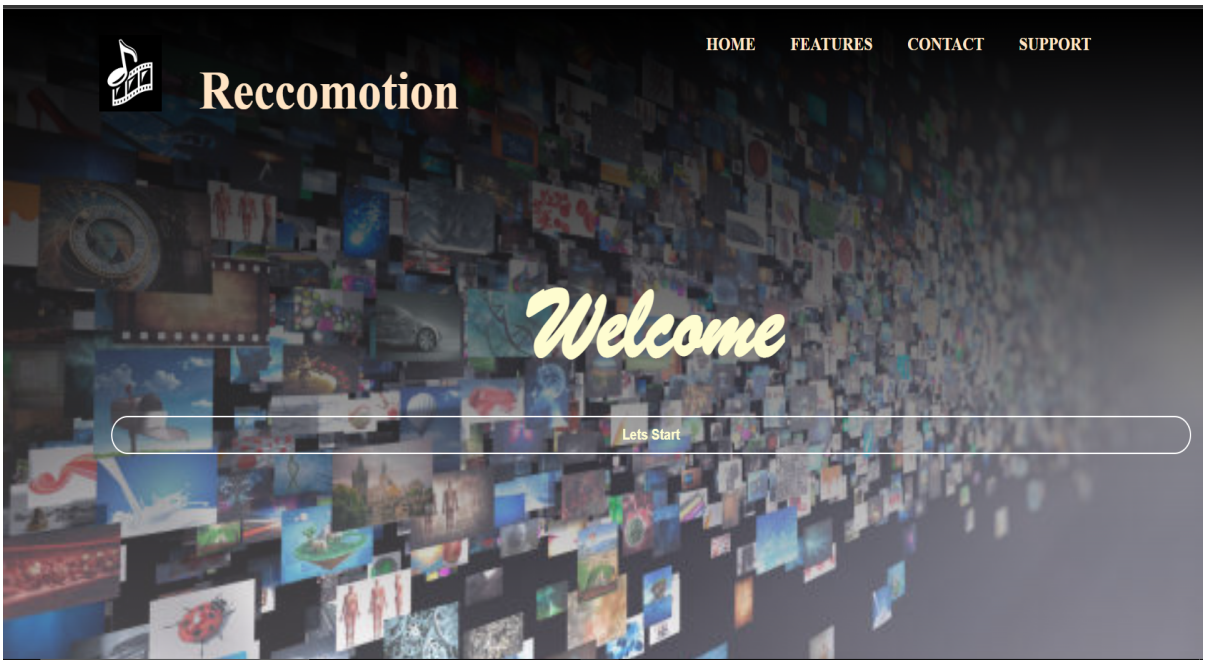

Fig 3.7 Home Page

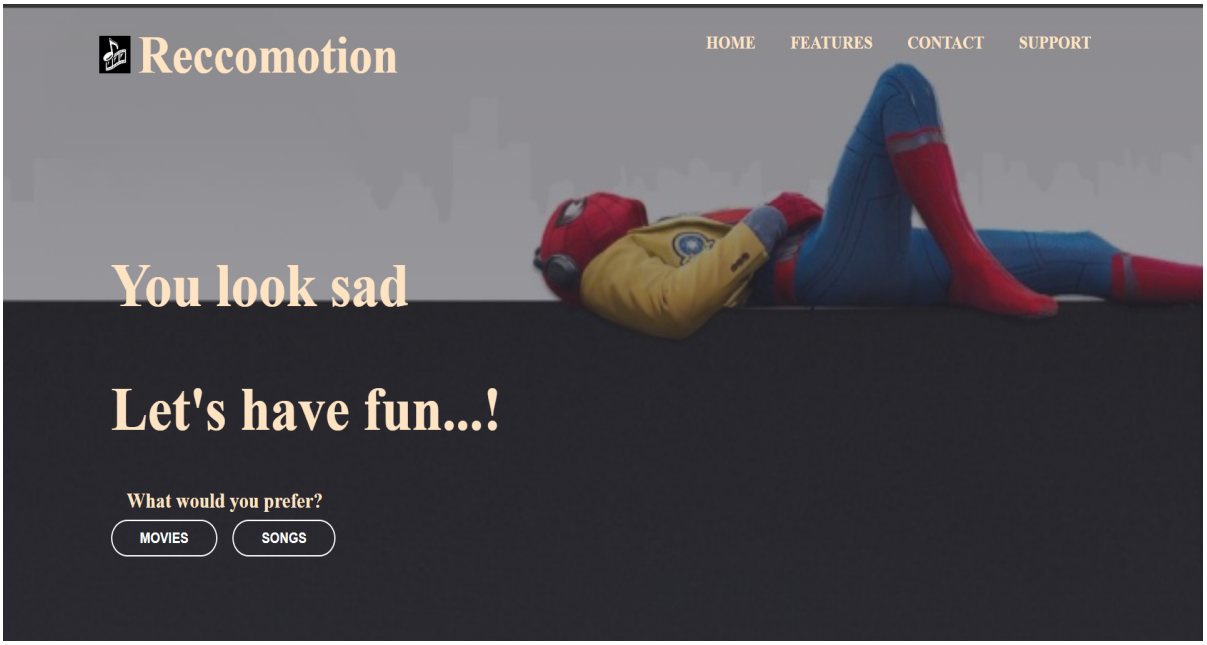

Fig 3.8 Selection Page

3.5 INTERFACE DESIGN

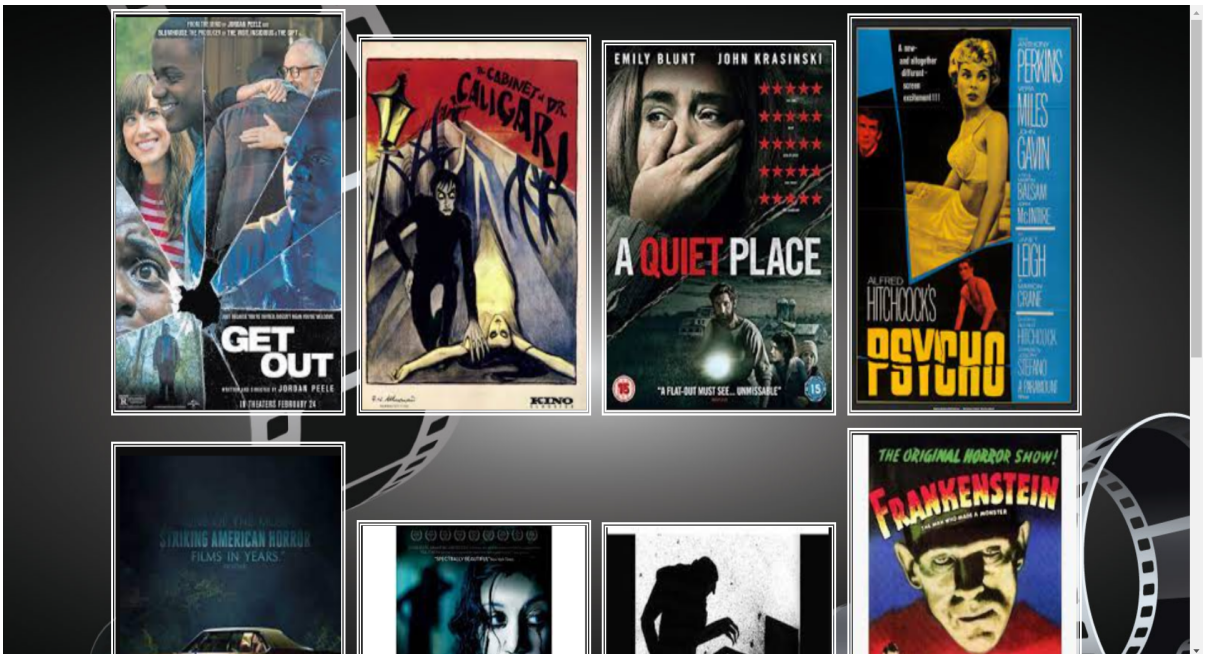

Fig 3.9 Movies List Page

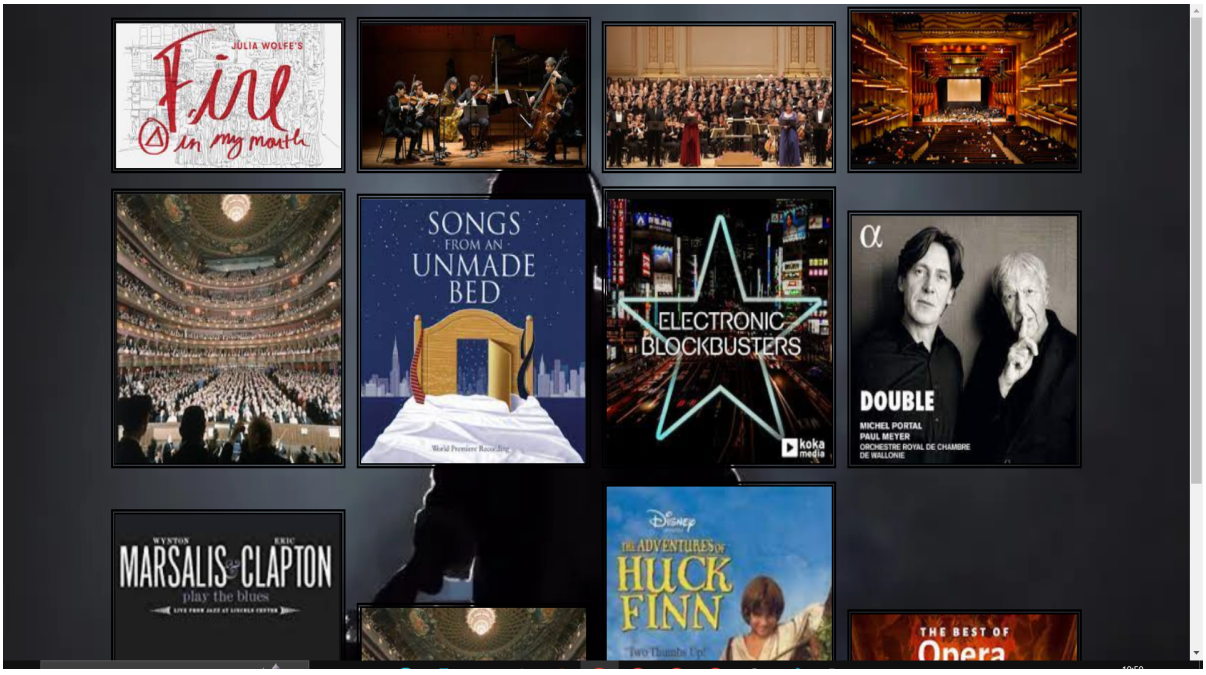

Fig 3.10 Songs List Page

3.6 DATASET DESCRIPTION

3.6.1 FER 2013 Dataset

The FER2013 database was introduced during the ICML 2013 Challenges in Representation Learning. FER2013 is a large-scale and unconstrained database collected automatically by the Google image search API. It is an open source dataset with 35.887 grayscale, 48x48 sized face images, and different labels such as:

- 4593 images- Angry
- 547 images- Disgust
- 5121 images- Fear
- 8989 images- Happy
- 6077 images- Sad
- 4002 images- Surprise
- 6198 images- Neutral

It contains 28,709 training images, 3,589 validation images and 3,589 test images.

Link : https://www.kaggle.com/datasets/msambare/fer2013

Following images to show all possible emotions are:

Fig 3.11 Angry Emotion from Dataset

Fig 3.12 Disgust Emotion from Dataset

Fig 3.10 Fear Emotion from Dataset

Fig 3.10 Happy Emotion from Dataset

Fig 3.10 Neutral Emotion from Dataset

Fig 3.10 Sad Emotion from Dataset

Fig 3.10 Surprise Emotion from Dataset

3.6.2 Movies and Songs Dataset

The dataset is scraped from the websites of spotify and imdb. Dataset contains songs of genre funk, classical, hip hop, metal, dance, rock, and top 200. It also contains movies with genre Comedy, motivational, Action, Adventure, Animation, drama, horror, suspense/mystery, romance, fiction/scifi.

4. IMPLEMENTATION

4.1 Implementation Approaches

The implementation of the system is done in on various steps:

1. Image Data Preparation
2. Convolution Neural Network Model Building
3. Real Time Face Detection
4. Facial Emotion Classification
5. Deployment Using Flask
6. Recommendation of Movies/Songs

4.1.1 Image Data Preparation

All Image Data Preparation is done using Image Data Generator.

Step1: Understanding Our Data

- All image data in the folder is divided into 7 classes of emotions which are fear, sad, disgust, happy, angry, surprise and neutral.
- Converting all images to a common size, which is (224,224,3).

Step2 : Visualizing Data

- Visualization of Data-Set on Data which is prepared using Data Generator.

4.1.2 Convolution Neural Network

- Creating a Convolution neural network.
- This CNN contains 24 layers.
- CNN network contains Convolution Layer, Batch Normalization, Max Pooling Layer, Flatten Layer, Dense Layer, Dropouts and Activation Function.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input (InputLayer)          [(None, 48, 48, 1)]       0

 conv1_1 (Conv2D)            (None, 48, 48, 64)        640

 batch_normalization (BatchN  (None, 48, 48, 64)       256
 ormalization)

 conv1_2 (Conv2D)            (None, 48, 48, 64)        36928

 batch_normalization_1 (Batc  (None, 48, 48, 64)       256
 hNormalization)

 pool1_1 (MaxPooling2D)      (None, 24, 24, 64)        0

 drop1_1 (Dropout)           (None, 24, 24, 64)        0

 conv2_1 (Conv2D)            (None, 24, 24, 128)       73856

 batch_normalization_2 (Batc  (None, 24, 24, 128)      512
 hNormalization)

...
Trainable params: 13,103,431
Non-trainable params: 7,936
_________________________________________________________________
None
```

Fig 4.1 Convolution Neural Network Model Building Summary

4.1.3 Real Time Face Detection

Using Haar Cascade Face we detect faces from real time video. Working mechanism of the haar cascade is its sum of all the image pixels lying in the darker area of the haar feature and the sum of all the image pixels lying in the lighter area of the haar feature. And then find out their differences. Now if the image has an edge separating dark pixels on the right and light pixels on the left, then the haar value will be closer to 1. That means, we say that there is an edge detected if the haar value is closer to 1.

So, the nose, ears, lips, and eyes are detected using this method.

4.1.4 Facial Emotion Classification

Once a model is completed we save our model and use this model for real-time facial emotion recognition. Using Haar Cascade we detect faces and corresponding to facial expressions and emotions are stored frame by frame in an array. Then a dominating emotion is taken and classified.

4.1.5 Deployment Using Flask

Now Using Flask we connect our Backend and Frontend. Model is deployed using a flask.

4.1.6 Recommendation of Movies/Songs

Once everything is done according to emotions we will be redirected to a web page to choose an option between multimedia like a movie or song. After we choose a movie/song we will be provided with a list of movies/songs corresponding to detected emotion.

4.2 Coding Details:

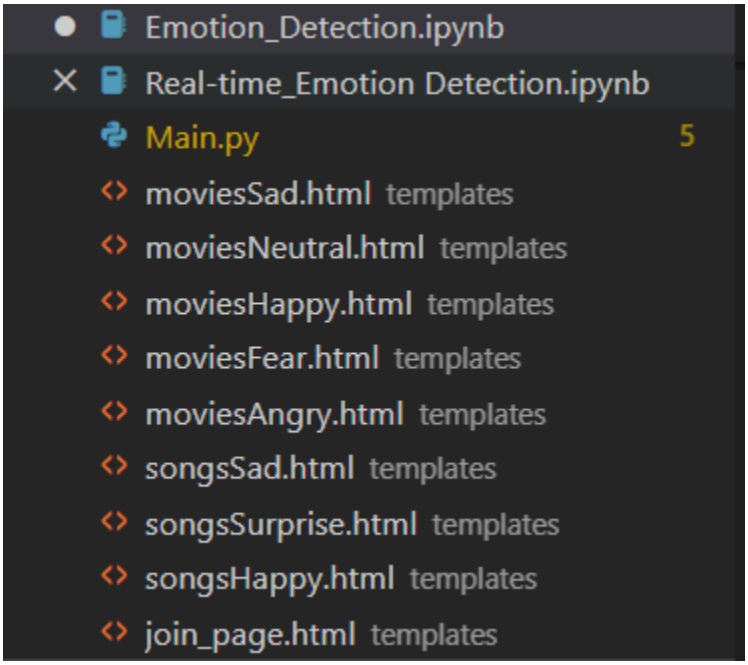

Fig 4.2 List of files for System

Project Name : Reccomotion

Convolution Neural Network Model Building : Emotion_Detection.ipynb

Real-Time Emotion Detection : Real-Time Emotion Detection.ipynb

Main Project Flask File : Main.py

Corresponding HTML Pages to Corresponding Emotions.

4.2.1 Coding Screenshots

4.2.1.1 Convolution Neural Network Model Building

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from collections.abc import MutableMapping

from keras.layers import Flatten, Dense
from keras.models import Model
from keras.preprocessing.image import ImageDataGenerator , img_to_array, load_img
from keras.applications.mobilenet import MobileNet, preprocess_input
from keras.losses import categorical_crossentropy
import cv2
import tensorflow as tf
from tensorflow.keras.applications import VGG19
import warnings
warnings.simplefilter(action='ignore', category=RuntimeWarning)
#import weightwatcher as ww
import torchvision
import torchvision.models as models
```

