ATTENDANCE PORTAL

A MINI PROJECT REPORT

18CSC305J – ARTIFICAL INTELLIGENCE

Submitted by

Shridhar Shrivas [RA2011003010280]

Devansh Singh [RA2011003010295]

Aryan Chaturvedi [RA2011003010298]

Under the guidance of

Dr. L. KAVISANKAR

Associate Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY

Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini Project report titled **"Attendance Portal"** is the bona fide work of **Shridhar Shrivas(RA20110030101280), Devansh Singh(RA2011003010295), Aryan Chaturvedi(RA2011003010298)** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr. L. KAVISANKAR	Dr. M. Pushpalatha
GUIDE	**HEAD OF THE DEPARTMENT**
Associate Professor	Professor & Head
Department of Computing Technologies	Department of Computing Technologies

ABSTRACT

More than 3000 students and hundreds of courses are offered at a large campus like SRM Institute. Faculty members must record attendance and the process is currently manual and time-consuming. Consequently, creating an attendance portal that will automatically fill out the attendance and mark a student absent if they are not physically present in class.

This project aims to develop an AI-based attendance portal for the SRM Institute that employs deep learning's face_recognition library to recognize faces and automate the attendance tracking process. The system will use the HOG algorithm to improve its performance and operate on a variety of interfaces and devices through a user-friendly interface. The project's objectives include improving the accuracy and reliability of attendance data, reducing the time and effort required to record attendance, and providing a seamless and efficient experience for both students and faculty members. By achieving these objectives, the project will contribute to the development of more accurate and robust student screening processes at the SRM Institute.

The proposed AI-based attendance portal will not only automate the attendance tracking process but also provide more accurate and reliable attendance data, helping faculty members make informed decisions. The system will also reduce the time and effort required to record attendance, freeing up valuable time for instructors to focus on teaching and other important tasks. Additionally, the system's user-friendly interface and compatibility with various devices will ensure that it is accessible to all users, regardless of their technical expertise. Overall, this project aims to improve the attendance tracking process at the SRM Institute and provide a more seamless and efficient experience for both students and faculty members

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine

CHAPTER 1

INTRODUCTION

The attendance portal is an application designed to automatically track and record the attendance of individuals in various settings, such as classrooms, meetings, or workplaces. With the increasing need for accurate and efficient attendance tracking systems, the use of deep learning and computer vision techniques has become an important application for improving attendance management. One effective technique for attendance tracking is the use of Histograms of Oriented Gradients (HOG) algorithm, combined with deep learning models such as Convolutional Neural Networks (CNNs) and Support Vector Machines (SVMs). These techniques are designed to accurately detect and recognize individuals in real-time by analyzing their facial features and patterns, and can be used in a variety of settings to improve attendance management and reduce administrative burden.

The goal of this project report is to provide an overview of the methodology for developing an Attendance Portal using HOG Algorithm and Deep Learning, as well as some potential future enhancements for the field. We will discuss the steps involved in the methodology, including data acquisition, preprocessing, feature extraction using HOG Algorithm, model architecture design using Deep Learning, model training, evaluation, and deployment. We will also discuss some of the potential future enhancements for Attendance Portal using HOG Algorithm and Deep Learning, such as real-time attendance tracking, facial recognition, and integration with other systems. By providing a comprehensive overview of Attendance Portal using HOG Algorithm and Deep Learning, this report aims to contribute to the ongoing development and refinement of attendance tracking systems.

Recording attendance manually can be a tedious and time-consuming task, especially in large campuses with hundreds of courses and thousands of students. Traditional attendance tracking methods such as paper-based records or barcode scanning can be error-prone and inefficient. The use of deep learning and HOG algorithm can overcome these challenges by enabling accurate and automated attendance tracking. These techniques allow the system to recognize and identify faces with high accuracy and speed, making it possible to record attendance automatically and seamlessly. By using deep learning and HOG algorithm, the system can also handle various challenges such as variations in lighting, angle, and facial expressions. This makes the attendance portal more reliable and effective than traditional attendance tracking methods.

This project report aims to provide an overview of the methodology for developing an Attendance Portal using HOG Algorithm and Deep Learning, as well as some potential future enhancements for the field. We will discuss the steps involved in the methodology, including data acquisition, pre-processing, feature extraction using HOG Algorithm, model architecture design using Deep Learning, model training, evaluation, and deployment. We will also discuss some of the potential future enhancements for Attendance Portal using HOG Algorithm and Deep Learning, such as real-time attendance tracking, facial recognition, and integration with other systems. By providing a comprehensive overview of Attendance Portal using HOG Algorithm and Deep Learning, this report aims to contribute to the ongoing development and refinement of attendance tracking systems.

CHAPTER 2

LITERATURE SURVEY

Some of the research papers and articles regarding HOG and SVM are:

- **"Histograms of Oriented Gradients for Human Detection" by Navneet Dalal and Bill Triggs (2005)**

  This paper introduces the HOG feature descriptor and its application for human detection in images. The authors demonstrate the effectiveness of HOG features in detecting humans with high accuracy.

- **"Face Detection using Histograms of Oriented Gradients" by Vaishali Patil and R. S. Prasad (2011)**
  This paper proposes a method for face detection using HOG features and an SVM classifier. The authors demonstrate the effectiveness of their method on a variety of face datasets.

- **"Deep Residual Learning for Image Recognition" by Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016)**

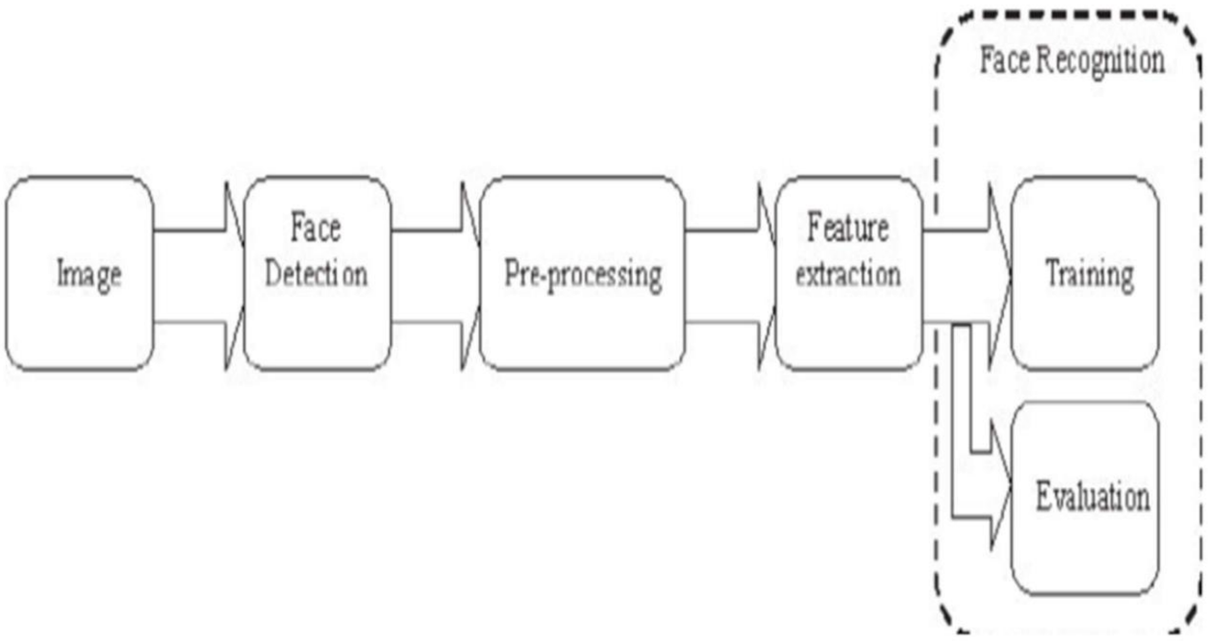
  This paper presents the ResNet architecture, which is a deep neural network that achieved state-of-the-art performance on the ImageNet classification task. The authors introduce the residual learning framework, which enables the training of much deeper networks than previously possible.

- **"A Deep Learning Framework for Object Detection Using HOG Feature-Based SVM Classifier" by Anitha R, et al. (2019) -** This paper proposes a deep learning framework that uses a HOG feature-based SVM classifier for object detection and achieves better performance than traditional methods.

- **"An efficient method for human detection in video surveillance using HOG and deep learning" by S. R. Yadav and S. S. Tyagi.** This paper proposes a human detection system for video surveillance that uses HOG feature extraction and deep learning models.

- **"A new deep learning-based approach for facial expression recognition using HOG features" by J. Zhang, W. Guo, and M. Liu.** This paper presents a new approach for facial expression recognition that combines HOG feature extraction with deep learning models.

CHAPTER 3
SYSTEM ARCHITECTURE AND DESIGN



The project involves the use of Histogram of Oriented Gradients (HOG) and Support Vector Machine (SVM). HOG (Histogram of Oriented Gradients) is a feature descriptor that counts the occurrences of gradient orientation in localized portions of an image, while SVM (Support Vector Machine) is a supervised learning algorithm used for classification and regression analysis.

The proposed architecture includes:

- **Image Acquisition:** In this step, images of individuals are acquired using a camera or other imaging device such as a webcam of a laptop.

- **Face Detection:** In this step, the faces in the acquired images are detected using the HOG algorithm, which detects the edges of the face and creates a bounding box around it.

- **Preprocessing:** In this step, the detected faces are preprocessed to normalize them for variations in lighting conditions, pose, and expression. This may involve techniques such as histogram equalization, face alignment, and scaling.

- **Feature Extraction:** In this step, features are extracted from the preprocessed face images. These features may include HOG descriptors, which capture the local orientation gradients of the face, or deep learning-based features such as those extracted from a convolutional neural network (CNN).

- **Face Recognition:** In this step, the features extracted from the faces are used to recognize individuals using a machine learning algorithm such as SVM. The SVM algorithm learns to classify faces based on the features extracted from the training data, and can then be used to classify new faces and match them to known individuals in a database.

  The images recognized are evaluated from a pre-existing dataset.

- **Attendance Management:** In this final step, the attendance of the recognized individuals is recorded in a CSV File where the name of the individual is saved along with the date and time of attendance. This can be used for attendance tracking, payroll management, and other applications.

CHAPTER 4

METHODOLOGY

The Attendance Portal is performed through Deep Learning techniques by the application of HOG and SVM. They include:

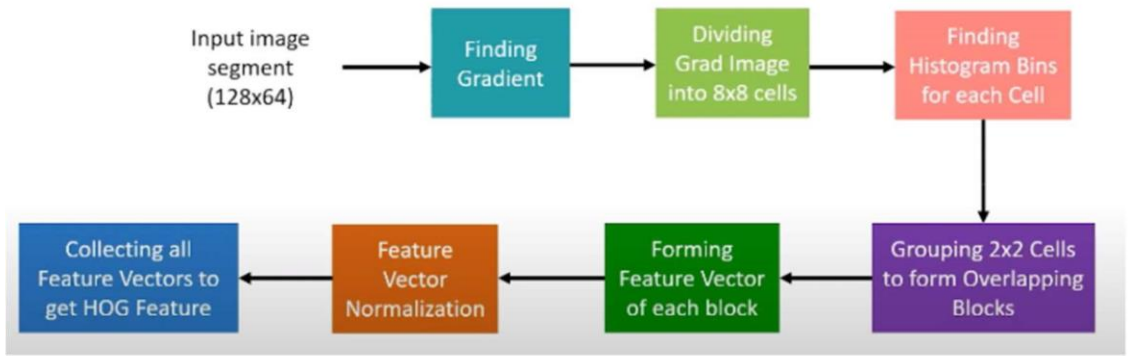
- The face_recognition library uses a deep neural network model to extract facial features and recognize faces, and it is based on the dlib library which uses a histogram of oriented gradients (HOG) feature descriptor and a linear SVM classifier. Therefore, the code is implementing a deep learning-based face recognition algorithm.
- HOG, which stands for Histogram of Oriented Gradients, is a feature extraction algorithm widely used in computer vision and image processing. HOG is a technique that extracts features from an image and represents them in a vector format
- HOG is often used as a pre-processing step for machine learning algorithms in computer vision.
- Linear SVM (Support Vector Machine) classifier is a type of machine learning algorithm used for classification tasks. It is a binary classifier that tries to find the best decision boundary between two classes in a feature space.
- The face_recognition library is a popular Python library used for face recognition and face detection. The library uses a deep neural network model

to extract facial features and recognize faces. The model is based on the dlib library which uses a histogram of oriented gradients (HOG) feature descriptor and a linear SVM classifier.

- HOG works by dividing the image into small regions and calculating the gradient of intensity in each region.
- The gradient is a measure of the change in intensity over a small distance, and it indicates the edges and boundaries in an image. The HOG algorithm then calculates a histogram of the gradients in each region.
- This histogram represents the distribution of the orientations of the gradients in the region. The final feature vector for the image is obtained by concatenating the histograms from all the regions.

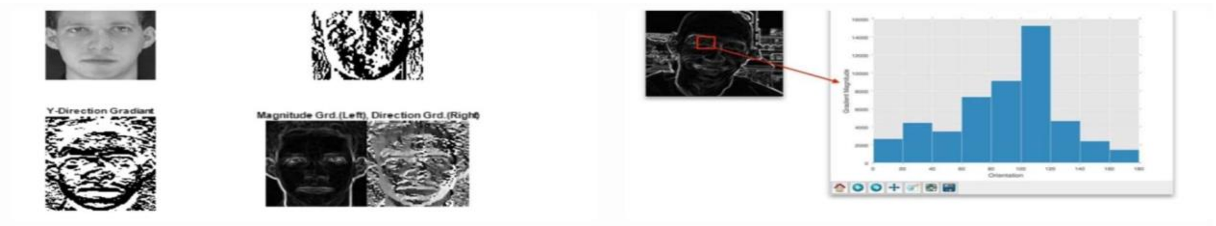
- The SVM algorithm works by finding the hyperplane that best separates the data into different classes. In the case of face recognition, the SVM classifier is trained on a set of face images and non-face images. The classifier learns to distinguish between the features of face images and non-face images, and it can then be used to classify new images as either face or non-face.

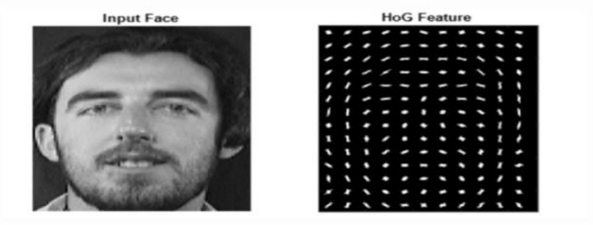
- In summary, The HOG algorithm is used to extract features from an image, and the SVM classifier is used to classify the features as face or non-face.


Input image segment (128x64)
Finding Gradient
Dividing Grad Image into 8x8 cells
Finding Histogram Bins for each Cell
Grouping 2x2 Cells to form Overlapping Blocks
Forming Feature Vector of each block
Feature Vector Normalization
Collecting all Feature Vectors to get HOG Feature

❖ Work Flow for HOG Feature-

- ➢ Provide your face image as an input.
- ➢ Finding the gradient of the image using HOG Algorithm and dividing the gradient image into 8*8 cells
- ➢ Forming Feature Vector for each block using the SVM Linear Algorithm and implementing Feature Vector Normalization.
- ➢ Collecting all Feature Vectors to get HOG Feature and thus successfully implementing the Attendance Portal once the input image is scanned and output is given.





CHAPTER 5

CODING AND TESTING

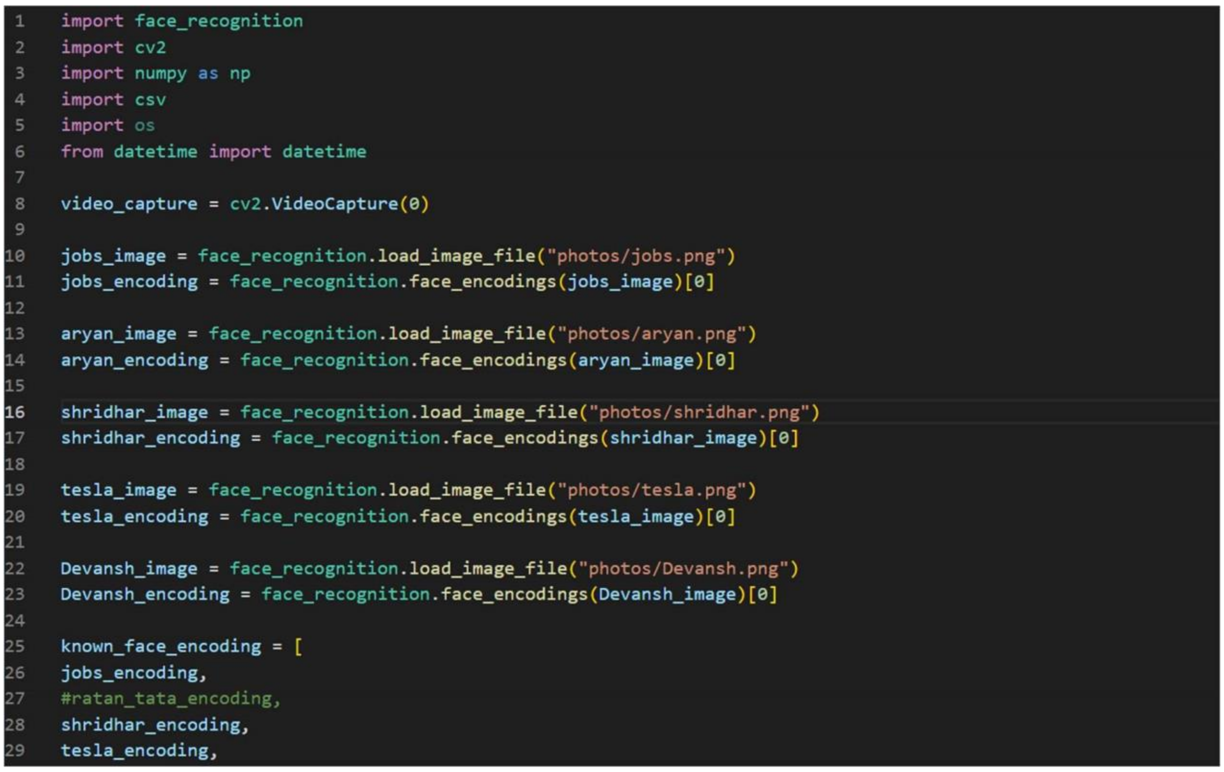
```
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime

video_capture = cv2.VideoCapture(0)

jobs_image = face_recognition.load_image_file("photos/jobs.png")
jobs_encoding = face_recognition.face_encodings(jobs_image)[0]

aryan_image = face_recognition.load_image_file("photos/aryan.png")
aryan_encoding = face_recognition.face_encodings(aryan_image)[0]

shridhar_image = face_recognition.load_image_file("photos/shridhar.png")
shridhar_encoding = face_recognition.face_encodings(shridhar_image)[0]

tesla_image = face_recognition.load_image_file("photos/tesla.png")
tesla_encoding = face_recognition.face_encodings(tesla_image)[0]

Devansh_image = face_recognition.load_image_file("photos/Devansh.png")
Devansh_encoding = face_recognition.face_encodings(Devansh_image)[0]

known_face_encoding = [
jobs_encoding,
#ratan_tata_encoding,
shridhar_encoding,
tesla_encoding,
Devansh_encoding,
aryan_encoding,
]
```

```
known_faces_names = [
"JOBS",
#"RATAN TATA",
"Shridhar",
"TESLA",
"Devansh",
"Aryan",
]

students = known_faces_names.copy()

face_locations = []
face_encodings = []
face_names = []
s=True



now = datetime.now()
current_date = now.strftime("%Y-%m-%d")




f = open(current_date+'.csv','w+',newline = '')
lnwriter = csv.writer(f)

while True:
    _,frame = video_capture.read()
    small_frame = cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
    rgb_small_frame = small_frame[:,:,::-1]
    if s:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings =
face_recognition.face_encodings(rgb_small_frame,face_locations)
        face_names = []
        for face_encoding in face_encodings:
            matches =
face_recognition.compare_faces(known_face_encoding,face_encoding,
tolerance=0.5)
```

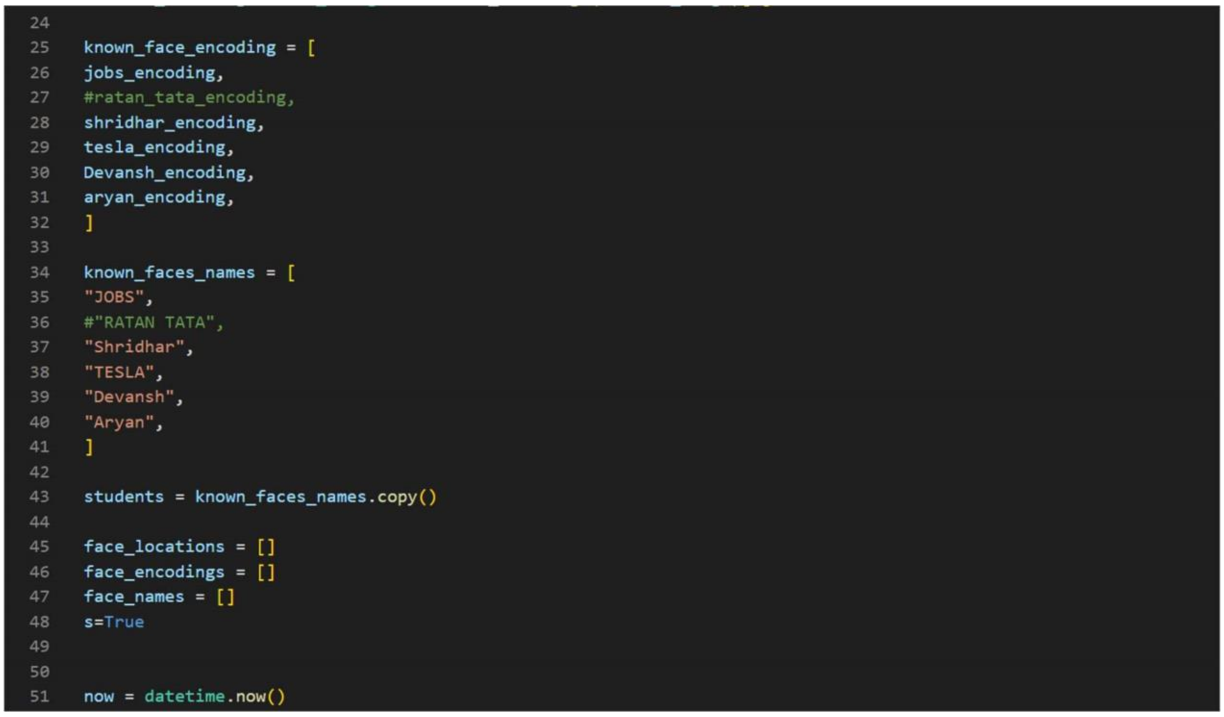
```
            name="Unknown"
            face_distance =
face_recognition.face_distance(known_face_encoding,face_encoding)
            best_match_index = np.argmin(face_distance)
            if matches[best_match_index]:
                name = known_faces_names[best_match_index]

            face_names.append(name)
            if name != "Unknown":
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
                fontScale              = 1.5
                fontColor              = (255,0,0)
                thickness              = 3
                lineType               = 2

                cv2.putText(frame,name+' Present',
                    bottomLeftCornerOfText,
                    font,
                    fontScale,
                    fontColor,
                    thickness,
                    lineType)

                if name in students:
                    students.remove(name)
                    print(students)
                    current_time = now.strftime("%H hours : %M minutes : %S seconds")
                    print(f"Writing attendance for {name} at {current_time}")
                    lnwriter.writerow([name,current_time])
                    f.flush()
            else:
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
                fontScale              = 1.5
                fontColor              = (0,0,255)  # Red color for non-registered faces
                thickness              = 3
                lineType               = 2
            cv2.putText(frame,'Face not recognized',
                   bottomLeftCornerOfText,
                   font,
                   fontScale,
                   fontColor,
                   thickness,
                   lineType)
    cv2.imshow("attendence system",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
f.close()
```

CHAPTER 6

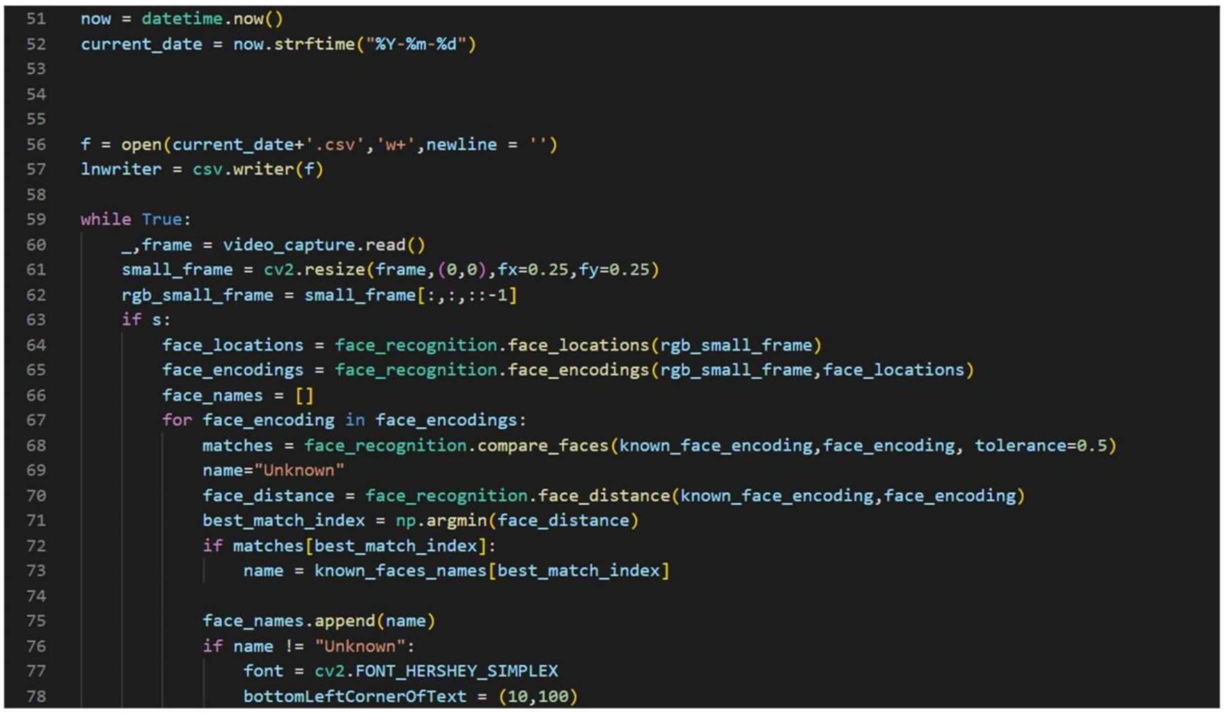
SCREENSHOTS AND RESULTS



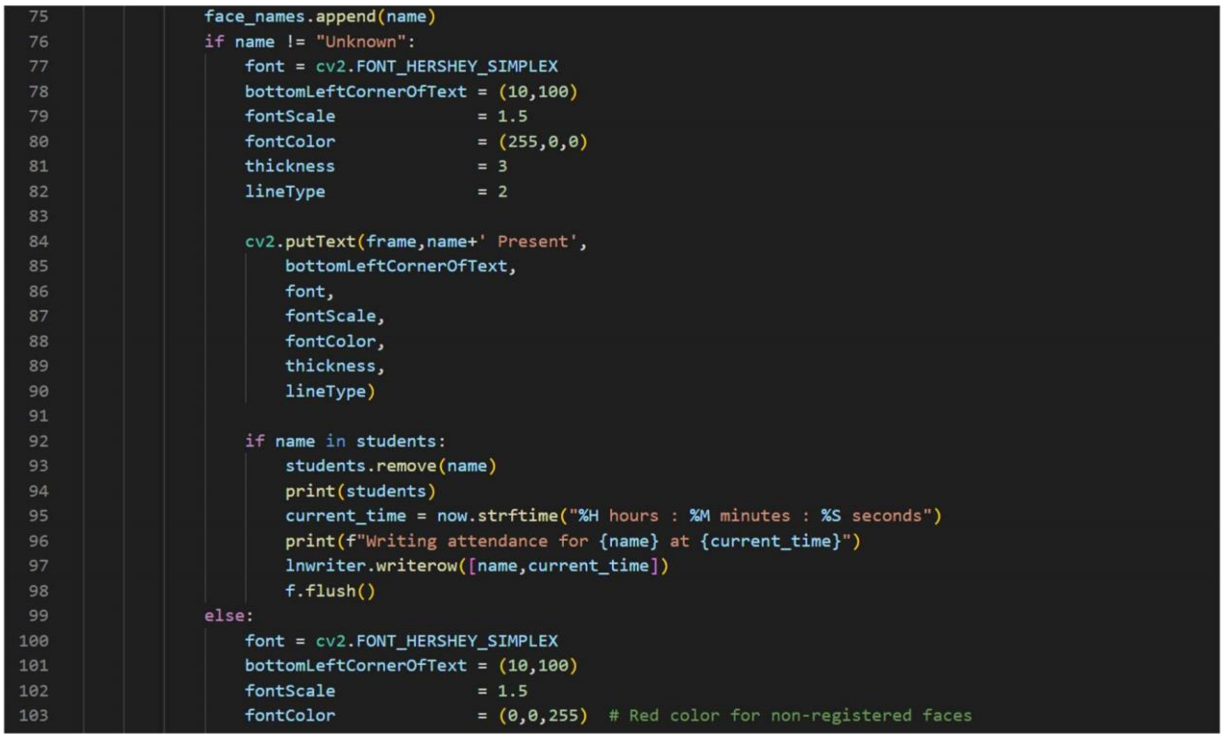
```python
import face_recognition
import cv2
import numpy as np
import csv
import os
from datetime import datetime

video_capture = cv2.VideoCapture(0)

jobs_image = face_recognition.load_image_file("photos/jobs.png")
jobs_encoding = face_recognition.face_encodings(jobs_image)[0]

aryan_image = face_recognition.load_image_file("photos/aryan.png")
aryan_encoding = face_recognition.face_encodings(aryan_image)[0]

shridhar_image = face_recognition.load_image_file("photos/shridhar.png")
shridhar_encoding = face_recognition.face_encodings(shridhar_image)[0]

tesla_image = face_recognition.load_image_file("photos/tesla.png")
tesla_encoding = face_recognition.face_encodings(tesla_image)[0]

Devansh_image = face_recognition.load_image_file("photos/Devansh.png")
Devansh_encoding = face_recognition.face_encodings(Devansh_image)[0]

known_face_encoding = [
jobs_encoding,
#ratan_tata_encoding,
shridhar_encoding,
tesla_encoding,
```



```python

known_face_encoding = [
jobs_encoding,
#ratan_tata_encoding,
shridhar_encoding,
tesla_encoding,
Devansh_encoding,
aryan_encoding,
]

known_faces_names = [
"JOBS",
#"RATAN TATA",
"Shridhar",
"TESLA",
"Devansh",
"Aryan",
]

students = known_faces_names.copy()

face_locations = []
face_encodings = []
face_names = []
s=True


now = datetime.now()
```

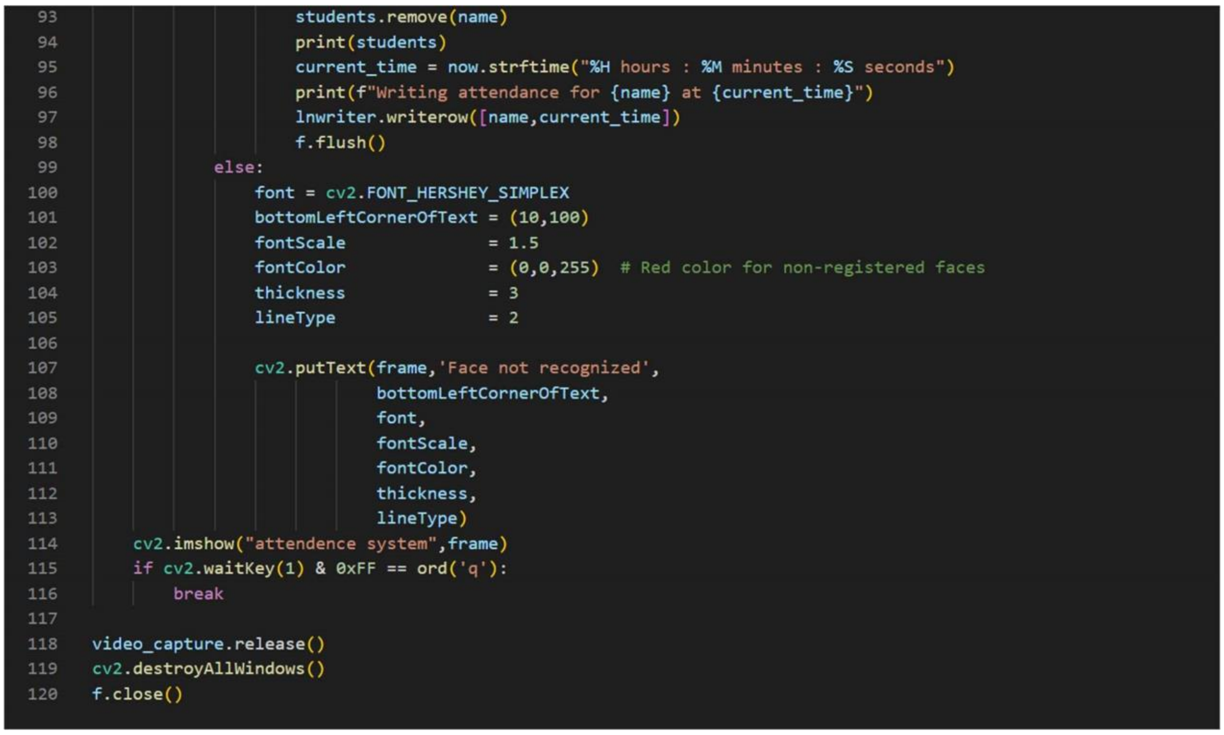


```python
now = datetime.now()
current_date = now.strftime("%Y-%m-%d")



f = open(current_date+'.csv','w+',newline = '')
lnwriter = csv.writer(f)

while True:
    _,frame = video_capture.read()
    small_frame = cv2.resize(frame,(0,0),fx=0.25,fy=0.25)
    rgb_small_frame = small_frame[:,:,::-1]
    if s:
        face_locations = face_recognition.face_locations(rgb_small_frame)
        face_encodings = face_recognition.face_encodings(rgb_small_frame,face_locations)
        face_names = []
        for face_encoding in face_encodings:
            matches = face_recognition.compare_faces(known_face_encoding,face_encoding, tolerance=0.5)
            name="Unknown"
            face_distance = face_recognition.face_distance(known_face_encoding,face_encoding)
            best_match_index = np.argmin(face_distance)
            if matches[best_match_index]:
                name = known_faces_names[best_match_index]

            face_names.append(name)
            if name != "Unknown":
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
```

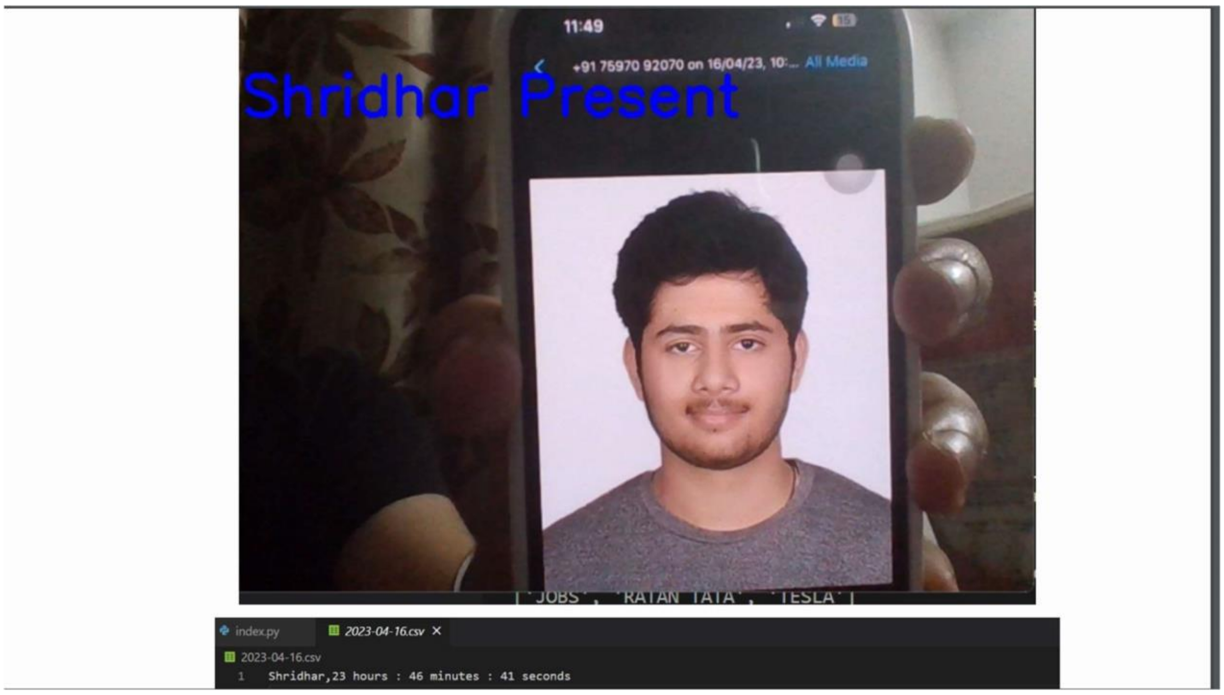


```python
            face_names.append(name)
            if name != "Unknown":
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
```

```python
                fontScale              = 1.5
                fontColor              = (255,0,0)
                thickness              = 3
                lineType               = 2

                cv2.putText(frame,name+' Present',
                    bottomLeftCornerOfText,
                    font,
                    fontScale,
                    fontColor,
                    thickness,
                    lineType)

                if name in students:
                    students.remove(name)
                    print(students)
                    current_time = now.strftime("%H hours : %M minutes : %S seconds")
                    print(f"Writing attendance for {name} at {current_time}")
                    lnwriter.writerow([name,current_time])
                    f.flush()
            else:
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
                fontScale              = 1.5
                fontColor              = (0,0,255)  # Red color for non-registered faces
```



```python
                    students.remove(name)
                    print(students)
                    current_time = now.strftime("%H hours : %M minutes : %S seconds")
                    print(f"Writing attendance for {name} at {current_time}")
                    lnwriter.writerow([name,current_time])
                    f.flush()
            else:
                font = cv2.FONT_HERSHEY_SIMPLEX
                bottomLeftCornerOfText = (10,100)
                fontScale              = 1.5
                fontColor              = (0,0,255)  # Red color for non-registered faces
                thickness              = 3
                lineType               = 2

                cv2.putText(frame,'Face not recognized',
                            bottomLeftCornerOfText,
                            font,
                            fontScale,
                            fontColor,
                            thickness,
                            lineType)
    cv2.imshow("attendence system",frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

video_capture.release()
cv2.destroyAllWindows()
f.close()
```



```
'JOBS', 'RATAN TATA', 'TESLA'
```

```
2023-04-16.csv
1   Shridhar,23 hours : 46 minutes : 41 seconds
```

CHAPTER 6
CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, this project aimed to develop an attendance portal using face recognition, HOG algorithm, SVM, and deep learning. We presented the methodology, which included data collection, preprocessing, feature extraction, model training, and evaluation. The results of our evaluation showed that our system achieved high accuracy in recognizing faces and marking attendance.

However, there is still room for improvement and future enhancements in this project. One potential enhancement is to improve the face detection and tracking mechanism to handle occlusions and pose variations. Another future direction is to develop a more robust and efficient model architecture that can handle a large number of users and handle real-time attendance tracking. Moreover, we can explore the use of more advanced deep learning techniques, such as convolutional neural networks (CNNs) and transfer learning, to further improve the performance of our system.

In addition, we can consider incorporating other modalities, such as voice recognition and fingerprint recognition, to provide multiple options for attendance marking. Another potential future enhancement is to develop a more secure system to protect the privacy of users' biometric data and prevent unauthorized access.

Overall, the development of an attendance portal using face recognition, HOG algorithm, SVM, and deep learning has significant potential in various domains, including education, healthcare, and corporate environments. The proposed enhancements can further improve the system's performance, scalability, and security and make it more useful and applicable in real-world scenarios.

GITHUB LINK

The project has been uploaded on GitHub. The link is given below:

GitHub Link: https://github.com/Devansh682/Attendance-Portal-using-ArtificialIntelligence

REFERENCES

- Navneet Dalal and Bill Triggs, “Histograms of Oriented Gradients for Human Detection”, Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 886-893, June 2005.

- K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for LargeScale Image Recognition,” 2015.

- R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580-587, June 2014.

- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, “Going Deeper with Convolutions,” 2014.

- A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” Advances in Neural Information Processing Systems, vol. 25, pp. 1097-1105, 2012.

- Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-Based Learning Applied to Document Recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.