MAJOR PROJECT REPORT

(Project Term January – April 2020)

OVAS - OPEN VISION ATTENDANCE SYSTEM

Submitted by:

Apoorv Taneja	**Nice Sebastian**
11710639	**11720056**
Diploma (CSE)	**Diploma (CSE)**
JK702	**JK702**

Under the Guidance of

ASST. PROF. AMANDEEP KAUR SANDHU

ASST. PROF. ANITA SHARMA

DR. S. JAHANGIR

ASST. PROF. MEHAK SHARMA

ASST. PROF. DEVDUTT BARESARY

DR. PRIYA SAHA

Department of Computer Science & Engineering

School of Polytechnic

Lovely Professional University, Phagwara

January – April 2020

DECLARATION

I hereby declare that the project work entitled “OVAS - OPEN VISION ATTENDANCE SYSTEM” is an authentic record of my own work carried out as requirements of Major Project for the award of degree of Diploma (CSE) from Lovely Professional University, Phagwara, under the guidance of Asst. Prof. Amandeep Kaur Sandhu, Asst. Prof. Anita Sharma, Dr. S. Jahangir,
Asst. Prof. Mehak Sharma, Asst. Prof. Devdutt Baresary and Dr. Priya Saha, during January to April 2020.

Apoorv Taneja	Nice Sebastian
11710639	11720056

Date: 2 April 2020

This is to certify that the above statement made by the student is correct to the best of my knowledge and belief.
Asst. Prof. Amandeep Kaur Sandhu

Asst. Prof. Anita Sharma

Dr. S. Jahangir

Asst. Prof. Mehak Sharma

Asst. Prof. Devdutt Baresary

Dr. Priya Saha

ACKNOWLEDGEMENT

Apart from the efforts of myself and my project partner, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to show my greatest appreciation to Ms. Amandeep Kaur Sandhu Mam for her expertise, and technical support in the implementation. I can't say thank you enough for her tremendous support and help. Without her encouragement, superior knowledge, experience, and guidance this project would not have materialized the way it did, and thus her support has been essential for the completion of this project.

INDEX

INTRODUCTION

This Project is about solving an age-old method of taking attendance with the integration of recent technologies. Our University has an already Robust Attendance taking System which is integrated into the UMS of every teacher. But we can further improve this System by the addition of AI based Facial Recognition.

We can implement this by installing a camera on the entrance of the Classrooms and have them recognise the faces of the students who enter the classroom.

The AI will only recognise the faces of students who are registered to that section, whose class is to be held in that classroom.

The AI can also Work as extension to a teacher, telling them what number of students are currently present in the classroom.

EXISTING SYSTEM

At present attendance marking involves manual attendance on UMS interface of Every teacher in our University. Although this is a pretty Robust System, but there also exists some disadvantages of using this system. There also exist attendance marking system such as RFID, Biometrics etc. but these systems are currently not so much popular in schools and classrooms for students as they have their own advantages and disadvantages. The Problem with current System is it takes a lot of time off the faculty member, the time that could've otherwise been spent teaching important lessons. Another Problem that arises with current Attendance implementation is that when the faculty Roll Calls the Students, sometimes students are distracted to answer their Roll call. And Another Problem is that after the faculty is done with the Roll call, he/she has Count the Number of Students that are present in the class. All this Process when combined can be quite time consuming and sometimes troubling for the faculty members. And Furthermore, During Examination an Attendance Sheet is passed Around, which can be quite distracting for the students who are trying to concentrate on their exam paper. To solve these Problems, or at least trying to improve upon them I propose the following system.

PROPOSED SYSTEM

The face recognition-based student attendance system emphasis its simplicity by eliminating the problems existing in current student attendance marking system. Problems such as calling student names or checking respective identification cards in an Examination Hall. The lecture classes specially the class with a large number of students can be chaotic to take attendance. Thus, face recognition student attendance system is proposed in order to replace the Current System.

Using Local Binary Patterns Histograms (LBPH) algorithm in Open CV the student's faces are trained and recognized. Python automates the tasks by providing for the execution of the programs in Computer Vision and GUI of the system along with managing the database of the student attendance. Furthermore, the face recognition based automated student attendance system able to overcome the problem of fraudulent approach and lecturers does not have to count the number of students several times to ensure the presence of the students.

