**MaskTrack: Real-Time Detection & Compliance Management**

**Subhashis Patbandha**

**Registration No.: 162/19**



**5 Year Integrated MCA**

**PG Department of Computer Science & Applications**

**Utkal University**

**2024**

**PROJECT REPORT**

**ON**

**MaskTrack: Real-Time Detection & Compliance Management**

**SUBMITTED IN PARTIAL FULFILLMENT FOR AWARD OF**

**DEGREE**

**IN**

**Integrated MCA**

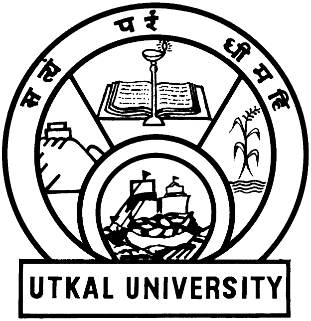
**(BATCH 2019-2024)**

**BY**

**Subhashis Patbandha – 19IMCA053**

**UNDER THE ESTEEMED GUIDANCE OF**

**Mrs. Pranati Satapathy**



**5 Year Integrated MCA**

**PG Department of Computer Science & Applications**

**Utkal University**

**2024**

**Declaration**

I hereby declare that the Project entitled “**MaskTrack: Real-Time Detection & Compliance Management”** submitted to the **PG Department of Computer Science and Applications**, **Utkal University, Bhubaneswar, Odisha** in partial fulfillment for the award of the degree of **Integrated MCA** in session 2019-2024 is an authentic record of my own work carried out under the guidance of **“Mrs. Pranati Satapathy”** and that the Project has not previously formed the basis for the award of any other degree. The report has been prepared in compliance to the guidelines specified by the University.

Place: Bhubaneswar

Date:

Subhashis Patbandha

[Reg. No.: 133/19]

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

GUIDE

Mrs. Pranati Satapathy

Lecturer

Dept. of Integrated MCA

**Acknowledgement**

I am very grateful to my project guide **Mrs. Pranati Satapathy** for giving her valuable time and constructive guidance in preparing the Synopsis/Project. It would not have been possible to complete this project in short period of time without his/her kind encouragement and valuable guidance.

Subhashis Patbandha

**CERTIFICATE**

This is to certify that this project entitled “MaskTrack Real-Time Detection & Compliance Management” submitted in partial fulfilment of the degree of Integrated MCA to the PG Department of Computer Science and Applications, Utkal University, Bhubaneswar, Odisha, done by Mr. Subhashis Patbandha, Registration No. 133/19 is an authentic work carried out by her at Integrated MCA, Utkal University under our guidance and is worthy of acceptance for award of the degree. The work fulfils the entire requirement as per the regulation of the University and in our opinion, it has reached the standard needed for submission.

**Mrs. Pranati Satapathy EXAMINER**

**GUIDE**

Lecturer

Dept. of Integrated MCA

**Course Coordinator**

5-year Integrated MCA, P.G. Dept of Computer Science and Application

Utkal University

**CONTRIBUTION OF INDIVIDUAL TEAM MEMBERS**

This project is a testament to the dedication and combined skillsets of our team. We are:

* Samir Kumar Choudhury
* Subhashis Patbandha
* Tusharkanta Bihari

Throughout the project's development, we each contributed significantly, drawing upon our expertise in various areas.

* **Machine Learning and Deep Learning:** Subhashis’s focused on developing and training the mask detection model using libraries like TensorFlow or OpenCV.
* **Computer Vision and Facial Recognition:** Samir’s expertise likely played a crucial role in implementing techniques for face detection and matching captured images with the student database.
* **Software Development and System Integration:** Tusharkanta's skills were likely instrumental in developing the core functionalities of the system, including the user interface for on-screen notifications and potential integration with the student database.
* **Project Management and Communication:** Ensuring smooth project execution through effective communication, collaboration, and problem-solving was a shared responsibility among all team members.
* **Report Writing and Documentation:** Preparing a comprehensive report detailing the project's functionalities, technical aspects, and future scope was a collaborative effort undertaken by the entire team.

The project's success wouldn't have been possible without our collaborative approach. We regularly brainstormed ideas, tackled challenges together, and ensured all aspects of the system were well-developed.

**Subhashis Patbandha**

**ABSTRACT**

This project introduces a real-time mask detection and enforcement system designed to promote responsible mask usage within educational institutions. The system analyses a live video feed to identify individuals entering the premises. Facial recognition technology then plays a crucial role by matching these detected individuals with a pre-existing student database. To encourage responsible behaviour, the system provides immediate feedback on screen and through audio cues, informing individuals about their mask-wearing status.

Furthermore, the system facilitates enforcement by capturing images of individuals not wearing masks. These captured images are stored and compared daily with the student database using facial recognition. This process allows for the identification of repeat offenders. Students exceeding a predefined threshold of mask violations within a designated period receive automated email notifications outlining potential consequences for non-compliance, such as fines or exam restrictions. This project aims to increase student awareness regarding the importance of mask-wearing and enforce mask-wearing regulations within the college environment, ultimately contributing to a safer learning environment.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NAME** | **FIGURE NO.** | **PAGE NO.** |
| FLOW DIAGRAM | 3.2.1 | 10 |
| USE-CASE DIAGRAM | 3.2.2 | 12 |
| ACTIVITY FLOW DIAGRAM | 3.2.3 | 13 |
| DATA FLOW DIAGRAM | 3.2.4 | 14 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NAME** | **TABLE NO.** | **PAGE NO.** |
| LIST OF TEST CASES | 4.8 | 27 |

