Biometric Routine Displayer

Open Source Programming
J-Component
Review - II

Team members :

→Ankush Rauniyar(20BIT0440)
→Jeeban Bhagat (20BIT0441)
→Taniya Shah(20BIT0442)

Submitted To :

Associate Prof. Sr.
Dr. John Singh K
School Of Information Technology and Engineering
VIT,Vellore,India

2.Abstract:

People working in large-scale organizations often face problems regarding their schedules and time-tables. For instance, in any university students and sometimes even teachers/professors tend to forget their class schedule. It results in failure in attending class. Similar problems are faced by employees in their workplace too. In order to solve this issue, we have come up with a technological solution that provides people information about their respective schedule at the particular time hour of their day through Biometric recognition. For example, if a student forgets his scheduled class at particular time, then he can get the information through our system just by showing his/hers face or by mechanically giving his/hers ID. This technology can be used even in hospitals by nurses and doctors for reminding the medical doses for a patient.

Biometric recognition, as one of the most successful applications of image analysis, has recently gained significant attention. It is due to availability of feasible technologies, including mobile solutions. Research in automatic face recognition has been conducted since the 1960s, but the problem is still largely unsolved. Last decade has provided significant progress in this area owing to advances in face modelling and analysis techniques. Although systems have been developed for face detection and tracking, reliable face recognition still offers a great challenge to computer vision and pattern

recognition researchers. There are several reasons for recent increased interest in face recognition, including rising public concern for security, the need for identity verification in the digital world, face analysis and modelling techniques in multimedia data management and computer entertainment. In this chapter, we have discussed face recognition processing, including major components such as face detection, tracking, alignment and feature extraction, and it points out the technical challenges of building a face recognition system. We focus on the importance of the most successful solutions available so far. The final summon can be how biometric recognition describes chosen face recognition methods and applications and their potential use in areas not related to face recognition.

3.Introduction :

In the project, we have presented an implementation of timetable scheduling problem using biometrics and also using classical method i.e. by providing registration number. The timetable scheduling problem is very common to all educational institutions. Main goal is to minimize the number of conflicts in the time table scheduling and help the students to efficiently find their classes.

We all know the initial timetabling problem with large number of binary variables has been reduced to the acceptable size by eliminating certain dimensions of the problem and incorporating those dimensions into constraints. To overcome this problem in todays world 'Biometric Routine Displayer' provides the users to minimize time consumption and makes it easier for them to have access of their time table and also deliver easier login in to time table using biometrics with OpenCV-Python according to time user requires and also maintains the details of the user's using 'Biometric Routine Displayer' in csv file. Users can also choose there mode i.e. biometrics or using registration number to log in and find their respective subject timing. This enables

the problem to be solved with efficiency and less time consumption.

This techniques can be used in businesses to minimize time consumption, improve efficiency of employees, manage their employees attendance, also just by using biometrics it becomes easily to know their daily projects or assignments on that particular day or check for meetings any time of the day. It can also be used in hospitals to manage check in and check out of patients and also the doctors and nurses can manage their time by biometrics that help them to find their day-to-day schedule in few seconds. There are many applications and wide use of this project and we are presenting one example of it.

4.Motivation :

Several incidents led us to work on the project 'Biometric Routine Displayer'. While attending online classes, students and teachers often tend to forget which class they have in future but due to continuous access to Internet in our home, we check our routine and attend the classes accordingly. But the scenario would be different in case of offline situation i.e. physical classes. It would be difficult and time-consuming for people to get access to their time schedule which might result either attending their classes late or missing the class. Thinking in wide perspective we figured out that same difficulty exists even in large organization for employees while figuring out their future tasks like meetings ,submitting files ,etc.

So we tried to come up with a solution for the existing situation. We realized the technology already exists for the current problem. In every organization there is a device that takes the attendance of the employee/teachers by scanning the fingerprint. We came up with an innovative idea of updating the technology. We thought of formulating a standalone- application that

would tell the people their current time schedule or even the day's time schedule just by the help of facial scanning. In this application we would either scan the people's face or take manual input of their unique identity number
(eg. Registration Number ,employee code ,etc) and then read their routine from the database and display either the current upcoming routine or the whole day's time schedule as the user wishes.

Hence, the above mentioned series of events and thoughts motivated us to work on the idea of "Biometric Routine Displayer" as it tend to solve a problem in an efficient manner.

5.Objectives:

Main objective of Biometric routine displayer are given below:

1) To provide easy access for students to view their time table at any given time.

2) To minimize constraints and find a feasible solution.

3) To store details of the students who view Biometrics Routine Displayer.

4) To increase efficiency and productivity of students and minimize time consumption.

5) To utilizes minimal processing or computing power and easy aspect of time table.

To achieve these objectives, the main step is finding the problems that our users face while finding these scheduling classes as it takes lots of time and biometrics generation takes few seconds to generate subject that

user has in that particular time. Then gathering and analyzing the required information for the problem was done.

6. Problem Statement:

Until recently, face recognition technology was commonly viewed as something straight out of science fiction. But over the past decade, this groundbreaking technology has not just become viable, it has become widespread. In fact, it's difficult to read technology news these days without seeing something about face recognition.
In this present scenario biometrics are plays key role on different tasks helping people do their job. By our project we are targeting those domains who have had issues of schedule. In the day of 24 hr., we have lots of task where we get involved in and sometimes, we miss very important stuffs which were meant to be done. So, domains can be workplace, hospital, school/college, bank, etc. In search of solving the issue, we have come up with this project where user can view their schedule in real-time by using their biometric to the system and plan the day accordingly. With the help of our project user will the

miss the stuffs he/she had at that specific time and don't encounter schedule issues.

7.EXISTING SYSTEM:

The existing system formulates users timetabling problem by considering that each lecture contained one group of students, one teacher, and any number of times which could be chosen freely. Since then the problem is being continuously studied using different conditions. Initially it was mostly applied to schools. Since the problem in schools and universities are relatively time consuming since their simple class structures, classical

methods, such as linear or integer programming approaches. However, the gradual consideration of the cases of , which contain different types of complicated class-structures, is increasing the complexity of the problem.

MODULES:

1. ***Attendance.csv:*** It contains details of users who have accessed Biometric Routine Display.

2. ***FacematchProject.py:*** It contains all the functions and code of the project.

3. ***ImageAttendence:*** It contains images of users that will be extracted during execution when user try to access routine using face recognition.

4. It contains all the details of the users such as registration number, name and course session details.

· Inside FacematchProject.py:

1. ***Import cv2 OpenCV***-Python is a library of Python bindings designed to solve computer vision problems. Importing cv2 makes it easier to integrate with other libraries.

2. ***Import numpy as np :***It can be used to call the numpy library and related functions and data types.

3. ***Import face_recognition:*** It is used to recognize faces in a photograph or folder full for photographs. We need to provide a folder with pictures of user to recognize it later.

4. ***Import os :*** It is to interact with the underlying operating system.

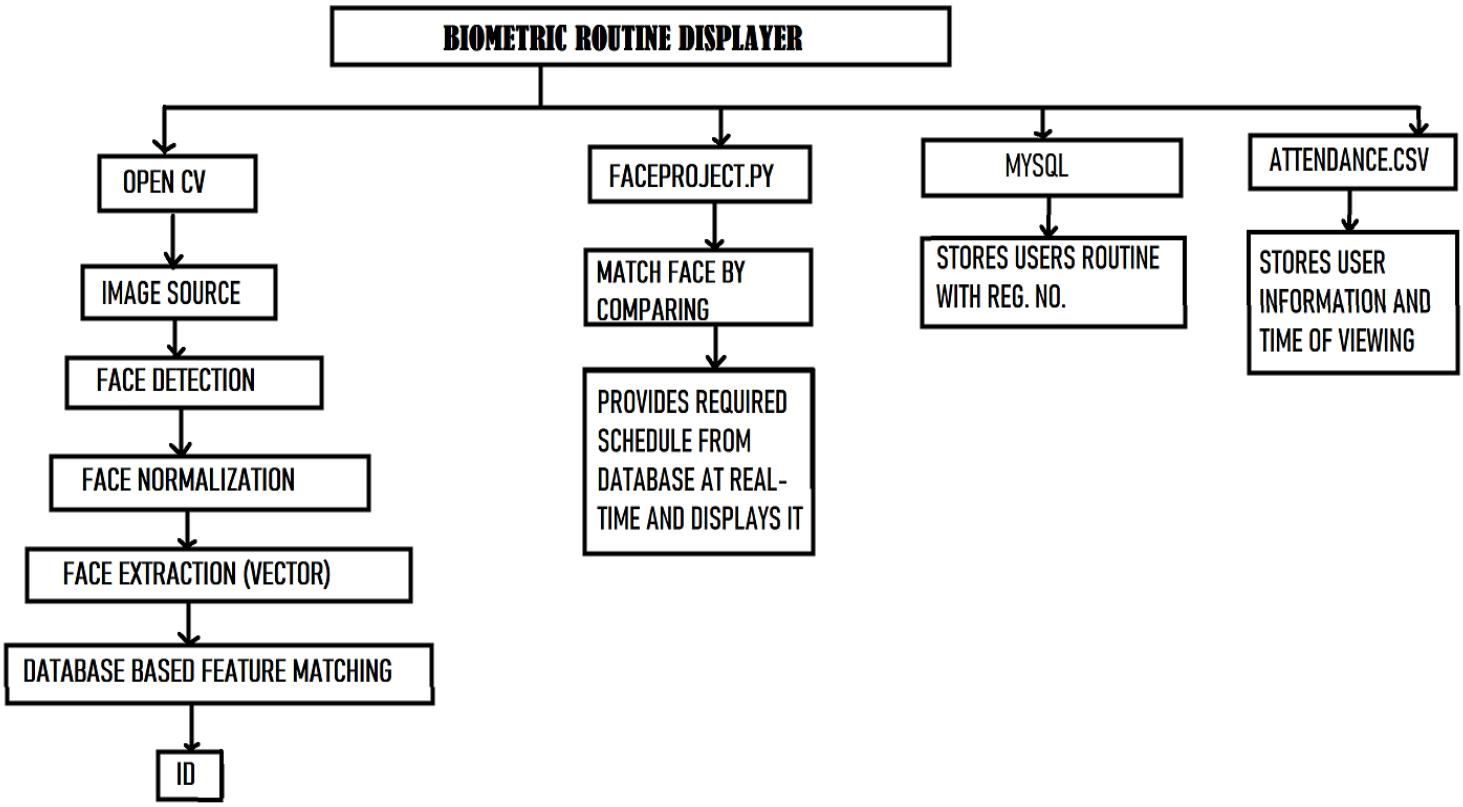
5. ***Import mysql.connector :*** It provide connectivity to the MySQL server for client programs.

6. ***Import calendar:*** It imports the date/day of the user viewing the Biometric Routine Display to provide accurate time table.

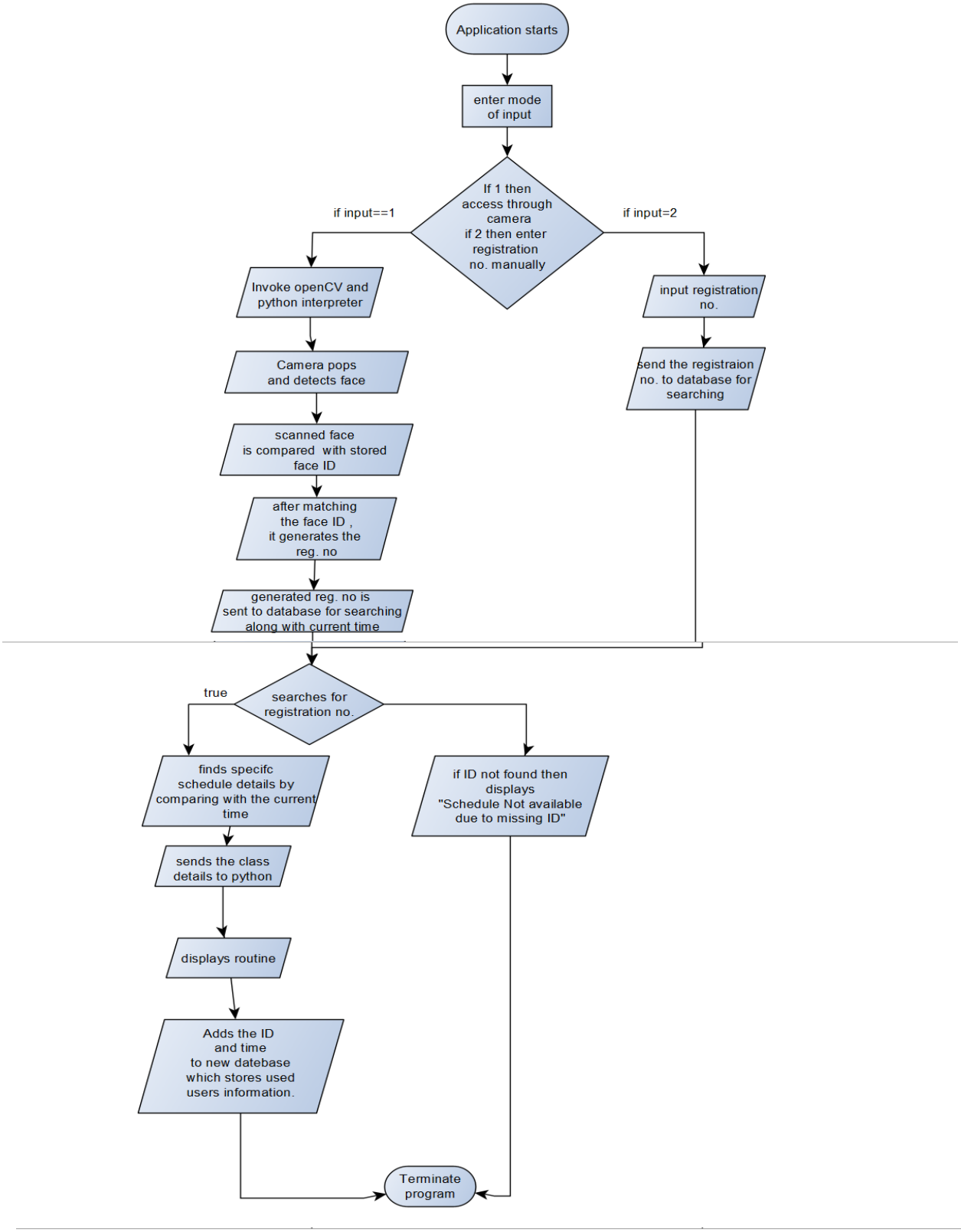
7. ***from pytz import timezone:*** It extracts the time of the user to provide him/her to access routine of that particular time.

8. ***from datetime import datetime:*** It imports all the content from the datetime module and provides reference to the class.

ARCHITECTURE:



Workflow :


Application starts
enter mode of input
If 1 then access through camera if 2 then enter registration no. manually
if input==1
if input=2
Invoke openCV and python interpreter
Camera pops and detects face
scanned face is compared with stored face ID
after matching the face ID , it generates the reg. no
generated reg. no is sent to database for searching along with current time
input registration no.
send the registraion no. to database for searching
searches for registration no.
true
finds specifc schedule details by comparing with the current time
sends the class details to python
displays routine
Adds the ID and time to new datebase which stores used users information.
if ID not found then displays "Schedule Not available due to missing ID"
Terminate program

8.Proposed Technique :

We have implemented our project with the help of python as scripting language and MySql as our relational database.

→In python we have imported different modules :

```
import cv2
import numpy as np
import face_recognition
import os
from datetime import datetime
import mysql.connector
import calendar
from pytz import timezone
from datetime import datetime
```

→we have connected our database with python by the help pf mysql.connector.

```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  password="ankush",
  database="project_data"
)
```

→ We have used the database which contains routines of all individuals for different day in different tables which are stored according to Day Name.

→The period is extracted by knowing the current time and accessing the respective column from the particular table.

```
def getdatabase_day(): #gets database name to use
  my_date = date.today()
  data_day = calendar.day_name[my_date.weekday()]
  return data_day.lower();
```

```
def get_time(): # gets time
  ind_time = datetime.now(timezone("Asia/Delhi"))
  intt=ind_time.strftime("%H");
  return intt;
```

```
def period_time(t): # columns extraction of databasse
  if(t=="8") :
    return "first";
  elif(t=="9"):
    return "second";
  elif(t=="10"):
    return "third";
  elif(t=="11"):
    return "fourth";
  elif(t=="12"):
    return "fifth";
  elif(t=="13"):
    return "sixth";
  elif(t=="14"):
    return "seventh";
  elif(t=="15"):
    return "eighth";
  elif(t=="16"):
    return "ninth";
  elif(t=="17") :
    return "tenth";
  elif(t=="18"):
    return "eleventh";
```

→We have used OpenCV module along with face_recognition module for recognizing the face and returning the unique ID number(i.e. Registration Number) by comparing the live face of the user with the stored data of the user's facial biometric.

```
# from PIL import ImageGrab
    path = 'ImagesAttendance'
    images = []
    classNames = []
    myList = os.listdir(path)
    print(myList)
    for cl in myList:
        curImg = cv2.imread(f'{path}/{cl}')
        images.append(curImg)
        classNames.append(os.path.splitext(cl)[0])
    print(classNames)


    def findEncodings(images):
        encodeList = []
        for img in images:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            encode = face_recognition.face_encodings(img)[0]
            encodeList.append(encode)
        return encodeList
    #### FOR CAPTURING SCREEN RATHER THAN WEBCAM
    # def captureScreen(bbox=(300,300,690+300,530+300)):
    #     capScr = np.array(ImageGrab.grab(bbox))
    #     capScr = cv2.cvtColor(capScr, cv2.COLOR_RGB2BGR)
    #     return capScr

    encodeListKnown = findEncodings(images)
    print('Encoding Complete')

    cap = cv2.VideoCapture(0)

while True:
        success, img = cap.read()
        # img = captureScreen()
```

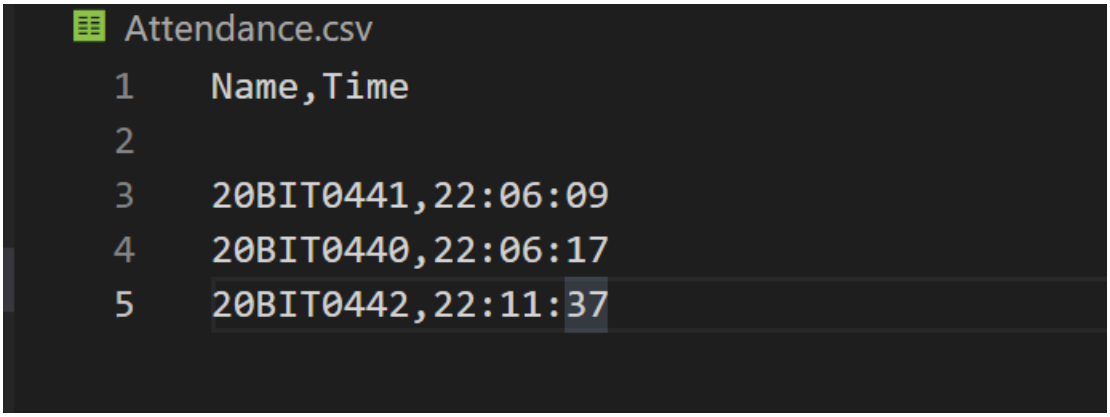
```
        imgS = cv2.resize(img, (0, 0), None, 0.25, 0.25)
        imgS = cv2.cvtColor(imgS, cv2.COLOR_BGR2RGB)

        facesCurFrame = face_recognition.face_locations(imgS)
        encodesCurFrame = face_recognition.face_encodings(imgS, facesCurFrame)

        for encodeFace, faceLoc in zip(encodesCurFrame, facesCurFrame):
            matches = face_recognition.compare_faces(encodeListKnown, encodeFace)
            faceDis = face_recognition.face_distance(encodeListKnown, encodeFace)
            # print(faceDis)
            matchIndex = np.argmin(faceDis)

            if matches[matchIndex]:
                name = classNames[matchIndex].upper()
                # print(name)
                y1, x2, y2, x1 = faceLoc
                y1, x2, y2, x1 = y1 * 4, x2 * 4, y2 * 4, x1 * 4
                cv2.rectangle(img, (x1, y1), (x2, y2), (0, 255, 0), 2)
                cv2.rectangle(img, (x1, y2 - 35), (x2, y2), (0, 255, 0),
cv2.FILLED)
                cv2.putText(img, name, (x1 + 6, y2 - 6),
cv2.FONT_HERSHEY_COMPLEX, 1, (255, 255, 255), 2)
                register=name
                markAttendance(name)
                print(register)

        cv2.imshow('Webcam', img)
        cv2.waitKey(1)
```

→after recognizing the face of the user, the program automatically adds the user's name along with the time of usage of platform to the attendance.csv file which can be used to clarify user's.

```
    def markAttendance(name):
        with open('Attendance.csv', 'r+') as f:
            myDataList = f.readlines()
            nameList = []
            for line in myDataList:
                entry = line.split(',')
                nameList.append(entry[0])
            if name not in nameList:
                now = datetime.now()
```

```
            dtString = now.strftime('%H:%M:%S')
            f.writelines(f'\n{name},{dtString}')
```

attendance.csv



→after using this application, the program also records the user's data into a separate database called attendance which is done by calling the function given below :

```
def store_in_db(register):
    mydb = mysql.connector.connect(
        host="localhost",
        user="root",
        password="password",
        database="project_osp"
    )
    mycursor = mydb.cursor()
    sql = "INSERT INTO attendance (register_no, time) VALUES (%s, %s)"
    # sql = "INSERT INTO view_info (register_no, time) VALUES (%s, %s)"
    now = datetime.now()
    current_time = now.strftime("%H:%M:%S")
    val = (register, current_time)
    mycursor.execute(sql, val)
    mydb.commit()
```

→After recognizing the user, the program accesses the Mysql database and finds for the particular output ie. Be it whole day's time table or current schedule.

```
print("Enter 1. For displaying whole day routine.")
print("Enter 2. For displaying current schedule")
mod = input()
if (mod == "1"): # for whole day's time table
  database_table_name = getdatabase_day()
  tim = "SELECT * FROM {} WHERE id =\"{}\" ".format(database_table_name,
register)
  mycursor.execute(tim)
  myresult = mycursor.fetchall()
  print("Your today's timetable is")
  for x in myresult:
    for y in x:
      print(y)
      condition = False
else:#for particular period
  database_table_name = a
  database_period = period_time(get_time())
  tim = "SELECT {} FROM {} WHERE id =\"{}\" ".format(database_period,
database_table_name, register)
  mycursor.execute(tim)
  myresult = mycursor.fetchall()
  for x in myresult:
    print("You have " + x[0] + " at " + get_time() + " o'clock," +
getdatabase_day())
    condition = False
```

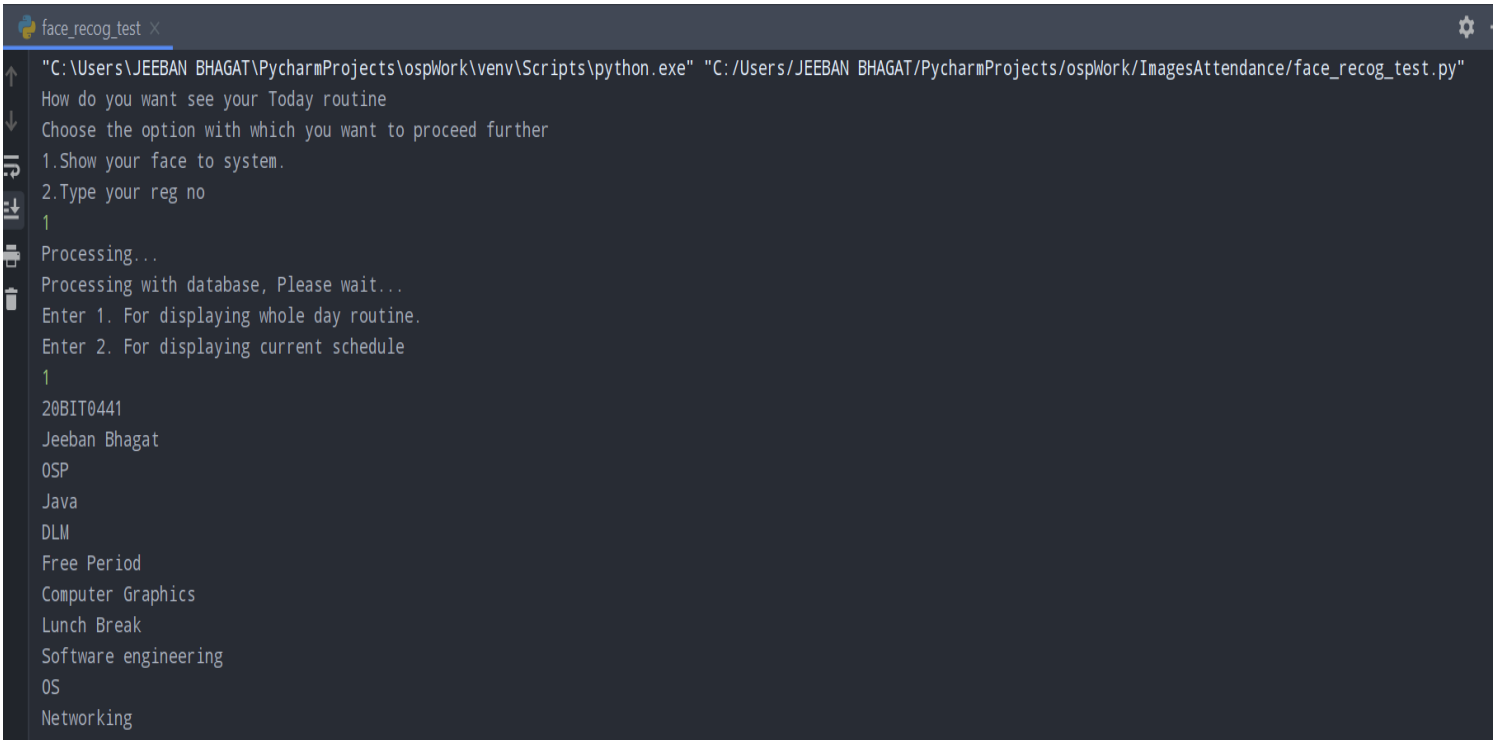
Same logic is used for Mechanical mode (physically entering the registration Number).

9. Results and Discussion

After running the code for the given project the output was as follows:

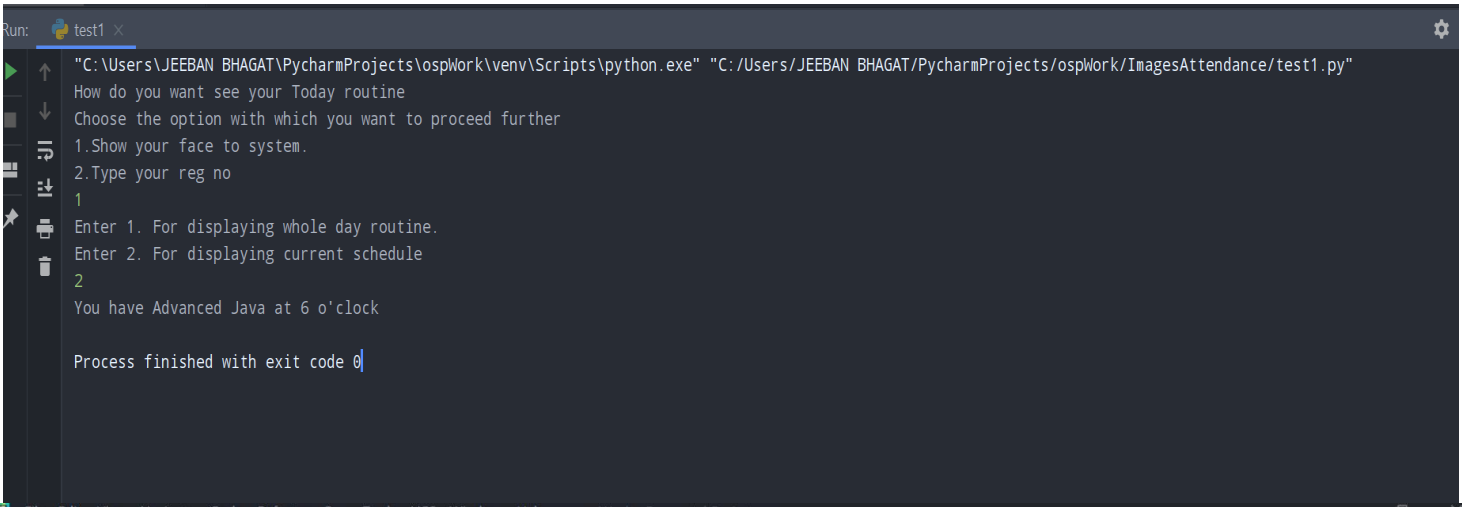
Case 1 : Facial Input for whole day's Routine :

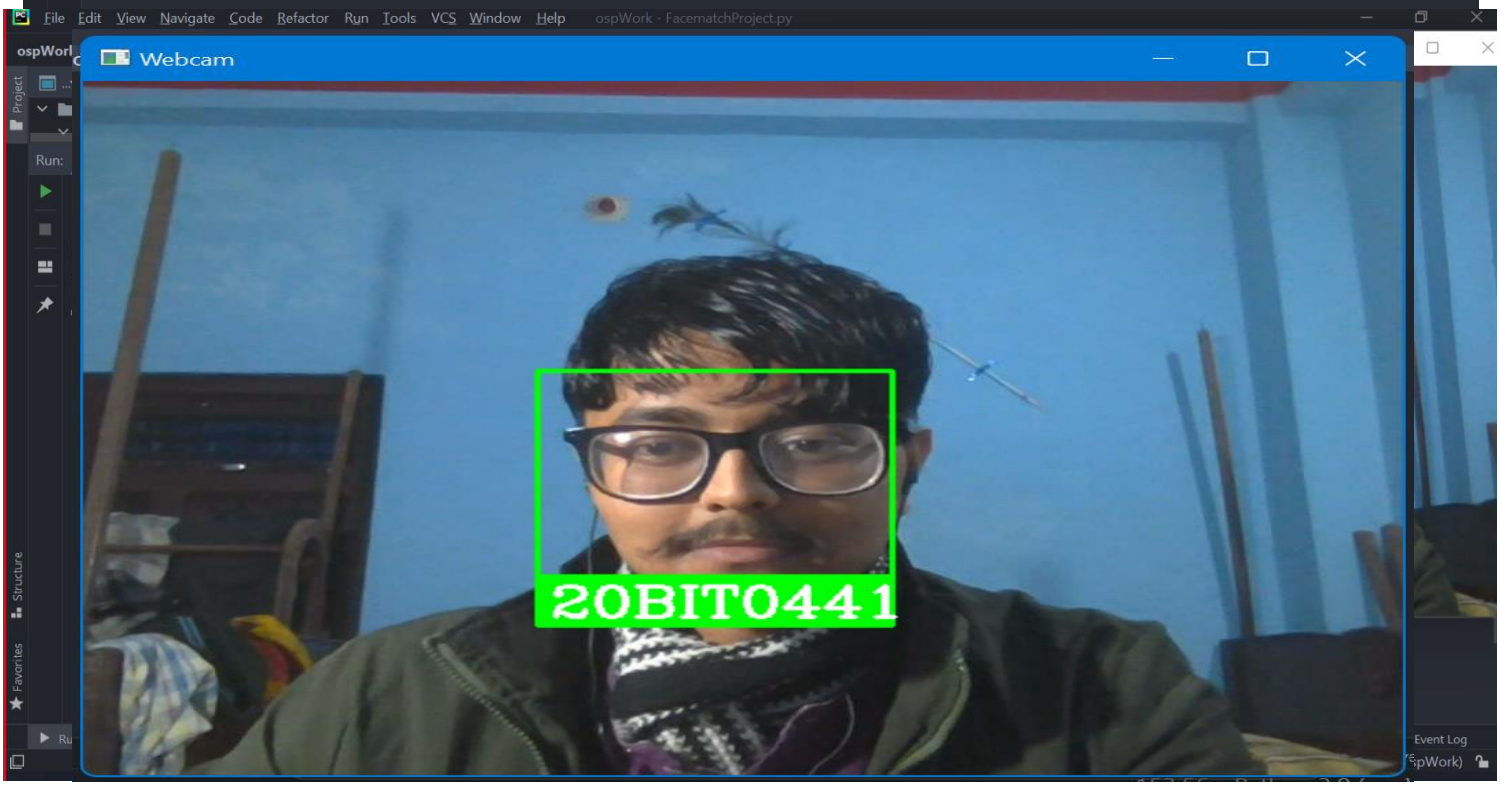
In this case, first it user inputs "1" for facial mode and then "1" again to get whole day's time table as output. It starts by scanning the face and then displays the required result.



Case 2 : Facial Input for particular Schedule display :

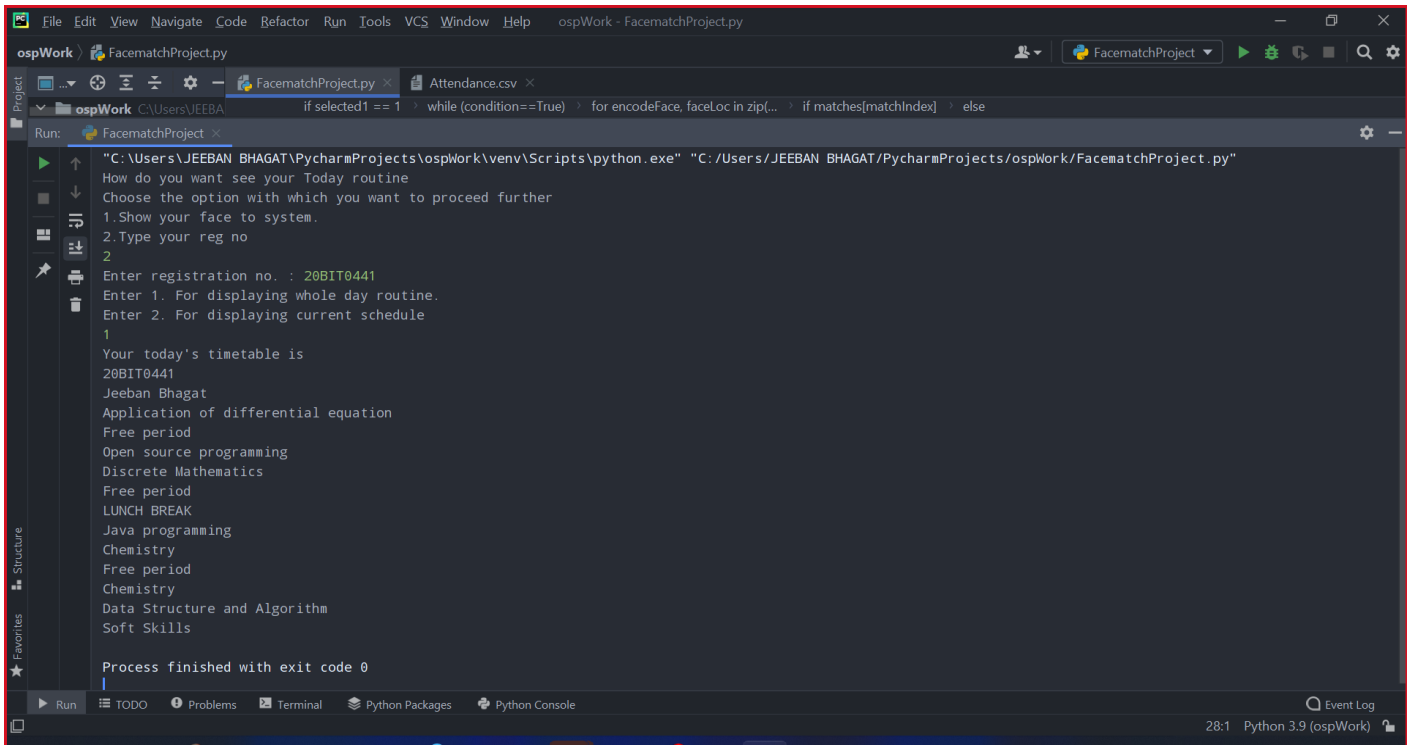
Here firstly the input is taken as"1" and then the mode as "2" which results in scanning the face and displaying the particular schedule at that point of time.





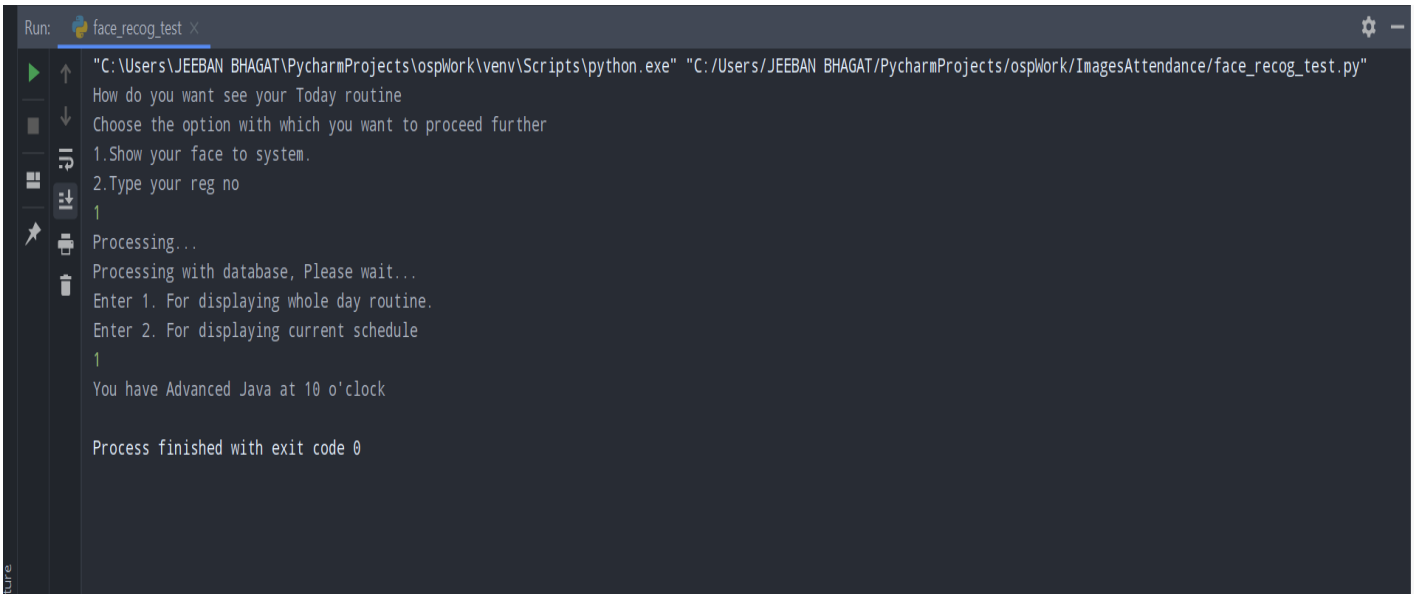
Case 3: Physical(Manual) Input for whole day's routine :

First the input is read as “2” and the mode as “1” which asks the user to manually enter the registration no. and then it displays the day's time table accordingly.



Case 4 : Physical (Manual) Input for particular Schedule:

Firstly the input is read as “2” and the mode of service as also “2”. Then the registration No. is entered manually and the particular schedule at that point of time is displayed by accessing from database.



Hence, the program runs in a user friendly manner along with the speed of performance being low. This application results in solving a major issue in large-scale organizations where working according to schedule is difficult to synchronize. Implementation of such system would eliminate their difficulty in remembering their time-schedule and would also help in not wasting time and effort as the usability is user-friendly and easy.

10.CONCLUSION/ FUTURE ENHANCEMENT:

‘Biometric Routine Displayer’ basically facilitates by accessing timetable using face recognition or registration number of the user. The transactions are executed off-line mode as well as on-line mode as per user’s convenience. It manages and stores the information for every user on daily basis and makes it easy for them by less consumption of time. It integrates all the records of daily subjects and displays them according to the timing student requires. It increases efficiency of students by saving their time. It also deals with monitoring the information and transactions of students which is saved in a database and can be viewed later. It can be accessed by multiple users at a particular time and does not display anything for unauthorized user.

In future we can enhance biometric techniques and add finger print scanner for more options for users.

The future enhancement that can be developed from the project is to generate the master timetable for the departments and to the entire college. This enhancement can be achieved my making further modifications keeping the approach and techniques used in this project.

The data of faculty in the data base can further be used to maintain record of faculty’s experience for particular subjects .

11.References :

https://www.w3schools.com/python/python_mysql_getstarted.asp#:~:text=Python%20needs%20a%20MySQL%20driver,installed%20in%20your%20Python%20environment.

https://stackoverflow.com/questions/46821936/open-cv-close-camera

https://stackoverflow.com/questions/52332268/pip-install-face-recognition-giving-error

https://www.w3schools.com/python/python_mysql_getstarted.asp#:~:text=Python%20needs%20a%20MySQL%20driver,installed%20in%20your%20Python%20environment.

https://pypi.org/project/opencv-python/

https://www.w3schools.com/python/python_mysql_select.asp

https://www.youtube.com/watch?v=sz25xxF_AVE