And furthermore, to avoid the risk of spoofing, the camera if installed inside the classroom can click an image or feed on the live video at a certain interval, say every 15 Minutes, it will detect the Number of Students and recognise their faces and punch the Attendance in the Last 10-15 Minutes of the Lecture. And if a student went out of the classroom for any reason, he will not be marked absent as this system will only punch in the attendance at the end of the lecture, and if it recognises the student in the classroom say 3 out of 4 times (Since taking attendance Every 15 Minutes), The Student will be Marked as Present.

FEASIBILITY STUDY

This Section will discuss that Whether the Project is Technically, Economically & Operationally Feasible.

1. **Economic Feasibility:**

   This new System can take high initial investment, for the installation of new cameras at the entrance of the classrooms, but these costs can be mitigated by using the current CCTV system that is present in the classrooms. But this also depends upon the quality of the recording taken by these cameras.

2. **Technical Feasibility:**

   This System can be fairly easily implanted by a small team of expert programmers. Taking into consideration the resources, that is needed to implement this system, most of the required libraries are open Source, i.e., free to use by all.

3. **Operational Feasibility:**

   This system after implementation would be fairly easy to maintain through the help of small number of people with basic skills. This System can provide a helping hand to the teachers & can also help in mitigating the time taken to take the attendance.

LOCAL BINARY PATTERN HISTOGRAM(LBPH)

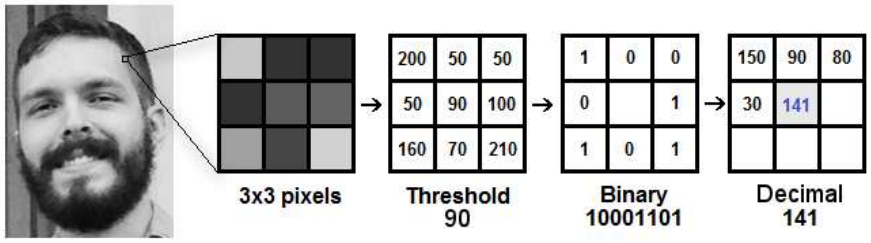
Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number.

It was first described in 1994 (LBP) and has since been found to be a powerful feature for texture classification. It has further been determined that when LBP is combined with histograms of oriented gradients (HOG) descriptor, it improves the detection performance considerably on some datasets. Using the LBP combined with histograms we can represent the face images with a simple data vector.

Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID.

The LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

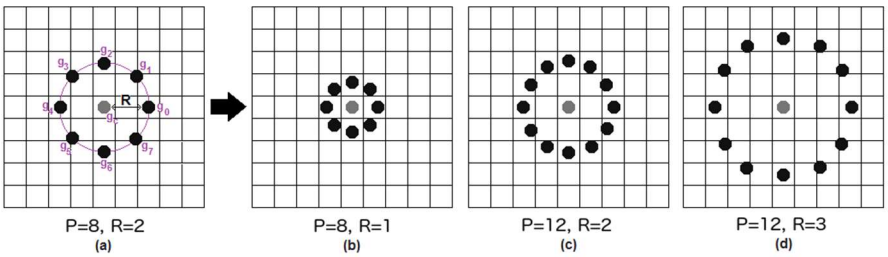
The image below shows this procedure:



Based on the image above, let's break it into several small steps so we can understand it easily:

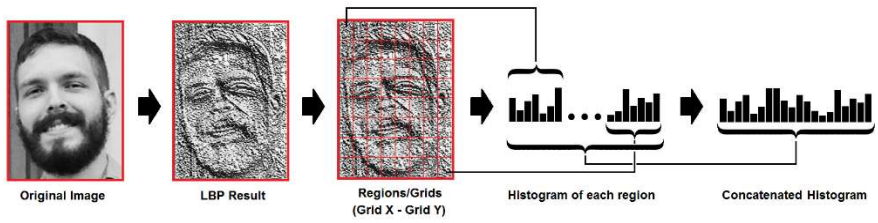
- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.

- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
- **Note**: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.



It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

Extracting the Histograms: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:



Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **Euclidean distance**, **chi-square**, **absolute value**, etc.

- So, the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a '**confidence**' measurement. **Note**: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

Conclusions

- LBPH is one of the easiest face recognition algorithms.
- It can represent local features in the images.
- It is possible to get great results (mainly in a controlled environment).
- It is robust against monotonic gray scale transformations.
- It is provided by the OpenCV library (Open Source Computer Vision Library).

FLASK (WEB FRAMEWORK)

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program.

Applications that use the Flask framework include Pinterest and LinkedIn.

FEATURES OF FLASK

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating
- Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

MODULES

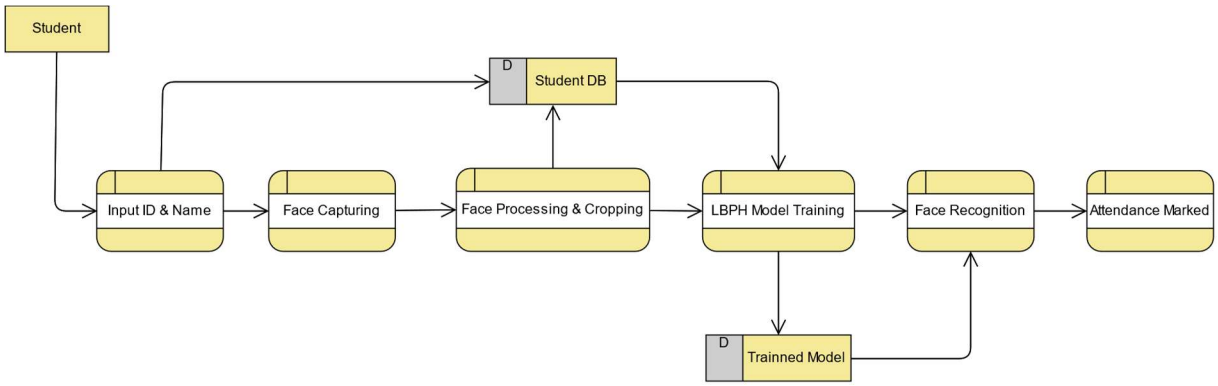
I. **Capturing the Image Data & Training the Data Set:** Taking the image data from the user, to be able to train the dataset. It can be done at the time of enrolment. The Image data will be stored in a folder, with the name of a file being the Name of the student and Registration number. At the time of scanning a data record of the student will also be created, so that so, the student can later be verified.

   Accessing the images from the dataset, in order to perform Training of the Recognition Model. The Model will use the Linear Binary Pattern Histogram (LBPH) algorithm.

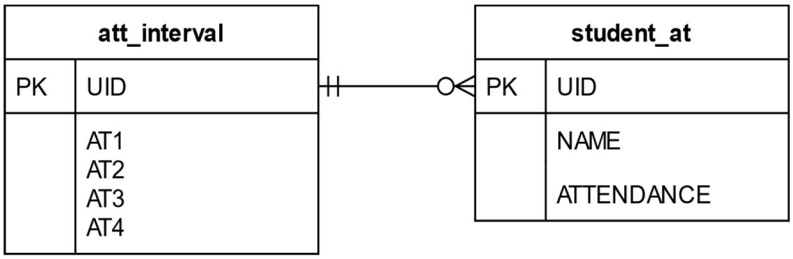
II. **Recognising and Marking the Attendance:** Taking the live feed through the camera and performing the operation of Marking Attendance. Here the Confidence of the Live image will be measured with the image data of already trained model. After certain threshold for the confidence is passed, the student than can be assigned a binary value, i.e. 1 for Present and 0 for Absent.

DIAGRAMS

DATA FLOW DIAGRAM



ENTITY RELATIONSHIP DIAGRAM



TECHNOLOGIES USED

These are following Software & Hardware technologies used to Implement the System.

1. Software:

- Visual Studio Code
- Atom
- Python 3.6
- Flask Web Framework
- SQLite – DB Browser
- Python Dependencies Requirements file is attached.
- Python Libraries
    - Flask
    - CSV
    - Pause
    - Numpy
    - OpenCV
    - Pandas
    - OS

2. Hardware:

- Laptop/Personal Computer
- Camera (HD)
- Internet Access (Not Must)

