# FACE RECOGNITION AND ATTENDANCE SYSTEM

- PRESENTED BY:V.KEERTHANA
- REG NO:613521104014
- DEPT:COMPUTER SCIENCE AND ENGINEERING

# PROBLEM STATEMENT

- Develop a robust face recognition system capable of accurately identifying individuals in real-time from a database of known faces. Integrate this system into an attendance-taking solution to automate the process of recording attendance in various settings such as classrooms, workplaces, or events. The system should be user-friendly, scalable, and capable of handling varying environmental conditions and facial expressions.

# Proposed System

- Here's a proposed system architecture for a face recognition and attendance-taking system:
- Face Detection Module: Utilize computer vision techniques to detect and locate faces within images or video streams.
- Face Recognition Module: Implement a deep learning-based model such as a convolutional neural network (CNN) to recognize and identify faces. Train the model on a dataset of known faces to learn unique facial features for each individual.
- Database Management: Maintain a database of known faces along with relevant information such as names and unique identifiers.
- Attendance Management System: Develop a user interface for administrators to manage attendance records, view attendance reports, and track attendance trends over time.
- Real-time Integration: Ensure that the system can perform face recognition and attendance tracking in real-time to provide immediate feedback to users.
- Scalability and Performance: Design the system to handle a large number of users and concurrent attendance-taking sessions efficiently.
- Security and Privacy: Implement measures to protect the privacy of individuals' facial data and ensure that the system complies with relevant data protection regulations.
- User Interface: Create user-friendly interfaces for both administrators and end-users to interact with the system seamlessly.
- Integration with Existing Systems: Provide options for integrating the face recognition and attendance-taking system with existing student information systems or human resource management systems.
- Testing and Validation: Conduct thorough testing and validation of the system to ensure accuracy, reliability, and robustness across different scenarios and environments.
- By incorporating these components into the system architecture, we can create a comprehensive face recognition and attendance-taking solution that meets the needs of various educational, organizational, and event management contexts

# SOLUTION

- Data Collection: Gather a diverse dataset of facial images representing individuals who will be recognized by the system. Ensure the dataset includes variations in lighting conditions, facial expressions, and poses.
- Preprocessing: Standardize the facial images by resizing them to a fixed size, converting them to grayscale, and performing normalization to enhance features.
- Feature Extraction: Utilize a pre-trained deep learning model (e.g., Convolutional Neural Network - CNN) to extract features from the facial images. This step transforms the raw pixel data into a compact representation that captures essential facial characteristics.
- Training: Train a machine learning model (e.g., Support Vector Machine - SVM, k-Nearest Neighbors - kNN) using the extracted features and corresponding labels (i.e., the identities of the individuals). Fine-tune the model parameters to optimize performance.
- Testing and Validation: Evaluate the trained model's performance on a separate validation dataset to assess its accuracy, precision, recall, and F1-score. Adjust the model and training process as needed to improve performance.
- Deployment: Integrate the trained model into the facial recognition system, ensuring it can efficiently process real-time facial images and make accurate predictions.Post-processing: Implement post-processing techniques to refine the recognition results, such as clustering similar faces or applying thresholds to filter out low-confidence predictions.
- Continual Improvement: Continuously update and refine the facial recognition system by collecting additional data, retraining the model with new samples, and incorporating feedback from users to enhance accuracy and adapt to evolving requirements.By following these steps, you can develop a robust facial recognition system capable of accurately identifying individuals in various real-world scenarios.

# ALGORITHM

```
import cv2
import pickle
import numpy as np
import os
video=cv2.VideoCapture(0)
facedetect=cv2.CascadeClassifier('C://Users/shali/Downloads/face_recognition_project-main/face_recognition_project-main/data/haarcascade_frontalface_default.xml')


faces_data=[]

i=0

name=input("Enter Your Name: ")

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.3 ,5)
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]
        resized_img=cv2.resize(crop_img, (50,50))
```

```
• if len(faces_data)<=100 and i%10==0:
•             faces_data.append(resized_img)
•         i=i+1
•         cv2.putText(frame, str(len(faces_data)), (50,50), cv2.FONT_HERSHEY_COMPLEX, 1, (50,50,255), 1)
•         cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)
•     cv2.imshow("Frame",frame)
•     k=cv2.waitKey(1)
•     if k==ord('q') or len(faces_data)==100:
•         break
• video.release()
• cv2.destroyAllWindows()

• faces_data=np.asarray(faces_data)
• faces_data=faces_data.reshape(100, -1)


• if 'n.pkl' not in os.listdir('C://Users/shali/Downloads/face_recognition_project-main/face_recognition_project-main/data/'):
•     names=[name]*100
```

# continue..

```
with open('n.pkl', 'wb') as f:
        pickle.dump(names, f)
else:
    with open('n.pkl', 'rb') as f:
        names=pickle.load(f)
    names=names+[name]*100
    with open('n.pkl', 'wb') as f:
        pickle.dump(names, f)

if 'f.pkl' not in os.listdir('C://Users/shali/Downloads/face_recognition_project-main/face_recognition_project-main/data/'):
    with open('f.pkl', 'wb') as f:
        pickle.dump(faces_data, f)
else:
    with open('f.pkl', 'rb') as f:
        faces=pickle.load(f)
    faces=np.append(faces, faces_data, axis=0)
    with open('f.pkl', 'wb') as f:
        pickle.dump(faces, f)
```

```
•  2.Test.py
•  from sklearn.neighbors import KNeighborsClassifier
•  import cv2
•  import pickle
•  import numpy as np
•  import os
•  import csv
•  import time
•  from datetime import datetime


•  from win32com.client import Dispatch

•  def speak(str1):
•      speak=Dispatch(("SAPI.SpVoice"))
•      speak.Speak(str1)

•  video=cv2.VideoCapture(0)
•  facedetect=cv2.CascadeClassifier('C://Users/shali/Downloads/face_recognition_project-main/face_recognition_project-main/data/haarcascade_frontalface_default.xml')
```

Continue…

```
with open('n.pkl', 'rb') as w:
    LABELS=pickle.load(w)
with open('f.pkl', 'rb') as f:
    FACES=pickle.load(f)

print('Shape of Faces matrix --> ', FACES.shape)

knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(FACES, LABELS)

imgBackground=cv2.imread("C://Users/shali/Downloads/face_recognition_project-main/face_recognition_project-main/background.png")

COL_NAMES = ['NAME', 'TIME']

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces=facedetect.detectMultiScale(gray, 1.3 ,5)
```

# Continue..

```
    for (x,y,w,h) in faces:
        crop_img=frame[y:y+h, x:x+w, :]
        resized_img=cv2.resize(crop_img, (50,50)).flatten().reshape(1,-1)
        output=knn.predict(resized_img)
        ts=time.time()
        date=datetime.now().strftime("%d-%m-%Y")
        timestamp=datetime.now().strftime("%H:%M-%S")
        exist=os.path.isfile("C://Users/shali/Downloads/face_recognition_project-main/attendance" + date + ".csv")
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)
        cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)
        cv2.putText(frame, str(output[0]), (x,y-15), cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255), 1)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (50,50,255), 1)
        attendance=[str(output[0]), str(timestamp)]
      imgBackground[162:162 + 480, 55:55 + 640] = frame
      cv2.imshow("Frame",imgBackground)
      k=cv2.waitKey(1)
      if k==ord('o'):
        speak("Attendance Taken..")
        time.sleep(5)
```

Continue…

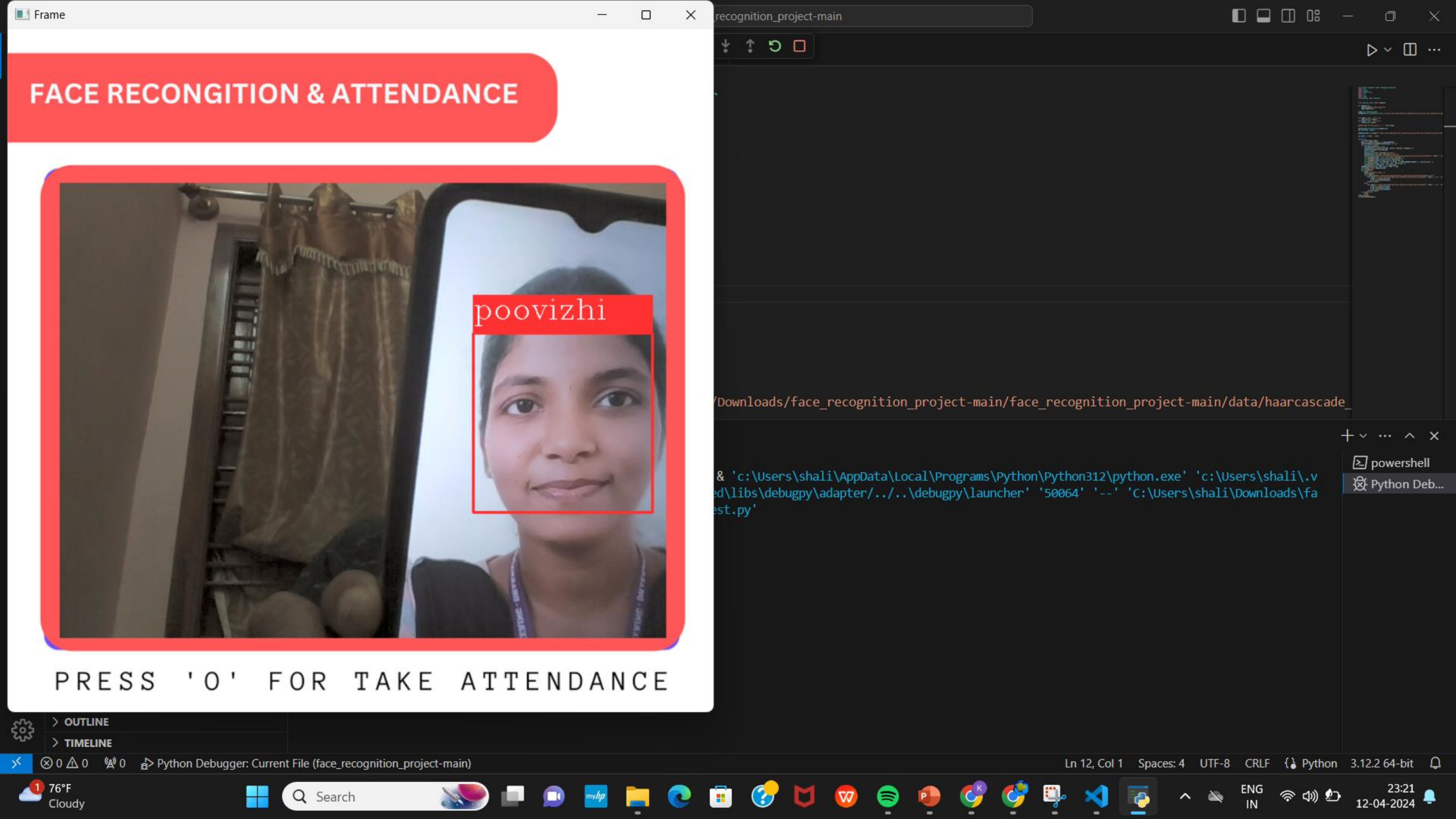
```
if os.path.exists("C://Users/shali/Downloads/face_recognition_project-main/attendance.csv"):
      with open("C://Users/shali/Downloads/face_recognition_project-main/attendance" + date
+ ".csv", "+a",newline='') as csvfile:
        writer=csv.writer(csvfile)
        writer.writerow(attendance)
      csvfile.close()
    else:
      with open("C://Users/shali/Downloads/face_recognition_project-main/attendance" + date
+ ".csv", "+a",newline='') as csvfile:
        writer=csv.writer(csvfile)
        writer.writerow(COL_NAMES)
        writer.writerow(attendance)
      csvfile.close()
  if k==ord('q'):
    break
video.release()
cv2.destroyAllWindows()
```

3.App.py

```
import streamlit as st
import pandas as pd
import time
from datetime import datetime

ts=time.time()
date=datetime.fromtimestamp(ts).strftime("%d-%m-%Y")
timestamp=datetime.fromtimestamp(ts).strftime("%H:%M-%S")

from streamlit_autorefresh import st_autorefresh

count = st_autorefresh(interval=2000, limit=100, key="fizzbuzzcounter")

if count == 0:
    st.write("Count is zero")
elif count % 3 == 0 and count % 5 == 0:
    st.write("FizzBuzz")
elif count % 3 == 0:
    st.write("Fizz")
elif count % 5 == 0:
    st.write("Buzz")
```

Continue…

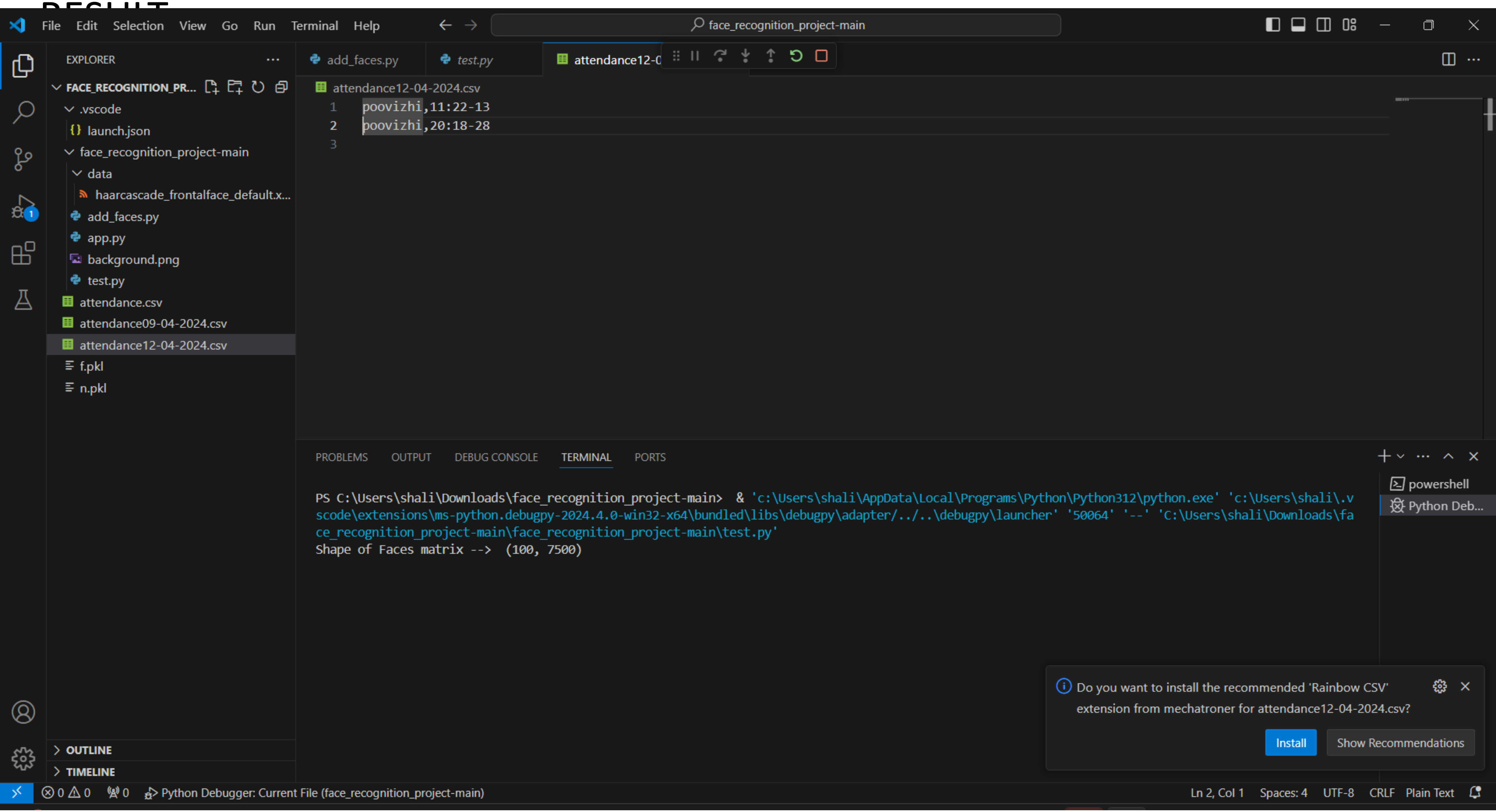
```
else:
    st.write(f"Count: {count}")



df=pd.read_csv("C://Users/shali/Downloads/face_recognition_project-main/attendance" + date + ".csv")


st.dataframe(df.style.highlight_max(axis=0))
```



```
/Downloads/face_recognition_project-main/face_recognition_project-main/data/haarcascade_
```

```
& 'c:\Users\shali\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\shali\.v
ed\libs\debugpy\adapter/../..\debugpy\launcher' '50064' '--' 'C:\Users\shali\Downloads\fa
est.py'
```

# RESULT



attendance12-04-2024.csv

```
poovizhi,11:22-13
poovizhi,20:18-28

```

```
PS C:\Users\shali\Downloads\face_recognition_project-main>  & 'c:\Users\shali\AppData\Local\Programs\Python\Python312\python.exe' 'c:\Users\shali\.v
scode\extensions\ms-python.debugpy-2024.4.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '50064' '--' 'C:\Users\shali\Downloads\fa
ce_recognition_project-main\face_recognition_project-main\test.py'
Shape of Faces matrix -->  (100, 7500)
```

Do you want to install the recommended 'Rainbow CSV' extension from mechatroner for attendance12-04-2024.csv?

- Conclusion

A face recognition and attendance taking system offers several advantages such as increased accuracy, efficiency, and security. By automating the attendance process, it reduces the likelihood of errors and eliminates the need for manual input. Additionally, it provides real-time tracking and monitoring capabilities. Overall, the implementation of such a system can significantly streamline administrative tasks and enhance overall operational efficiency.

- REFERENCES

- https://github.com/KeerthanaV20/Face-recognition-and-attendance-system-