FACE EMOTION RECOGNITION SYSTEM USING STREAMLIT & DEEPFACE

ABSTRACT

Face Emotion Recognition is an important application of Artificial Intelligence and Computer Vision that focuses on identifying human emotions from facial expressions. Emotions such as happiness, sadness, anger, fear, surprise, and neutrality play a vital role in human–computer interaction. This project implements a **Face Emotion Recognition System** using **Python, Streamlit, OpenCV, and DeepFace**. The system supports both **image upload** and **webcam-based image capture**, detects one or more faces, predicts the dominant emotion for each detected face, and stores the results in an Excel-compatible CSV file for further analysis.

1: INTRODUCTION

Emotion recognition from facial expressions is a rapidly growing research area in computer vision and artificial intelligence. Facial expressions provide essential non-verbal cues that help in understanding human behavior and emotions. Automated face emotion recognition systems are widely used in surveillance systems, mental health analysis, human–computer interaction, customer feedback analysis, and smart monitoring systems.

This project titled **"Face Emotion Recognition System"** demonstrates a practical implementation of emotion recognition using pre-trained deep learning models provided by the DeepFace library along with a simple and interactive web interface developed using Streamlit.

2: OBJECTIVES OF THE PROJECT

The main objectives of this project are:

- To detect human faces from images and webcam captures.
- To recognize and classify facial emotions.
- To support both image upload and webcam-based input.
- To visualize emotion confidence scores.
- To store emotion recognition results in an Excel-compatible file.

3: SCOPE OF THE PROJECT

The scope of this project includes:

- Emotion recognition from static images.
- Emotion recognition from webcam-captured images.
- Detection of multiple faces in a single frame.
- Storage of results for further analysis using Excel.

4: SOFTWARE AND HARDWARE REQUIREMENTS

4.1 Software Requirements

- Operating System: Windows 10 / Windows 11
- Programming Language: Python 3.x
- IDE: Visual Studio Code
- Libraries and Frameworks:
    - Streamlit
    - OpenCV
    - DeepFace
    - NumPy
    - Pandas

4.2 Hardware Requirements

- Laptop/Desktop Computer
- Minimum 4 GB RAM
- Webcam (for webcam capture mode)

5: TECHNOLOGIES USED

5.1 Streamlit

Streamlit is used to develop a web-based graphical user interface that allows users to interact with the application easily.

5.2 OpenCV

OpenCV is used for image processing and face detection using Haar Cascade classifiers.

5.3 DeepFace

DeepFace is a deep learning-based facial analysis library that provides pre-trained models for emotion recognition.

6: SYSTEM ARCHITECTURE

The system architecture consists of the following components:

1. Streamlit User Interface
2. Input Selection Module (Image Upload / Webcam)
3. Face Detection Module (Haar Cascade)
4. Emotion Recognition Module (DeepFace)
5. Result Visualization Module
6. Result Storage Module (CSV / Excel)

System Flow:

User → Select Input Mode → Image/Webcam Input → Face Detection → Emotion Recognition → Display Output → Save Results

7: MODULE DESCRIPTION

7.1 Input Selection Module

Allows the user to choose between image upload and webcam capture modes using radio buttons.

7.2 Face Detection Module

Detects one or more faces from the input using OpenCV Haar Cascade classifier.

7.3 Emotion Recognition Module

Uses DeepFace to analyze detected faces and predict the dominant emotion along with confidence scores.

7.4 Result Visualization Module

Displays detected faces with bounding boxes and emotion labels. Emotion confidence scores are shown using tables and bar charts.

