In [1]:
```
from keras.utils import to_categorical
from keras_preprocessing.image import load_img
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Dropout, Flatten, MaxPooling2D
import os
import pandas as pd
import numpy as np
```

In [2]:
```
TRAIN_DIR = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\train'
TEST_DIR = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test'
```

In [3]:
```
def createdataframe(dir):
    image_paths = []
    labels = []
    for label in os.listdir(dir):
        for imagename in os.listdir(os.path.join(dir,label)):
            image_paths.append(os.path.join(dir,label,imagename))
            labels.append(label)
        print(label, "completed")
    return image_paths,labels
```

In [4]:
```
train = pd.DataFrame()
train['image'], train['label'] = createdataframe(TRAIN_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

In [5]:
```
print(train)
```

```
                                                   image     label
0      C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
1      C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
2      C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
3      C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
4      C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
...                                                  ...       ...
28816  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
28817  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
28818  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
28819  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
28820  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise

[28821 rows x 2 columns]
```

In [6]:
```
test = pd.DataFrame()
test['image'], test['label'] = createdataframe(TEST_DIR)
```

```
angry completed
disgust completed
fear completed
happy completed
neutral completed
sad completed
surprise completed
```

In [7]:
```
print(test)
print(test['image'])
```

```
                                                  image     label
0     C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
1     C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
2     C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
3     C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
4     C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...     angry
...                                                 ...       ...
7061  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
7062  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
7063  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
7064  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise
7065  C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...  surprise

[7066 rows x 2 columns]
0       C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
1       C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
2       C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
3       C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
4       C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
                              ...
7061    C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
7062    C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
7063    C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
7064    C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
7065    C:\Users\mdaud\OneDrive\Desktop\AI Project\ima...
Name: image, Length: 7066, dtype: object
```

In [8]:
```
from tqdm.notebook import tqdm
```

In [9]:
```
def extract_features(images):
    features = []
    for image in tqdm(images):
        img = load_img(image,grayscale =  True )
        img = np.array(img)
        features.append(img)
    features = np.array(features)
    features = features.reshape(len(features),48,48,1)
    return features
```

In [10]:
```
train_features = extract_features(train['image'])
```

```
100%                                                 28821/28821 [05:39<00:00, 112.84it/s]

C:\Users\mdaud\anaconda3\Lib\site-packages\keras_preprocessing\image\utils.py:107: UserWarning: gra
yscale is deprecated. Please use color_mode = "grayscale"
  warnings.warn('grayscale is deprecated. Please use '
```

In [11]:
```
test_features = extract_features(test['image'])
```

```
100%                                                 7066/7066 [01:06<00:00, 82.01it/s]
```

In [12]:
```
x_train = train_features/255.0
x_test = test_features/255.0
```

In [13]:
```
from sklearn.preprocessing import LabelEncoder
```

In [14]:
```
le = LabelEncoder()
le.fit(train['label'])
```

Out[14]:
```
LabelEncoder
LabelEncoder()
```

In [15]:
```
y_train = le.transform(train['label'])
y_test = le.transform(test['label'])
```

In [16]:
```
y_train = to_categorical(y_train,num_classes = 7)
y_test = to_categorical(y_test,num_classes = 7)
```

In [18]:
```
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense, Input

model = Sequential()
# input layer
model.add(Input(shape=(48,48,1)))
# convolutional layers
model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(256, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Conv2D(512, kernel_size=(3,3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.4))

model.add(Flatten())
# fully connected layers
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.4))
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.3))
# output layer
model.add(Dense(7, activation='softmax'))
```

In [20]:
```
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

In [21]:
```
model.fit(x= x_train,y = y_train, batch_size = 128, epochs = 100, validation_data = (x_test,y_test))
```

```
Epoch 62/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 219s 969ms/step - accuracy: 0.6695 - loss: 0.8894 - val_accuracy:
0.6217 - val_loss: 1.0409
Epoch 63/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 219s 969ms/step - accuracy: 0.6612 - loss: 0.9015 - val_accuracy:
0.6211 - val_loss: 1.0285
Epoch 64/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 218s 966ms/step - accuracy: 0.6698 - loss: 0.8933 - val_accuracy:
0.6165 - val_loss: 1.0414
Epoch 65/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 222s 983ms/step - accuracy: 0.6689 - loss: 0.8919 - val_accuracy:
0.6230 - val_loss: 1.0260
Epoch 66/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 218s 964ms/step - accuracy: 0.6753 - loss: 0.8905 - val_accuracy:
0.6298 - val_loss: 1.0250
Epoch 67/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 219s 967ms/step - accuracy: 0.6764 - loss: 0.8736 - val_accuracy:
0.6302 - val_loss: 1.0209
Epoch 68/100
226/226 ━━━━━━━━━━━━━━━━━━━━ 218s 963ms/step - accuracy: 0.6827 - loss: 0.8504 - val_accuracy:
```

In [23]:
```
# Save the model
model.save("emotiondetector.keras")
```

In [25]:
```
# Load the model
from keras.models import load_model
model = load_model("emotiondetector.keras", compile=False)

# Compile the model with the same optimizer
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])
```

In [26]:
```
# Define the function to preprocess the image
def ef(image):
    img = load_img(image, grayscale = True)
    feature = np.array(img)
    feature = feature.reshape(1, 48, 48, 1)
    return feature / 255.0
```

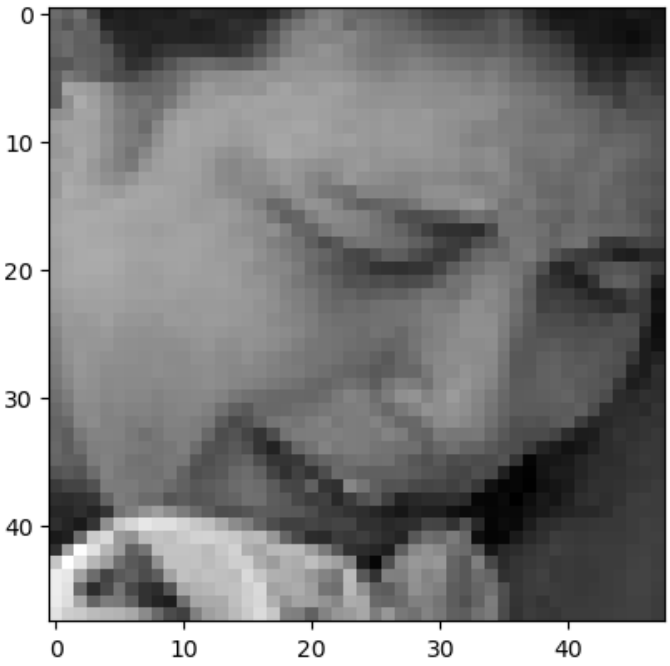
In [27]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

In [29]:
```
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

In [30]:
```
# Use the loaded model for prediction
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\sad\350.jpg'
print("Original image is of sad")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("Model prediction is ", pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
Original image is of sad
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 31ms/step
Model prediction is  sad
```

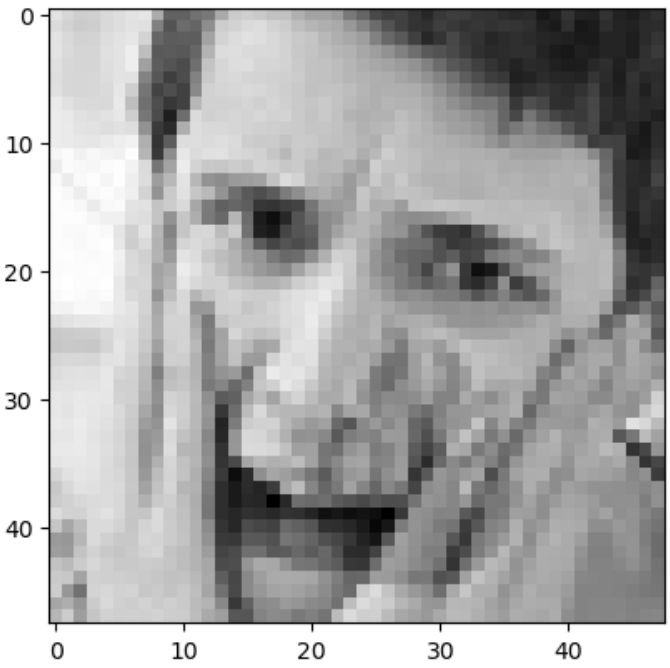
Out[30]: <matplotlib.image.AxesImage at 0x2ba95170e90>



In [31]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\fear\73.jpg'
print("original image is of fear")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of fear
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step
model prediction is  fear
```

