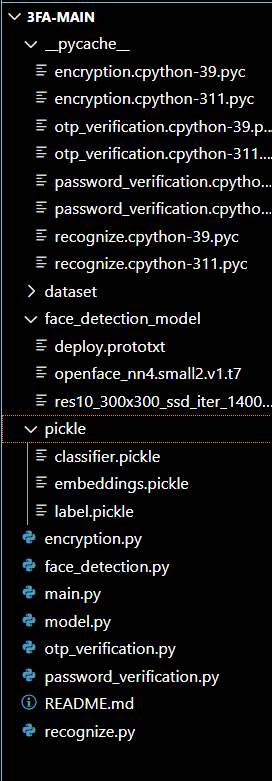
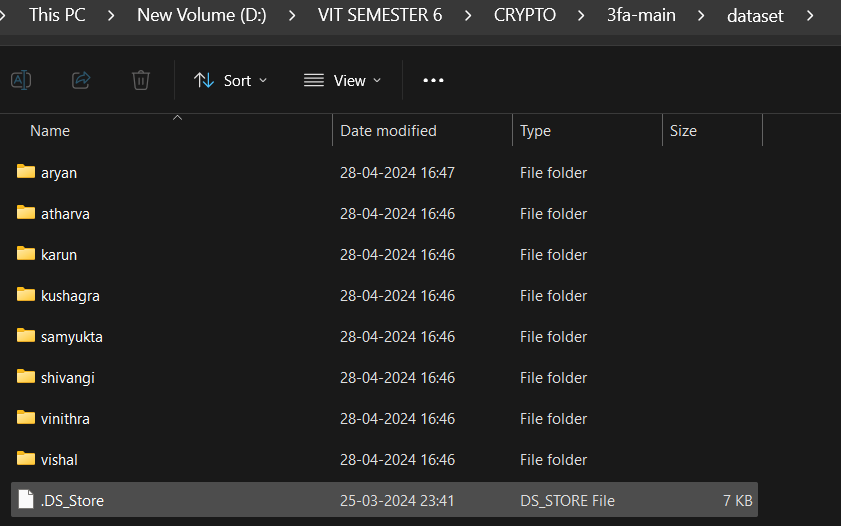
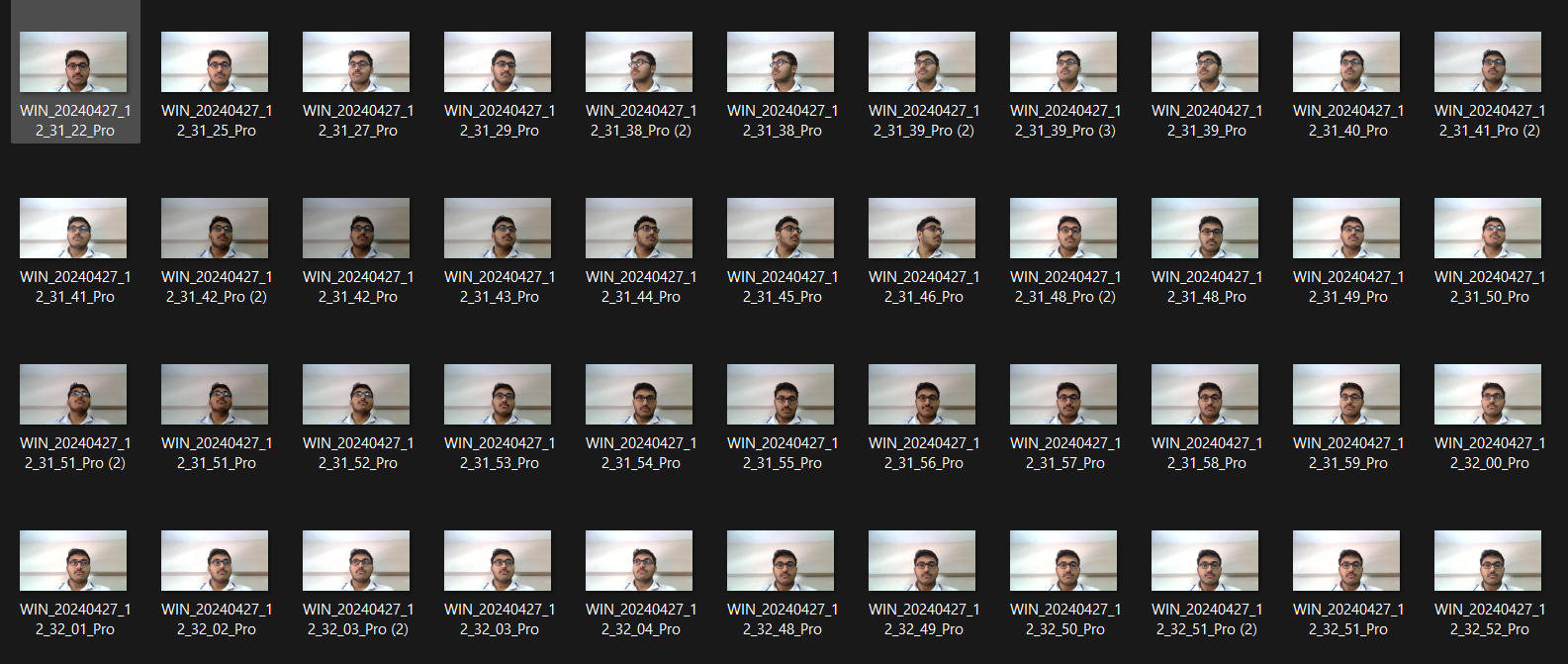
ALL FILES:

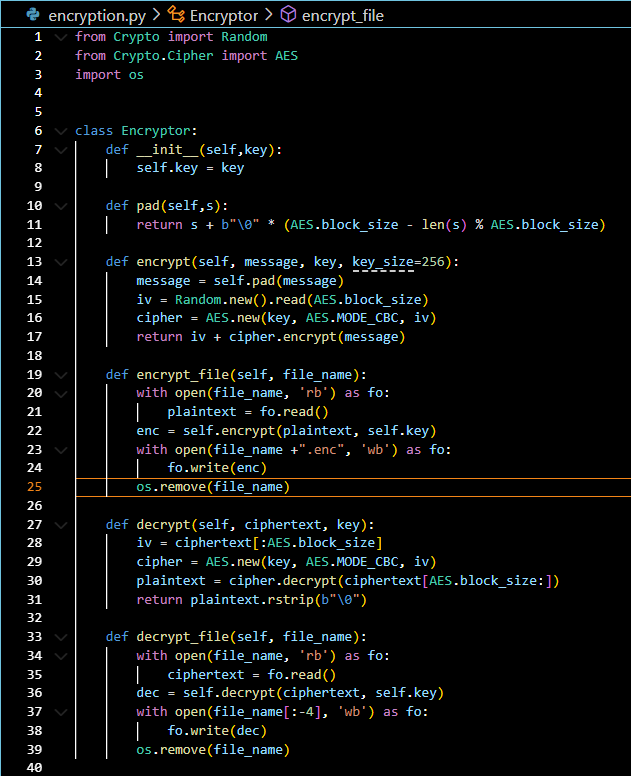


DATASET:



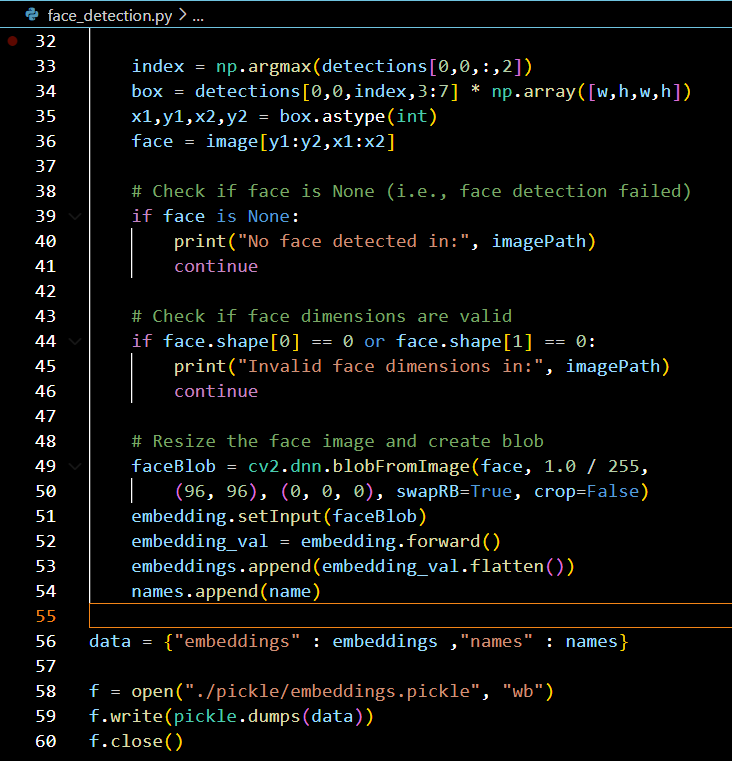


**ENCRYPTION.py**

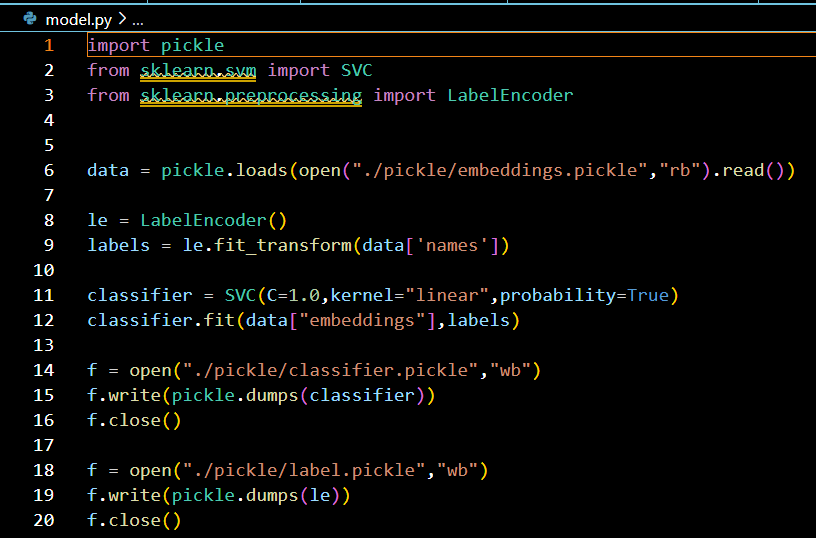


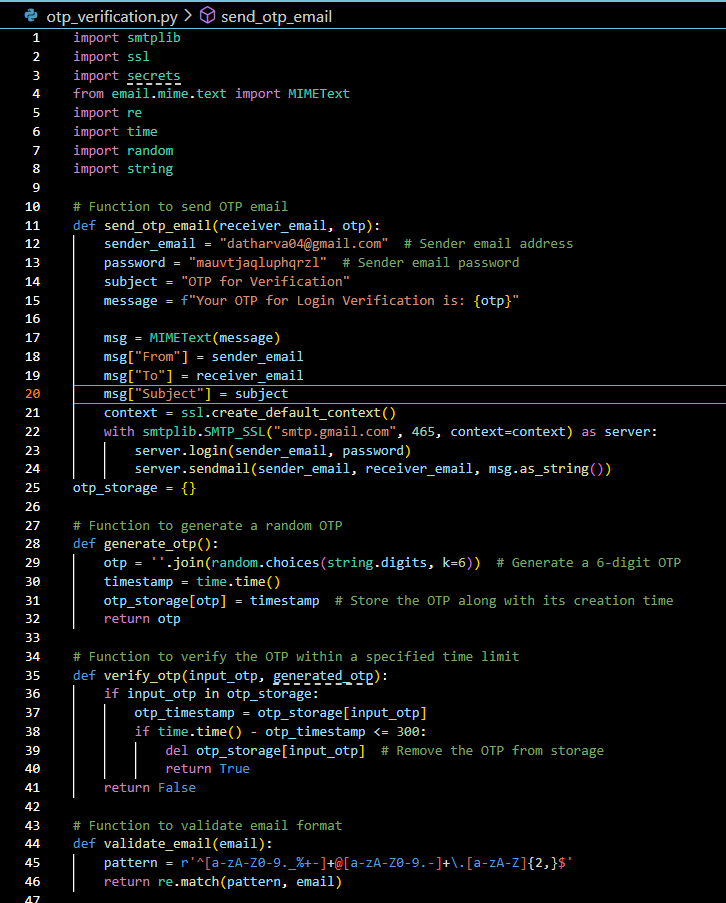
**FACE DETECTION**



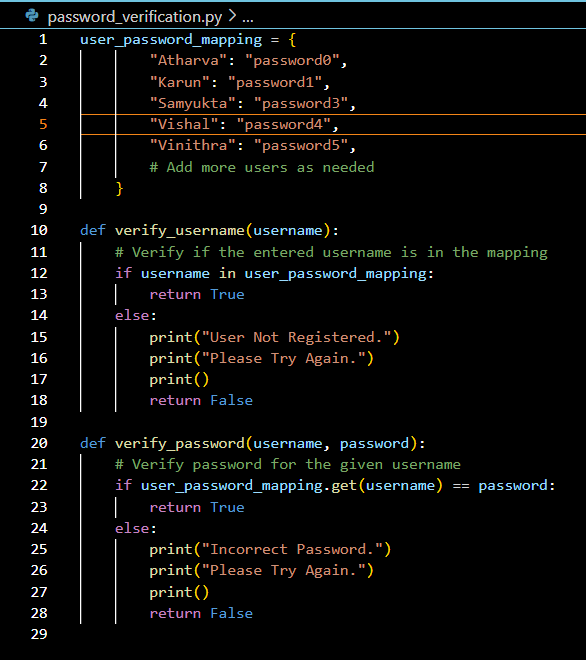


**MODEL**



**OTP VERIFICATION  
**

**PASSWORD VERIFICATION**



**RECOGNIZE.py**

import numpy as np

import cv2

import pickle

import time

class Recognizer:

def \_\_init\_\_(self, user\_mapping):

self.capture = cv2.VideoCapture(0)

time.sleep(2.0)

ProtoPath = "face\_detection\_model/deploy.prototxt"

ModelPath = "face\_detection\_model/res10\_300x300\_ssd\_iter\_140000.caffemodel"

EmbedderPath = "face\_detection\_model/openface\_nn4.small2.v1.t7"

self.detector = cv2.dnn.readNetFromCaffe(ProtoPath, ModelPath)

self.embedder = cv2.dnn.readNetFromTorch(EmbedderPath)

self.classifier = pickle.loads(open("./pickle/classifier.pickle", "rb").read())

self.labels = pickle.loads(open("./pickle/label.pickle", "rb").read())

self.user\_mapping = user\_mapping

def recognize(self):

total = 0

user\_count = {user: 0 for user in self.user\_mapping.keys()}

unknown = 0

# Create a named window with appropriate properties

cv2.namedWindow("face\_detection", cv2.WINDOW\_NORMAL)

cv2.setWindowProperty("face\_detection", cv2.WND\_PROP\_TOPMOST, 1)

while total < 20:

ret, image = self.capture.read()

(h, w) = image.shape[:2]

image\_blob = cv2.dnn.blobFromImage(

cv2.resize(image, (300, 300)), 1.0, (300, 300),

(104.0, 177.0, 123.0), swapRB=False, crop=False)

self.detector.setInput(image\_blob)

detections = self.detector.forward()

index = np.argmax(detections[0, 0, :, 2])

box = detections[0, 0, index, 3:7] \* np.array([w, h, w, h])

x1, y1, x2, y2 = box.astype(int)

face\_input = image[y1:y2, x1:x2]

(fH, fW) = face\_input.shape[:2]

if fW < 20 or fH < 20:

continue

faceBlob = cv2.dnn.blobFromImage(face\_input, 1.0 / 255, (96, 96), (0, 0, 0), swapRB=True, crop=False)

self.embedder.setInput(faceBlob)

embedding = self.embedder.forward()

preds = self.classifier.predict\_proba(embedding)[0]

j = np.argmax(preds)

proba = preds[j]

label = self.labels.classes\_[j]

text = "{}".format(label)

y = y1 - 10 if y1 - 10 > 10 else y1 + 10

face = cv2.rectangle(image, (x1, y1), (x2, y2), (255, 0, 0), 2)

face = cv2.putText(face, text, (x1, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (0, 255, 0), 2)

cv2.imshow("face\_detection", face)

for user, user\_label in self.user\_mapping.items():

if label == user\_label:

user\_count[user] += 1

break

else:

unknown += 1

total += 1

if cv2.waitKey(1) & 0xFF == ord('q'):

break

self.capture.release()

cv2.destroyAllWindows()

max\_user = max(user\_count, key=user\_count.get)

if user\_count[max\_user] > unknown:

return max\_user

else:

return None

**MAIN.py**

import argparse

from encryption import Encryptor

from recognize import Recognizer

from otp\_verification import send\_otp\_email, generate\_otp, verify\_otp, validate\_email

from password\_verification import verify\_password, verify\_username

import time

import threading

# User-label mapping

user\_mapping = {

"Atharva": "user0",

"Karun":"user1",

"Samyukta": "user2",

"Vishal": "user3",

"Vinithra": "user4"

# Add more users as needed

}

# User-email mapping

user\_email\_mapping = {

"Atharva": "atharvamangesh.d2021@vitstudent.ac.in",

"Karun": "karun.pramod2021@vitstudent.ac.in",

"Samyukta": "samyukta.pathak2021@vitstudent.ac.in",

"Vishal": "vishalbommisetty@gmail.com",

"Vinithra": "vinithra@example.com",

# Add more users as needed

}

# Function to authenticate user based on password

def authenticate\_user\_password():

global username

attempts = 3

while attempts > 0:

username = input("Enter Your Username: ")

if verify\_username(username):

password\_attempts = 3

while password\_attempts > 0:

password = input("Enter Your Password: ")

if verify\_password(username, password):

print("Password Verified.")

print()

return username # Return the username upon successful authentication

password\_attempts -= 1

print("Too Many Incorrect Attempts.")

print("Please try again later.")

return None

attempts -= 1

print("Authentication Failed. Too Many Attempts.")

return None

def update\_time\_left(start\_time, event, time\_limit):

while not event.is\_set():

time\_passed = time.time() - start\_time

time\_left = max(0, time\_limit - time\_passed) # Calculate time left

if time\_left == 0:

break

print(f"\rTime Left: {time\_left} seconds", end='', flush=True)

time.sleep(1)

def authenticate\_user\_otp(username):

attempts = 3

receiver\_email = user\_email\_mapping.get(username)

if not validate\_email(receiver\_email):

print("Your Registered Email is Invalid.")

print("Contact Admin to Update.")

exit()

generated\_otp = generate\_otp()

send\_otp\_email(receiver\_email, generated\_otp)

print("OTP has been sent to your Registered Email.")

start\_time = time.time()

time\_limit = 300 # Time limit for OTP verification

while attempts > 0:

timer\_event = threading.Event()

time\_thread = threading.Thread(target=update\_time\_left, args=(start\_time, timer\_event, time\_limit))

time\_thread.start()

print("\n")

timer\_event.set()

input\_otp = input("Enter the OTP received: ")

time\_thread.join()

if verify\_otp(input\_otp, generated\_otp):

print("\nOTP Verified.")

print()

print("Please Look at the Camera.")

return True

else:

print("\nInvalid OTP.")

print("Please Try Again.")

print()

attempts -= 1

print("OTP Verification Failed.")

return False

# Function to authenticate user based on detected faces

def authenticate\_user\_face():

attempts = 2

while attempts > 0:

recog = Recognizer(user\_mapping)

user\_name = recog.recognize()

if user\_name == username:

return user\_name

else:

if(attempts!=1 or attempts!=2):

print()

print("User Not Recognized.")

print("Please Try Again.")

attempts -= 1

return None

# Main functionality

def main():

# Parse command line arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-f", "--file", required=True, help="Path to file")

ap.add\_argument("-m", "--mode", required=True, help="Enter 'encrypt' or 'decrypt'")

args = vars(ap.parse\_args())

# Extract arguments

file\_path = args['file']

mode = args['mode']

# Encryption key

key = b'[EX\xc8\xd5\xbfI{\xa2$\x05(\xd5\x18\xbf\xc0\x85)\x10nc\x94\x02)j\xdf\xcb\xc4\x94\x9d(\x9e'

enc = Encryptor(key)

# Main functionality

if mode == "encrypt":

try:

enc.encrypt\_file(file\_path)

print("File Encrypted")

except:

print("Incorrect File Path")

exit()

elif mode == "decrypt":

# Authenticate user using password

if not authenticate\_user\_password():

exit()

# Authenticate user using otp

if not authenticate\_user\_otp(username):

exit()

# Authenticate user using face detection

authenticated\_user = None

while not authenticated\_user:

authenticated\_user = authenticate\_user\_face()

if authenticated\_user:

break

break

if authenticated\_user:

try:

enc.decrypt\_file(file\_path)

print()

print("File Decrypted.")

print(f"User Authenticated: {authenticated\_user}")

except Exception as e:

print("Decryption Failed:", e)

else:

print()

print("Face Authentication Failed. Exiting.")

else:

print("Incorrect Mode")

if \_\_name\_\_ == "\_\_main\_\_":

main()