Anexă – Codul sursă al aplicației

Main.py

```
import pickle
import cv2
import os
import cvzone
import face recognition
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import numpy
from datetime import datetime
import time
from memory_profiler import profile
import psutil
import matplotlib.pyplot as plt
from test import test
import smtplib
from email.message import EmailMessage

def send_email(receiver, subject, message):
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    # Make sure to give app access in your Google account
    server.login('cristipopesco25@gmail.com', 'tzai ttvs hhse mmcu')
    email = EmailMessage()
    email['From'] = 'cristipopesco25@gmail.com'
    email['To'] = receiver
    email['Subject'] = subject
    email.set_content(message)
    server.send_message(email)

cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual/serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendancerealtime-5f9d2-default-rtdb.firebaseio.com/",
    'storageBucket':"faceattendancerealtime-5f9d2.appspot.com"
})
bucket=storage.bucket()
cap=cv2.VideoCapture(0)
cap.set(3,640)
cap.set(4,480)
#capDetectie = cv2.VideoCapture(0)
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
smile_cascade = cv2.CascadeClassifier('haarcascade_smile.xml')
#face=face_cascade.detectMultiScale(cap,1.3,5)
imgBackground = cv2.imread('D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual\Resources/background.png')
#Importarea imaginilor din Modes in lista
folderModePath ='D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\Resources\Modes'
modePathList = os.listdir(folderModePath)
imgModeList = [ ]
for path in modePathList:
    imgModeList.append(cv2.imread(os.path.join(folderModePath,path))) #se adauga in lista
pozele 1-4.png indexate in FolderModePath+path
print(len(imgModeList))
#incarcarea fisierului de encoding
print("Loading Encode File...")
file=open('EncodeFile.p','rb')
encodeListKnownWithIds = pickle.load(file)
file.close()
encodeListKnown,studentIds = encodeListKnownWithIds
#print(studentIds)
print("Encode File Loaded")
modeType=0
counter=0
id=-1
ok=0
imgStudent=[ ]
while True:
    #_, frame = capDetectie.read()
    success, img = cap.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
    face = face_cascade.detectMultiScale(gray, 1.3, 5)
    imgS = cv2.resize(img, (640, 480))
    imgColor = imgS # backup imagine color
    imgS = cv2.cvtColor(imgS, cv2.COLOR BGR2RGB)
    faceCurFrame = face_recognition.face_locations(imgS)
    encodeCurFrame =face_recognition.face_encodings(imgS,faceCurFrame)
    imgBackground[162:162+480,55:55+640] =img
    imgBackground[44:44+633 , 808:808 + 414] = imgModeList[modeType]
    if faceCurFrame: #daca se detecteaza o fata cunoscuta
        for encodeFace, faceLoc in zip(encodeCurFrame, faceCurFrame):
            matches=face_recognition.compare_faces(encodeListKnown,encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            #print("matches",matches)
            #print("faceDis",faceDis)
            matchIndex=numpy.argmin(faceDis)#se extrage indexul cu valoarea minima, in cazul
nostru ce fata se potriveste
            if matches[matchIndex]:
                #for x,y,w,h in face:
                #print("Known face detected")
                #print(studentIds[matchIndex])
                y1,x2,y2,x1 = faceLoc
                for c1, c2, c3, c4 in face:
                    cv2.rectangle(img, (c1, c2), (c1 + c3, c2 + c4), (0, 255, 255), 2)
                    face_roi = img[c2:c2 + c4, c1:c1 + c3]
                    gray_roi = gray[c2:c2 + c4, c1:c1 + c3]
                    smile = smile_cascade.detectMultiScale(face_roi, 1.3, 25)
                    for s1, s2, s3, s4 in smile:
                        cv2.rectangle(face_roi, (s1, s2), (s1 + s3, s2 + s4), (0, 0, 255), 2)
                #y1, x2, y2, x1 = y1*4, x2*4, y2*4, x1*4
                        bbox= 55+x1, 162+y1, x2-x1, y2-y1 # la sfarsit avem width and height
                        label=test(image=imgS,model_dir='D:\\ETTI\\An IV\\Licenta\\curs 2h
CV\\cod python - scris manual\\Silent-Face-Anti-Spoofing-
master\\resources\\anti_spoof_models',device_id=0)
                        if label == 1:
                            imgBackground = cvzone.cornerRect(imgBackground, bbox, rt=0)#
dreptunghi in jurul fetei de grosime=0
                            id=studentIds[matchIndex]
                            if counter == 0:
                                cvzone.putTextRect(imgBackground,"Procesare",(275,400))
                                cv2.imshow("Identificare chip", imgBackground)
                                cv2.waitKey(1)
                                counter=1
                                modeType=1
                    else: ok=ok+1
                    if ok==1:
                        send_email('cristianirimescu25@gmail.com', 'TENTATIVA FRAUDARE',
'ATENTIE, se incearcă o fraudare!')

        if counter!=0:
            #preia datele despre individ
            if counter==1:
                studentInfo=db.reference(f'Students/{id}').get()
                print(studentInfo)
            #preia imaginea individului
                blob=bucket.get_blob(f'D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual\Images/{id}.png')
                array=numpy.frombuffer(blob.download_as_string(),numpy.uint8)
                imgStudent=cv2.imdecode(array,cv2.COLOR_BGRA2BGR) #se descarca imaginea prin
decodificare in format BGR
                #actualizeaza datele despre prezenta
                datetimeObject=datetime.strptime(studentInfo['last_attendance_time'],
                                                 "%Y-%m-%d %H:%M:%S")
                secondsElapsed=(datetime.now()-datetimeObject).total_seconds()
                print(secondsElapsed)
                if secondsElapsed>30:
                    ref=db.reference(f'Students/{id}')
                    studentInfo['total_attendance']=studentInfo['total_attendance']+1
#actualizam numar prezente
                    ref.child('total_attendance').set(studentInfo['total_attendance'])
                    ref.child('last_attendance_time').set(datetime.now().strftime("%Y-%m-%d
%H:%M:%S")) #actualizam data noua de prezenta
                else:
                    modeType=3
                    counter=0
                    imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
            if modeType!=3:
                if 10<counter<20:
```

```
                    modeType=2
                imgBackground[44:44 + 633, 808:808 + 414] = imgModeList[modeType]
                if counter<=10:
                 cv2.putText(imgBackground,str(studentInfo['total attendance']),(861,125),
                                                 cv2.FONT_HERSHEY_COMPLEX,1,(255,255,255),1)
#pozitie, font, culoare, grosime pentru a afisa numarul de prezente
                 cv2.putText(imgBackground, str(studentInfo['major']), (1006, 550),
                             cv2.FONT_HERSHEY_COMPLEX, 0.5, (255, 255, 255), 1)
                 cv2.putText(imgBackground, str(id), (1006, 493),
                             cv2.FONT HERSHEY COMPLEX, 0.5, (255, 255, 255), 1)

                 (w,h),_ =cv2.getTextSize(studentInfo['name'],cv2.FONT_HERSHEY_COMPLEX,1,1)
                 offset=(414-w)//2
                 cv2.putText(imgBackground, str(studentInfo['name']), (808+offset, 445),
                             cv2.FONT_HERSHEY_COMPLEX, 1, (50, 50, 50), 1)
                 imgBackground[175:175+216,909:909+216]=imgStudent
            counter=counter+1
            if counter>=20:
                counter=0
                modeType=0
                studentInfo=[]
                imgStudent=[]
                imgBackground[44:44+633 , 808:808 + 414] = imgModeList[modeType]
    else:
        modeType=0
        counter=0
    #cv2.imshow("Webcam", img)
    cv2.imshow("Identificare chip", imgBackground)
    cv2.imwrite('ultima_poza.jpg', imgColor)
    if cv2.waitKey(10) == ord('q'):
        break
```

EncodeGenerator.py

```
import cv2
import face_recognition
import pickle
import os
import time
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
from firebase_admin import storage
import psutil
import matplotlib.pyplot as plt
from memory_profiler import profile
# Înregistrați memoria utilizată înainte de execuție
memory_before = psutil.virtual_memory().percent
# Înregistrați timpul de început
start_time = time.time()
cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual/serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendancerealtime-5f9d2-default-rtdb.firebaseio.com/",
    'storageBucket':"faceattendancerealtime-5f9d2.appspot.com"
})
#Importarea imaginilor din Images in lista
folderPath ='D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris manual\Images'
pathList = os.listdir(folderPath)
print(pathList)
imgList = [ ]
studentIds=[ ]
for path in pathList:
    imgList.append(cv2.imread(os.path.join(folderPath,path)))
    studentIds.append(os.path.splitext(path)[0])
    fileName=f'{folderPath}/{path}'
    bucket=storage.bucket()
    blob=bucket.blob(fileName)
    blob.upload_from_filename(fileName)
    #print(path)
    #print(os.path.splitext(path)[0])
print(studentIds)

def findEncodings(imagesList):
```

```
    encodeList=[ ]
    for img in imgList:
        img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB) #conversie imagine din format BGR in format
RGB suportat
        encode=face_recognition.face_encodings(img)[0] #variabila in care se petrece encodarea
        encodeList.append(encode)

    return encodeList
print("Encoding started...")
encodeListKnown = findEncodings(imgList)
encodeListKnownWithIds = [encodeListKnown,studentIds]
#print(encodeListKnown)
print("Encoding Complete")
file=open("EncodeFile.p",'wb')
pickle.dump(encodeListKnownWithIds,file)
file.close()
print("File saved")
end_time = time.time()
elapsed_time = end_time - start_time
memory_after = psutil.virtual_memory().percent
print(f"Timpul de execuție: {elapsed_time:.2f} secunde")
print(f"Memorie înainte de execuție: {memory_before:.2f}%")
print(f"Memorie după execuție: {memory_after:.2f}%")
labels = ['Memorie înainte', 'Memorie după']
values = [memory_before, memory_after]

plt.bar(labels, values)
plt.ylim(0, 90)  # Seteaza limita pentru axa Y între 0 și 90 (procente)
plt.title('Utilizarea memoriei înainte și după execuție')
plt.ylabel('Memorie (%)')
plt.show()
labels2 = ['Timp de executie']
values2=elapsed_time
plt.bar(labels2, values2)
plt.ylim(8,10 )
plt.title('Timp de executie')
plt.ylabel('Timp (s)')
plt.show()
```

AddDataToDatabase.py

```
import firebase_admin
from firebase_admin import credentials
from firebase_admin import db
cred = credentials.Certificate("D:\ETTI\An IV\Licenta\curs 2h CV\cod python - scris
manual/serviceAccountKey.json")
firebase_admin.initialize_app(cred,{
    'databaseURL':"https://faceattendancerealtime-5f9d2-default-rtdb.firebaseio.com/"
})

ref=db.reference('Students')
data = {
   "852741":   #cheia
        {   #valoarea - compusa din toate campurile de jos
            "name": "Emily Blunt",
            "major": "Economics",
            "starting_year":2018,
            "total_attendance":12,
            "standing":"B",
            "year":2,
            "last_attendance_time":"2022-12-11 00:54:34"
        },
    "963852":   #cheia
        {   #valoarea - compusa din toate campurile de jos
            "name": "Elon Musk",
            "major": "Physics",
            "starting_year":2020,
            "total_attendance":7,
            "standing":"G",
            "year":2,
            "last_attendance_time":"2022-12-11 00:54:34"
        },
    "973853":   #cheia
        {   #valoarea - compusa din toate campurile de jos
            "name": "Irimescu Ovidiu",
```

```
        "major": "Electronics",
        "starting_year":2020,
        "total_attendance":5,
        "standing":"G",
        "year":4,
        "last_attendance_time":"2021-12-9 00:54:34"
    },
"973854":   #cheia
    {   #valoarea - compusa din toate campurile de jos
        "name": "Irimescu Teodora",
        "major": "Pupil",
        "starting_year":2020,
        "total_attendance":20,
        "standing":"G",
        "year":4,
        "last attendance time":"2021-12-9 00:54:34"
    },
"973855":   #cheia
    {   #valoarea - compusa din toate campurile de jos
        "name": "Irimescu Ionut",
        "major": "Mechanics",
        "starting_year":2020,
        "total_attendance":5,
        "standing":"D",
        "year":4,
        "last_attendance_time":"2021-12-9 00:54:34"
    },
"973856":   #cheia
    {   #valoarea - compusa din toate campurile de jos
        "name": "Irimescu Ion",
        "major": "Management",
        "starting_year":2005,
        "total_attendance":20,
        "standing":"D",
        "year":"F",
        "last_attendance_time":"2021-12-9 00:54:34"
    },
"973857":
    { "name": "Ashton Kutcher",
    "major": "Management",
    "starting_year": 2005,
    "total_attendance": 20,
    "standing": "D",
    "year": "F",
    "last_attendance_time":"2021-12-9 00:54:34"
    },
"973858":
    { "name": "David Schwimmer",
    "major": "Management",
    "starting_year": 2005,
    "total_attendance": 20,
    "standing": "D",
    "year": "F",
    "last_attendance_time":"2021-12-9 00:54:34"
    },
"973859":
    { "name": "Michael C Hall",
    "major": "Management",
    "starting_year": 2005,
    "total_attendance": 20,
    "standing": "D",
    "year": "F",
    "last_attendance_time":"2021-12-9 00:54:34"
    },
"9738510":
    { "name": "Topher Grace",
    "major": "Management",
    "starting_year": 2005,
    "total_attendance": 20,
    "standing": "D",
    "year": "F",
    "last attendance time":"2021-12-9 00:54:34"
    },
"9738511":
    { "name": "Elizabeth Mitchell",
    "major": "Management",
    "starting_year": 2005,
```

```
        "total_attendance": 20,
        "standing": "D",
        "year": "F",
        "last attendance time":"2021-12-9 00:54:34"
        },
    "9738512":
        { "name": "Kenneth Branagh",
        "major": "Management",
        "starting_year": 2005,
        "total attendance": 20,
        "standing": "D",
        "year": "F",
        "last_attendance_time":"2021-12-9 00:54:34"
        },
    "9738513":
        {"name": "Julia Roberts",
         "major": "Management",
         "starting_year": 2005,
         "total_attendance": 20,
         "standing": "D",
         "year": "F",
         "last_attendance_time": "2021-12-9 00:54:34"
         },
    "9738514":
        {"name": "Bruce Boxleitner",
         "major": "Management",
         "starting_year": 2005,
         "total_attendance": 20,
         "standing": "D",
         "year": "F",
         "last_attendance_time": "2021-12-9 00:54:34"
         }
}
for key, value in data.items():
   ref.child(key).set(value)
```