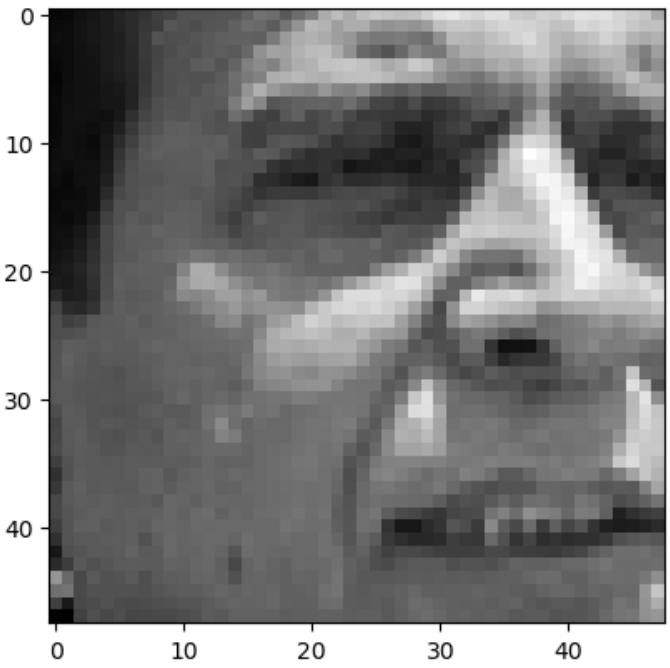
Out[31]: <matplotlib.image.AxesImage at 0x2ba98930d10>



In [32]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\angry\23.jpg'
print("original image is of angry")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of angry
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step
model prediction is  angry
```

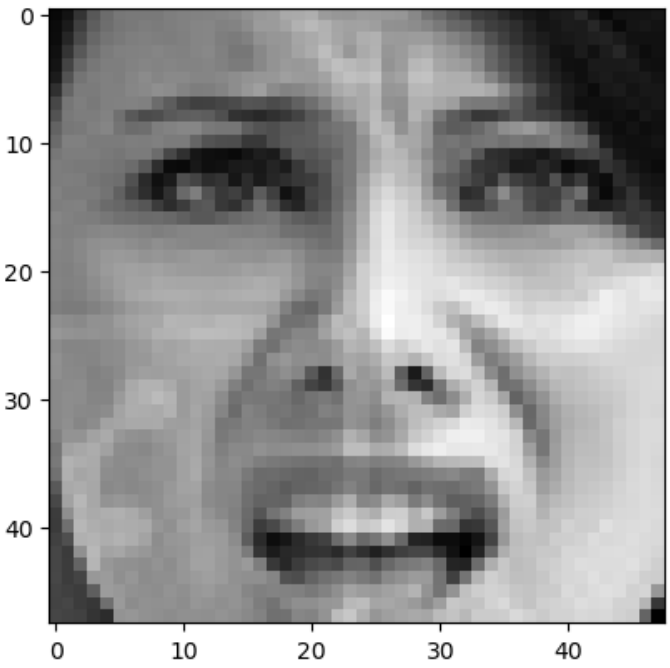
Out[32]: <matplotlib.image.AxesImage at 0x2bb31621250>



In [33]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\disgust\533.jpg'
print("original image is of disgust")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of disgust
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step
model prediction is  disgust
```

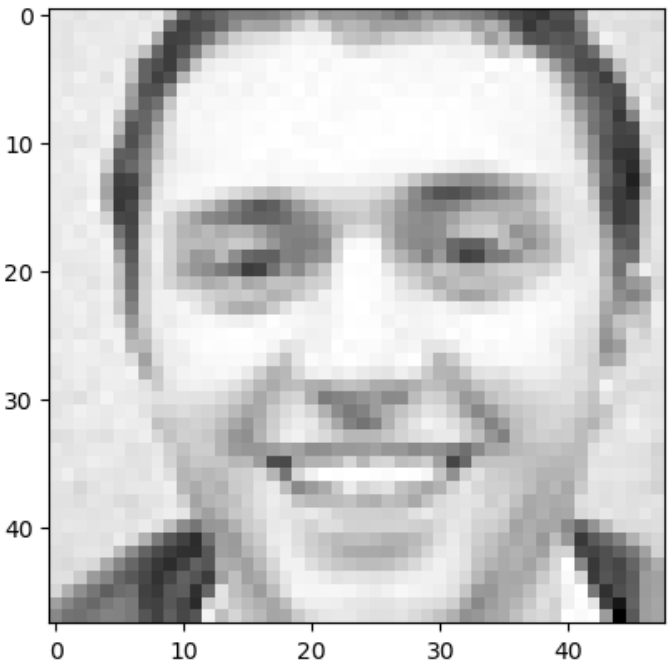
Out[33]: <matplotlib.image.AxesImage at 0x2ba98930e90>



In [34]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\happy\30.jpg'
print("original image is of happy")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of happy
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
model prediction is  happy
```

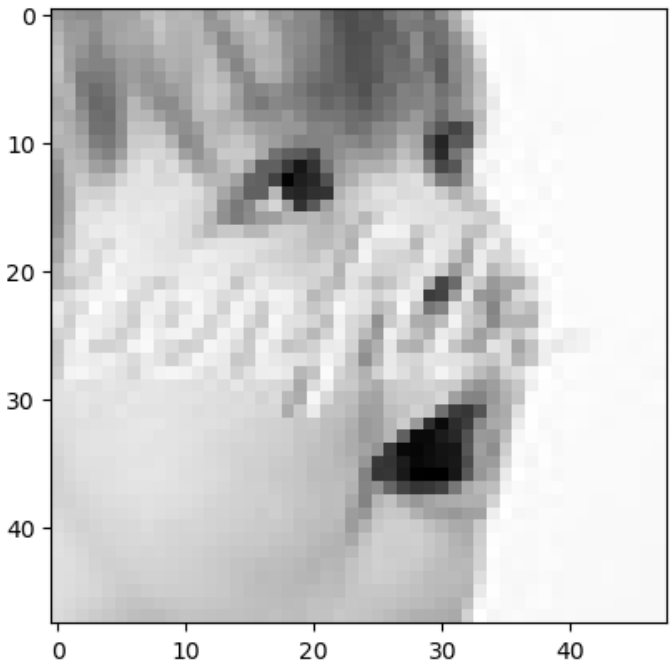
Out[34]: <matplotlib.image.AxesImage at 0x2ba92501250>



In [35]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\surprise\256.jpg'
print("original image is of surprise")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of surprise
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
model prediction is  surprise
```

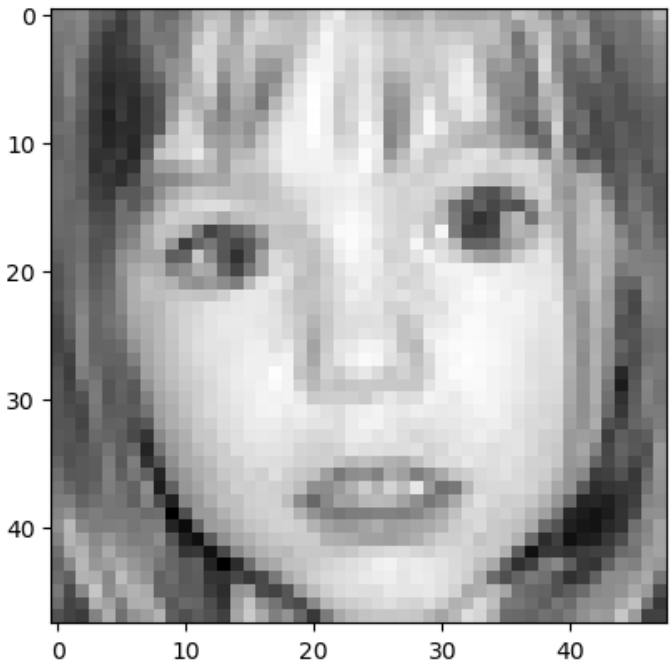
Out[35]: <matplotlib.image.AxesImage at 0x2ba8b563390>



In [37]:
```
image = r'C:\Users\mdaud\OneDrive\Desktop\AI Project\images\test\neutral\60.jpg'
print("original image is of neutral")
img = ef(image)
pred = model.predict(img)
pred_label = label[pred.argmax()]
print("model prediction is ",pred_label)
plt.imshow(img.reshape(48,48),cmap='gray')
```

```
original image is of neutral
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 30ms/step
model prediction is  neutral
```

Out[37]: <matplotlib.image.AxesImage at 0x2ba92759550>



In [38]:
```
pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\mdaud\appdata\roaming\python\python311\sit
e-packages (4.10.0.82)
Requirement already satisfied: numpy>=1.21.2 in c:\users\mdaud\anaconda3\lib\site-packages (from op
encv-python) (1.24.3)
Note: you may need to restart the kernel to use updated packages.
```

In [39]:
```
pip install --user opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\mdaud\appdata\roaming\python\python311\sit
e-packages (4.10.0.82)
Requirement already satisfied: numpy>=1.21.2 in c:\users\mdaud\anaconda3\lib\site-packages (from op
encv-python) (1.24.3)
Note: you may need to restart the kernel to use updated packages.
```

In [40]:
```
import cv2

# Try to release the camera if it's being used
webcam = cv2.VideoCapture(0)
if webcam.isOpened():
    webcam.release()
cv2.destroyAllWindows()

print("Camera released")
```

```
Camera released
```

In [56]:
```
import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing.image import img_to_array
from IPython.display import display, Image, clear_output
import time

# Load the model
from keras.models import load_model
model = load_model("emotiondetector.keras", compile=False)

# Compile the model with the same optimizer
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Haarcascade file for face detection
haar_file = cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
face_cascade = cv2.CascadeClassifier(haar_file)

# Function to preprocess the image
def extract_features(image):
    feature = img_to_array(image)
    feature = feature.reshape(1, 48, 48, 1)
    feature = feature.astype('float32') / 255.0
    return feature

# Initialize webcam
webcam = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not webcam.isOpened():
    print("Error: Could not open webcam.")
else:
    labels = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neutral', 5: 'sad', 6: 'surprise'
    frames = []

    # Capture frames for 2-3 seconds
    start_time = time.time()
    while time.time() - start_time < 3:  # Capture for 3 seconds
        ret, frame = webcam.read()
        if not ret:
            print("Error: Could not read frame.")
            break
        frames.append(frame)

    # Process captured frames
    for frame in frames:
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            face = gray[y:y + h, x:x + w]
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            face = cv2.resize(face, (48, 48))
            face = extract_features(face)
            pred = model.predict(face)
            prediction_label = labels[np.argmax(pred)]
            cv2.putText(frame, f'{prediction_label}', (x, y - 10), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2

        # Display the processed frame in the notebook
        _, jpeg = cv2.imencode('.jpeg', frame)
        display(Image(data=jpeg.tobytes()))
        clear_output(wait=True)

    # Release the webcam and close windows
    webcam.release()
```

```
cv2.destroyAllWindows()
```

In [55]:
```
import cv2
import numpy as np
from tensorflow.keras.models import model_from_json
from tensorflow.keras.preprocessing.image import img_to_array
from IPython.display import display, Image, clear_output

# Load the model
from keras.models import load_model
model = load_model("emotiondetector.keras", compile=False)

# Compile the model with the same optimizer
model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['accuracy'])

# Haarcascade file for face detection
haar_file = cv2.data.haarcascades + 'haarcascade_frontalface_default.xml'
face_cascade = cv2.CascadeClassifier(haar_file)

# Function to preprocess the image
def extract_features(image):
    feature = img_to_array(image)
    feature = feature.reshape(1, 48, 48, 1)
    feature = feature.astype('float32') / 255.0
    return feature

# Initialize webcam
webcam = cv2.VideoCapture(0)

# Check if the webcam is opened correctly
if not webcam.isOpened():
    print("Error: Could not open webcam.")
else:
    labels = {0: 'angry', 1: 'disgust', 2: 'fear', 3: 'happy', 4: 'neutral', 5: 'sad', 6: 'surprise'

    while True:
        ret, frame = webcam.read()
        if not ret:
            print("Error: Could not read frame.")
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            face = gray[y:y + h, x:x + w]
            cv2.rectangle(frame, (x, y), (x + w, y + h), (255, 0, 0), 2)
            face = cv2.resize(face, (48, 48))
            face = extract_features(face)
            pred = model.predict(face)
            prediction_label = labels[np.argmax(pred)]
            cv2.putText(frame, f'{prediction_label}', (x, y - 10), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2

        # Display the processed frame in the notebook
        _, jpeg = cv2.imencode('.jpeg', frame)
        display(Image(data=jpeg.tobytes()))
        clear_output(wait=True)

    # Release the webcam and close windows
    webcam.release()
    cv2.destroyAllWindows()
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[55], line 48
     46 face = cv2.resize(face, (48, 48))
     47 face = extract_features(face)
---> 48 pred = model.predict(face)
     49 prediction_label = labels[np.argmax(pred)]
     50 cv2.putText(frame, f'{prediction_label}', (x, y - 10), cv2.FONT_HERSHEY_COMPLEX_SMALL,
2, (0, 0, 255))

File ~\anaconda3\Lib\site-packages\keras\src\utils\traceback_utils.py:117, in filter_traceback.<
locals>.error_handler(*args, **kwargs)
    115 filtered_tb = None
    116 try:
--> 117     return fn(*args, **kwargs)
    118 except Exception as e:
    119     filtered_tb = _process_traceback_frames(e.__traceback__)

File ~\anaconda3\Lib\site-packages\keras\src\backend\tensorflow\trainer.py:504, in TensorFlowTra
```