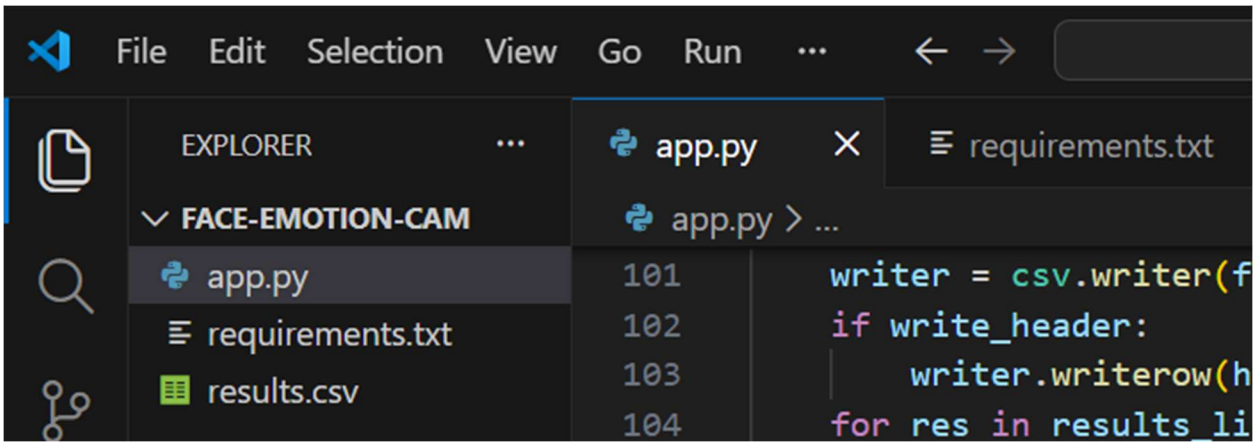
7.5 Result Storage Module

Stores emotion recognition results in a CSV file (results.csv), which can be opened using Microsoft Excel.

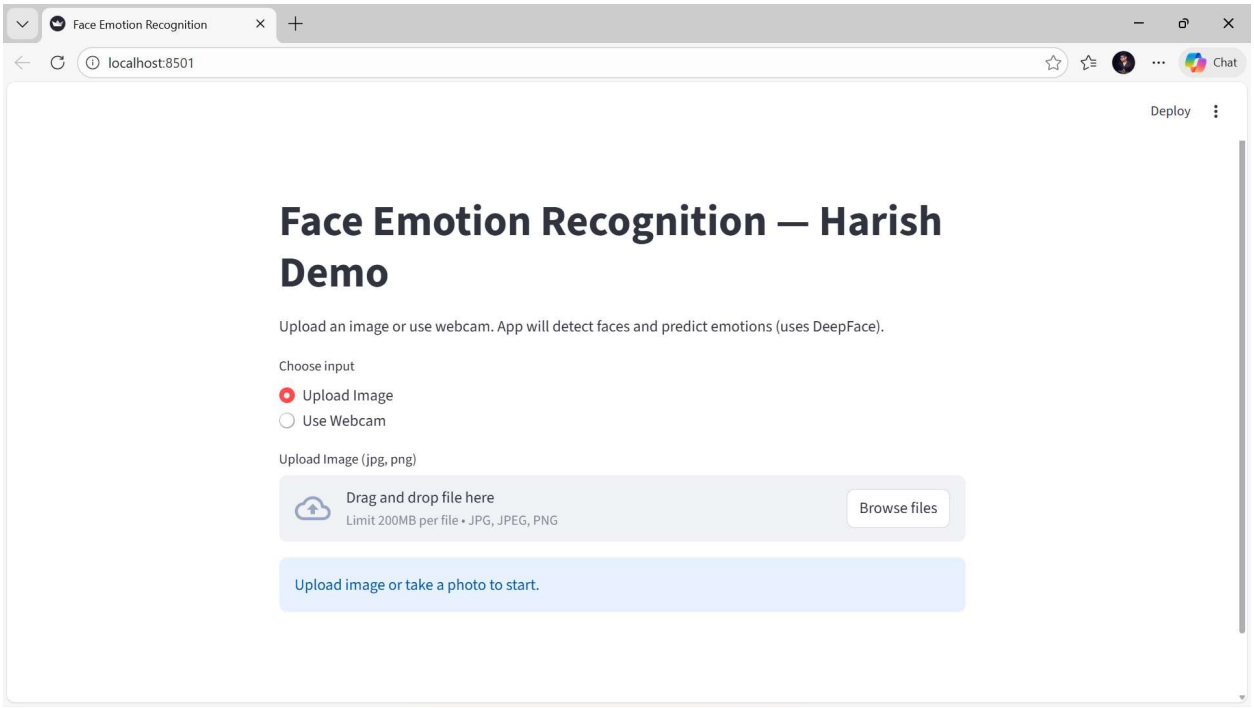
8: IMPLEMENTATION

This chapter describes the actual working of the system.

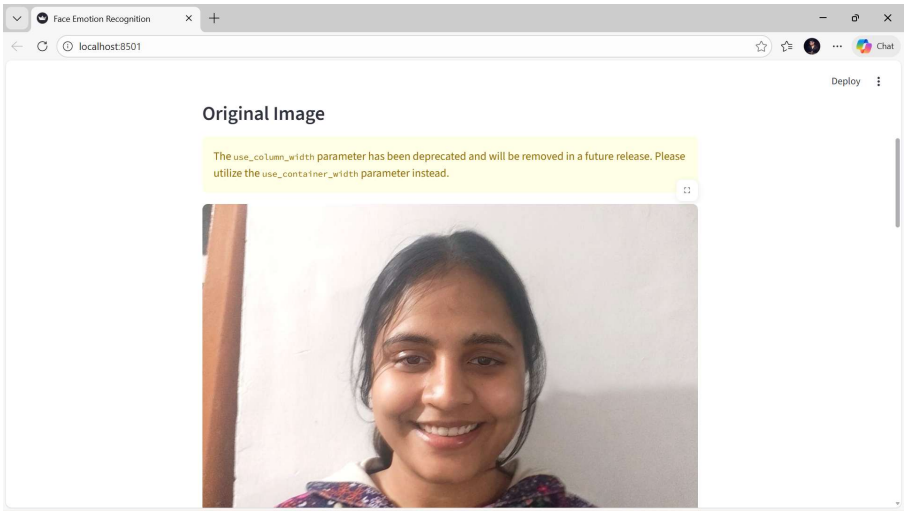
8.1 Project Folder Structure

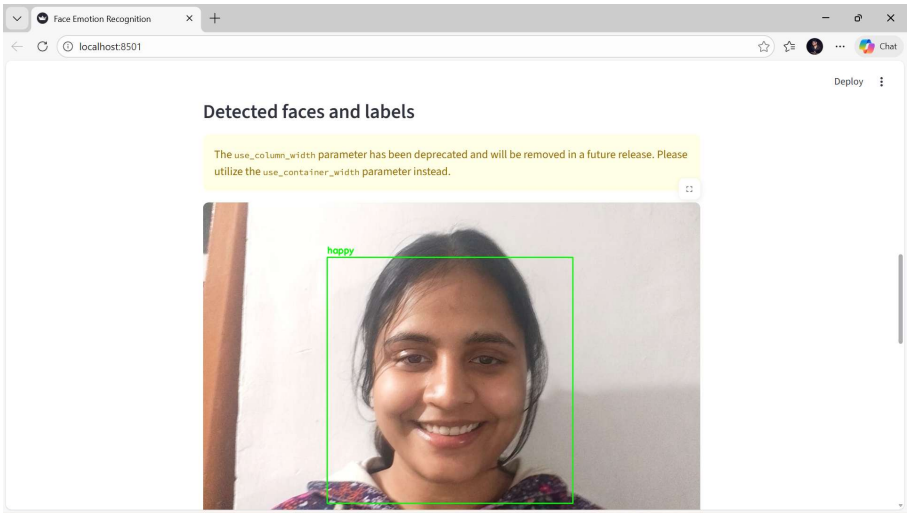


8.2 Streamlit User Interface



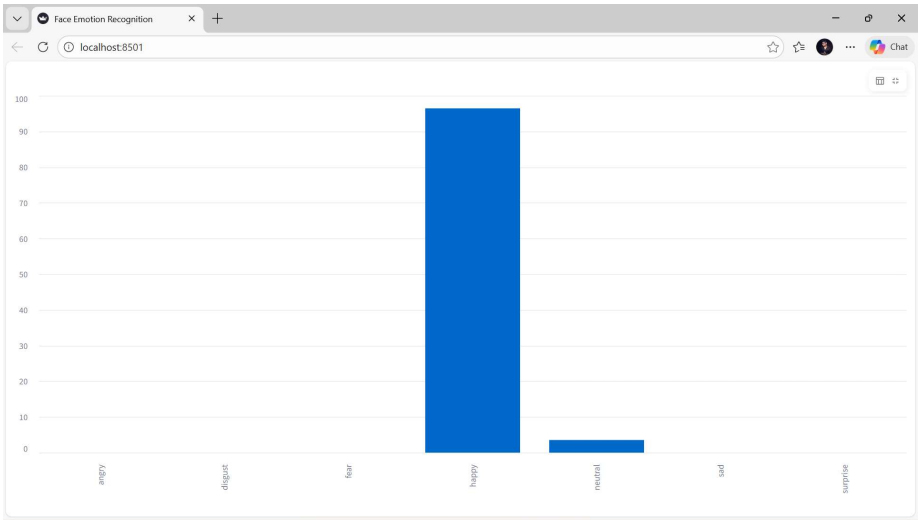
8.3 Image Upload Based Emotion Recognition



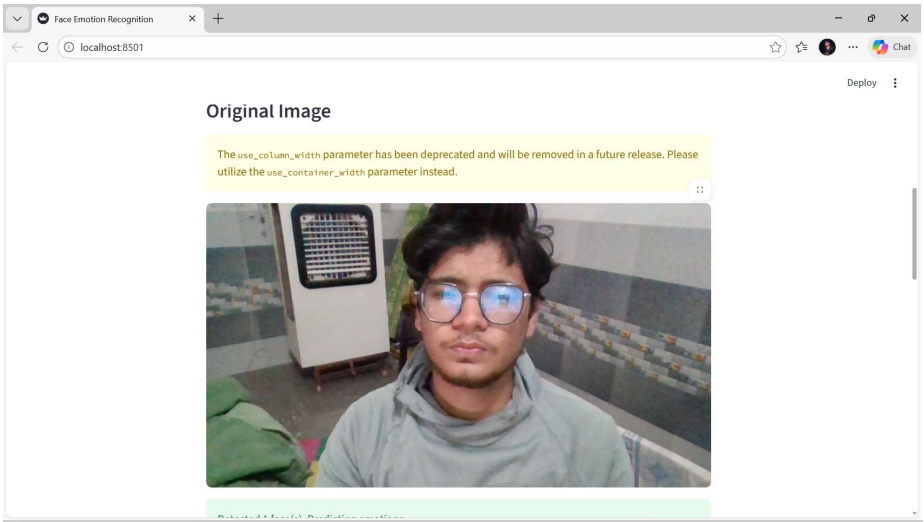


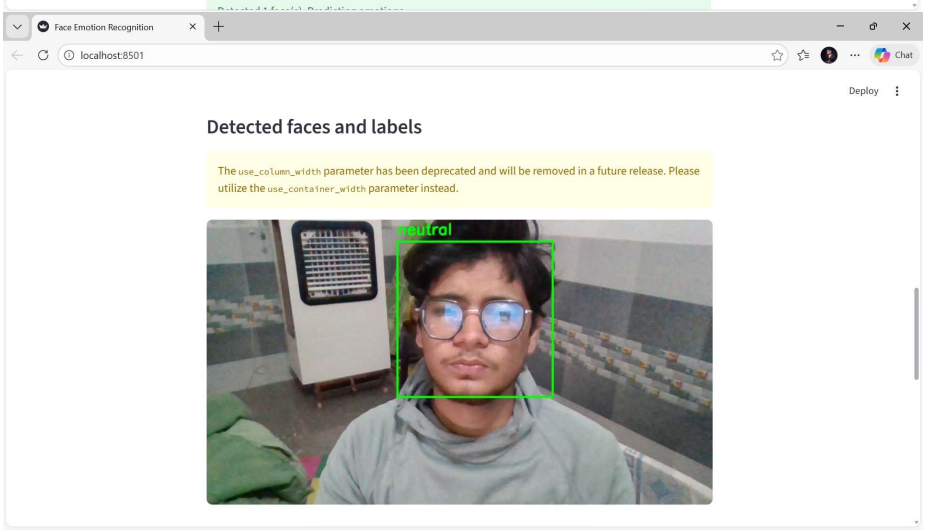
Face #1: happy

	emotion	score
0	happy	96.4654
1	neutral	3.5346
2	surprise	0.0000
3	sad	0.0000
4	angry	0.0000
5	fear	0.0000
6	disgust	0.0000



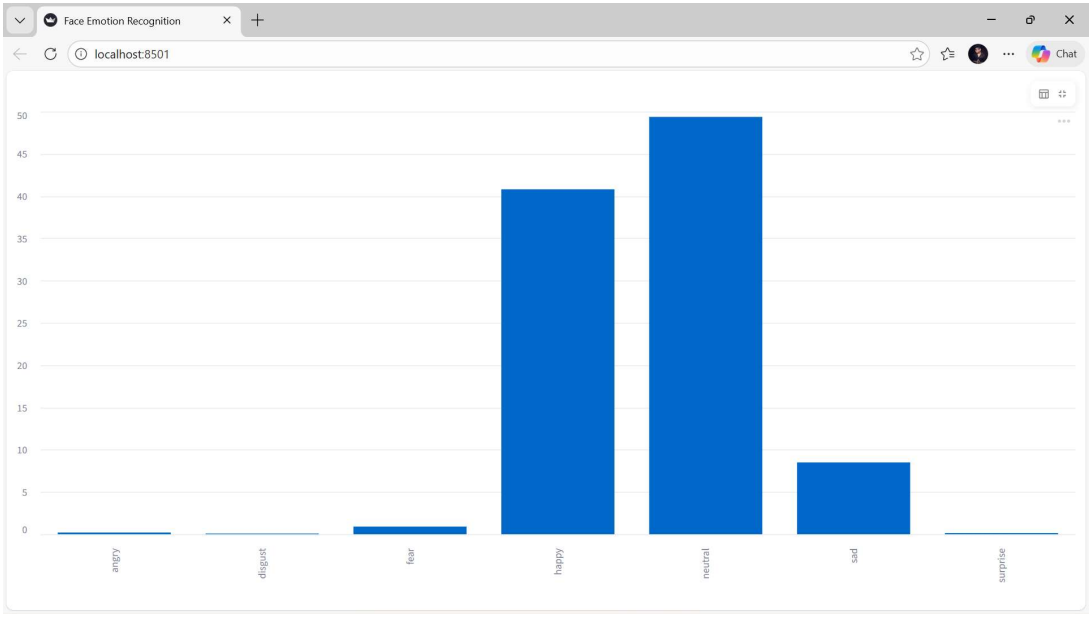
8.4 Webcam Based Emotion Recognition





Face #1: neutral

	emotion	score
0	neutral	49.3752
1	happy	40.8074
2	sad	8.4924
3	fear	0.9041
4	angry	0.2009
5	surprise	0.1378
6	disgust	0.0822



9: RESULTS AND DISCUSSION

The system was tested using multiple images and webcam captures. Faces were accurately detected, and emotions were correctly predicted under proper lighting conditions. The system supports multiple face detection and provides confidence-based emotion visualization.

All detected emotions are automatically stored in a CSV file named **results.csv**, which can be opened in Microsoft Excel for further analysis.

10: ADVANTAGES AND LIMITATIONS

Advantages

- User-friendly web interface
- Supports multiple input modes
- Uses pre-trained deep learning models
- Automatic result storage

Limitations

- Performance depends on image quality
- Accuracy may reduce in poor lighting conditions

11: CONCLUSION

The Face Emotion Recognition System successfully demonstrates the application of computer vision and deep learning techniques for emotion analysis. The project fulfills its objectives by providing accurate emotion detection along with a user-friendly interface and data storage capability.

12: FUTURE SCOPE

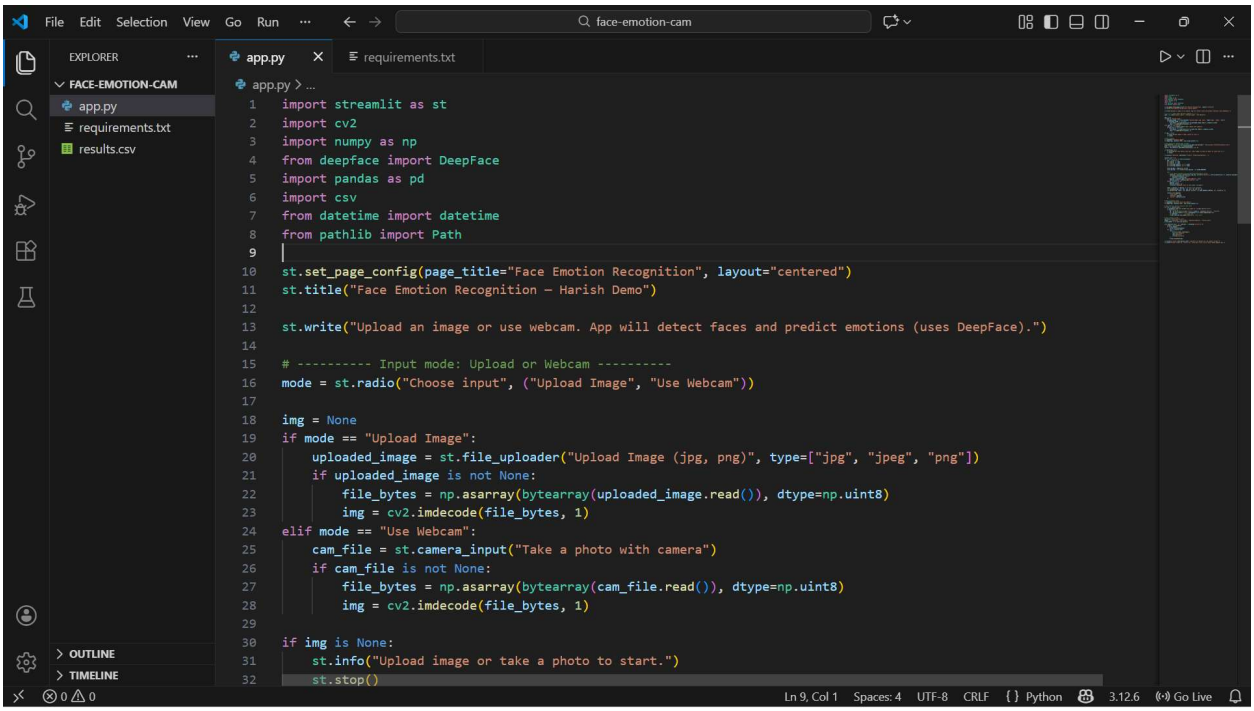
- Real-time video-based emotion recognition
- Integration with databases
- Emotion analytics dashboards
- Mobile and cloud deployment

13: REFERENCES

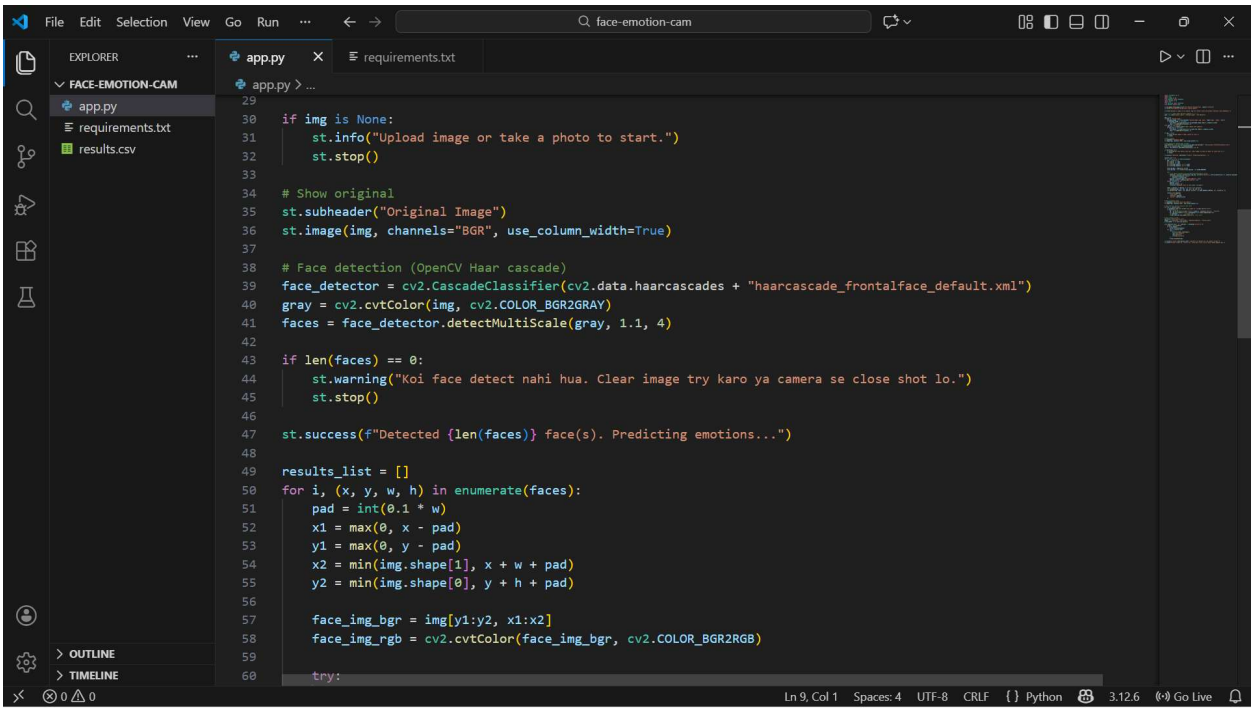
1. OpenCV Documentation
2. DeepFace Documentation
3. Streamlit Documentation
4. Research papers on Facial Emotion Recognition

SOURCE CODE

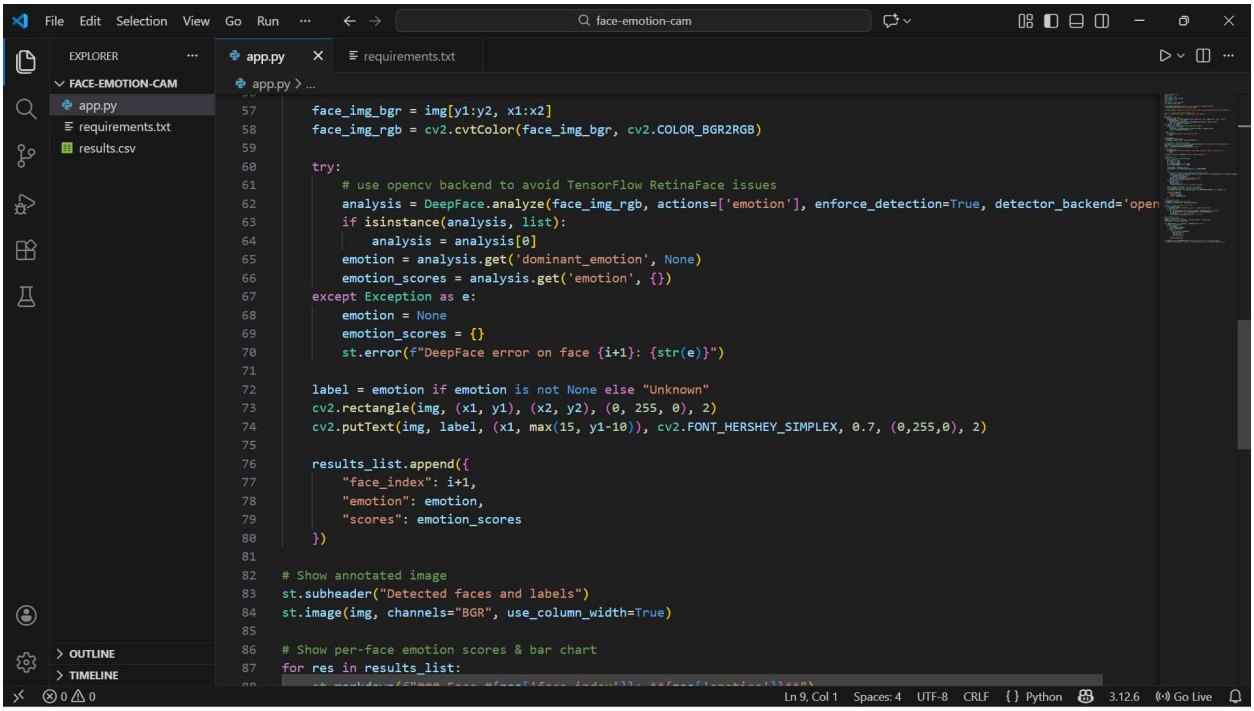
File: app.py

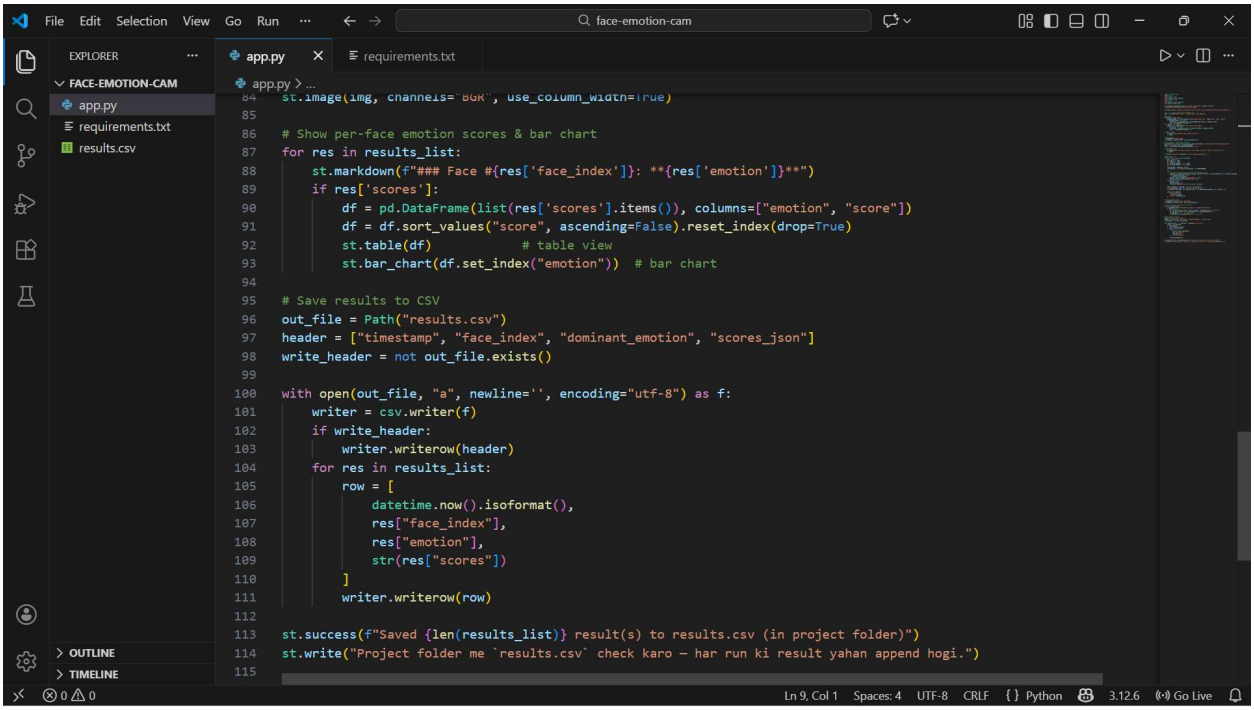


```python
import streamlit as st
import cv2
import numpy as np
from deepface import DeepFace
import pandas as pd
import csv
from datetime import datetime
from pathlib import Path

st.set_page_config(page_title="Face Emotion Recognition", layout="centered")
st.title("Face Emotion Recognition – Harish Demo")

st.write("Upload an image or use webcam. App will detect faces and predict emotions (uses DeepFace).")

# ---------- Input mode: Upload or Webcam ----------
mode = st.radio("Choose input", ("Upload Image", "Use Webcam"))

img = None
if mode == "Upload Image":
    uploaded_image = st.file_uploader("Upload Image (jpg, png)", type=["jpg", "jpeg", "png"])
    if uploaded_image is not None:
        file_bytes = np.asarray(bytearray(uploaded_image.read()), dtype=np.uint8)
        img = cv2.imdecode(file_bytes, 1)
elif mode == "Use Webcam":
    cam_file = st.camera_input("Take a photo with camera")
    if cam_file is not None:
        file_bytes = np.asarray(bytearray(cam_file.read()), dtype=np.uint8)
        img = cv2.imdecode(file_bytes, 1)

if img is None:
    st.info("Upload image or take a photo to start.")
    st.stop()
```



```python
if img is None:
    st.info("Upload image or take a photo to start.")
    st.stop()

# Show original
st.subheader("Original Image")
st.image(img, channels="BGR", use_column_width=True)

# Face detection (OpenCV Haar cascade)
face_detector = cv2.CascadeClassifier(cv2.data.haarcascades + "haarcascade_frontalface_default.xml")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_detector.detectMultiScale(gray, 1.1, 4)

if len(faces) == 0:
    st.warning("Koi face detect nahi hua. Clear image try karo ya camera se close shot lo.")
    st.stop()

st.success(f"Detected {len(faces)} face(s). Predicting emotions...")

results_list = []
for i, (x, y, w, h) in enumerate(faces):
    pad = int(0.1 * w)
    x1 = max(0, x - pad)
    y1 = max(0, y - pad)
    x2 = min(img.shape[1], x + w + pad)
    y2 = min(img.shape[0], y + h + pad)

    face_img_bgr = img[y1:y2, x1:x2]
    face_img_rgb = cv2.cvtColor(face_img_bgr, cv2.COLOR_BGR2RGB)

    try:
```


File Edit Selection View Go Run
face-emotion-cam
EXPLORER
FACE-EMOTION-CAM
app.py
requirements.txt
results.csv
app.py
requirements.txt
app.py > ...
face_img_bgr = img[y1:y2, x1:x2]
face_img_rgb = cv2.cvtColor(face_img_bgr, cv2.COLOR_BGR2RGB)
try:
# use opencv backend to avoid TensorFlow RetinaFace issues
analysis = DeepFace.analyze(face_img_rgb, actions=['emotion'], enforce_detection=True, detector_backend='open
if isinstance(analysis, list):
analysis = analysis[0]
emotion = analysis.get('dominant_emotion', None)
emotion_scores = analysis.get('emotion', {})
except Exception as e:
emotion = None
emotion_scores = {}
st.error(f"DeepFace error on face {i+1}: {str(e)}")
label = emotion if emotion is not None else "Unknown"
cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
cv2.putText(img, label, (x1, max(15, y1-10)), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0,255,0), 2)
results_list.append({
"face_index": i+1,
"emotion": emotion,
"scores": emotion_scores
})
# Show annotated image
st.subheader("Detected faces and labels")
st.image(img, channels="BGR", use_column_width=True)
# Show per-face emotion scores & bar chart
for res in results_list:
OUTLINE
TIMELINE
Ln 9, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.6 Go Live


File Edit Selection View Go Run
face-emotion-cam
EXPLORER
FACE-EMOTION-CAM
app.py
requirements.txt
results.csv
app.py
requirements.txt
app.py > ...
st.image(img, channels="BGR", use_column_width=True)
# Show per-face emotion scores & bar chart
for res in results_list:
st.markdown(f"### Face #{res['face_index']}: **{res['emotion']}**")
if res['scores']:
df = pd.DataFrame(list(res['scores'].items()), columns=["emotion", "score"])
df = df.sort_values("score", ascending=False).reset_index(drop=True)
st.table(df) # table view
st.bar_chart(df.set_index("emotion")) # bar chart
# Save results to CSV
out_file = Path("results.csv")
header = ["timestamp", "face_index", "dominant_emotion", "scores_json"]
write_header = not out_file.exists()
with open(out_file, "a", newline='', encoding="utf-8") as f:
writer = csv.writer(f)
if write_header:
writer.writerow(header)
for res in results_list:
row = [
datetime.now().isoformat(),
res["face_index"],
res["emotion"],
str(res["scores"])
]
writer.writerow(row)
st.success(f"Saved {len(results_list)} result(s) to results.csv (in project folder)")
st.write("Project folder me `results.csv` check karo — har run ki result yahan append hogi.")
OUTLINE
TIMELINE
Ln 9, Col 1 Spaces: 4 UTF-8 CRLF { } Python 3.12.6 Go Live

File: requirements.txt

streamlit

deepface

opencv-python

numpy

pandas

File: results.csv

This file stores emotion recognition results in CSV format and can be opened using Microsoft Excel.