Cyber Security Project : Web-Based Facial Authentication Systems

Yosri Zayani
Nada Hermi
Firas Amara
Mohammad Hajaj

I. Programming language , libraries and frameworks:

- *Python*: The core programming language
- *Tkinter*: Provides a user-friendly graphical interface (GUI) for user interaction using elements like buttons, text boxes, and labels.
- *OpenCV* : Open source computer vision library used for face detection, landmark identification, and image processing.
- *face_recognition*: Built on top of OpenCV, simplifies facial recognition tasks like face location, feature extraction, and face recognition using pre-trained models.
- pyMongo : Stores python objects in MongoD

OpenCV:

- **Functionality**: OpenCV is primarily used for image and video processing tasks in computer vision applications.

Video Capture: Use OpenCV to capture video feed from the webcam.
Python code :

```
cap = cv2.VideoCapture(0)

ret, frame = cap.read()
```

Face Detection: Implement face detection using OpenCV's pre-trained *Haar cascades* (machine learning-based object detection algorithm that identifies objects in images or video streams by analyzing patterns known as Haar-like features within sliding windows.) or

deep learning-based models.
Python code :

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades
+ 'haarcascade_frontalface_default.xml')

faces = face_cascade.detectMultiScale(gray,
scaleFactor=1.1, minNeighbors=5)
```

Drawing Bounding Boxes: Visualize the detected faces by drawing bounding boxes around them.
python

```
for (x, y, w, h) in faces:

    cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
```

Recognizing Faces: Compare the face encodings of detected faces with known faces to recognize them.
python

```
face_locations = face_recognition.face_locations(rgb_frame)

face_encodings = face_recognition.face_encodings(rgb_frame,
face_locations)
```

TensorFlow / PyTorch

Functionality: Deep learning frameworks for building, training, and deploying neural networks.

II. Development Process:

1. **Deep Learning Models**: Train and deploy deep learning models for face recognition tasks.
   Python code :

```
import tensorflow as tf

# or

import torch
```

Development Phases:

1. **Setup Environment**:
   - Install Python and required libraries (OpenCV, face recognition library).
   - Connect and test your webcam to ensure it's working properly.
2. **Face Detection**:
   - Use OpenCV to capture video feed from the webcam.
   - Implement face detection using pre-trained models provided by OpenCV or other libraries.
   - Draw bounding boxes around detected faces to visualize the detection process.
3. **Face Recognition**:
   - Collect and preprocess images of intended users for training.
   - Train a face recognition model using pre-trained embeddings (face_recognition librairy) or train from scratch using your dataset.
   - Implement face recognition on the live video feed to recognize known faces.
   - Associate names or IDs with recognized faces and display them.
   - Detect whether the face in front of the machine is fake or real (Silent-Face-Anti-Spoofing technology)
4. **Geometry Analysis**:
   - Extract facial landmarks using the face detection model.
   - Compute geometric features such as distances between eyes, nose width, etc.
   - Use these features to create a faceprint or numerical code representing each face.
   - Save detected faces into a database (using pickle serialization)
5. **Matching and Authentication**:

   - Compare the faceprints of detected faces with the faceprints stored in the database.
   - Implement a matching algorithm (e.g., Euclidean distance, cosine similarity) to determine the similarity between faceprints.
   - Set a threshold for similarity scores to authenticate users.
6. **Integration and Testing**:
   - Integrate all components into a unified system.
   - Test the system rigorously under various lighting conditions, angles, and environments to ensure robustness.
   - Handle edge cases and fine-tune parameters as necessary.
7. **Deployment and Scaling**:
   - Deploy the system on your desired platform (e.g., local machine, server, cloud).
   - Ensure scalability and efficiency for handling multiple users simultaneously.
   - Implement logging and monitoring for tracking system performance and user activities.
8. **Documentation and Maintenance**:
   - Document the project including setup instructions, usage guidelines, and code documentation.
   - Provide support and maintenance for the deployed system, including updates and bug fixes.