OVAS FILE STRUCTURE

Major Project\OVAS

```
|   app.py
|   haarcascade_frontalface_default.xml
|   Procfile
|   Recognize.py
|   requirements.txt
|   Resultant.txt
|   StudentDetails.db
|
+---.vscode
|       settings.json
|
+---ImagesUnknown
|       Image1.jpg
|       Image2.jpg
|
+---static
|   \---css
|           style.css
|
+---StudentDetails
|       StudentDetails.csv
|
+---templates
|       dashboard.html
|       index.html
|       marked.html
|       recognize.html
|       register.html
|
+---TrainingImage
|        Apurv Taneja.11710639.1.jpg
|        Apurv Taneja.11710639.10.jpg
|        Apurv Taneja.11710639.11.jpg
|        Apurv Taneja.11710639.12.jpg
|        Apurv Taneja.11710639.13.jpg
|
\---TrainnedModel
        Trainner.yml
```

CODE

OVAS\app.py

```
from flask import Flask, render_template, url_for, request, redirect
from datetime import datetime
import cv2
import csv
import numpy as np
import sqlite3
import os
import pause
from Recognize import TrackImages, getImagesAndId

app = Flask(__name__)

@app.route('/', methods=['POST', 'GET'])
def index():
    return render_template('index.html')

@app.route('/register', methods=['POST', 'GET'])
def register():
    if request.method == 'POST':
        conn = sqlite3.connect('StudentDetails.db')
        c = conn.cursor()
        Id= request.form['uid']
        name = request.form['name']
        c.execute("INSERT INTO student_at(Id,Name) VALUES (?, ?)",(Id,name))
        c.execute("INSERT INTO att_interval(Id) VALUES ("+ Id +")")
        conn.commit()
        c.close()
        conn.close()
        cam = cv2.VideoCapture(0)
        detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml"
)
        noOfImages = 0
        while True:
            ret, img = cam.read()
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            faces = detector.detectMultiScale(gray, 1.05, 3)
            for x, y, w, h in faces:
                cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
                # incrementing sample number
                noOfImages = noOfImages+1
                # saving the captured face in the dataset folder TrainingImage
```

```
                cv2.imwrite("TrainingImage\ "+name +"."+Id +'.'+ str(noOfImage
s) + ".jpg", gray[y:y+h,x:x+w])
                # display the frame
                cv2.imshow('Taking Images...', img)
            # wait for esc key or q
            if cv2.waitKey(100) and 0xFF == ord('q'):
                break
            # break if the sample number is more than 60. Meaning Images are m
ore than 60.
            elif noOfImages >= 20:
                break
        cam.release()
        cv2.destroyAllWindows()
        row = [Id, name]

        with open('StudentDetails\StudentDetails.csv','a+') as csvFile:
            writer = csv.writer(csvFile)
            writer.writerow(row)
        csvFile.close()

        # It is used to Recognise Face Data
        recognizer = cv2.face_LBPHFaceRecognizer.create()
        # Passing the Id & the Faces Received from getImagesAndId
        faces, Id = getImagesAndId("TrainingImage")
        # Training the LBPH Model
        recognizer.train(faces, np.array(Id))
        # Saving the Trained Model
        recognizer.save("TrainnedModel\Trainner.yml")

        return redirect(url_for('register'))
    return render_template('register.html')

@app.route('/dashboard', methods=['POST', 'GET'])
def dashboard():
    if request.method == 'POST':
        return redirect(url_for('dashboard'))

    return render_template('dashboard.html')

@app.route('/marked', methods=['POST', 'GET'])
def marked():
    conn = sqlite3.connect('StudentDetails.db')
    c = conn.cursor()
    sqlite_select_query = """SELECT * from student_at"""
    c.execute(sqlite_select_query)
    records = c.fetchall()
    conn.commit()
    conn.close()
```

```
    if request.method == 'POST':
        return redirect(url_for('marked'))
    return render_template('marked.html',records=records)

@app.route('/recognize', methods=['POST', 'GET'])
def recognize():
    if request.method == 'POST':
        return redirect(url_for('recognize'))
    else:
        TrackImages(1)

    return render_template('recognize.html')

if __name__ == "__main__":
    app.run(debug=True)
```

OVAS\Recognise.py

```
import cv2, os
import numpy as np
import pandas as pd
import sqlite3
import pause

def TrackImages(at):
    recognizer = cv2.face.LBPHFaceRecognizer_create()  # cv2.createLBPHFaceRec
ognizer()
    recognizer.read("TrainnedModel\Trainner.yml")
    harcascadePath = "haarcascade_frontalface_default.xml"
    faceCascade = cv2.CascadeClassifier(harcascadePath)
    StudentDetails = pd.read_csv("StudentDetails\StudentDetails.csv")
    noOfImages = 0
    font = cv2.FONT_HERSHEY_SIMPLEX
    cam = cv2.VideoCapture(0)

    conn = sqlite3.connect('StudentDetails.db')
    c = conn.cursor()

    while True:
        ret, img = cam.read()
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        faces = faceCascade.detectMultiScale(gray, scaleFactor=1.05, minNeighb
ors=3, minSize=(30, 30))
        noOfImages = noOfImages + 1
        #cv2.imwrite("Test\ "+str(noOfImages) + ".jpg", gray)
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x + w, y + h), (225, 0, 0), 2)
            Id, conf = recognizer.predict(gray[y:y + h, x:x + w])

            if conf < 50:
                idName = StudentDetails.loc[StudentDetails['Id'] == Id]['Name'
].values
                VideoTag = str(Id) + "-" + idName
                c.execute('UPDATE att_interval SET at' + str(at) + ' = 1 WHERE
  Id = '+str(Id))
                conn.commit()
            else:
                Id = 'Unknown'
                VideoTag = str(Id)
            if conf > 75:
                noOfFile = len(os.listdir("ImagesUnknown")) + 1
                cv2.imwrite("ImagesUnknown\Image" + str(noOfFile) + ".jpg", im
g[y:y + h, x:x + w])
```

```
            cv2.putText(img, str(VideoTag), (x, y + h), font, 1, (255, 255, 25
5), 2)
            cv2.putText(img, 'Number of Faces : ' + str(len(faces)), (40, 40),
 font, 1, (255, 255, 255), 2)
        cv2.imshow('Recognizing Face!!!', img)
        if cv2.waitKey(100) and 0xFF == ord('q'):
            break
        # break if the sample number is more than 60. Meaning Images are more
than 60.
        elif noOfImages > 20:
            break
    cam.release()
    cv2.destroyAllWindows()
    if at < 4:
        pause.seconds(1)
        TrackImages(at+1)
    else:
        sqlite_select_query = """SELECT * from att_interval"""
        c.execute(sqlite_select_query)
        records = c.fetchall()
        for row in records:
            Id_At = row[0]
            at1_At = row[1]
            at2_At = row[2]
            at3_At = row[3]
            at4_At = row[4]
            At = row[1] + row[2] + row[3] + row[4]
            if At >= 3:
                c.execute('UPDATE student_at SET Attendance  = 1 WHERE  Id = '
+str(Id_At))
        print("Attendence Updated")
        conn.commit()
        conn.close()


def getImagesAndId(path):
    # get the path of all the files in the folder
    imagePaths = [os.path.join(path, f)
                  for f in os.listdir(path)]
    faces = []  # create empty face list
    Ids = []  # create empty ID list
    # now looping through all the image paths and loading the Ids and the imag
es
    for imagePath in imagePaths:
        # loading the image and converting it to gray scale
        grayImage = cv2.imread(imagePath,0)
        # Now we are converting the image into numpy array
        imageNp = np.array(grayImage, 'uint8')
        # getting the Id from the image
```

```
        Id = int(os.path.split(imagePath)[-1].split(".")[1])
        # extract the face from the training image sample
        faces.append(imageNp)
        Ids.append(Id)
    return faces, Ids
```

OVAS\templates\index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>OVAS - Open Vison Attendance System</title>

  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.4.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/style.css'
) }}">

</head>

<body>
  <div class="container">
    <div class="row">
      <div class="col-sm-9 col-md-7 col-lg-5 mx-auto">
        <div class="card card-signin my-5">
          <div class="card-body">
            <h2 class="text-center">Welcome to</h2>
            <h1 class="text-center"><b>Open Vison Attendance System</b></h1>
            <br />
            <form class="form-signin">
              <a  href="{{ url_for('dashboard') }}" id="signin" class="btn btn
-lg btn-primary btn-block text-uppercase">Go to the dashboard</a>
              <hr class="my-4">
              <a href="{{ url_for('register') }}" class="btn btn-lg btn-
primary btn-block text-uppercase">Register</a>
            </form>
          </div>
        </div>
      </div>
    </div>
  </div>

</body>

</html>
```

OVAS\templates\dashboard.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>OVAS - Dashboard</title>
  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css"
    integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.4.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/style.css'
) }}">
  <style>
    a {
      color: black;
      text-decoration: none;
    }
    a:hover {
      color: gold;
      text-decoration: none;
    }

    a:active {
      color: black;
    }

    a:visited {
      color: black;
    }
  </style>
</head>

<body>
  <center>

    <div class="dash">
      <div class="card-signin dashin">
        <a href="{{ url_for('index') }}">
          <h1 class="text-center"><B>OVAS</B></h1>
          <h2 class="text-center"><b>Open Vison Attendance System</b></h2>
        </a>
        <div class='dashinn card-signin'>
```

```
          <h2 class="text-center"><b>Dashboard</b></h2>
          <br />
          <label for="section" style="font-size:16px;">Section:</label>
          <select id="section">
            <option value="JK701" disabled>JK701</option>
            <option value="JK702">JK702</option>
            <option value="JK703" disabled>JK703</option>
          </select>
          <div class="attendance card-signin">
            <form action="{{ url_for('recognize') }}" method="GET">
              <button id="mark" class="btn btn-lg btn-primary btn-block text-
uppercase att" type="submit">Start
                Recognising</button>
            </form>
            <br/>
            <form action="{{ url_for('marked') }}">
              <button id="mark" class="btn btn-lg btn-primary btn-block text-
uppercase att" type="submit">View
                Attendance Record</button>
            </form>
          </div>
        </div>
      </div>
    </div>

  </center>

</body>

</html>
```

OVAS\templates\register.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>OVAS - Register a Student</title>
  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css"
    integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.4.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/style.css'
) }}">
  <style>
    a {
      color: black;
      text-decoration: none;
    }
    a:hover {
      color: gold;
      text-decoration: none;
    }

    a:active {
      color: black;
    }

    a:visited {
      color: black;
    }
  </style>
</head>

<body>
  <center>
    <form action="/register" method="POST">
      <div class="dash">
        <div class="dashin card-signin">
          <a href="{{ url_for('index') }}">
            <h1 class="text-center"><B>OVAS</B></h1>
            <h2 class="text-center"><b>Open Vison Attendance System</b></h2>
          </a>
          <div class='dashinn card-signin'>
            <h2 class="text-center"><b>Register a Student</b></h2>
```

```
            <div class="flex-container">
              <div>

                <p style="font-size:22px;">UID :
                  <input name="uid" type="text" id="inputUID" class="form-
control" placeholder="UID" required />
                </p>

              </div>
              <div>

                <p style="font-size:22px; float:right;">Name :
                  <input name="name" type="text" id="inputName" class="form-
control" placeholder="Name" required />
                </p>

              </div>
            </div>

            <label for="section" style="font-size:22px;">Section:</label>
            <select id="section">
              <option value="JK701" disabled>JK701</option>
              <option value="JK702">JK702</option>
              <option value="JK703" disabled>JK703</option>
            </select>
            <div class="attendance card-signin">
              <input type="submit" id="mark" class="btn btn-lg btn-
primary btn-block text-uppercase att" value="Take Images & Train">
              </div>
          </div>
        </div>
      </div>
    </form>
  </center>

</body>

</html>
```

OVAS\templates\marked.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>OVAS - Dashboard</title>
  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css" integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.4.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/style.css'
) }}">
  <style>
    a {
      color: black;
      text-decoration: none;
    }
    a:hover {
      color: gold;
      text-decoration: none;
    }

    a:active {
      color: black;
    }

    a:visited {
      color: black;
    }
  </style>
</head>

<body>
  <center>

    <div class="dash">
      <div class="dashin card-signin">
        <a href="{{ url_for('index') }}">
          <h1 class="text-center"><B>OVAS</B></h1>
          <h2 class="text-center"><b>Open Vison Attendance System</b></h2>
        </a>
        <div class='dashinn card-signin'>
            <h2>Student Attendance Record</h2>
            <table class="table">
```

```
                <thead>
                  <tr>
                    <th scope="col">ID</th>
                    <th scope="col">Name</th>
                    <th scope="col">Attendance</th>
                  </tr>
                </thead>
                <tbody>
                    {% for item in records %}
                    <tr>
                        <td>{{item[0]}}</td>
                        <td>{{item[1]}}</td>
                        <td>{{item[2]}}</td>
                    </tr>
                    {% endfor %}

                </tbody>
              </table>
        </div>
      </div>
    </div>

  </center>

</body>

</html>
```

OVAS\templates\recognise.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <title>OVAS - Dashboard</title>
  <!-- Bootstrap -->
  <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.
4.1/css/bootstrap.min.css"
    integrity="sha384-
Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuhOf23Q9Ifjh" crossorigin=
"anonymous">
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/font-
awesome/4.4.0/css/font-awesome.min.css">
  <link rel="stylesheet" href="{{ url_for('static', filename = 'css/style.css'
) }}">
  <style>
    a {
      color: black;
      text-decoration: none;
    }
    a:hover {
      color: gold;
      text-decoration: none;
    }

    a:active {
      color: black;
    }

    a:visited {
      color: black;
    }
  </style>
</head>

<body>
  <center>

    <div class="dash">
      <div class="dashin card-signin">
        <a href="{{ url_for('index') }}">
          <h1 class="text-center"><B>OVAS</B></h1>
          <h2 class="text-center"><b>Open Vison Attendance System</b></h2>
        </a>
        <div class='dashinn card-signin'>
```

```
          <h2 class="text-center"><b>Dashboard</b></h2>
          <br />
          <label for="section" style="font-size:16px;">Section:</label>
          <select id="section">
            <option value="JK701" disabled>JK701</option>
            <option value="JK702">JK702</option>
            <option value="JK703" disabled>JK703</option>
          </select>
          <div class="attendance card-signin">
            <form action="{{ url_for('recognize') }}" method="GET">
              <button id="mark" class="btn btn-lg btn-primary btn-block text-
uppercase att" type="submit">Start
                Recognising</button>
            </form>
            <br/>
            <form action="{{ url_for('marked') }}">
              <button id="mark" class="btn btn-lg btn-primary btn-block text-
uppercase att" type="submit">View
                Attendance Record</button>
            </form>
          </div>
        </div>
      </div>
    </div>

  </center>

</body>

</html>
```

OVAS\static\css\style.css

```
:root {
  --input-padding-x: 1.5rem;
  --input-padding-y: .75rem;
}

body {
  background: #ffae3d;
  background: linear-gradient(to right,  #ffcd78,#ffce85);
  font-family: "Times New Roman", Times, serif;
}


.dashin{
  background-color: #ffffff;
  width : 75%;
  height: 75%;
  margin: 5rem;
  border-radius: 15px;
  padding: 2rem;
}


.flex-container{
  margin-top: 2rem;
  display: flex;
  justify-content: space-around;
}

td{
  padding: 5rem 0rem 2rem 0rem;
}

.card-signin {
  border: 0;
  border-radius: 1rem;
  box-shadow: 0 0.5rem 1rem 0 rgba(0, 0, 0, 0.1);
}

.card-signin .card-title {
  margin-bottom: 2rem;
  font-weight: 300;
  font-size: 1.5rem;
}

.card-signin .card-body {
  padding: 2rem;
}
```

```css
.attendance{
  border-radius: 15px;
  background-color: #fffdf7;
  padding: 2rem;
  width : 75%;
  margin-top:2rem;
}

.dashinn{
  border-radius: 15px;
  background-color: #fff9eb;
  padding: 2rem;
  width : 75%;
  margin-top:2rem;
}

.form-control{
  border-radius: 20px;
}
.att{
  background-color: #ff0000;
  font-size: 80%;
  border-radius: 5rem;
  letter-spacing: .1rem;
  font-weight: bold;
  padding: 1rem;
  transition: all 0.2s;
  width: 50%;
}
.att:hover{
  background: #de0000;
}

.at {
  background: #de0000;
}

#signin{
  background-color: #ff0000;
  /* background: linear-gradient(to right, #d91111, #ff4242); */
}

#signin:hover{
  background: #de0000;
}

.form-signin {
```

```
  width: 100%;
}

.form-signin .btn {
  font-size: 80%;
  border-radius: 5rem;
  letter-spacing: .1rem;
  font-weight: bold;
  padding: 1rem;
  transition: all 0.2s;
}

.form-label-group {
  position: relative;
  margin-bottom: 1rem;
}

.form-label-group input {
  height: auto;
  border-radius: 2rem;
}

.form-label-group>input,
.form-label-group>label {
  padding: var(--input-padding-y) var(--input-padding-x);
}

.form-label-group>label {
  position: absolute;
  top: 0;
  left: 0;
  display: block;
  width: 100%;
  margin-bottom: 0;
  /* Override default `<label>` margin */
  line-height: 1.5;
  color: #495057;
  border: 1px solid transparent;
  border-radius: .25rem;
  transition: all .1s ease-in-out;
}

.form-label-group input::-webkit-input-placeholder {
  color: transparent;
}

.form-label-group input:-ms-input-placeholder {
  color: transparent;
```

```
}

.form-label-group input::-ms-input-placeholder {
  color: transparent;
}

.form-label-group input::-moz-placeholder {
  color: transparent;
}

.form-label-group input::placeholder {
  color: transparent;
}

.form-label-group input:not(:placeholder-shown) {
  padding-top: calc(var(--input-padding-y) + var(--input-padding-
y) * (2 / 3));
  padding-bottom: calc(var(--input-padding-y) / 3);
}

.form-label-group input:not(:placeholder-shown)~label {
  padding-top: calc(var(--input-padding-y) / 3);
  padding-bottom: calc(var(--input-padding-y) / 3);
  font-size: 12px;
  color: #777;
}
/* Fallback for Edge
-------------------------------------------------- */
@supports (-ms-ime-align: auto) {
  .form-label-group>label {
    display: none;
  }
  .form-label-group input::-ms-input-placeholder {
    color: #777;
  }
}
/* Fallback for IE
-------------------------------------------------- */
@media all and (-ms-high-contrast: none),
(-ms-high-contrast: active) {
  .form-label-group>label {
    display: none;
  }
  .form-label-group input:-ms-input-placeholder {
    color: #777;
  }
}
```

OVAS\requirements.txt

```
dlib==19.19.0
dodgy==0.2.1
Flask==1.1.1
Flask-SQLAlchemy==2.4.1
gunicorn==20.0.4
Jinja2==2.11.1
jsonschema==3.2.0
matplotlib==3.2.1
notebook==6.0.3
numpy==1.17.4
opencv-contrib-python==4.2.0.32
opencv-python==4.1.2.30
pandas==1.0.1
pause==0.2
python-dateutil==2.8.1
qtconsole==4.7.0
QtPy==1.9.0
scipy==1.4.1
seaborn==0.10.0
SQLAlchemy==1.3.15
sqlparse==0.3.0
terminado==0.8.3
testpath==0.4.4
tornado==6.0.3
tqdm==4.43.0
traitlets==4.3.3
typed-ast==1.4.1
typing-extensions==3.7.4.1
virtualenv==20.0.15
wcwidth==0.1.8
webencodings==0.5.1
Werkzeug==1.0.1
widgetsnbextension==3.5.1
wrapt==1.11.2
```

REFERENCES

- **Learn Flask for Python - Full Tutorial –** (https://www.youtube.com/watch?v=Z1RJmh_OqeA)
- **Flask Documentation** – (https://flask.palletsprojects.com/en/1.1.x/)
- **Python for Computer Vision with OpenCV and Deep Learning -** UDEMY (https://www.udemy.com/course/python-for-computer-vision-with-opencv-and-deep-learning/)
- **Deep Learning A-Z™: Hands-On Artificial Neural Networks -** UDEMY (https://www.udemy.com/course/deeplearning/)
- **W3SCHOOLS** (https://www.w3schools.com/python/)
- **Python Documentation** (https://docs.python.org/3/tutorial/)
- **OpenCV** (https://docs.opencv.org/2.4/doc/tutorials/tutorials.html)
- **GeeksForGeeks** (https://www.geeksforgeeks.org/python-programming-language/)
- **Basics of the Classic CNN –** Towards Data Science (https://towardsdatascience.com/basics-of-the-classic-cnn-a3dce1225add)
- **Applied Deep Learning - Part 4: Convolutional Neural Networks –** Towards Data Science (https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2)
- **Deploying Simple Flask App with Google Cloud Instance -** (https://www.youtube.com/watch?v=QUYCiIkzZlA)