**Table of Contents**

|  |  |  |
| --- | --- | --- |
| **SL NO.** | **TOPICS** | **PAGE NO.** |
|  | Cover Page as per format | i |
|  | Declaration of the Student | ii |
|  | Acknowledgement Certificate | iii |
|  | Certificate | iv |
|  | Contribution of Individual Team members | v |
|  | Abstract | vi |
|  | List of Figures | vii |
|  | List of Tables | viii |
|  |  |  |
| **1.** | **INTRODUCTION** | **1** |
|  | * 1. Problem Definition and Objective(s)   2. Motivations(s)   3. Project Overview / Specifications   4. Hardware Specification   5. Software Specification   6. Organization of the Report | 1  2  3  5  5  5 |
|  |  |  |
| **2.** | **LITERATURE SURVEY** | **6** |
|  | 2.1 Existing System  2.2 Proposed System  2.3 Feasibility Study | 6  6  7 |
|  |  |  |
| **3.** | **SYSTEM ANALYSIS & DESIGN** | **8** |
|  | 3.1 Requirement Specification  3.1.1 Functional Requirements  3.1.2 Non-functional Requirements  3.2 Flowcharts / Use Case / Activity Flow /DFD Diagrams  3.3 Design Steps / Criteria  3.4 Algorithms and Pseudo Code | 8  8  9  12  16  18 |
|  |  |  |
| **4.** | **TESTING PROCESS** | **25** |
|  | 4.1 Unit Testing  4.2 Integration Testing | 25  26 |
|  |  |  |
| **5.** | **RESULTS AND DISCUSSIONS** | **28** |
|  | 5.1 Results and Outputs | 31 |
|  |  |  |
| **6.** | **CONCLUSIONS AND FUTURE SCOPE** | **31** |
|  | 6.1 Conclusion  6.2 Future Scope | 31  32 |
|  |  |  |
|  | **REFERENCES** | **43** |
|  |  |  |

# INTRODUCTION

## Problem Definition and Objective

**Global Challenge**: The COVID-19 pandemic has significantly impacted public health and safety. Wearing masks in public spaces has proven to be an effective mitigation strategy to reduce transmission. However, ensuring consistent mask adherence remains a challenge, especially in high-traffic areas like educational institutions.

**Specific Problem:**

* Difficulty in achieving consistent mask usage compliance within educational settings.
* Lack of immediate feedback and awareness regarding mask-wearing status for individuals entering these areas.
* Manual and time-consuming processes for identifying repeat mask offenders and enforcing mask-wearing regulations.

**Consequences:**

* Increased risk of viral transmission within schools and colleges.
* Potential outbreaks and disruption of educational activities.
* Difficulty in maintaining a safe learning environment.

**Objective:**

This project proposes a real-time mask detection and enforcement system utilizing the following functionalities:

* **Live Video Feed Analysis:** Analyze a live video feed to detect individuals entering the premises.
* **Mask Detection:** Employ facial recognition technology to identify whether an individual is wearing a mask or not.
* **Immediate Feedback:** Provide real-time audio and visual cues on screen, informing individuals about their mask-wearing status.
* **Data Collection for Repeat Offenders**: Capture images of individuals not wearing masks for further processing.
* **Automated Daily Matching:** Compare captured images with a pre-existing student database using facial recognition.
* **Counter Increment for Violations:** Maintain a record of mask violations for each student identified without a mask.
* **Automated Enforcement Action:** Trigger automated email notifications for students exceeding a predefined threshold of mask violations within a designated period (e.g., month). These emails would communicate potential consequences of non-compliance, such as fines or exam restrictions.

**Expected Outcomes:**

* Enhance student awareness regarding the importance of mask-wearing.
* Encourage responsible mask usage behaviour within the college environment.
* Reduce the risk of viral transmission and potential outbreaks.
* Facilitate efficient enforcement of mask-wearing regulations through automation.

**Benefits:**

* Creates a safer learning environment for students and faculty.
* Contributes to public health protection within the college community.
* Offers a scalable and automated solution for mask-wearing enforcement.

This problem definition clearly outlines the existing challenge, its specific impact within the educational setting, and how your project proposes a solution with its functionalities, expected outcomes, and potential benefits. It focuses on the importance of mask usage for public health and student safety.

## Motivation

The COVID-19 pandemic has posed an unprecedented global health crisis. Educational institutions play a crucial role in maintaining a safe learning environment for students and faculty. Wearing masks in public spaces has been identified by health authorities as a vital measure in controlling the spread of the virus.

Our motivation for developing this real-time mask detection and enforcement system stems from the following key factors:

* **Importance of Mask-Wearing in Educational Settings:** Schools and colleges are high-traffic environments where maintaining social distancing can be challenging. Consistent mask usage becomes even more critical to minimize the risk of transmission within these settings.
* **Challenges in Achieving Compliance:** Ensuring consistent adherence to mask-wearing regulations can be difficult, especially with large student populations. Traditional methods rely on manual observation, which can be time-consuming and prone to human error.
* **Need for Immediate Feedback and Awareness:** Providing real-time feedback regarding mask-wearing status to individuals entering educational premises can significantly improve compliance. This immediate awareness serves as a gentle reminder and reinforces responsible behaviour.
* **Enhancing Enforcement Efficiency:** Manual identification and tracking of repeated mask offenders are inefficient and resource-intensive. Automating this process through facial recognition and data analysis streamlines enforcement efforts.

This project is driven by the desire to leverage technology for a positive public health impact. We believe this real-time mask detection and enforcement system can play a significant role in promoting responsible mask usage within educational institutions, fostering a safer learning environment for everyone.

## Project Overview and Specifications

The MaskTrack is a comprehensive system designed to enforce mask-wearing regulations in public environments, particularly educational institutions like colleges. In the wake of the COVID-19 pandemic, the importance of wearing masks in social settings has been underscored, prompting the development of technologies to ensure compliance. This project integrates machine learning, computer vision, and email communication to monitor, identify, and enforce mask-wearing rules within the college premises.

**Key Features:**

1. Real-time Mask Detection:

- Utilizes a machine learning model to detect whether individuals within the college premises are wearing masks or not.

- Provides immediate feedback through on-screen text and audio alerts to notify individuals about their mask-wearing status.

2. Mask Violation Tracking:

- Captures images of individuals who violate the mask-wearing rule using video feed.

- Stores images for further analysis and enforcement purposes.

3. Violation Counting and Enforcement:

- Matches captured images with student profiles in the database to identify violators.

- Increments the violation count for each identified individual.

- If a student violates the mask-wearing rule more than a predefined threshold (e.g., 10 times in a month), they receive an automatic email notification indicating a fine payment requirement or exam participation restrictions.

4. Face Recognition for Identification:

- Utilizes facial recognition to match violators with their student profiles.

- Compares facial embeddings extracted from captured images with those stored in the student database.

- Enhances identification accuracy and enables personalized enforcement actions.

**Components:**

1. Mask Detection Module:

- Utilizes a MobileNetV2-based model to detect masks in real-time video feed.

- Provides immediate feedback to individuals through on-screen text and audio alerts.

2. Violation Tracking Module:

- Captures images of individuals without masks and stores them for further analysis.

- Utilizes computer vision techniques to identify violators and track violation counts.

3. Face Recognition Module:

- Utilizes the MTCNN and InceptionResnetV1 models for face detection and recognition.

- Extracts facial embeddings from captured images and compares them with stored embeddings for identification.

4. Email Notification Module:

- Sends automatic email notifications to violators who exceed the specified violation threshold.

- Communicates fine payment requirements or exam participation restrictions.

**Specifications:**

- Technology Stack:

* + Python for backend development.
  + TensorFlow and Keras for machine learning and deep learning tasks.
  + OpenCV for computer vision and image processing.
  + SMTP protocol for email communication.

- Hardware Requirements:

* + Cameras for video feed capture.
  + Computer system for running the monitoring and enforcement software.

- Integration and Deployment:

* + Integration with existing surveillance systems or installation of dedicated cameras at college entrances.
  + Deployment on a local server or cloud infrastructure for real-time monitoring and enforcement.

- User Interface:

* + Minimalistic user interface for system administrators to monitor violation statistics, manage student profiles, and configure enforcement rules.
  + On-screen alerts for individuals to receive immediate feedback on their mask-wearing status.

- Security and Privacy:

* + Secure storage and encryption of student profiles and captured images.
  + Compliance with data protection regulations to ensure user privacy and confidentiality.

- Scalability and Maintenance:

* + Scalable architecture to accommodate varying numbers of users and surveillance cameras.
  + Regular maintenance and updates to address performance improvements, security patches, and evolving compliance requirements.

## Hardware Specification

The section of hardware plays a very important role in the existence and working of any system. This system has the following hardware specifications:

* A high-performance GPU and CPU.
* Embedded Systems
* Core Processor i5 or higher
* 8 GB RAM
* 512 GB SSD

## Software Specification

* OPERATING SYSTEM: Windows 10/11
* LANGUAGE: Python 3.7
* TOOLS: VSCode

LIBRARIES: Pandas, NumPy, Scikit-Learn, TensorFlow, Matplotlib, Keras, SciPy, Imutils, OpenCV-Python, matplotlib, ARGPARSE, PILLOW, STREAMLIT, ONNX, TF2ONNX, cosine\_similarity, facenet\_pytorch, MTCNN, torch, PIL

## Organization of the Report

The report consists of 6 chapters:

Chapter 1 introduce our project, explained what motivated us to build such a system, its objectives, overview and software and hardware specifications.

Then in chapter 2, we talk about analysing the existing system, what are some modifications which can improve the existing system are implemented in our proposed system and what are the feasibility study performed to build this project.

Coming to chapter 3, here are specified requirements (functional and non-functional) of this project, the designed step which is followed, diagrams explaining different aspects of our project, algorithms used are written.

Chapter 4 holds the testing process of the system.

Chapter 5, explains the interface design which has been made followed by the results.

Chapter 6, gives a conclusion to this project and put light on next step with stating some future scopes. And the report ends with the references from which we gathered the information and other resources for the making of this project.

# LITERATURE SURVEY

## Existing System

Several recent studies and projects have focused on developing mask detection systems using computer vision and deep learning techniques. Here's an overview of some key approaches:

* **Deep Learning-based Mask Detection:** Convolutional Neural Networks (CNNs) are a popular approach for mask detection. Researchers have developed various CNN architectures trained on large datasets of masked and unmasked faces. These systems achieve high accuracy in controlled environments but might struggle with variations in lighting, pose, and mask types.
* **Rule-based and Pre-trained Models:** Some systems utilize simpler rule-based approaches or leverage pre-trained models for facial landmark detection. These methods can be less computationally expensive but might have lower accuracy compared to deep learning models.

**Limitations of Existing Systems:**

* **Focus on Accuracy:** Many existing systems prioritize high detection accuracy, potentially neglecting factors like real-time performance or ease of deployment.
* **Limited Scope**: Some systems might not consider integration with existing infrastructure or functionalities like facial recognition for enforcement purposes.
* **Privacy Concerns:** Data collection and storage practices in existing systems raise privacy concerns that need to be addressed.

## Proposed System

Our proposed real-time mask detection and enforcement system addresses some limitations of existing systems while incorporating valuable functionalities:

* **Real-time Performance:** The system prioritizes real-time processing of video feeds for immediate feedback and enforcement actions.
* **Integration with Facial Recognition:** The system leverages facial recognition technology to identify repeat offenders and personalize enforcement actions.
* **Data Privacy Measures:** The system incorporates anonymization techniques or user consent mechanisms to address data privacy concerns.
* **Enforcement Mechanism:** The system implements an automated email notification system with potential consequences for non-compliance, promoting responsible behaviour.

Your proposed system aims to not only detect mask usage but also create a comprehensive solution for promoting awareness and enforcing regulations within an educational institution.

## Feasibility Study

The feasibility of the project is analysed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are,

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

### Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

### Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# SYSTEM ANALYSIS & DESIGN

## Requirement Specification

The requirement specification defines the detailed functionalities and performance expectations of the mask detection and enforcement system. It outlines both functional and non-functional requirements essential for the system's design, implementation, and operation.

**3.1.1. Functional Requirements**

Functional requirements specify the system's expected behaviour and interactions with users and external systems.

* **Mask Detection and Notification**
  + The system must be capable of real-time detection of individuals wearing masks through live video feeds.
  + Upon detecting a violation, the system must promptly display an on-screen notification and emit an audible alert to notify the individual.
* **Image Capture and Storage**
  + The system should capture images of individuals detected without masks for further processing and record-keeping.
  + Captured images must be time-stamped and associated with relevant metadata, including location and camera ID.
* **Matching with Student Database**
  + The system shall compare captured images with images stored in the student database to identify individuals violating mask-wearing rules.
  + Matching should be performed using facial recognition techniques with predefined similarity thresholds.
* **Rule Violation Tracking**
  + The system must maintain a count of instances where individuals are detected without masks.
  + Violation counts should be recorded for each individual and updated in real-time to track compliance.
* **Automatic Email Notification**
  + The system must automatically send email notifications to individuals exceeding the predefined limit of mask rule violations.
  + Email notifications should contain details of the violation and instructions for rectification.

**3.1.2. Non-Functional Requirements**

Non-functional requirements specify quality attributes such as performance, security, reliability, and usability.

* Performance
  + The system should detect mask violations and issue notifications in real-time with a latency of less than 1 second to ensure timely response.

* Accuracy
  + The mask detection algorithm must achieve a minimum accuracy rate of 95% in identifying mask-wearing individuals to minimize false positives and negatives.

* Security
  + Access to sensitive data, including captured images and student information, must be encrypted and restricted to authorized personnel only to ensure data confidentiality and integrity.

* Scalability
  + The system must be designed to handle up to 100 concurrent video feeds and accommodate a student population of 10,000 without compromising performance, allowing for scalability as the user base grows.

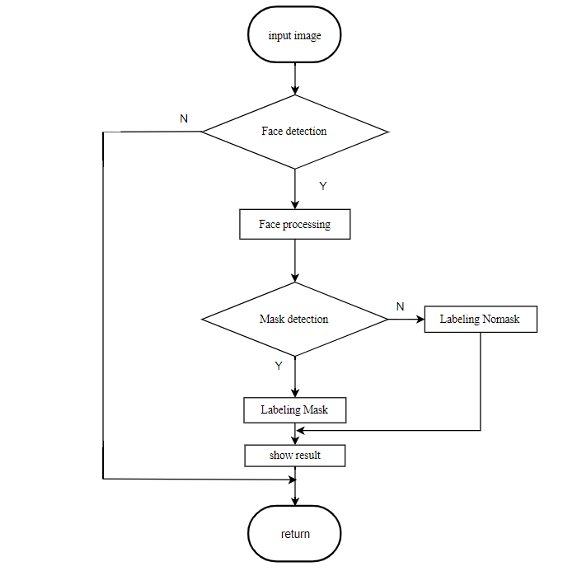
* Reliability
  + The system should operate continuously with a mean time between failures (MTBF) of at least 30 days, ensuring minimal downtime and uninterrupted service.

* Usability
  + The user interface must be intuitive and accessible, requiring minimal training for users to operate effectively, thereby enhancing user adoption and satisfaction.

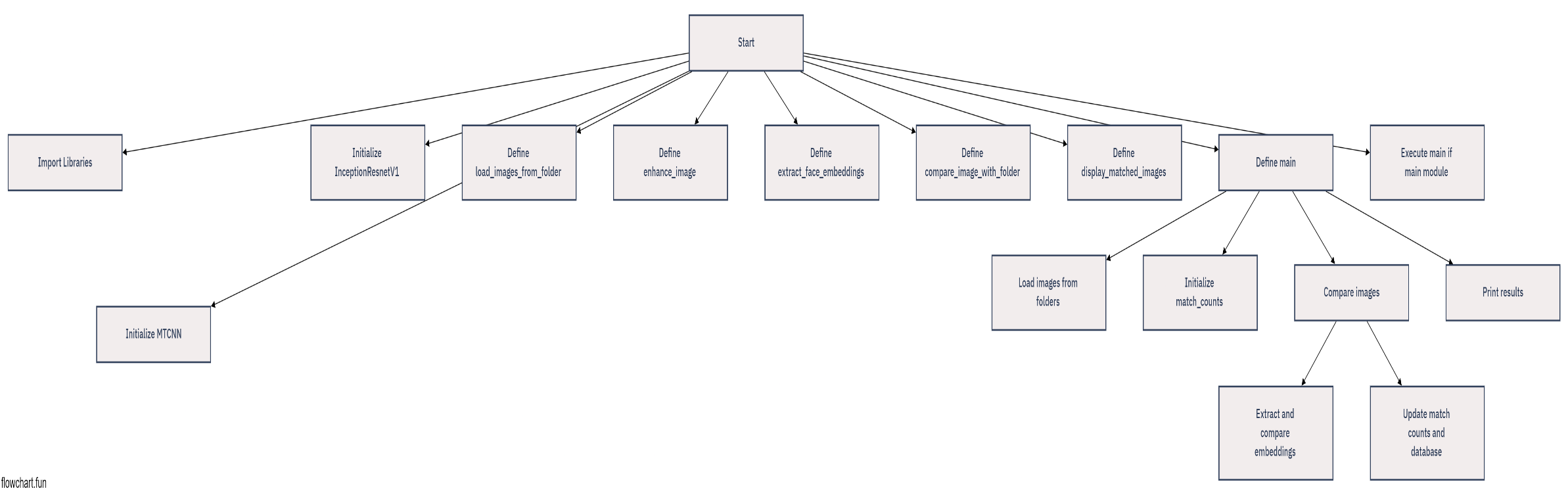
* Maintainability
  + The system should be designed with modular components and well-documented code to facilitate easy maintenance and future updates, ensuring long-term sustainability and supportability.

## 3.2 Diagram

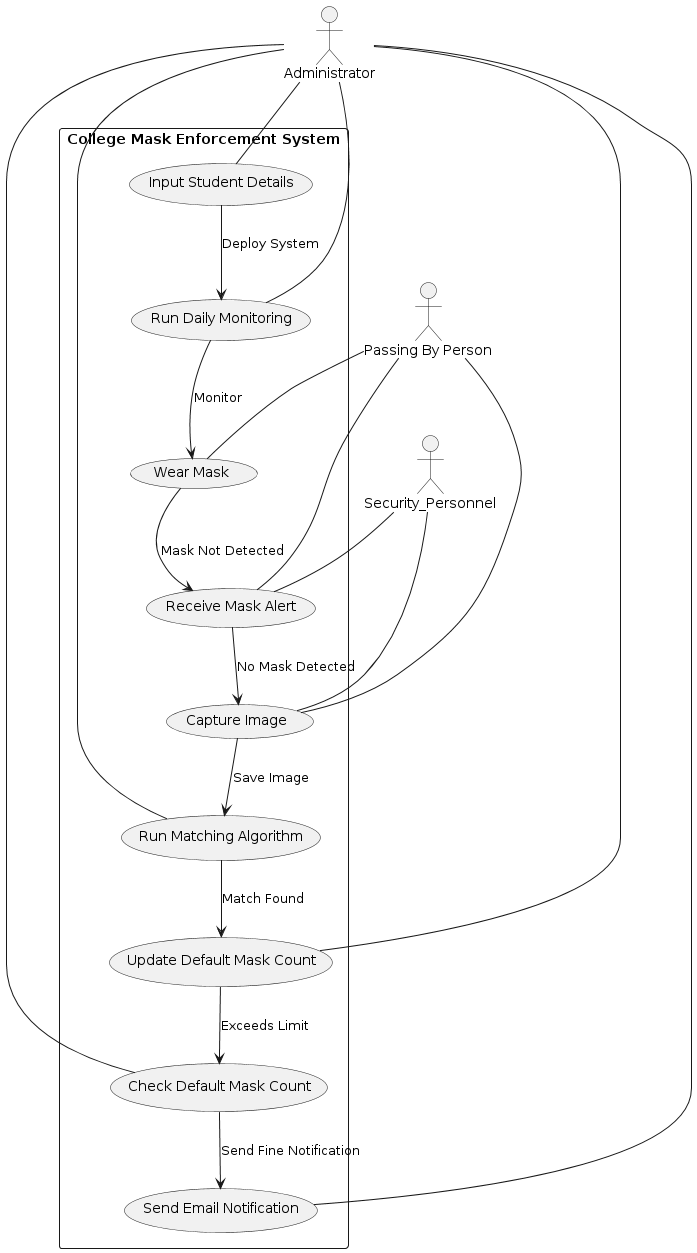
### Flow Diagram

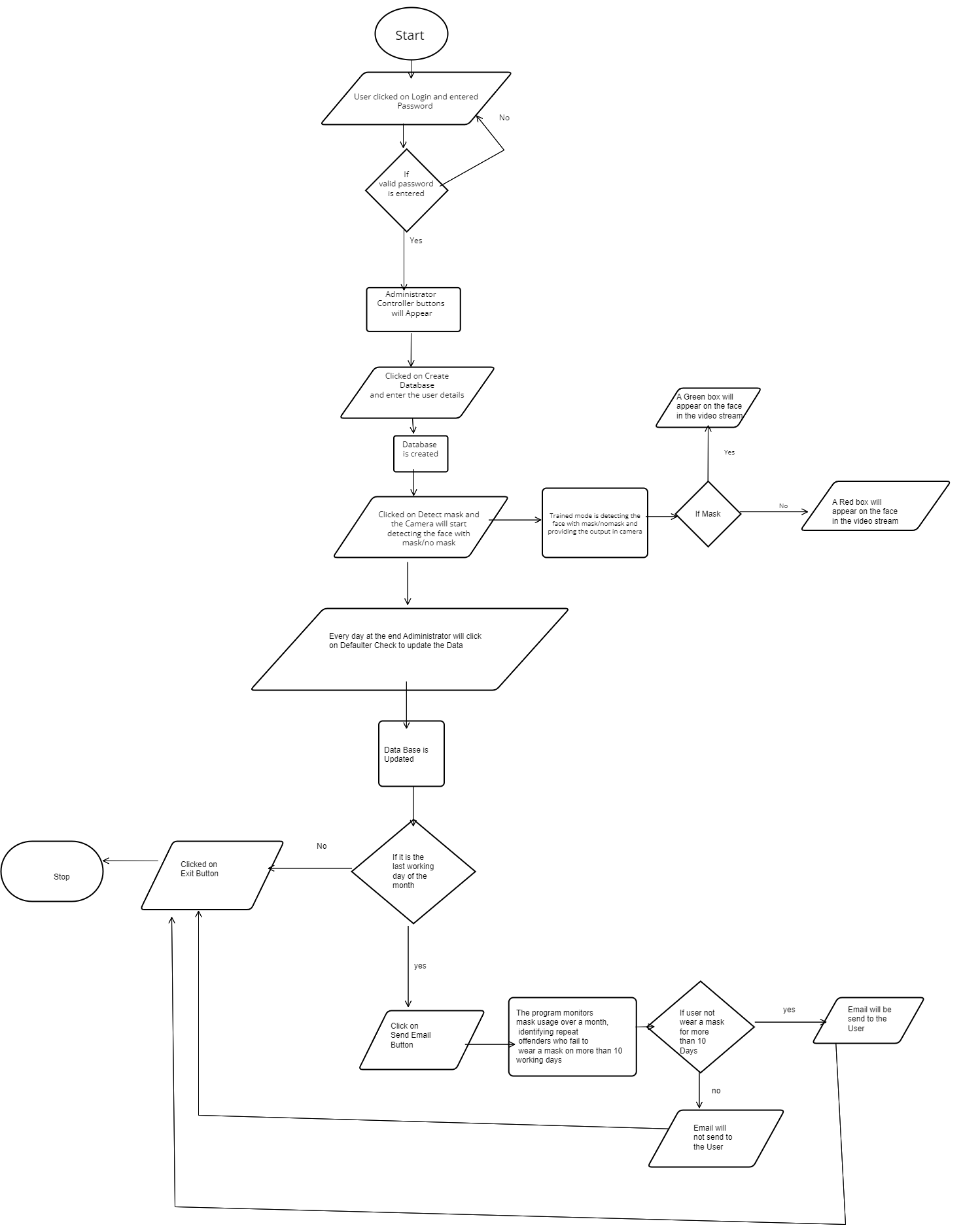


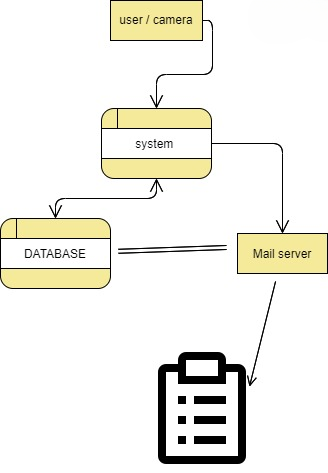
Mask Detection and Output



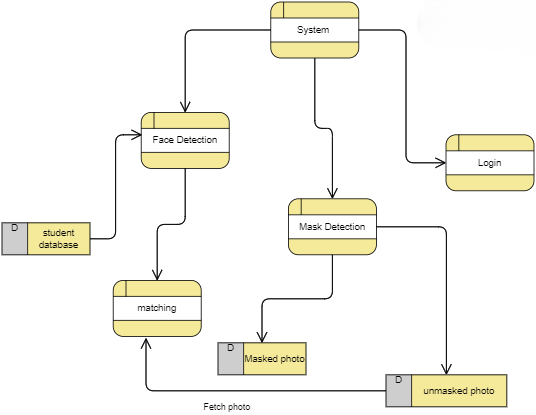
Matching of Images in Student Images and No Mask folder and update of database

* + 1. **Use Case Diagram**

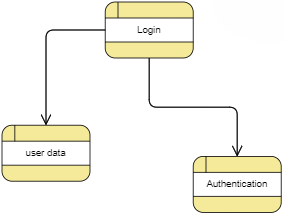
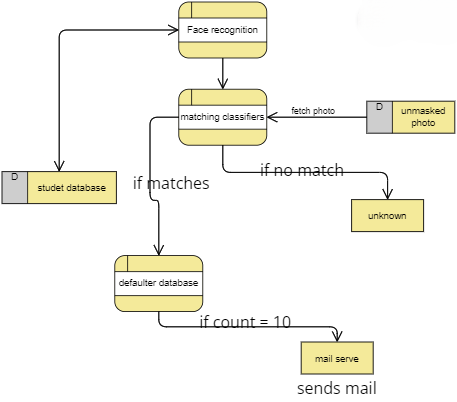
****

* + 1. **Activity Flow Diagram**
    2. **Data Flow Diagram**

**Level 0**

****

**Level 1**

****

**Level 2**

**3.3 Design Steps**

* Step 1: Download and Visualize the Dataset of Masked and Unmasked images with its Labels
* Step 2: Preprocess the Dataset
* Step 3: Split the Data into Train and Test Set
* Step 4.1: Utilize a pre-trained MobileNetV2 model for feature extraction, then add a custom head for classification. Transfer learning fine-tunes the model for mask detection.
* Step 4.2: Train for 20 epochs with a batch size of 32. Initialize learning rate to 0.0001 and decay over epochs.
* Step 4.3: Save the trained model & Training loss, validation loss, training accuracy, and validation accuracy are plotted against the number of epochs to visualize the training process.
* Step 5.1: Load Models: Load the pre-trained face detection model (SSD) and mask detection model (MobileNetV2) from disk.
* Step 5.2: Capture Video: Start capturing video frames from the webcam or another video source.
* Step 5.3: Process Each Frame: For each frame captured: Detect faces in the frame using the face detection model (SSD),

For each detected face: Extract the face region, Preprocess the face region, Pass the pre-processed face region through the mask detection model (MobileNetV2), Determine whether the face is wearing a mask or not based on the prediction, Draw bounding boxes and labels on the frame indicating the detection result (mask or no mask).

* Step 6: Face Detection and Alignment: Utilized MTCNN (Multi-task Cascaded Convolutional Networks) for face detection and alignment. MTCNN is known for its accuracy and efficiency in detecting faces in images.
* Step 7: Face Recognition Model Selection: Chose InceptionResnetV1 for face recognition. This model is widely used for face recognition tasks due to its strong performance in generating discriminative embeddings for faces.
* Step 8.1: Image Loading: Loaded images from specified folders (student\_images and default\_no\_mask\_images). This step ensures that the system has access to the necessary images for comparison.
* Step 8.2: Image Enhancement: Enhanced image brightness before processing to improve the quality of input images. This preprocessing step helps in achieving better face detection and recognition results, especially in varying lighting conditions.
* Step 8.3: Face Embeddings Extraction: Extracted face embeddings from images using the InceptionResnetV1 model. Face embeddings represent unique numerical vectors that capture essential features of a face, enabling direct comparison between faces.
* Step 8.4: Cosine Similarity Calculation: Calculated cosine similarity between embeddings to compare faces. Cosine similarity is a metric commonly used in face recognition systems to quantify the similarity between two face embeddings.
* Step 8.5: Match Counting: Implemented logic to count the number of matches found for each student ID. This step involves iterating through the embeddings of student images and comparing them with embeddings of default no-mask images.
* Step 8.6: Database Update: Updated a JSON student database with the number of matches found for each student ID. This database serves as a record of students who have been detected without masks and allows for tracking their match counts over time.

## Algorithms and Pseudo Code

**Detecting Mask or No Mask from Live Video Feed**

1. Import necessary libraries

2. Initialize pygame mixer for sound playback

3. Define a function to detect and predict masks on faces in a given frame

4. Parse command-line arguments for face detector model and mask detector model paths

5. Load the face detector model and mask detector model

6. Initialize the video stream from the webcam

7. Set up full-screen display window

8. Define directories to save images with and without masks

9. Set alert intervals for no mask and mask detected alerts

10. Initialize variables for last alert times

11. Loop over frames from the video stream

12. Read the frame from the video stream and resize it

13. Detect faces in the frame and predict if they are wearing masks or not

14. Loop over the detected faces

15. Extract face bounding box and predictions

16. Determine the label (mask or no mask) and colour for the bounding box

17. Display the label and bounding box on the frame

18. If no mask is detected, play a sound alert and save the frame as an image

19. If a mask is detected, play a sound alert and save the frame as an image

20. If no unmasked faces are detected, reset the timer for the no mask alert

21. If no masked faces are detected, reset the timer for the mask alert

22. Show the output frame

23. Wait for the 'q' key to be pressed to exit the loop

24. Cleanup resources (close windows, stop the video stream)

# USAGE

# python detect\_mask\_video.py

# import the necessary packages

from tensorflow.keras.applications.mobilenet\_v2 import preprocess\_input

from tensorflow.keras.preprocessing.image import img\_to\_array

from tensorflow.keras.models import load\_model

from imutils.video import VideoStream

import numpy as np

import argparse

import imutils

import time

import cv2

import os

import pygame

import re

pygame.mixer.init()

def detect\_and\_predict\_mask(frame, faceNet, maskNet):

# grab the dimensions of the frame and then construct a blob

# from it

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

blob = cv2.dnn.blobFromImage(frame, 1.0, (300, 300),

(104.0, 177.0, 123.0))

# pass the blob through the network and obtain the face detections

faceNet.setInput(blob)

detections = faceNet.forward()

# initialize our list of faces, their corresponding locations,

# and the list of predictions from our face mask network

faces = []

locs = []

preds = []

# loop over the detections

for i in range(0, detections.shape[2]):

# extract the confidence (i.e., probability) associated with

# the detection

confidence = detections[0, 0, i, 2]

# filter out weak detections by ensuring the confidence is

# greater than the minimum confidence

if confidence > args["confidence"]:

# compute the (x, y)-coordinates of the bounding box for

# the object

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

(startX, startY, endX, endY) = box.astype("int")

# ensure the bounding boxes fall within the dimensions of

# the frame

(startX, startY) = (max(0, startX), max(0, startY))

(endX, endY) = (min(w - 1, endX), min(h - 1, endY))

# extract the face ROI, convert it from BGR to RGB channel

# ordering, resize it to 224x224, and preprocess it

face = frame[startY:endY, startX:endX]

if face.any():

face = cv2.cvtColor(face, cv2.COLOR\_BGR2RGB)

face = cv2.resize(face, (224, 224))

face = img\_to\_array(face)

face = preprocess\_input(face)

# add the face and bounding boxes to their respective

# lists

faces.append(face)

locs.append((startX, startY, endX, endY))

# only make a predictions if at least one face was detected

if len(faces) > 0:

