

Project Report
On
FACE RECOGNITION FOR BANKING SECURITY
Submitted by

ID NO: R171049, P.PAVANA NARASIMHA PRATHAP,

ID NO: R170396, C.LOKESH KUMAR REDDY,

ID: R170481, J.SUCHITRA.

Under the guidance of
RATNA KUMARI CHALLA

Assistant Professor, CSE



Rajiv Gandhi University of Knowledge and Technologies(RGUKT),

R.K.Valley, Kadapa, Andhra Pradesh



**Rajiv Gandhi University of Knowledge and Technologies(RGUKT),
R.K.Valley, Kadapa(Dist), Andhra Pradesh, 516330.**

CERTIFICATE

This is to certify that the project work titled “**FACE RECOGNITION FOR BANKING SECURITY**” is a bonafide project work submitted by **C.LOKESH KUMAR REDDY, P.PAVANA NARASIMHA PRATHAP ,J.SUCHITRA** in the department of COMPUTER SCIENCE AND ENGINEERING in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Computer science and engineering for the year 2021-2022 carried out the work under the supervision

GUIDE

RATNA KUMARI CHALLA

HEAD OF THE DEPARTMENT

P HARINADHA

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of the people who made it possible and whose constant guidance and encouragement crown all the efforts success. We are extremely grateful to our respected Director, Prof. K. SANDHYA RANI for fostering an excellent academic climate in our institution. We also express my sincere gratitude to our respected Head of the Department Mr. P HARINADHA for his encouragement, overall guidance in viewing this project as a good asset and effort in bringing out this project. We would like to convey thanks to our guide at college Mrs. RATNA KUMARI CHALLA for her guidance, encouragement, cooperation and kindness during the entire duration of the course and academics. Our sincere thanks to all the members who helped us directly and indirectly in the completion of project work. We express our profound gratitude to all our friends and family members for their encouragement.

INDEX

S.NO	INDEX	PAGE NUMBER
1	Abstract	5
2	Introduction	6-7
3	Literature Review	8-9
4	Preliminaries	10-12
5	Working Model	13-19
6	Input and Output	20-22
7	Conclusion	22
8	References	23

Abstract

Face Recognition is a computer application that is capable of detecting, tracking, identifying or verifying human faces from an image or video captured using a digital camera. Although a lot of progress has been made in the domain of face detection and recognition for security, identification and attendance purposes, there are still issues hindering the progress to reach or surpass human level accuracy. These issues are variations in human facial appearance such as; varying lighting condition, noise in face images, scale, pose etc. Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library.

Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line

INTRODUCTION

What is Face Recognition?

The human face is a sophisticated multidimensional structure that can convey a lot of information about the individual, including expression, feeling, facial features. Effectively and efficiently analyzing the features related to facial information is a challenging task that requires a lot of time and effort. Recently, many facial recognition-based algorithms for banking security systems have been proposed, successfully implemented and also new algorithms developed or some existing algorithms improved or combined with other methods, techniques, or algorithms to build facial recognition systems or applications.

APPLICATIONS

- Automobile Security
- Access Control
- Immigration
- Education
- Retail
- Healthcare

ALGORITHMS AVAILABLE

Traditional Face Recognition Algorithms:

During the 1990s holistic approaches were used for face recognition. Handcrafted local descriptors became popular in the early 1990s, and then the local feature learning approaches were followed in the late 2000s. Nowadays algorithms that are widely used and are implemented in OpenCV are as follows:

- [Eigenfaces](#) (1991)
- [Local Binary Patterns Histograms \(LBPH\)](#) (1996)
- [Fisherfaces](#) (1997)

- [Scale Invariant Feature Transform \(SIFT\)](#) (1999)
- [Speed Up Robust Features \(SURF\)](#) (2006)

These algorithms are not faster compared to modern day face-recognition algorithms. Traditional algorithms can't be trained only by taking a single picture of a person.

Deep Learning for Face Recognition:

Some of the widely used Deep Learning-based Face Recognition systems are as follows:

- DeepFace
- DeepID series of systems
- VGGFace
- FaceNet

Face recognizers generally take face images and find the important points such as the corner of the mouth, an eyebrow, eyes, nose, lips, etc. Coordinates of these points are called facial-features points; there are 66 points. In this way, a different technique for finding feature points give different results

PURPOSE

The Purpose of Face Recognition Technology

The main objective of facial recognition is to identify individuals, whether individually or collectively. The number of false positives can vary, depending on the technology used for facial recognition. The best face identification algorithm has an error rate of 0.08%. Facial recognition systems that operate with [liveness detection](#), have higher rates of accuracy.

CONTRIBUTION

We used Face Recognition for Banking Security to provide two step authentication where the user can login using username and password and additional to the password we use face for biometric authentication. If any one who was unauthorized tries to access the account it will close automatically.

LITERATURE REVIEW

Why build such a system?

The number of identity fraud cases increased 16 percent between 2015 and 2016, according to Javelin Strategy & Research. The finance sector still struggles to combat fraudsters. Despite increased adoption of EMV cards and robust password creation policies, banking customers are still falling victim to fraudsters, and it's costing banks big. U.S. financial institutions alone lost \$16 billion last year as a result of fraud.

The problem goes beyond North America. ACI Worldwide found 49 percent of Brazilians and 56 percent of Mexicans fell victim to card fraud last year.

What's the solution? Pioneering financial institutions are starting to improve upon conventional authentication methods such as passwords and PINs. Many feel that facial recognition software and other biometric solutions are the keys to improving banking security.

The Problem with Passwords

Using passwords comes with a serious caveat: They're based on what people know. Hackers can use any number of tactics to obtain that knowledge.

Furthermore, the more complex they become, the easier they are to forget. When a banking customer forgets his password, he may receive a temporary code via email to reset it. The problem is, someone could use a man-in-the-middle attack to intercept that email and use the code himself.

Even security questions aren't completely foolproof. A cybercriminal could peruse a customer's social media profile to learn key information. So, in an attempt to change a customer's password, a hacker may be able to answer questions such as "Where were you born?" or "What was the name of your first pet?"

Social engineering is also a popular tactic among hackers. Symantec noted **how**

fraudsters trick Gmail users into disclosing verification codes by creating messages that look like their from Google. passwords aren't enough to deter fraudsters. What makes facial recognition different?

How Facial Recognition Combats Fraud?

Facial recognition technology on a mobile device authenticates customers based on who they *are* as opposed to what they know. Facial recognition on a mobile device offers a second factor of authentication, with the first being possession of the device itself, and the second being a live facial image. Multi-factor authentication presents more barriers to fraudsters.

While facial recognition can be considered among the most convenient of biometric modalities, it does pose a higher risk of impersonation, given the higher availability of facial images of a given victim. A hacker could try to use a picture of someone they're impersonating, often called a "spoof". This is why **it's important to employ means to detect spoofs** by assessing the "liveness" of the facial image. This is often called "liveness detection" and applies to many biometric modalities, including fingerprint.

Facial biometrics can also be used to access accounts from a computer. Many computers, for example, have built-in webcams. Every time a banking customer logs into his online account, they can use their facial biometrics as an additional security factor to login to their accounts and to request transactions.

PRELIMINARIES

STREAMLIT

Streamlit is a free and open-source framework to rapidly build and share beautiful machine learning and data science web apps. It is a Python-based library specifically designed for machine learning engineers. Data scientists or machine learning engineers are not web developers and they're not interested in spending weeks learning to use these frameworks to build web apps. Instead, they want a tool that is easier to learn and to use, as long as it can display data and collect needed parameters for modeling.

STREAMLIT AUTHENTICATOR

A secure authentication module to validate user credentials in a Streamlit application. Using Streamlit-Authenticator is as simple as importing the module and calling it to verify your predefined users' credentials.

1. Hashing passwords
2. Creating a login widget
3. Authenticating users
 - You can then use the returned name and authentication status to allow your verified user to proceed to any restricted content. In addition, you have the ability to add an optional logout button at any location on your main body or sidebar.

FACE RECOGNITION

Recognize and manipulate faces from Python or from the command line with the world's simplest face recognition library. Built using dlib's state-of-the-art face recognition built with deep learning. The model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. This also provides a simple face_recognition command line tool that lets you do face recognition on a folder of images from the command line!

Applications

Find faces in pictures

Find all the faces that appear in a picture:



Input



Output

Find and manipulate facial features in pictures

Get the locations and outlines of each person's eyes, nose, mouth and chin.



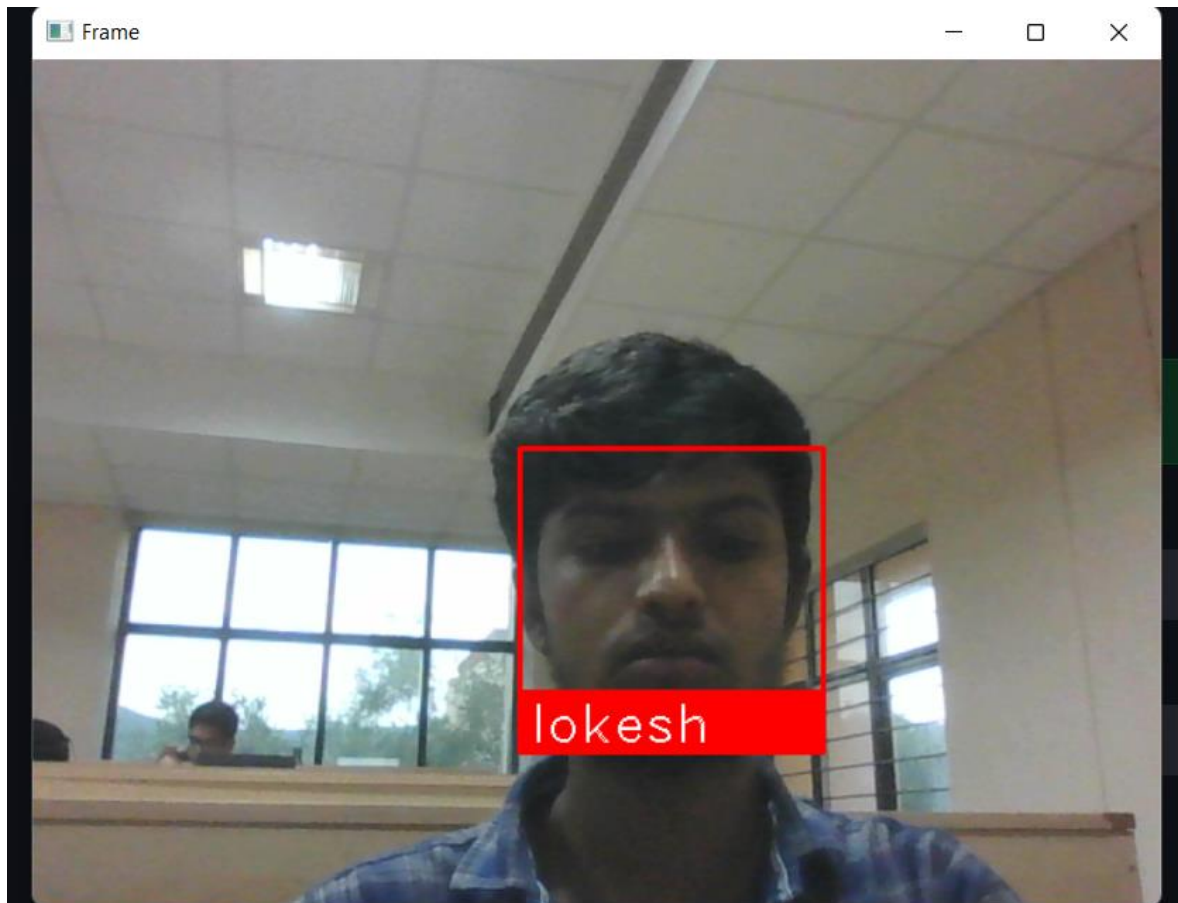
Input



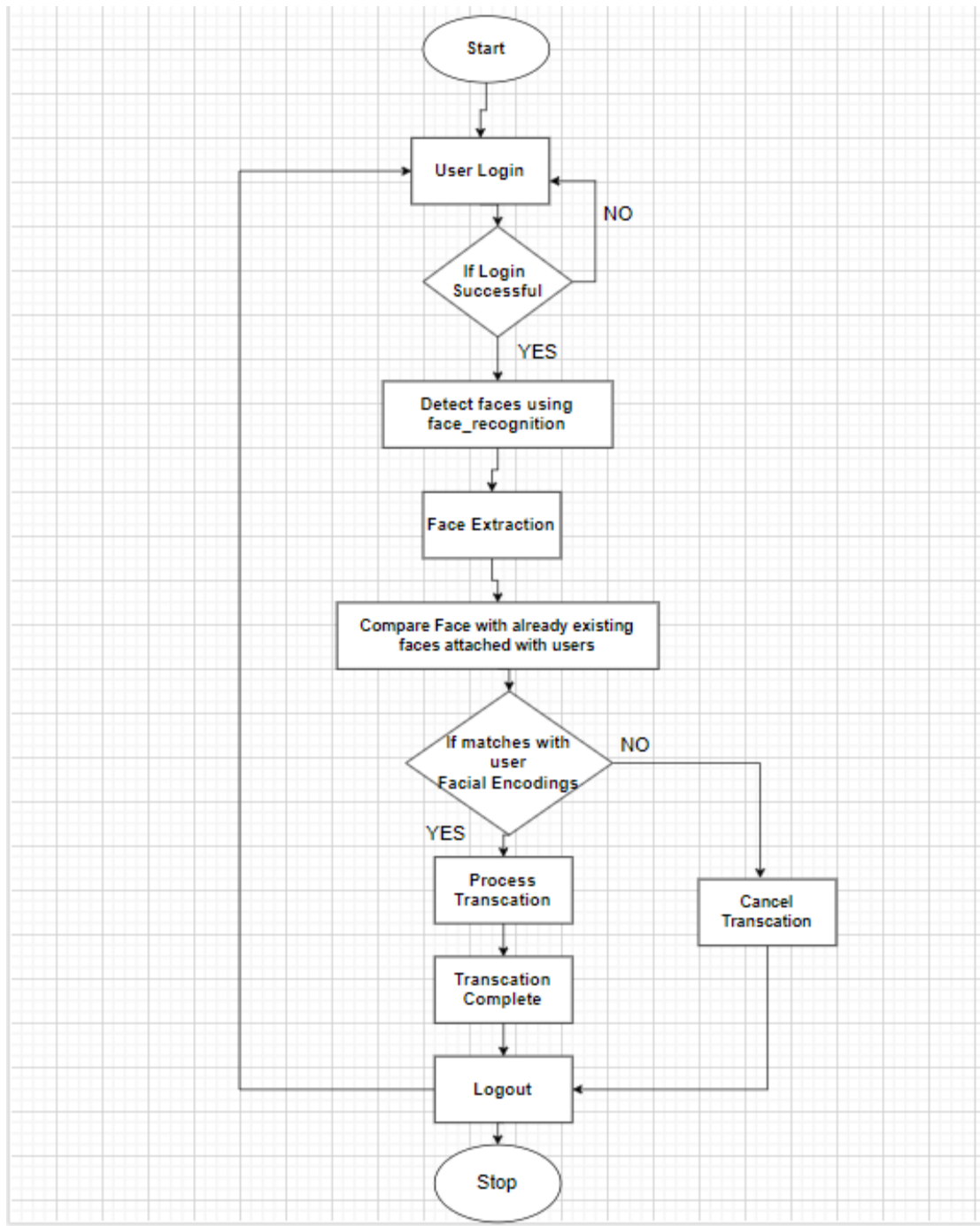
Output

OPENCV

OpenCV (*Open Source Computer Vision Library*) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross-platform and free for use under the open-source Apache 2 License. Starting in 2011, OpenCV features GPU acceleration for real-time operations.



WORKING MODEL



It Contains three modules: **data**, **generate_keys** and **final**.

DATA Module: It is used for storing the facial embeddings of the customers of the bank. After importing the necessary modules and loading the images using `load_image_file` and converting them into RGB and storing the embeddings with their usernames.

Filename: data.py

```
# -- Importing necessary modules

import face_recognition

import cv2

# -- Loading Images

img = face_recognition.load_image_file('Dataset/loki2.jpg')

img1 = face_recognition.load_image_file('Dataset/suchitra.jpeg')

img2 = face_recognition.load_image_file('Dataset/prathap.jpeg')

img3 = face_recognition.load_image_file('Dataset/eswar1.png')

# -- Converting Images into RGB from BGR

rgb_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

rgb_img1 = cv2.cvtColor(img1, cv2.COLOR_BGR2RGB)

rgb_img2 = cv2.cvtColor(img2, cv2.COLOR_BGR2RGB)

rgb_img3 = cv2.cvtColor(img3, cv2.COLOR_BGR2RGB)

# -- Storing all the facial embeddings as list

lokeshr = face_recognition.face_encodings(rgb_img)[0]

suchitra = face_recognition.face_encodings(rgb_img1)[0]

prathap = face_recognition.face_encodings(rgb_img2)[0]

eswar = face_recognition.face_encodings(rgb_img3)[0]
```

GENERATE_KEYS module: In this module we will store names, usernames, and passwords and encrypt the passwords using bcrypt hashing technique and store them in a pickle file.

Filename: generate_keys.py

```
# -- Importing necessary modules

import pickle

import streamlit_authenticator as stauth

# -- Names, Usernames and passwords

names = ["lokesh", "prathap", "suchitra", "eswar"]
usernames = ["lokeshkr", "prathap", "suchitra", 'eswar']
passwords = ["lokesh123", "prathap123", "suchitra123", "eswar123"]

# -- Hashing the passwords which uses bcrypt

hashed_passwds = stauth.Hasher(passwords).generate()

with open('passwords.pkl', 'wb') as file:

    pickle.dump(hashed_passwds, file)
```

Streamlit_authenticator contains Hasher which encrypts the passwords which were passed to that function as an iterable(list).

Final module: In this module we create login widget using streamlit_authenticator. If the user entered valid username and password he can able to login into main page and there user will be checked with the facial_encodings stored with his username if it was

valid he will be allowed to transfer funds if the person who was not authorized to access the account, the page will be stopped after showing unauthorized access to the screen.

We will scale the image by 75 percent for faster processing and we will rescale it to normal size, we continuously show that frames. Once any person who was unauthorized tries to access it will release the webcam and destroy all the windows.

Filename: final.py

```
# -- Importing Necessary Modules

import pickle
import time
import cv2
import face_recognition
from streamlit_autorefresh import st_autorefresh
import data
from data import *
import streamlit as st
import streamlit_authenticator as stauth
from numpy import dot
import numpy as np

# -- Function when the image matches with image attached to that account

def valid():
    st.success("Login Success")
    st.text_input("UserID", key="id")
    st.number_input("Amount", key="amt")
    out = st.button("Transfer")
    if out:
        st.success("Transaction success")

# -- Function when the image doesn't match with image attached to that account

def invalid():
    st.error('Unauthorized Access')
    video.release()
    cv2.destroyAllWindows()
    st.stop()

names = ["lokes", "prathap", "suchitra", "eswar"]
```



```

usernames = ["lokeshkr", "prathap", "suchitra", "eswar"]

# -- Loading passwords with was encrypted and stored in the pickle file

with open('passwords.pkl', 'rb') as file:
    hashed_passwords = pickle.load(file)

# -- Configuring page
st.set_page_config(page_title="Banking secure", layout='wide')
st.header("Welcome to Rkv Bank")
authenticator = stauth.Authenticate(names, usernames, hashed_passwords,
'cookie_name', 'signature_key',
                                cookie_expiry_days=0)
name, authentication_status, username = authenticator.login("Login", "main")
flag = 0

if authentication_status == False:
    st.error("Username/Password is incorrect")

if authentication_status is None:
    st.warning("please enter username and password")

if authentication_status:
    authenticator.logout("Logout", 'sidebar')
    st.sidebar.write(f"Welcome {name}")
    video = cv2.VideoCapture(cv2.CAP_DSHOW)
    result = None
    val = None
    while True:
        ret, frame = video.read()
        small_frame = cv2.resize(frame, (0, 0), fx=0.25, fy=0.25)
        rgb_small_frame = small_frame[:, :, :-1]
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,
face_locations)
        if face_encodings:
            if username == 'lokeshkr':
                result=face_recognition.compare_faces(data.lokeshkr,
face_encodings)
                if result[0]:
                    if flag == 0:
                        valid()
                    flag = 1
                else:
                    invalid()

            elif username == 'suchitra':
                result=face_recognition.compare_faces(data.suchitra,
face_encodings)

```

```

        if result[0]:
            if flag == 0:
                valid()
                flag = 1
            else:
                invalid()
    elif username == 'eswar':
        result = face_recognition.compare_faces(data.eswar, face_encodings)
        if result[0]:
            if flag == 0:
                valid()
                flag = 1
            else:
                invalid()
    else:
        result = face_recognition.compare_faces(data.prathap,
        face_encodings)
        if result[0]:
            if flag == 0:
                valid()
                flag = 1
            else:
                invalid()

    top, right, bottom, left = face_locations[0][0], face_locations[0][1],
    face_locations[0][2], face_locations[0][3]
    top *= 4
    right *= 4
    bottom *= 4
    left *= 4

    cv2.rectangle(frame, (left, top), (right, bottom), (0, 0, 255), 2)

    cv2.rectangle(frame, (left, bottom - 35), (right, bottom), (0, 0, 255),
    cv2.FILLED)
    font = cv2.FONT_HERSHEY_DUPLEX
    if result[0]:
        cv2.putText(frame, name, (left + 6, bottom - 6), font, 1.0, (255,
    255, 255), 1)
    else:
        cv2.putText(frame, "Unknown", (left + 6, bottom - 6), font, 1.0, (255,
    255, 255), 1)

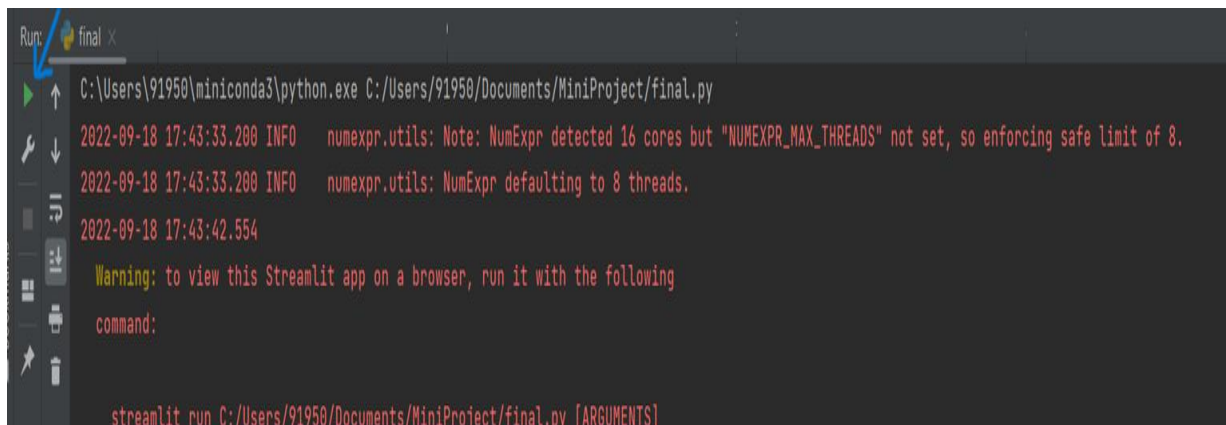
    cv2.imshow("Frame", frame)

    if cv2.waitKey(10) & 0xFF == ord('q'):
        break

```

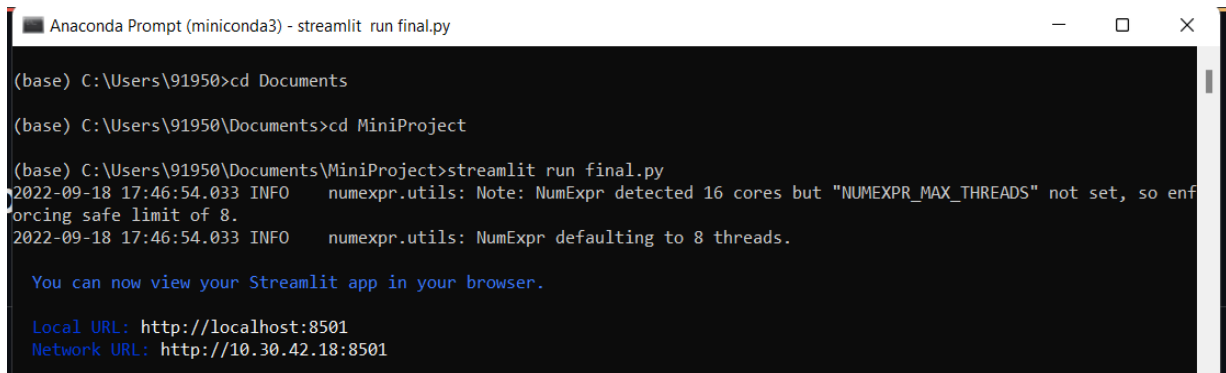
Steps to run the software

- 1.Install the necessary packages
- 2.Open the python files in an IDE.
- 3.click on the run button and open the terminal/ command prompt and type streamlit run filename.py after entering into the directory.



A screenshot of a code editor interface. At the top, a tab labeled 'final' is visible. Below it, a terminal window shows the execution of a Python script. The command 'C:\Users\91950\miniconda3\python.exe C:/Users/91950/Documents/MiniProject/final.py' is entered. The output includes two INFO messages from 'numexpr.utils' regarding thread limits and a warning to view the app in a browser. The command 'streamlit run C:/Users/91950/Documents/MiniProject/final.py [ARGUMENTS]' is shown at the bottom of the terminal.

```
Run: final x
C:\Users\91950\miniconda3\python.exe C:/Users/91950/Documents/MiniProject/final.py
2022-09-18 17:43:33.200 INFO    numexpr.utils: Note: NumExpr detected 16 cores but "NUMEXPR_MAX_THREADS" not set, so enforcing safe limit of 8.
2022-09-18 17:43:33.200 INFO    numexpr.utils: NumExpr defaulting to 8 threads.
2022-09-18 17:43:42.554
Warning: to view this Streamlit app on a browser, run it with the following
command:
streamlit run C:/Users/91950/Documents/MiniProject/final.py [ARGUMENTS]
```



A screenshot of an Anaconda Prompt terminal window titled 'Anaconda Prompt (miniconda3) - streamlit run final.py'. The terminal shows the user navigating to the 'Documents\MiniProject' directory and running 'streamlit run final.py'. The output is identical to the previous screenshot, including the INFO messages and the warning. At the bottom, the local and network URLs for the Streamlit app are displayed.

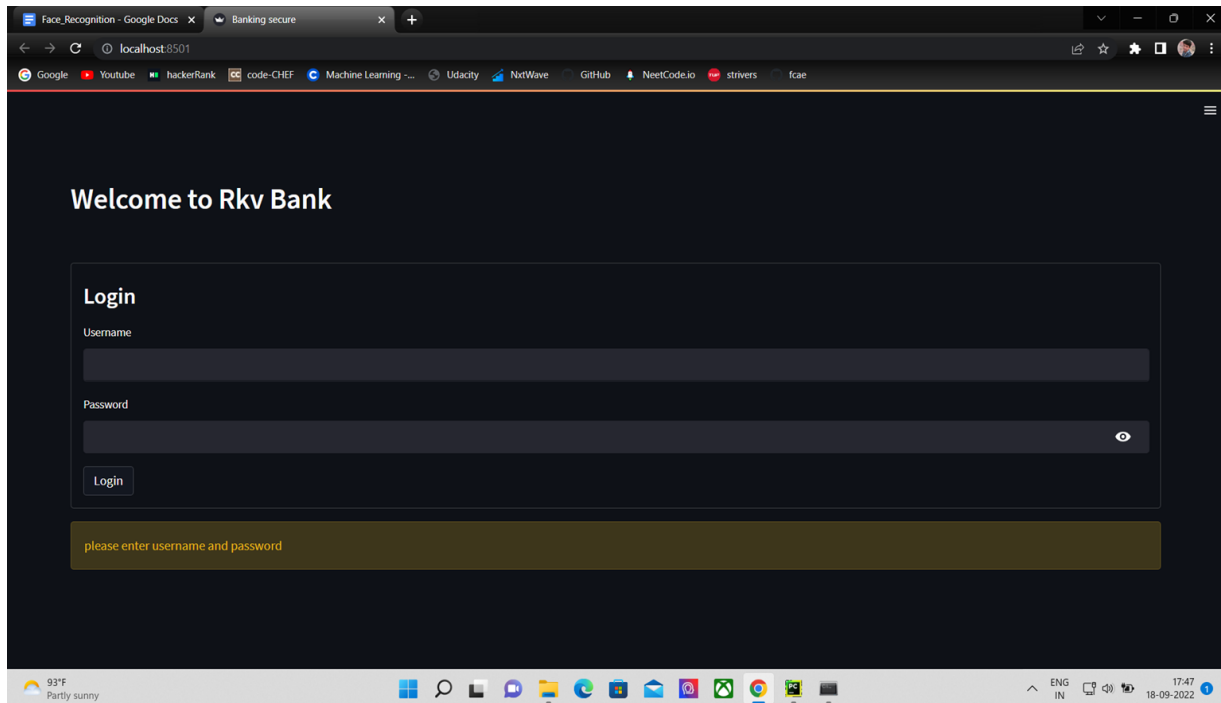
```
Anaconda Prompt (miniconda3) - streamlit run final.py
(base) C:\Users\91950>cd Documents
(base) C:\Users\91950\Documents>cd MiniProject
(base) C:\Users\91950\Documents\MiniProject>streamlit run final.py
2022-09-18 17:46:54.033 INFO    numexpr.utils: Note: NumExpr detected 16 cores but "NUMEXPR_MAX_THREADS" not set, so enforcing safe limit of 8.
2022-09-18 17:46:54.033 INFO    numexpr.utils: NumExpr defaulting to 8 threads.

You can now view your Streamlit app in your browser.

Local URL: http://localhost:8501
Network URL: http://10.30.42.18:8501
```

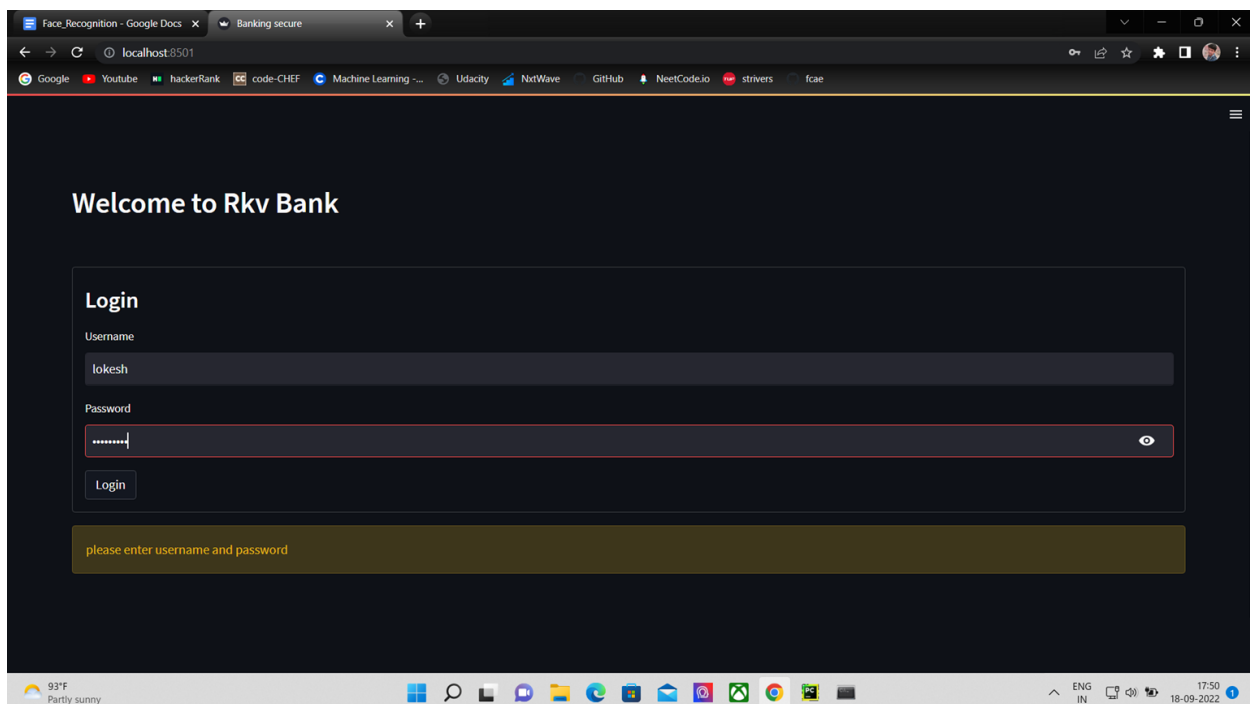
INPUT AND OUTPUT

Steps to login

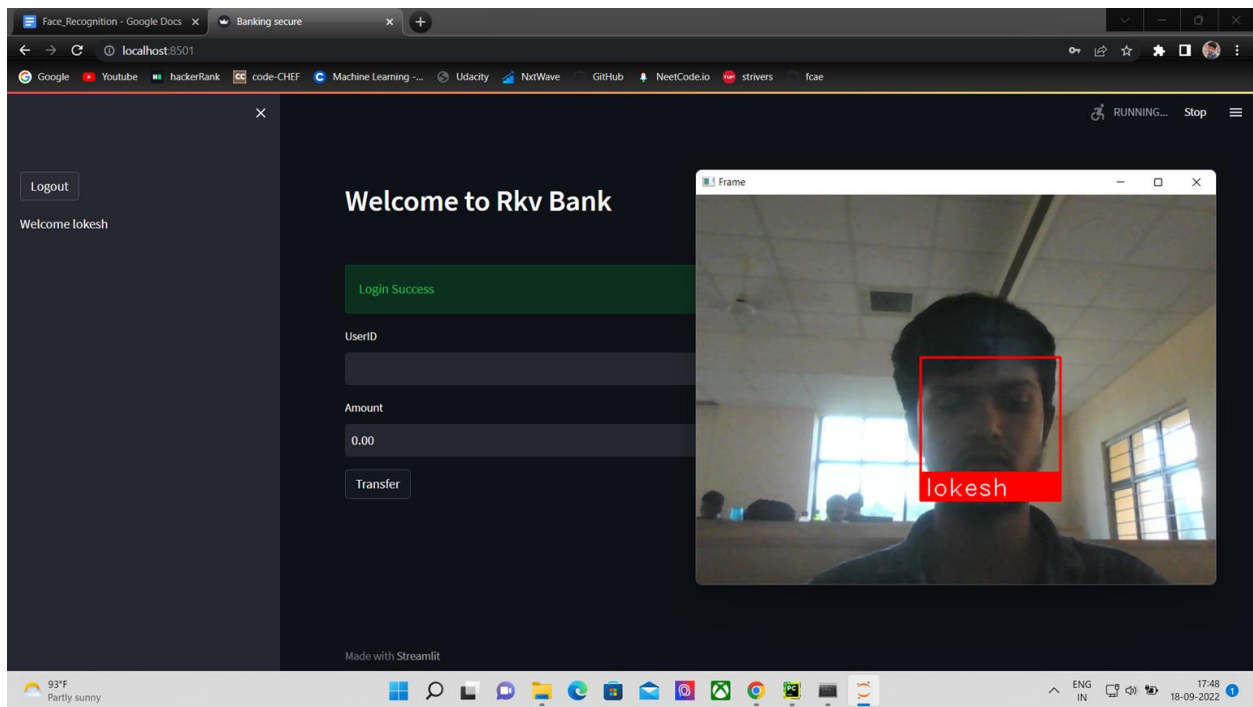


1. Enter username and password

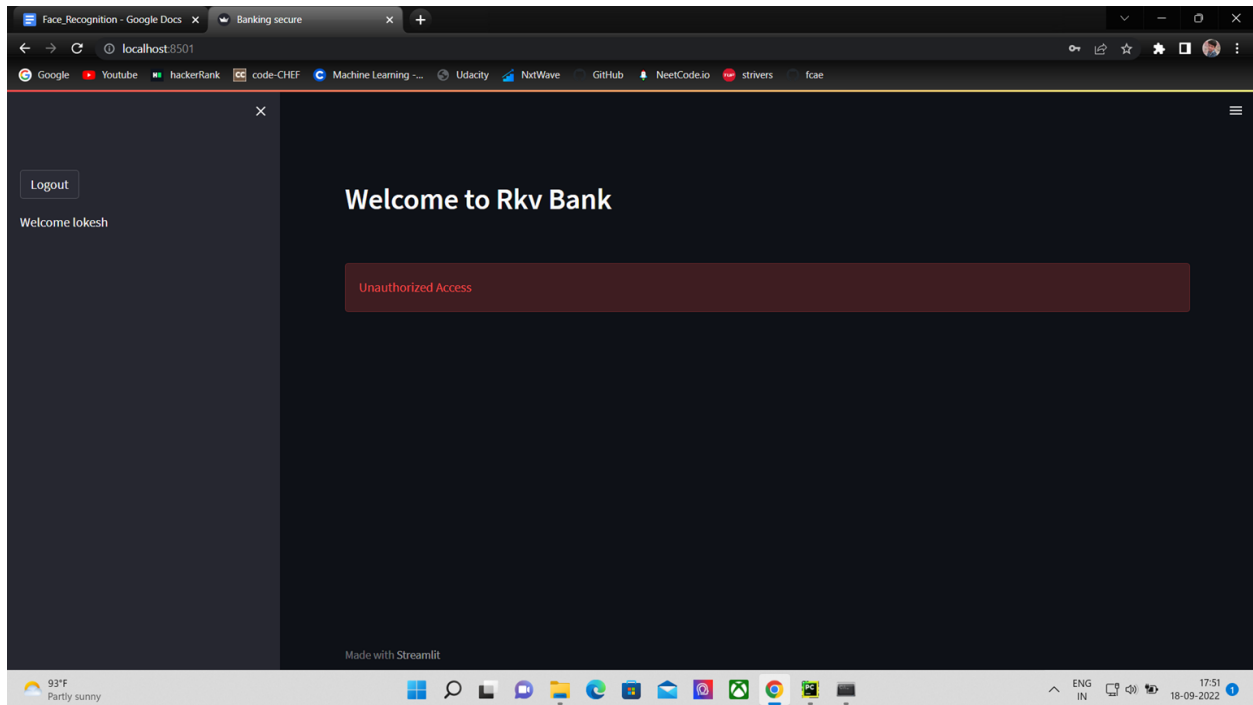
2. click on Login



When the credentials are true and facial_encodings match to encodings saved in the data module.



When the unauthorized person with valid credentials tries to access but their facial encodings will be mismatched and it shows unauthorized access and the user will not be able to access the application.



CONCLUSION

Face Recognition for Banking Security can be helpful to prevent online Fraud. Many computers, for example, have built-in webcams. Every time a banking customer logs into his online account, they can use their facial biometrics as an additional security factor to login to their accounts and to request transactions.

- Two factor authentication provides better security.
- Easy to integrate, it can be easily integrated into existing systems.
- Automated identification. Face detection lets facial identification be automated, thus increasing efficiency alongside a heightened rate of accuracy.
- Facial detection will be done throughout the transaction.
- Even though your credentials were available without facial encodings transactions can't be done.

REFERENCES

For face_recognition

https://github.com/ageitgey/face_recognition

For streamlit

<https://docs.streamlit.io/>

For streamlit_authenticator

<https://github.com/mkhorasani/Streamlit-Authenticator>

For opencv

<https://docs.opencv.org/4.x/>

Other references

<https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/>