Task 06

Explanation of code Face Profiling System:

1. Introduction

The **Face Profiling System** is a web-based application that analyzes facial features from an uploaded image and predicts personality traits based on predefined rules. The system uses:

- **Flask (Python)** for backend processing
- **OpenCV** for face detection
- **dlib** for facial landmark extraction
- **HTML, CSS, and JavaScript** for a user-friendly frontend

2. System Overview

The system follows these steps:

1. **User uploads an image** via the frontend.
2. **Face detection** is performed using OpenCV's **Haar Cascade classifier**.
3. **Facial landmarks are extracted** using **dlib's shape predictor**.
4. **Facial feature ratios** (e.g., width-height ratio, eye distance, mouth width) are calculated.
5. **Personality traits** are predicted based on these facial measurements.
6. **Results are displayed** on the frontend.

3. Key Components

Flask Backend (app.py)

The backend is responsible for:
Processing uploaded images
Detecting facial landmarks
Extracting key facial features
Predicting personality traits
Returning results as a JSON response

Main Functions:

- detect_face(image): Identifies a face in the image using OpenCV.
- get_landmarks(image, face_rect): Extracts **68 facial landmarks** using dlib.
- calculate_features(landmarks): Computes facial proportions.
- predict_personality(features): Maps facial features to personality traits.
- analyze_face(): API route that handles image processing and returns results.

3. Frontend (index.html)

The frontend provides an **interactive UI** for users to upload images and view results.

Features:

- File upload interface
- Live image preview
- Loading animation while processing
- Display of facial feature analysis and personality predictions

JavaScript Functionalities:

- Reads uploaded images and previews them.
- Sends images to Flask backend via fetch() API.
- Displays results dynamically after processing.

4. Technologies Used

Component	Technology Used
Backend	Flask (Python)
Face Detection	OpenCV Haar Cascade
Landmark Detection	dlib Shape Predictor
Frontend	HTML, CSS, JavaScript

5. Face Detection & Landmark Extraction

Haar Cascade Classifier (Face Detection)

- The system uses OpenCV's Haar Cascade classifier (haarcascade_frontalface_default.xml) to detect faces.
- It identifies face regions by detecting patterns of light and dark pixels.
- The classifier is pre-trained and included in OpenCV's default data.

Shape Predictor (Facial Landmark Detection)

- **Dlib's shape_predictor_68_face_landmarks.dat** extracts 68 key facial landmarks.
- These points define **eyes, nose, mouth, and jawline**.
- This allows precise measurement of facial ratios.

6. Personality Prediction Mapping

The system maps facial features to personality traits based on predefined logic.

Facial Feature	Influenced Traits
Face Width/Height Ratio	Extraversion, Neuroticism
Eye Distance Ratio	Openness, Conscientiousness
Mouth Width Ratio	Extraversion, Neuroticism
Jaw Width Ratio	Agreeableness, Conscientiousness
Chin Prominence	Conscientiousness, Neuroticism

7. Conclusion

This Face Profiling System demonstrates computer vision techniques for analyzing facial features and estimating personality traits. While the system is not a scientific psychological tool, it effectively showcases face detection, landmark extraction, and rule-based personality prediction using Flask, OpenCV, and dlib.

Output: