ARTIFICIAL INTELLIGENCE LAB PROJECT REPORT

UCS411 -Artificial Intelligence

Fourth – Semester

Submitted to: Ms. Chandni Kumari

Submitted by:

Daksh Kumar Nahar 102103481

Stuti Mittal 102103485

Selina Varshney 102103496

Batch: 2CO18

BE Second Year, COE

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Problem statement:

To develop a security system to authenticate the fingerprint of the user from the dataset

Description of problem:

Fingerprint recognition is a type of biometric technology that uses artificial intelligence to identify individuals based on their unique fingerprints. This project aims to authenticate access to the user, if the fingerprint matches the dataset provided by the organization. It displays the sample fingerprint which the user inputs and then compares it to the dataset, providing the fingerprint it matches to, hence creating an efficient security system. It also prompts a message when access is given, and when access is denied.

Applications of fingerprint recognition in AI include security systems for access control, criminal identification, and forensic analysis. Fingerprint recognition can also be used in mobile devices for authentication purposes, such as unlocking a smartphone or making a payment.

Approach to the problem:

- Generate a dataset for the organization.
- Collect Sample fingerprints to be tested
- Import the required python modules and write the appropriate code.
- Compare the sample to the dataset.
- Print result.

Code and Explanation

The project typically involves the following steps:

- Data collection: A large dataset of fingerprints is collected. This dataset is used to train the AI model We have used the following dataset from Kaggle: https://www.kaggle.com/datasets/ruizgara/socofing
- Feature extraction: Key features of the fingerprint, such as ridge endings and bifurcations, are extracted using image processing techniques.
- Model training: A deep learning model, such as a convolutional neural network (CNN), is trained on the extracted features to identify unique patterns in the fingerprint.
- Evaluation: The trained model is evaluated on a test dataset to measure its accuracy and performance.

IMPORTING USED MODULES

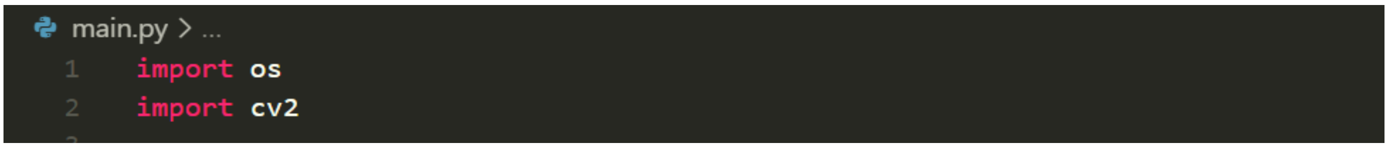
Using `pip install opencv-python`

`pip install os_sys`

Brief description of the imported modules:

1. **OS:** OS comes under Python's standard utility modules. This module provides a portableway of using operating system-dependent functionality. The OS module in Python provides functions for creating and removing a directory (folder), fetching its contents, changing and identifying the current directory, etc.

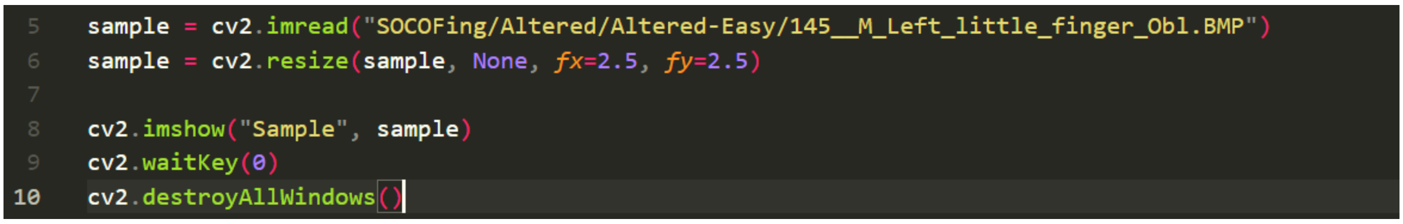
2. **CV2:** OpenCV-Python is a library of Python bindings designed to solve computer vision problems. cv2. imshow() method is used to display an image in a window. The window automatically fits to the image size.



main.py > ...

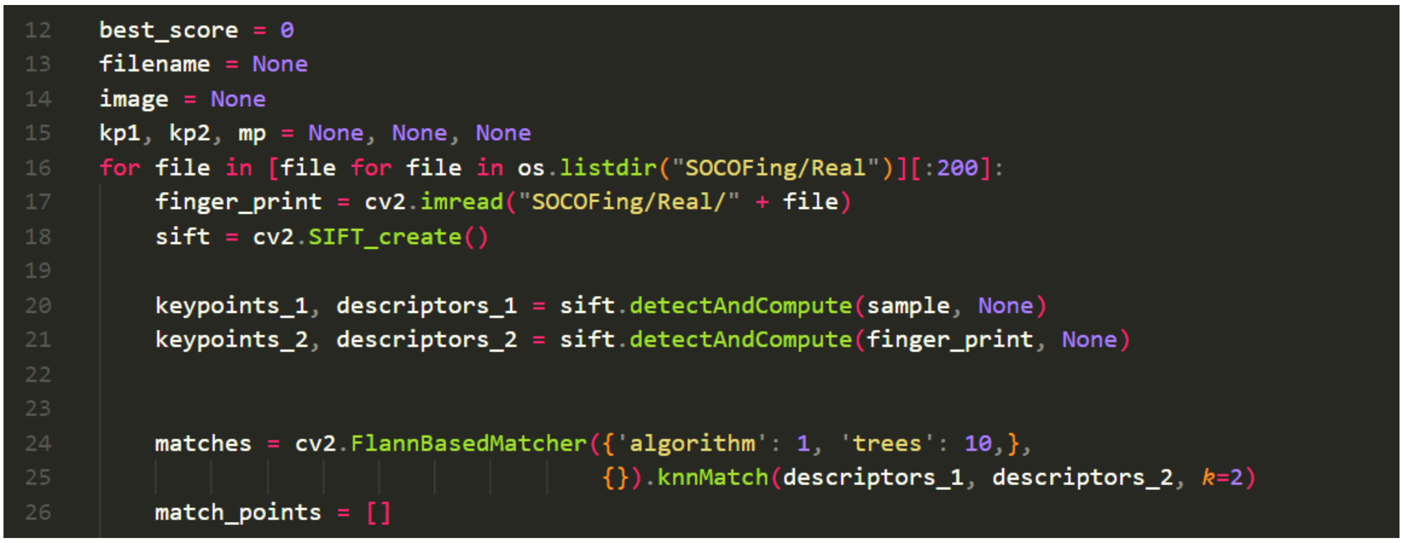
```python
import os
import cv2
```

Program:



```python
sample = cv2.imread("SOCOFing/Altered/Altered-Easy/145__M_Left_little_finger_Obl.BMP")
sample = cv2.resize(sample, None, fx=2.5, fy=2.5)

cv2.imshow("Sample", sample)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Taking a sample from the altered dataset, which contains rotated, blurred , faded versions of the real fingerprints, and using cv2 .imread() to read and cv2.resize() to resize it, then using cv2.imshow() to display it to the user.



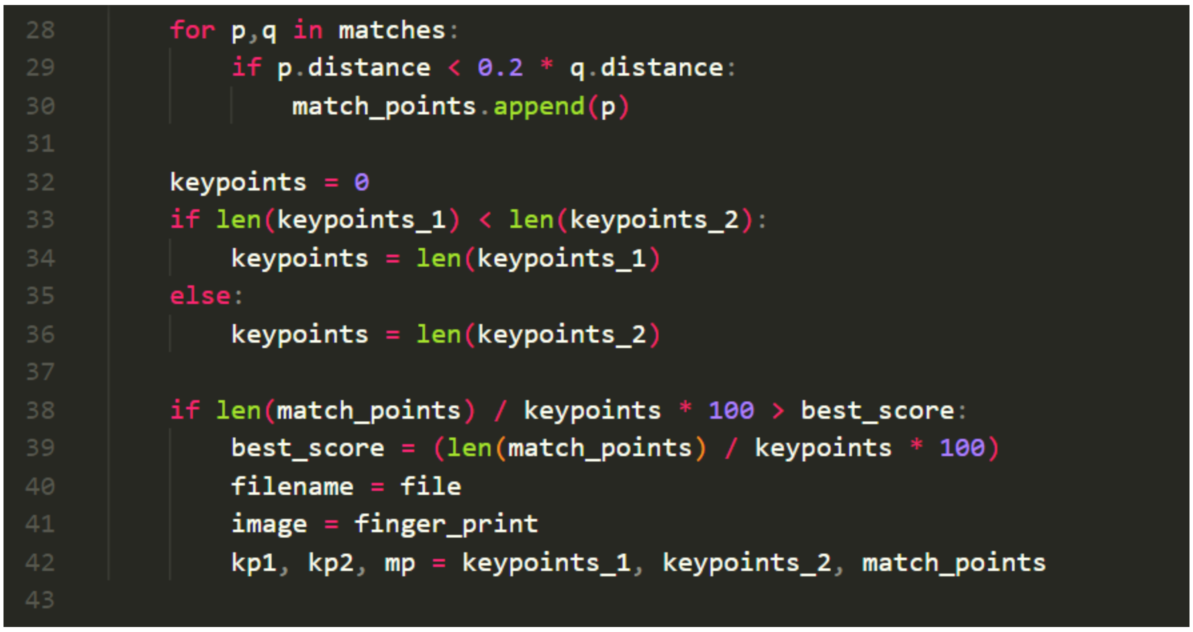
```python
best_score = 0
filename = None
image = None
kp1, kp2, mp = None, None, None
for file in [file for file in os.listdir("SOCOFing/Real")][:200]:
    finger_print = cv2.imread("SOCOFing/Real/" + file)
    sift = cv2.SIFT_create()

    keypoints_1, descriptors_1 = sift.detectAndCompute(sample, None)
    keypoints_2, descriptors_2 = sift.detectAndCompute(finger_print, None)


    matches = cv2.FlannBasedMatcher({'algorithm': 1, 'trees': 10,},
                                    {}).knnMatch(descriptors_1, descriptors_2, k=2)
    match_points = []
```

These lines initialize variables that will be used in the subsequent loop that iterates through all the fingerprint images in the dataset. best_score is initialized to 0, filename, image, kp1, kp2, and mp are initialized to None. These variables will be used to store the best matching fingerprint image and corresponding key points and match points.

Here SIFT (Scale-Invariant Feature Transform) object is used using the cv2.SIFT_create() function. SIFT is a feature detection algorithm used to detect and describe local features in images.

Next lines detects and compute the key points and descriptors for both the sample and finger_print images using the detectAndCompute() function of the SIFT object.

Next it matches the key points and descriptors of the sample and finger_print images using a Flann-based matcher. Flann is a fast approximate nearest neighbour algorithm. The knnMatch() function takes in the descriptors of the two images and matches them based on their Euclidean distance. The {'algorithm': 1, 'trees': 10} argument specifies the parameters for the Flann algorithm used.



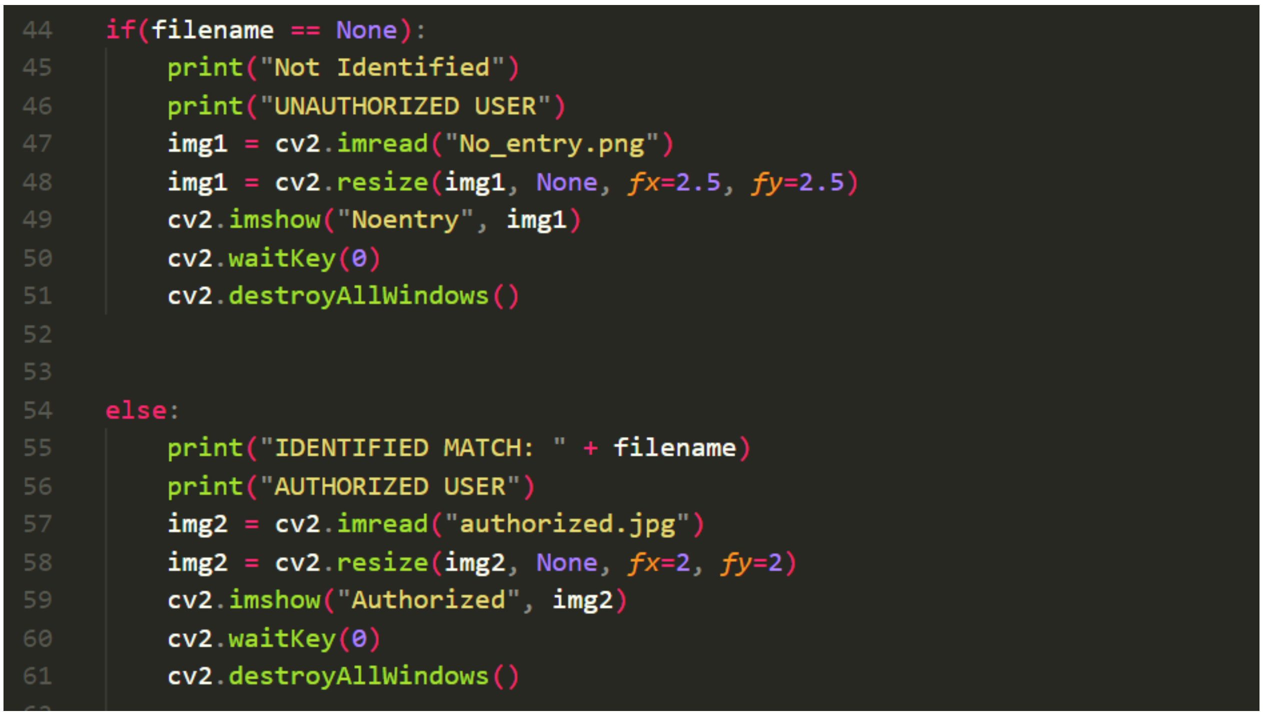
```
    for p,q in matches:
        if p.distance < 0.2 * q.distance:
            match_points.append(p)

    keypoints = 0
    if len(keypoints_1) < len(keypoints_2):
        keypoints = len(keypoints_1)
    else:
        keypoints = len(keypoints_2)

    if len(match_points) / keypoints * 100 > best_score:
        best_score = (len(match_points) / keypoints * 100)
        filename = file
        image = finger_print
        kp1, kp2, mp = keypoints_1, keypoints_2, match_points
```

Here on first line, it iterates over each match in "matches" and assigns the two nearest matches to the variables "p" and "q".

This loop iterates over the matches found between two sets of keypoints (features) extracted from two fingerprint images. For each match, it checks whether the distance between the two matched points (p and q) is less than 20% of the distance between the next closest match. If this condition is met, the current match point (p) is added to a list called match_points.

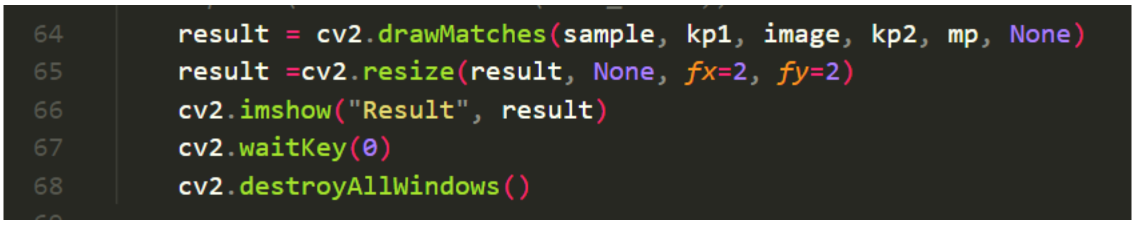
Next the code initializes a variable called keypoints to 0 and then sets its value to the number of keypoints in the image with fewer keypoints. This is done so that the similarity score is not biased towards images with more keypoints.

This if statement computes the similarity score between two fingerprint images based on the number of keypoints matched. If the similarity score is greater than the current best score, it updates the best score and saves the current image, keypoints, and matched points.



```python
if(filename == None):
    print("Not Identified")
    print("UNAUTHORIZED USER")
    img1 = cv2.imread("No_entry.png")
    img1 = cv2.resize(img1, None, fx=2.5, fy=2.5)
    cv2.imshow("Noentry", img1)
    cv2.waitKey(0)
    cv2.destroyAllWindows()


else:
    print("IDENTIFIED MATCH: " + filename)
    print("AUTHORIZED USER")
    img2 = cv2.imread("authorized.jpg")
    img2 = cv2.resize(img2, None, fx=2, fy=2)
    cv2.imshow("Authorized", img2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

This block of code displays a message and an image based on whether the fingerprint image was matched or not. If the image was not matched, it displays a "Not Identified" message, along with an image indicating that the user is unauthorized. If the image was matched, it displays an "IDENTIFIED MATCH" message, along with an image indicating that the user is authorized.



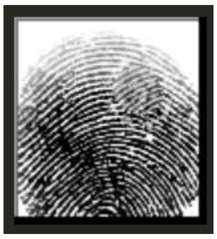
```python
    result = cv2.drawMatches(sample, kp1, image, kp2, mp, None)
    result =cv2.resize(result, None, fx=2, fy=2)
    cv2.imshow("Result", result)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

This code block creates an image called result that shows the matched keypoints between the sample image and the fingerprint image. It then resizes the image and displays it.

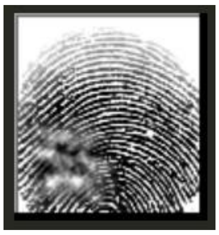
OUTPUT:

Our database contains a set of altered images:

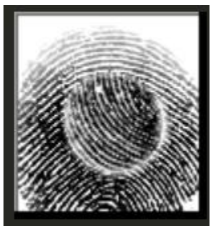
Altered-easy:



Altered-medium:



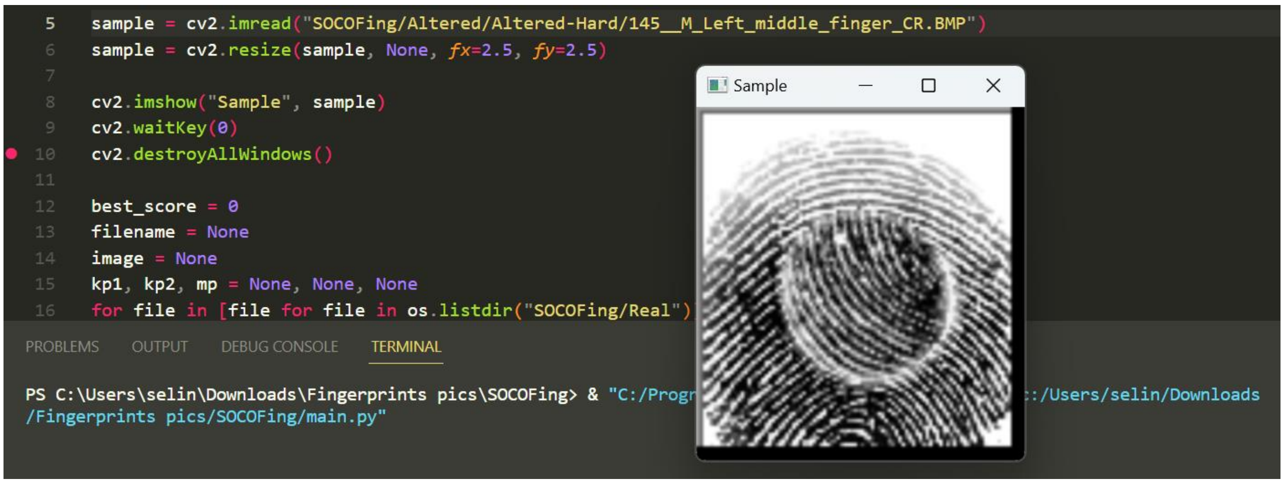
Altered-hard:



These images are faded, blurred, or rotated versions of the real images- which has a variety of fingerprints-left thumb, middle finger, right little finger, etc.

AUTHORIZED ACCESS:

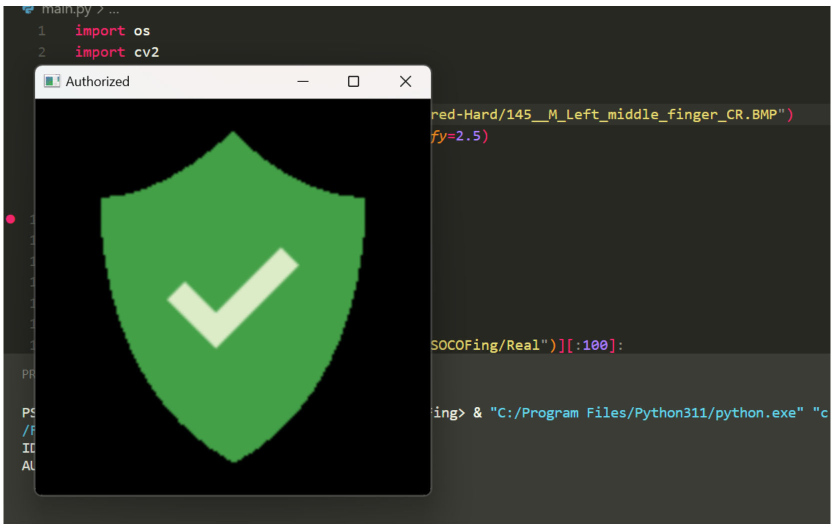
Taking an altered hard image as the sample, which is present in the real database:



```python
sample = cv2.imread("SOCOFing/Altered/Altered-Hard/145__M_Left_middle_finger_CR.BMP")
sample = cv2.resize(sample, None, fx=2.5, fy=2.5)

cv2.imshow("Sample", sample)
cv2.waitKey(0)
cv2.destroyAllWindows()

best_score = 0
filename = None
image = None
kp1, kp2, mp = None, None, None
for file in [file for file in os.listdir("SOCOFing/Real")
```

```
PS C:\Users\selin\Downloads\Fingerprints pics\SOCOFing> & "C:/Progr                     :/Users/selin/Downloads
/Fingerprints pics/SOCOFing/main.py"
```

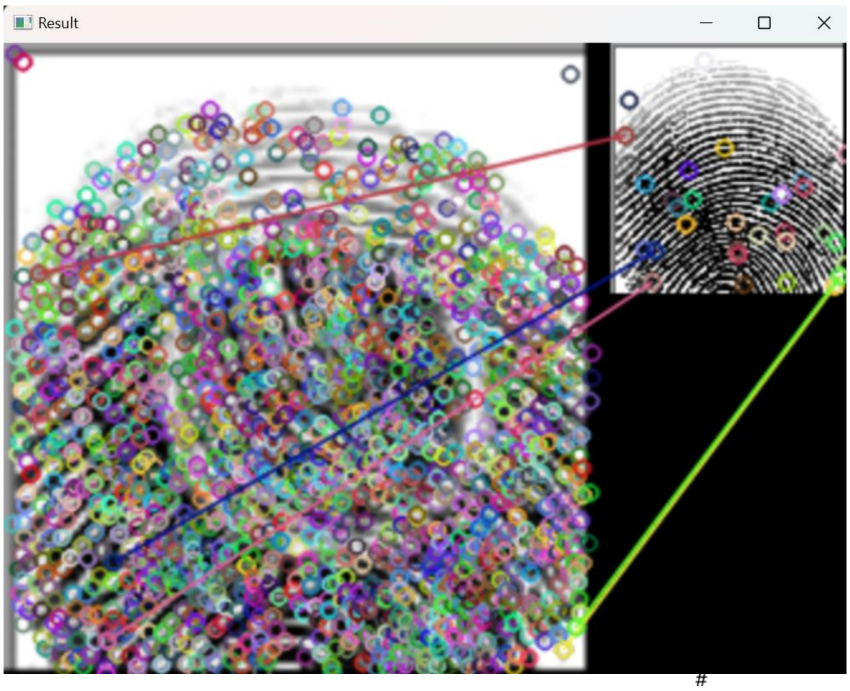
The program displays:



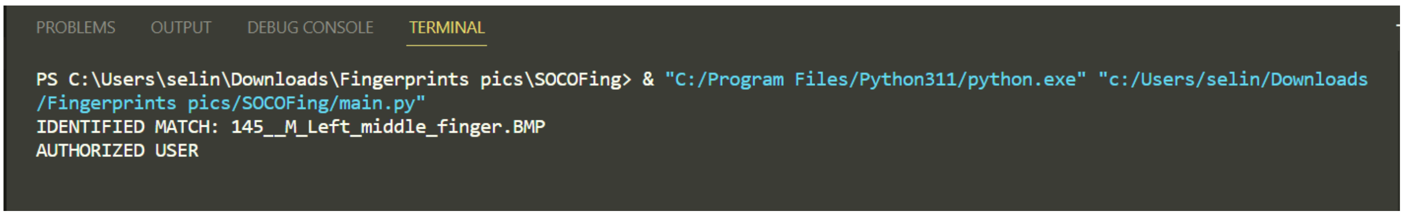
```python
import os
import cv2
```

```
red-Hard/145__M_Left_middle_finger_CR.BMP")
fy=2.5)
SOCOFing/Real")][:100]:
ing> & "C:/Program Files/Python311/python.exe" "c
```

It also displays matching of points:



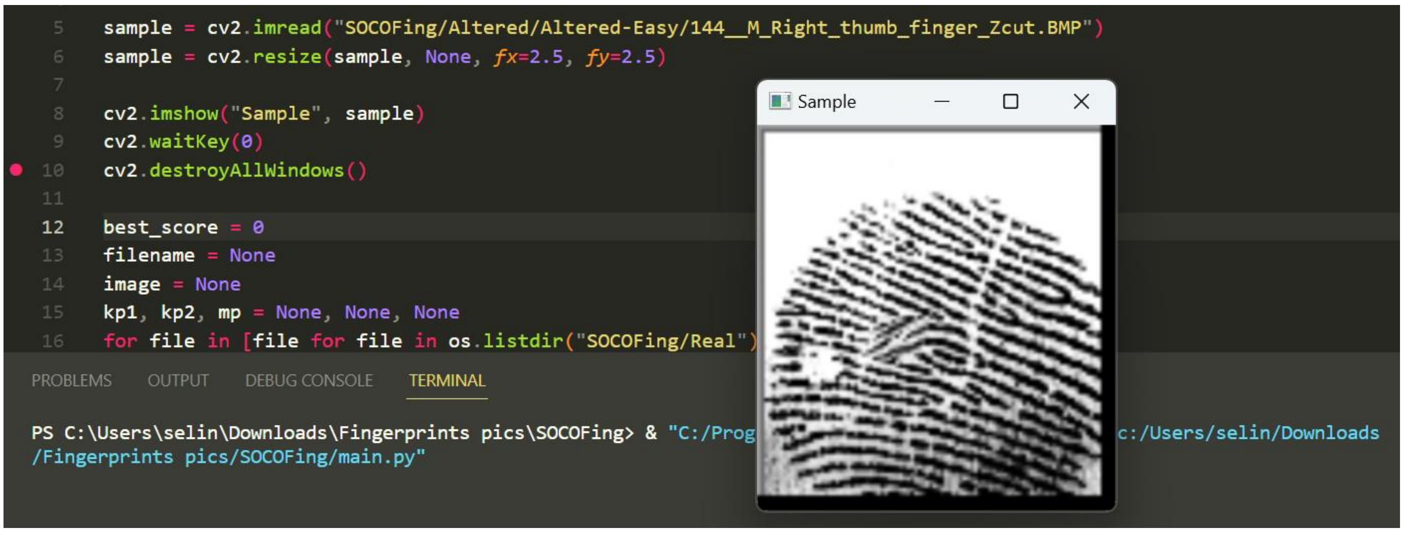
It further displays on the terminal, which image in the real dataset, the sample matches to and shows its name as IDENTIFIED MATCH. It also displays a message in the terminal that the user is authorized.



```
PS C:\Users\selin\Downloads\Fingerprints pics\SOCOFing> & "C:/Program Files/Python311/python.exe" "c:/Users/selin/Downloads
/Fingerprints pics/SOCOFing/main.py"
IDENTIFIED MATCH: 145__M_Left_middle_finger.BMP
AUTHORIZED USER
```

UNAUTHORIZED ACCESS:

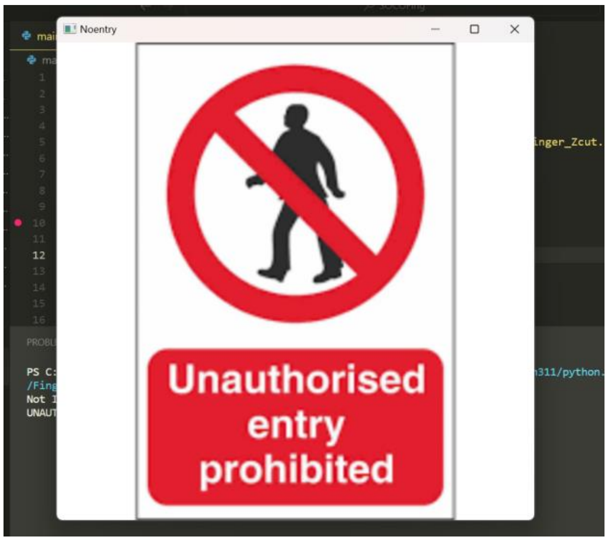
Taking an image, which is not present in the real dataset:



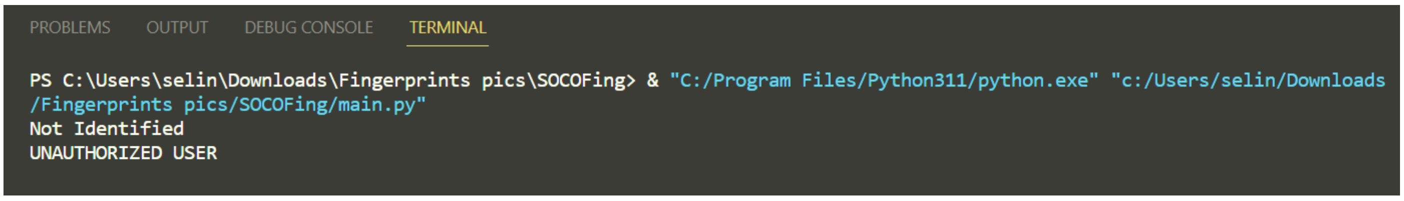
```python
sample = cv2.imread("SOCOFing/Altered/Altered-Easy/144__M_Right_thumb_finger_Zcut.BMP")
sample = cv2.resize(sample, None, fx=2.5, fy=2.5)

cv2.imshow("Sample", sample)
cv2.waitKey(0)
cv2.destroyAllWindows()

best_score = 0
filename = None
image = None
kp1, kp2, mp = None, None, None
for file in [file for file in os.listdir("SOCOFing/Real")
```

```
PS C:\Users\selin\Downloads\Fingerprints pics\SOCOFing> & "C:/Prog                    c:/Users/selin/Downloads
/Fingerprints pics/SOCOFing/main.py"
```

The program displays:



It further displays “Not identified, UNAUTHORIZED USER” on the terminal.



```
PS C:\Users\selin\Downloads\Fingerprints pics\SOCOFing> & "C:/Program Files/Python311/python.exe" "c:/Users/selin/Downloads
/Fingerprints pics/SOCOFing/main.py"
Not Identified
UNAUTHORIZED USER
```

RESULT:

The Program successfully identifies authorized and unauthorized users, hence providing an efficient solution to security.

We as Computer Engineering students, thoroughly enjoyed exploring python by the means of this project.

Project link:

Provided below is the drive link for our project, with all the dataset and the main python file- main.py which can be downloaded and run on applications like VS code:

https://drive.google.com/drive/folders/1nzwnYUcr_e8rxTTQcAO1G7SuAmvVqnEm?usp=share_link