✓ 36.7s Python

Fig 4.3 Importing Required Modules for Building Convolution Neural Network Model

Building our Model To train the data

Preparing our data using data generator

```
train_datagen = ImageDataGenerator(
    zoom_range = 0.2,
    shear_range = 0.2,
    horizontal_flip=True,
    rescale = 1./255
)

train_data = train_datagen.flow_from_directory(directory= "train",
                                               target_size=(224,224),
                                               batch_size=32,
                                              )


train_data.class_indices
```

[6] ✓ 3.4s Python

Fig 4.4 Preparing Data Using Data Generator

4.2.1 Coding Screenshots

4.2.1.1 Convolution Neural Network Model Building

```
Found 28709 images belonging to 7 classes.

{'angry': 0,
 'disgust': 1,
 'fear': 2,
 'happy': 3,
 'neutral': 4,
 'sad': 5,
 'surprise': 6}
```

Fig 4.5 Facial Emotions Present in Dataset

visualizaing the data that is fed to train data gen

```python
# to visualize the images in the traing data denerator

t_img , label = train_data.next()

#-----------------------------------------------------------------------
# function when called will prot the images
def plotImages(img_arr, label):
  """
  input  :- images array
  output :- plots the images
  """
  count = 0
  for im, l in zip(img_arr,label) :
    plt.imshow(im)
    plt.title(im.shape)
    plt.axis = False
    plt.show()

    count += 1
    if count == 10:
      break

#-----------------------------------------------------------------------
# function call to plot the images
plotImages(t_img, label)
```

[17] ✓ 5.3s Python

Fig 4.6 Code to Visualize Prepared Data

4.2.1 Coding Screenshots

Fig 4.7 Visualizing Prepared Data for different Emotions

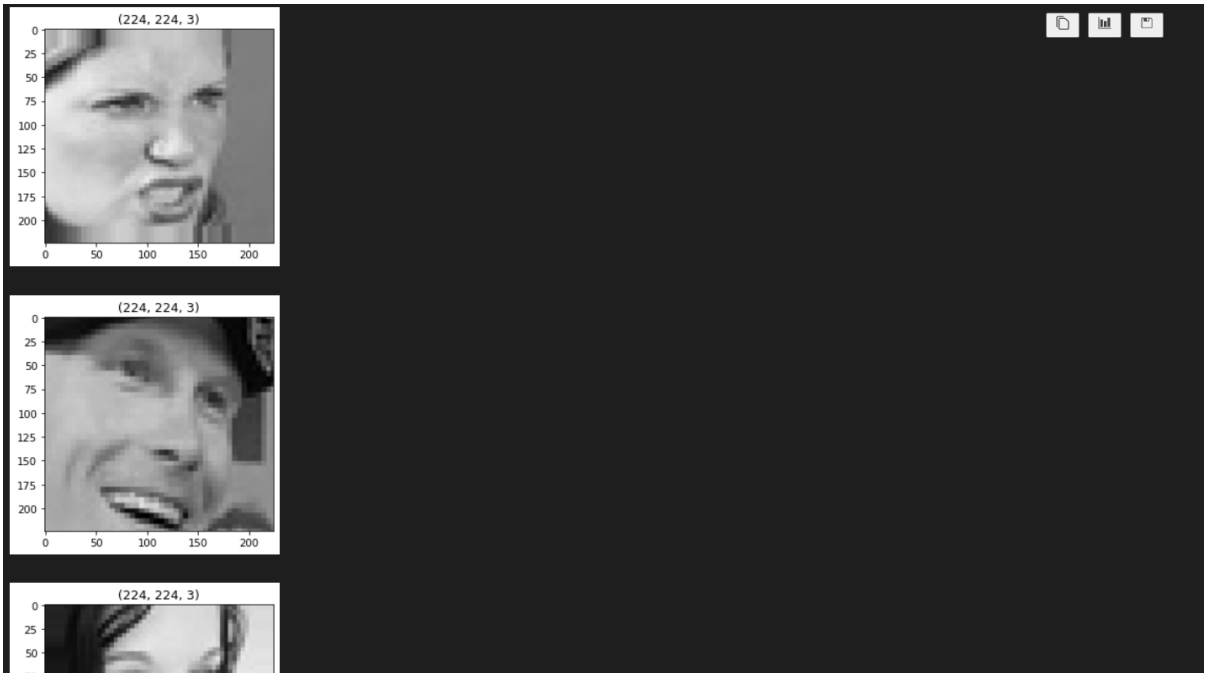

Fig 4.8 Visualizing Prepared Data for different Emotions

4.2.1 Coding Screenshots

```python
def FER_Model(input_shape=(48,48,1)):
    # first input model
    visible = Input(shape=input_shape, name='input')
    num_classes = 7
    #the 1-st block
    conv1_1 = Conv2D(64, kernel_size=3, activation='relu', padding='same', name = 'conv1_1')(visible)
    conv1_1 = BatchNormalization()(conv1_1)
    conv1_2 = Conv2D(64, kernel_size=3, activation='relu', padding='same', name = 'conv1_2')(conv1_1)
    conv1_2 = BatchNormalization()(conv1_2)
    pool1_1 = MaxPooling2D(pool_size=(2,2), name = 'pool1_1')(conv1_2)
    drop1_1 = Dropout(0.3, name = 'drop1_1')(pool1_1)#the 2-nd block
    conv2_1 = Conv2D(128, kernel_size=3, activation='relu', padding='same', name = 'conv2_1')(drop1_1)
    conv2_1 = BatchNormalization()(conv2_1)
    conv2_2 = Conv2D(128, kernel_size=3, activation='relu', padding='same', name = 'conv2_2')(conv2_1)
    conv2_2 = BatchNormalization()(conv2_2)
    conv2_3 = Conv2D(128, kernel_size=3, activation='relu', padding='same', name = 'conv2_3')(conv2_2)
    conv2_2 = BatchNormalization()(conv2_3)
    pool2_1 = MaxPooling2D(pool_size=(2,2), name = 'pool2_1')(conv2_3)
    drop2_1 = Dropout(0.3, name = 'drop2_1')(pool2_1)#the 3-rd block
    conv3_1 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_1')(drop2_1)
    conv3_1 = BatchNormalization()(conv3_1)
    conv3_2 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_2')(conv3_1)
    conv3_2 = BatchNormalization()(conv3_2)
    conv3_3 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_3')(conv3_2)
    conv3_3 = BatchNormalization()(conv3_3)
    conv3_4 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv3_4')(conv3_3)
    conv3_4 = BatchNormalization()(conv3_4)
    pool3_1 = MaxPooling2D(pool_size=(2,2), name = 'pool3_1')(conv3_4)
    drop3_1 = Dropout(0.3, name = 'drop3_1')(pool3_1)#the 4-th block
    conv4_1 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv4_1')(drop3_1)
    conv4_1 = BatchNormalization()(conv4_1)
    conv4_2 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv4_2')(conv4_1)
    conv4_2 = BatchNormalization()(conv4_2)
    conv4_3 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv4_3')(conv4_2)
    conv4_3 = BatchNormalization()(conv4_3)
    conv4_4 = Conv2D(256, kernel_size=3, activation='relu', padding='same', name = 'conv4_4')(conv4_3)
```

Fig 4.9 Convolution Neural Network Model Building

```python
    conv4_4 = BatchNormalization()(conv4_4)
    pool4_1 = MaxPooling2D(pool_size=(2,2), name = 'pool4_1')(conv4_4)
    drop4_1 = Dropout(0.3, name = 'drop4_1')(pool4_1)

    #the 5-th block
    conv5_1 = Conv2D(512, kernel_size=3, activation='relu', padding='same', name = 'conv5_1')(drop4_1)
    conv5_1 = BatchNormalization()(conv5_1)
    conv5_2 = Conv2D(512, kernel_size=3, activation='relu', padding='same', name = 'conv5_2')(conv5_1)
    conv5_2 = BatchNormalization()(conv5_2)
    conv5_3 = Conv2D(512, kernel_size=3, activation='relu', padding='same', name = 'conv5_3')(conv5_2)
    conv5_3 = BatchNormalization()(conv5_3)
    conv5_4 = Conv2D(512, kernel_size=3, activation='relu', padding='same', name = 'conv5_4')(conv5_3)
    conv5_3 = BatchNormalization()(conv5_3)
    pool5_1 = MaxPooling2D(pool_size=(2,2), name = 'pool5_1')(conv5_4)
    drop5_1 = Dropout(0.3, name = 'drop5_1')(pool5_1)#Flatten and output
    flatten = Flatten(name = 'flatten')(drop5_1)
    ouput = Dense(num_classes, activation='softmax', name = 'output')(flatten)# create model
    model = Model(inputs =visible, outputs = ouput)
    # summary layers
    print(model.summary())

    return model
```

[18] ✓ 0.7s Python

```python
model = FER_Model()
opt = Adam(lr=0.0001, decay=1e-6)
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy'])
```

Fig 4.10 Model Defining

4.2.1 Coding Screenshots

```python
num_epochs = 100
history = model.fit_generator(train_flow,
                              steps_per_epoch=len(X_train) / batch_size,
                              epochs=num_epochs,
                              verbose=1,
                              validation_data=test_flow,
                              validation_steps=len(X_test) / batch_size)
```

```
C:\Users\zaida\AppData\Local\Temp\ipykernel_13084\2811330767.py:2: UserWarning: `Model.fit_generator` is deprecated and will be removed in a
future version. Please use `Model.fit`, which supports generators.
  history = model.fit_generator(train_flow,

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/100
448/448 [==============================] - 2803s 6s/step - loss: 2.0212 - accuracy: 0.2185 - val_loss: 1.9799 - val_accuracy: 0.2494
Epoch 2/100
448/448 [==============================] - 2935s 7s/step - loss: 1.7899 - accuracy: 0.2510 - val_loss: 1.9168 - val_accuracy: 0.2536
Epoch 3/100
448/448 [==============================] - 2891s 6s/step - loss: 1.7564 - accuracy: 0.2703 - val_loss: 1.8781 - val_accuracy: 0.2689
Epoch 4/100
448/448 [==============================] - 2806s 6s/step - loss: 1.7087 - accuracy: 0.3092 - val_loss: 1.9568 - val_accuracy: 0.2853
Epoch 5/100
448/448 [==============================] - 2849s 6s/step - loss: 1.6428 - accuracy: 0.3549 - val_loss: 1.8562 - val_accuracy: 0.3561
Epoch 6/100
448/448 [==============================] - 2986s 7s/step - loss: 1.5790 - accuracy: 0.3816 - val_loss: 1.5355 - val_accuracy: 0.4255
Epoch 7/100
448/448 [==============================] - 2829s 6s/step - loss: 1.5122 - accuracy: 0.4151 - val_loss: 1.4837 - val_accuracy: 0.4425
Epoch 8/100
448/448 [==============================] - 2869s 6s/step - loss: 1.4466 - accuracy: 0.4447 - val_loss: 1.6566 - val_accuracy: 0.4369
Epoch 9/100
448/448 [==============================] - 2967s 7s/step - loss: 1.3844 - accuracy: 0.4721 - val_loss: 1.3961 - val_accuracy: 0.4798
```

Fig 4.11 Running Model with 100 Epoch

```python
# Loading the best fit model
from keras.models import load_model
model = load_model("best_model.h5")
```

```python
h =  hist.history
h.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

Fig 4.12 Saving Model

4.2.1 Coding Screenshots

4.2.1.2 Real-Time Emotion Classifier

```python
import os
import cv2
import numpy as np
from keras.preprocessing import image
import warnings
warnings.filterwarnings("ignore")
from keras.preprocessing.image import load_img, img_to_array
from keras.models import  load_model
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import statistics as st

# load model
model = load_model("best_model.h5")

face_haar_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
```

Python

Fig 4.13 Importing Required Modules for Real-Time Emotion Detection

```python
i=0
GR_dict={0:(0,255,0),1:(0,0,255)}

model = tf.keras.models.load_model('best_model.h5')
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

output=[]
cap = cv2.VideoCapture(0)

while (i<=300):
    ret, img = cap.read()
    #gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(img,1.05,5)

    for x,y,w,h in faces:

        face_img = img[y:y+h,x:x+w]

        resized = cv2.resize(face_img,(224,224))
        reshaped=resized.reshape(1, 224,224,3)/255
        predictions = model.predict(reshaped)

        # find max indexed array
        max_index = np.argmax(predictions[0])

        emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'neutral', 'surprise')
        predicted_emotion = emotions[max_index]
        output.append(predicted_emotion)



        cv2.rectangle(img,(x,y),(x+w,y+h),GR_dict[1],2)
        cv2.rectangle(img,(x,y-40),(x+w,y),GR_dict[1],-1)
        cv2.putText(img, predicted_emotion, (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
    i = i+1

```

Fig 4.14 Using Computer Vision Classifying Emotions for Real-Time

4.2.1 Coding Screenshots

```
    cv2.imshow('LIVE', img)
    key = cv2.waitKey(1)
    if key == 100:
        cap.release()
        cv2.destroyAllWindows()
        break
print(output)
cap.release()
cv2.destroyAllWindows()
final_output1 = st.mode(output)
```

Python

```
['fear', 'sad', 'sad', 'sad', 'sad', 'angry', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'neutral', 'sad', 'neutral', 'sad', 'sad',
'sad', 'fear', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'fear', 'sad', 'sad', 'sad',
'fear', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'fear', 'sad', 'sad',
'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'fear', 'sad', 'sad', 'sad', 'sad', 'fear', 'fear', 'sad', 'sad', 'fear', 'sad', 'sad', 'sad',
'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'sad', 'fear', 'sad', 'sad', 'sad', 'sad', 'fear', 'sad', 'fear', 'fear', 'fear', 'sad', 'fear',
'fear', 'fear', 'fear', 'happy', 'sad', 'happy', 'sad']
```

```
final_output = st.mode(output)
final_output
```

Fig 4.14 Saving Emotions Frame by Fame in an array and taking Dominant Emotion

4.2.1.3 Deployment Using Flask

```
from __future__ import division, print_function
#import sys
import os
import cv2
#import re
import numpy as np
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from flask import Flask, request, render_template
from werkzeug.utils import secure_filename
import statistics as st


app = Flask(__name__)

@app.route("/")
def home():
    return render_template("Home.html")

@app.route('/camera', methods = ['GET', 'POST'])
def camera():
    i=0

    GR_dict={0:(0,255,0),1:(0,0,255)}
    model = tf.keras.models.load_model('best_model.h5')
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
    output=[]
    cap = cv2.VideoCapture(0)
    while (i<=50):
        ret, img = cap.read()
        faces = face_cascade.detectMultiScale(img,1.05,5)

        for x,y,w,h in faces:

            face_img = img[y:y+h,x:x+w]
```

Fig 4.16 Importing Required Modules and setting up flask

4.2.1 Coding Screenshots

4.2.1.3 Deployment Using Flask

```
            face_img = img[y:y+h,x:x+w]

            resized = cv2.resize(face_img,(224,224))
            reshaped=resized.reshape(1, 224,224,3)/255
            predictions = model.predict(reshaped)

            max_index = np.argmax(predictions[0])

            emotions = ('angry', 'disgust', 'fear', 'happy', 'sad', 'neutral', 'surprise')
            predicted_emotion = emotions[max_index]
            output.append(predicted_emotion)



            cv2.rectangle(img,(x,y),(x+w,y+h),GR_dict[1],2)
            cv2.rectangle(img,(x,y-40),(x+w,y),GR_dict[1],-1)
            cv2.putText(img, predicted_emotion, (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)
        i = i+1

        cv2.imshow('LIVE', img)
        key = cv2.waitKey(1)
        if key == 27:
            cap.release()
            cv2.destroyAllWindows()
            break
    print(output)
    cap.release()
    cv2.destroyAllWindows()
    final_output1 = st.mode(output)
    return render_template("buttons.html",final_output=final_output1)


@app.route('/templates/buttons', methods = ['GET','POST'])
def buttons():
    return render_template("buttons.html")
```

Fig 4.17 Detecting Emotions Using Model

```
@app.route('/songs/disgust', methods = ['GET', 'POST'])
def songsDisgust():
    return render_template("songsDisgust.html")

@app.route('/songs/happy', methods = ['GET', 'POST'])
def songsHappy():
    return render_template("songsHappy.html")

@app.route('/songs/fear', methods = ['GET', 'POST'])
def songsFear():
    return render_template("songsFear.html")

@app.route('/songs/neutral', methods = ['GET', 'POST'])
def songsNeutral():
    return render_template("songsSad.html")

@app.route('/templates/join_page', methods = ['GET', 'POST'])
def join():
    return render_template("join_page.html")

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

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

Fig 4.18 Requesting Web Pages according to classified emotion

5. TESTING

5.1 Test Cases

5.1.1 Test Cases for Emotion Detection

Here we demonstrated a sample using a web-cam as a test subject.

Trying to Detect Different Emotions using Model

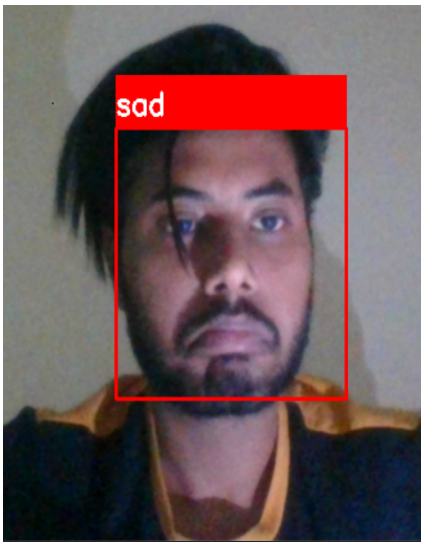

Fig 5.1 Sad Emotion Detection

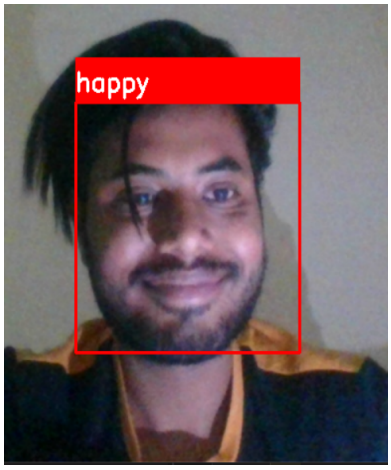

Fig 5.2 Happy Emotion Detection

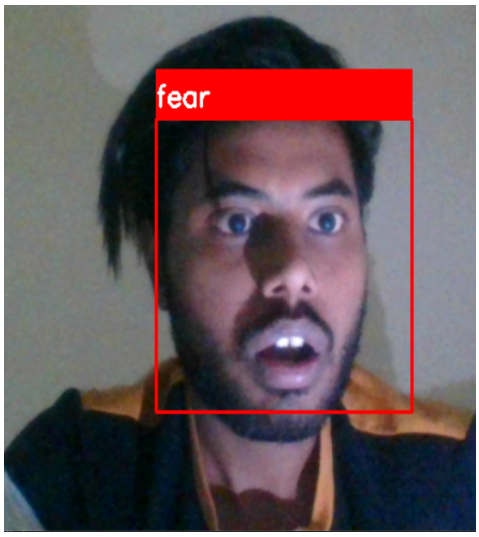

Fig 5.3 Fear Emotion Detection

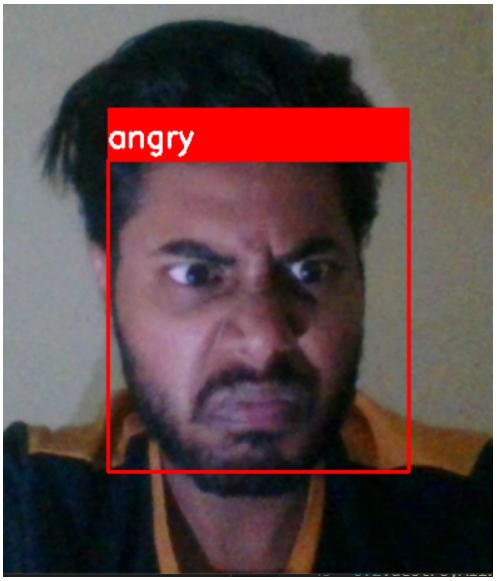

Fig 5.4 Angry Emotion Detection

Source of Image : Captured Using Web-Cam

5.1.2 Output Recommendation

Output for Sad Emotion will be a list of movies/songs:

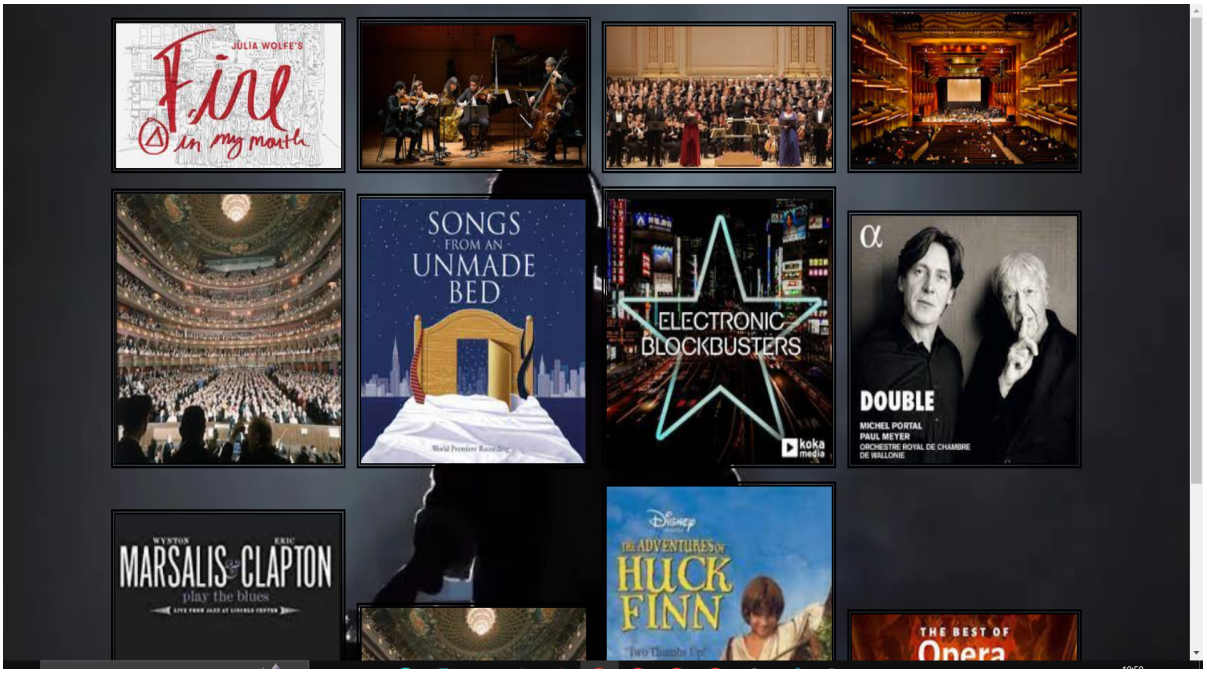

Fig 5.5 List of Songs for Detected Emotion

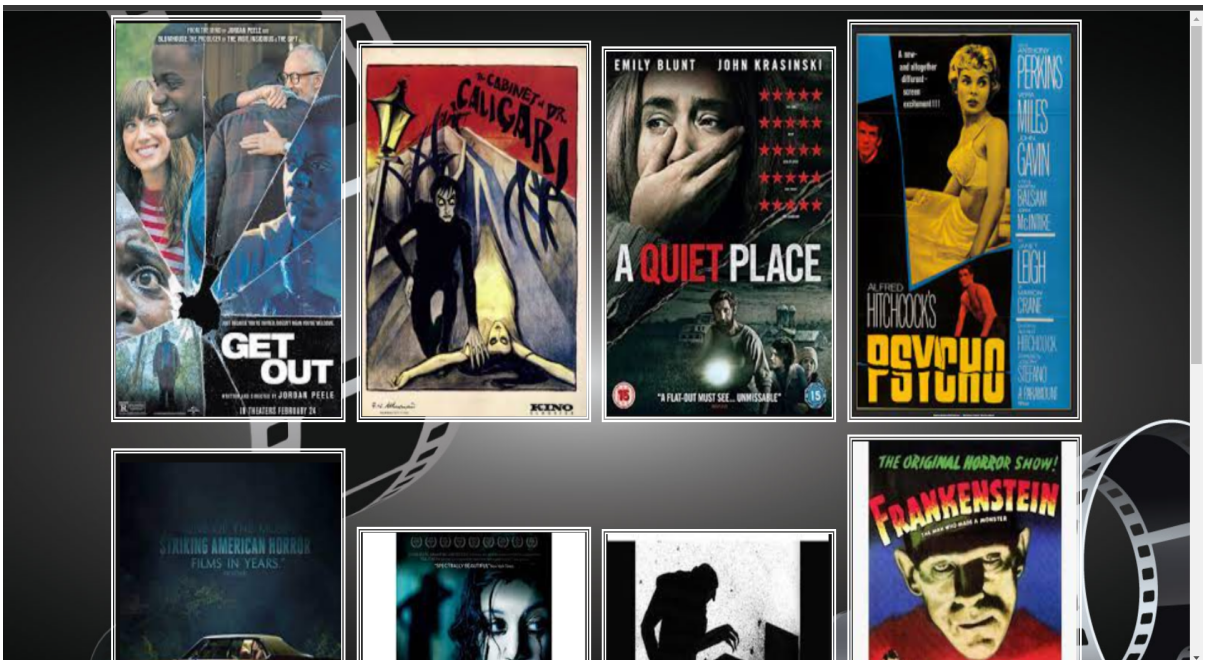

Fig 5.6 List of Movies for Detected Emotion

6. Conclusion

6.1 Design and implementation issues

- The capturing is frame by frame not scene.
- To design an efficient user interface for multimedia.
- The accuracy using CNN model is 69% but still trying to make customized neural networks for improvement in model.
- Making it more user friendly.
- Creating a Database to store images of users for different emotions to improve model for further training.

6.2 Advantages and Limitations

- Providing a web app for movie or music fans with a modern platform.
- Offering users entertainment resources
- The system still is not able to record all the emotions correctly due to the less availability of the images in the image dataset being used.
- The image that is fed into the classifier should be taken in a well-lit atmosphere for the classifier to give accurate results.
- The quality of the image should be at least higher than 320p for the classifier to predict the emotion of the user accurately.

6.3 Future Scope of the Project

- Reduce the classifier's training period.
- EEG signals can be used to further optimize the project and identify the user's precise emotional state.
- It is clear that more information may be gathered and used to build the grouping model in order to increase the precision of the arrangement framework.

7. References

1. Deny, J., & Sundararajan, M. (2014). Survey of Texture Analysis UsingHistogram in Image Processing. International Journal of Applied Engineering Research, 9(26), 8737- 8739. https://www.researchgate.net/publication/304091674_Survey_of_Texture_Analysis_Using_Histogram_in_Image_Processing
2. Deny, J., & Sundhararajan, M. (2015). Multi Modal Biometric Security for MANET Military Application-Face and Fingerprint. Journal of Computational and Theoretical Nanoscience, 12(12), 5949-5953. https://www.researchgate.net/publication/304380954_Multi_Modal_Biometric_Security_for_MANET_Military_Application-Face_and_Fingerprint
3. Deny, J., Muthukumaran, E., Ramkumar, S., & Kartheesawaran, S. (2018). Extraction Of Respiratory Signals And Motion Artifacts From PPG Signal Using Modified Multi Scale Principal Component Analysis. International Journal of Pure and Applied Mathematics, 119(12), 13719-13727. https://www.researchgate.net/publication/325465991_Extraction_Of_Respiratory_Signals_And_Motion_Artifacts_From_PPG_Signal_Using_Modified_Multi_Scale_Principal_Component_Analysis
4. Fan, X., Zhang, F., Wang, H., & Lu, X. (2012). The System of Face Detection Based on OpenCV. In 24th Chinese Control and Decision Conference (CCDC), Taiyuan, China. IEEE.
    a. https://doi.org/10.1109/CCDC.2012.6242980
    b.
5. Gupta, S. (2018). Facial emotion recognition in real-time and static images. In 2nd International Conference on Inventive Systems and Control (ICISC), Coimbatore, India. IEEE.
    a. https://doi.org/10.1109/ICISC.2018.8398861
6. Knyazev, B., Shvetsov, R., Efremova, N., & Kuharenko, A. (2018). Leveraging large face recognition data for emotion classification. In 13th IEEE International

Conference on Automatic Face & Gesture Recognition (FG 2018), Xi'an, China. IEEE.

a. https://doi. org/10.1109/FG.2018.00109

7. Youssif, A. A. A., & Wesam, A. A. A. (2011). Automatic Facial Expression Recognition System based on Geometric and Appearance Features. Computer and Information Science, 4(2), 115-124.
   a. https://doi.org/10.5539/cis.v4n2p115
8. G. Sailaja, V. Hima Deepthi, Facial Expression Recognition Complications with the Stages of Face Detection and Recognition, (IJRTE, 2019).
9. H. Immanuel James, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan, R. Saranya, "EMOTION BASED MUSIC RECOMMENDATION SYSTEM", (IRJET, 2019)
10. S.Nithya Roopa, "Research on Face Expression Recognition", (IJITEE, 2019).
11. Leo Pauly, Deepa Sankar, A Novel Online Product Recommendation System Based on Face Recognition and Emotion Detection, (ICCICCT, 2015).