# for faster inference we'll make batch predictions on \*all\*

# faces at the same time rather than one-by-one predictions

# in the above `for` loop

faces = np.array(faces, dtype="float32")

preds = maskNet.predict(faces, batch\_size=1)

# return a 2-tuple of the face locations and their corresponding

# locations

return (locs, preds)

# construct the argument parser and parse the arguments

ap = argparse.ArgumentParser()

ap.add\_argument("-f", "--face", type=str,

default="face\_detector",

help="path to face detector model directory")

ap.add\_argument("-m", "--model", type=str,

default="mask\_detector.model",

help="path to trained face mask detector model")

ap.add\_argument("-c", "--confidence", type=float, default=0.5,

help="minimum probability to filter weak detections")

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

# load our serialized face detector model from disk

print("[INFO] loading face detector model...")

prototxtPath = os.path.sep.join([args["face"], "deploy.prototxt"])

weightsPath = os.path.sep.join([args["face"],

"res10\_300x300\_ssd\_iter\_140000.caffemodel"])

faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)

# load the face mask detector model from disk

print("[INFO] loading face mask detector model...")

maskNet = load\_model(args["model"])

frame\_num=0

# initialize the video stream and allow the camera sensor to warm up

print("[INFO] starting video stream...")

vs = VideoStream(src=1).start()

time.sleep(2.0)

#full screen

# Get the screen resolution

screen\_width, screen\_height = 1920, 1080 # Update with your screen resolution

# Create a full-screen display window

cv2.namedWindow("Frame", cv2.WND\_PROP\_FULLSCREEN)

cv2.setWindowProperty("Frame", cv2.WND\_PROP\_FULLSCREEN, cv2.WINDOW\_FULLSCREEN)

# Define the paths for the directories to save images

with\_mask\_dir = "./results/with\_mask/"

without\_mask\_dir = "./results/without\_mask/"

# Create the directories if they don't exist

os.makedirs(with\_mask\_dir, exist\_ok=True)

os.makedirs(without\_mask\_dir, exist\_ok=True)

last\_alert\_time = 0 # Store the last time the no mask alert was played

alert\_interval = 5 # Interval in seconds to wait before repeating the sound

last\_mask\_alert\_time = 0 # Store the last time the mask alert was played

mask\_alert\_interval = 5 # Interval in seconds to wait before repeating the sound

# loop over the frames from the video stream

while True:

# grab the frame from the threaded video stream and resize it

# to have a maximum width of 400 pixels

frame = vs.read()

frame = imutils.resize(frame, width=400)

# detect faces in the frame and determine if they are wearing a

# face mask or not

(locs, preds) = detect\_and\_predict\_mask(frame, faceNet, maskNet)

current\_time = time.time()

any\_face\_detected = False

any\_no\_mask\_detected = False

any\_mask\_detected = False

# loop over the detected face locations and their corresponding

# locations

for (box, pred) in zip(locs, preds):

# unpack the bounding box and predictions

(startX, startY, endX, endY) = box

(mask, withoutMask) = pred

# determine the class label and color we'll use to draw

# the bounding box and text

label = "Mask" if mask > withoutMask else "No Mask"

color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

# include the probability in the label

label = "{}: {:.2f}%".format(label, max(mask, withoutMask) \* 100)

# display the label and bounding box rectangle on the output

# frame

cv2.putText(frame, label, (startX, startY - 10),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, color, 2)

cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)

# save the frame as an image

# frame\_num += 1

frame = cv2.resize(frame, (screen\_width, screen\_height))

number = re.search(r'\d+\.\d+', label)

# print(label)

if label.find("No Mask") != -1:

any\_face\_detected = True

any\_no\_mask\_detected = True

if current\_time - last\_alert\_time >= alert\_interval:

nomask\_sound = pygame.mixer.Sound("audio/no\_mask.wav") # Path to your sound file

# Play the sound

nomask\_sound.play()

last\_alert\_time = current\_time # Update the last alert time

extracted\_number = float(number.group())

if extracted\_number>99:

frame\_num += 1

cv2.imwrite(os.path.join(without\_mask\_dir, "frame\_{}.jpg".format(frame\_num)), frame)

elif number and label.find("Mask") != -1:

extracted\_number = float(number.group())

if extracted\_number>99:

any\_face\_detected = True

any\_mask\_detected = True

if current\_time - last\_mask\_alert\_time >= mask\_alert\_interval:

# Play accepted sound if mask is detected

accepted\_sound=pygame.mixer.Sound("audio/accepted.wav")# Path to your sound file

# Play the sound

accepted\_sound.play()

last\_mask\_alert\_time = current\_time # Update the last mask alert time

cv2.imwrite(os.path.join(with\_mask\_dir, "frame\_{}.jpg".format(frame\_num)), frame)

# If no unmasked faces are detected, reset the timer

if not any\_no\_mask\_detected:

last\_alert\_time = current\_time - alert\_interval # Allow immediate alert if no mask is detected next time

if not any\_mask\_detected:

last\_mask\_alert\_time = current\_time - mask\_alert\_interval # Allow immediate alert if mask is detected next time

# show the output frame

cv2.imshow("Frame", frame)

key = cv2.waitKey(3) & 0xFF

# if the `q` key was pressed, break from the loop

if key == ord("q" or "Q"):

break

# do a bit of cleanup

cv2.destroyAllWindows()

vs.stop()

**Matching Student Images from JSON file with No Mas Images to identifty Students and Updating JSON file**

1. Import necessary libraries

2. Initialize MTCNN for face detection and alignment

3. Initialize InceptionResnetV1 for face recognition

4. Define a function to load images from a folder

5. Define a function to enhance image brightness

6. