Zidio: Realtime Video Emotion Detection

In [4]:

```
import cv2
from deepface import DeepFace
import pyttsx3  # For text-to-speech
import datetime

# Initialize text-to-speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Speed of speech
engine.setProperty('voice', 'english+f3')  # Set voice to female (ensure system has this voice)

# Initialize global variables for historical mood tracking and stress alerts
mood_history = []
stress_alerts = []

# Task Recommendation with specific tasks for each emotion
def recommend_task(emotion):
    tasks = {
        "happy": "You're feeling happy! How about working on creative tasks like brainstorming ne
        "sad": "You're feeling sad. It may help to take a break, listen to calming music, or enga
        "angry": "You're feeling angry. Try calming down with deep breathing exercises, taking a
        "neutral": "You're feeling neutral. This is the perfect time to focus on routine tasks or
        "surprise": "You're surprised! Take advantage of this unexpected energy to explore new id
        "fear": "You're feeling fearful. Focus on planning and prioritizing your tasks. Break dow
        "disgust": "You're feeling disgusted. It might help to reflect on what's bothering you, e
    }
    return tasks.get(emotion, "Please take a moment to relax and reflect on your feelings.")

# Stress Management Alerts with specific alerts for each emotion
def check_stress_level(emotion):
    alerts = {
        "happy": "You're in a great mood! Enjoy the positivity, keep spreading joy, and make the
        "sad": "You're feeling sad. Consider reaching out to someone you trust, or engage in some
        "angry": "You seem angry. Try focusing on your breath, stepping away from the situation f
        "neutral": "You seem neutral. Stay on track with your tasks and focus on maintaining a ca
        "surprise": "You seem surprised! Embrace the change, and consider adapting quickly to any
        "fear": "You're feeling fearful. Break down your tasks into smaller steps and focus on po
        "disgust": "You seem disgusted. Take a moment to process your feelings, reflect on what's
    }
    return alerts.get(emotion, None)

# Historical Mood Tracking
def track_mood_history(emotion):
    mood_history.append((datetime.datetime.now(), emotion))

# Real-Time Video Emotion Detection
def real_time_emotion_detection():
    cap = cv2.VideoCapture(0)
    if not cap.isOpened():
        print("Error: Could not open webcam.")
        return

    last_emotion = None  # Track the last detected emotion

    while True:
        ret, frame = cap.read()
        if not ret:
            print("Failed to grab frame")
            break
```

```
        try:
            # Detect emotions using DeepFace
            result = DeepFace.analyze(frame, actions=['emotion'], enforce_detection=False)
            emotion = result[0]['dominant_emotion'].lower()  # Convert emotion to lowercase for c

            # Only update task and stress alerts if the emotion changes
            if emotion != last_emotion:
                print(f"Detected Emotion: {emotion}")

                # Task recommendation
                task = recommend_task(emotion)
                print(f"Recommended Task: {task}")

                # Stress management alerts
                alert = check_stress_level(emotion)
                if alert:
                    print(alert)
                    stress_alerts.append(alert)

                # Track mood history
                track_mood_history(emotion)

                # Speak the detected emotion and recommended task
                engine.say(f"Detected emotion: {emotion}. Recommended task: {task}")
                engine.runAndWait()

                # Update last_emotion
                last_emotion = emotion

            # Display emotion and task on the frame
            cv2.putText(frame, f'Emotion: {emotion}', (50, 50), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,
            cv2.putText(frame, f'Task: {task}', (50, 100), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0,

        except Exception as e:
            print(f"Error in emotion detection: {e}")

        # Show the frame
        cv2.imshow('Real-Time Emotion Detection', frame)

        # Break loop on pressing 'q'
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    # Release the webcam and close the window
    cap.release()
    cv2.destroyAllWindows()
    engine.stop()  # Stop the text-to-speech engine

# Main Function
def main():
    print("Starting Real-Time Video Emotion Detection...")
    real_time_emotion_detection()

    # Display Historical Mood Tracking
    print("\nHistorical Mood Tracking:")
    for timestamp, mood in mood_history:
        print(f"{timestamp}: {mood}")

    # Display Stress Alerts
    print("\nStress Alerts:")
    for alert in stress_alerts:
        print(alert)
```

```python

# Run the program
if __name__ == "__main__":
    main()
```

```
WARNING:tensorflow:From C:\Users\fawad\anaconda3\Lib\site-packages\tf_keras\src\losses.py:2976: T
he name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.spar
se_softmax_cross_entropy instead.

Starting Real-Time Video Emotion Detection...
Detected Emotion: angry
Recommended Task: You're feeling angry. Try calming down with deep breathing exercises, taking a
walk, or journaling to process your emotions.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
Detected Emotion: fear
Recommended Task: You're feeling fearful. Focus on planning and prioritizing your tasks. Break do
wn challenges into smaller steps to reduce anxiety.
You're feeling fearful. Break down your tasks into smaller steps and focus on positive action to
overcome your fears. You've got this!
Detected Emotion: neutral
Recommended Task: You're feeling neutral. This is the perfect time to focus on routine tasks or o
rganize your environment. Stay productive!
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
Detected Emotion: angry
Recommended Task: You're feeling angry. Try calming down with deep breathing exercises, taking a
walk, or journaling to process your emotions.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
Detected Emotion: neutral
Recommended Task: You're feeling neutral. This is the perfect time to focus on routine tasks or o
rganize your environment. Stay productive!
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
Detected Emotion: angry
Recommended Task: You're feeling angry. Try calming down with deep breathing exercises, taking a
walk, or journaling to process your emotions.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
Detected Emotion: neutral
Recommended Task: You're feeling neutral. This is the perfect time to focus on routine tasks or o
rganize your environment. Stay productive!
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
Detected Emotion: angry
Recommended Task: You're feeling angry. Try calming down with deep breathing exercises, taking a
walk, or journaling to process your emotions.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
Detected Emotion: neutral
Recommended Task: You're feeling neutral. This is the perfect time to focus on routine tasks or o
rganize your environment. Stay productive!
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.

Historical Mood Tracking:
2025-02-19 15:26:00.229355: angry
2025-02-19 15:26:14.992197: fear
2025-02-19 15:26:29.818734: neutral
2025-02-19 15:26:45.298093: angry
2025-02-19 15:26:59.510923: neutral
2025-02-19 15:27:13.433671: angry
2025-02-19 15:27:27.616427: neutral
2025-02-19 15:27:41.889138: angry
2025-02-19 15:27:56.077656: neutral
```

```
Stress Alerts:
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
You're feeling fearful. Break down your tasks into smaller steps and focus on positive action to
overcome your fears. You've got this!
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
You seem angry. Try focusing on your breath, stepping away from the situation for a moment, or us
ing mindfulness techniques to relax.
You seem neutral. Stay on track with your tasks and focus on maintaining a calm and productive mi
ndset throughout the day.
```

In [ ]:
```
Zidio: Realtime Speech Emotion Detection
```

In [6]:
```
import speech_recognition as sr
from textblob import TextBlob
import pyttsx3  # For text-to-speech
import datetime

# Initialize text-to-speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Speed of speech
engine.setProperty('voice', 'english+f3')  # Set voice to female

# Initialize global variables for historical mood tracking and stress alerts
mood_history = []
stress_alerts = []

# Task Recommendation
def recommend_task(emotion):
    tasks = {
        "Happy": "Work on creative tasks or brainstorming.",
        "Sad": "Take a break or engage in a relaxing activity.",
        "Angry": "Practice mindfulness or take a short walk.",
        "Neutral": "Focus on routine tasks.",
        "Surprise": "Explore new ideas or challenges.",
        "Fear": "Plan and prioritize tasks to reduce anxiety.",
        "Disgust": "Reflect on what's bothering you and address it.",
    }
    return tasks.get(emotion, "No specific recommendation.")

# Stress Management Alerts
def check_stress_level(emotion):
    alerts = {
        "Happy": "You seem happy. Keep up the good work!",
        "Sad": "You seem sad. Consider talking to someone or taking a break.",
        "Angry": "You seem angry. Try deep breathing or stepping away for a moment.",
        "Neutral": "You seem neutral. Stay focused and maintain your routine.",
        "Surprise": "You seem surprised. Embrace the unexpected and adapt.",
        "Fear": "You seem fearful. Focus on planning and prioritizing tasks.",
        "Disgust": "You seem disgusted. Reflect on what's bothering you.",
    }
    return alerts.get(emotion, None)
```

```python
# Historical Mood Tracking
def track_mood_history(emotion):
    mood_history.append((datetime.datetime.now(), emotion))

# Speech Emotion Detection
def detect_speech_emotion():
    try:
        recognizer = sr.Recognizer()
        with sr.Microphone() as source:
            print("Listening for speech...")
            recognizer.adjust_for_ambient_noise(source, duration=2)  # Reduce background noise
            try:
                # Listen for speech
                audio = recognizer.listen(source, timeout=10, phrase_time_limit=5)
                print("Processing speech...")
                text = recognizer.recognize_google(audio)
                print(f"Detected Speech: {text}")

                # Analyze sentiment using TextBlob
                blob = TextBlob(text)
                sentiment = blob.sentiment.polarity
                if sentiment > 0.2:
                    emotion = "Happy"
                elif sentiment < -0.2:
                    emotion = "Sad"
                else:
                    emotion = "Neutral"

                # Task recommendation
                task = recommend_task(emotion)
                print(f"Recommended Task: {task}")

                # Stress management alerts
                alert = check_stress_level(emotion)
                if alert:
                    print(alert)
                    stress_alerts.append(alert)

                # Track mood history
                track_mood_history(emotion)

                # Speak the detected emotion and recommended task
                engine.say(f"Detected emotion: {emotion}. Recommended task: {task}")
                engine.runAndWait()

                return emotion
            except sr.UnknownValueError:
                return "Unknown (No speech detected)"
            except sr.RequestError:
                return "Error (Speech service unavailable)"
            except sr.WaitTimeoutError:
                return "Error (No speech detected within timeout)"
    except ImportError:
        print("PyAudio not installed. Skipping speech emotion detection.")
        return "PyAudio not installed"

# Main Function
def main():
    print("Starting Speech Emotion Detection...")
    emotion = detect_speech_emotion()
    print(f"Detected Speech Emotion: {emotion}")
```

```
    # Display Historical Mood Tracking
    print("\nHistorical Mood Tracking:")
    for timestamp, mood in mood_history:
        print(f"{timestamp}: {mood}")

    # Display Stress Alerts
    print("\nStress Alerts:")
    for alert in stress_alerts:
        print(alert)

    engine.stop()  # Stop the text-to-speech engine

# Run the program
if __name__ == "__main__":
    main()
```

```
Starting Speech Emotion Detection...
Listening for speech...
Processing speech...
Detected Speech: Mustafa is a good
Recommended Task: Work on creative tasks or brainstorming.
You seem happy. Keep up the good work!
Detected Speech Emotion: Happy

Historical Mood Tracking:
2025-02-19 15:30:52.369874: Happy

Stress Alerts:
You seem happy. Keep up the good work!
```

In [ ]:
```
Zidio: Realtime Text Emotion Detection
```

In [8]:
```
from textblob import TextBlob
import pyttsx3  # For text-to-speech
import datetime

# Initialize text-to-speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Speed of speech
engine.setProperty('voice', 'english+f3')  # Set voice to female

# Initialize global variables for historical mood tracking and stress alerts
mood_history = []
stress_alerts = []

# Task Recommendation with short tasks for each emotion
def recommend_task(emotion):
    tasks = {
        "Happy": "Work on creative projects or connect with others.",
        "Sad": "Take a break, listen to calming music, or talk to a friend.",
        "Angry": "Try deep breathing or take a short walk.",
        "Neutral": "Focus on routine tasks or organize your space.",
        "Surprise": "Explore new ideas or take on a challenge.",
        "Fear": "Break down tasks and prioritize your day.",
        "Disgust": "Reflect on what's bothering you and relax.",
    }
    return tasks.get(emotion, "Relax and reflect.")

# Stress Management Alerts with short alerts for each emotion
def check_stress_level(emotion):
    alerts = {
```

```
        "Happy": "Keep up the good work and enjoy the positive energy!",
        "Sad": "Consider talking to someone or meditating.",
        "Angry": "Focus on your breath and stay calm.",
        "Neutral": "Stay productive and keep a calm mindset.",
        "Surprise": "Adapt quickly and embrace change.",
        "Fear": "Take action and break tasks into steps.",
        "Disgust": "Take a moment to process and reflect.",
    }
    return alerts.get(emotion, None)

# Historical Mood Tracking
def track_mood_history(emotion):
    mood_history.append((datetime.datetime.now(), emotion))

# Text Emotion Detection
def detect_text_emotion(text):
    # Basic sentiment analysis
    blob = TextBlob(text)
    sentiment = blob.sentiment.polarity

    # Classify the sentiment polarity into basic emotion categories
    if sentiment > 0.2:
        emotion = "Happy"
    elif sentiment < -0.2:
        emotion = "Sad"
    else:
        emotion = "Neutral"

    # Check for specific keywords in the text to refine emotions
    keywords = {
        "angry": "Angry", "rage": "Angry", "furious": "Angry",
        "sad": "Sad", "depressed": "Sad", "down": "Sad",
        "fear": "Fear", "scared": "Fear", "anxious": "Fear",
        "surprise": "Surprise", "shocked": "Surprise", "unexpected": "Surprise",
        "disgust": "Disgust", "dislike": "Disgust", "horrible": "Disgust",
    }

    # Search for keywords that could change the emotion detection
    for word, emotion_keyword in keywords.items():
        if word in text.lower():
            emotion = emotion_keyword
            break

    # Task recommendation
    task = recommend_task(emotion)
    print(f"Recommended Task: {task}")

    # Stress management alerts
    alert = check_stress_level(emotion)
    if alert and alert not in stress_alerts:  # Prevent duplicate stress alert
        print(alert)
        stress_alerts.append(alert)

    # Track mood history
    track_mood_history(emotion)

    # Speak the detected emotion and recommended task
    engine.say(f"Detected emotion: {emotion}. Recommended task: {task}")
    engine.runAndWait()

    return emotion
```

```python
# Main Function
def main():
    print("Starting Text Emotion Detection...")
    text = input("Enter your text: ")
    emotion = detect_text_emotion(text)
    print(f"Detected Text Emotion: {emotion}")

    # Display Historical Mood Tracking
    print("\nHistorical Mood Tracking:")
    for timestamp, mood in mood_history:
        print(f"{timestamp}: {mood}")

    # Display Stress Alerts
    print("\nStress Alerts:")
    for alert in stress_alerts:
        print(alert)

    engine.stop()  # Stop the text-to-speech engine

# Run the program
if __name__ == "__main__":
    main()
```

```
Starting Text Emotion Detection...
Recommended Task: Take a break, listen to calming music, or talk to a friend.
Consider talking to someone or meditating.
Detected Text Emotion: Sad

Historical Mood Tracking:
2025-02-19 15:33:30.112533: Sad

Stress Alerts:
Consider talking to someone or meditating.
```

In [ ]: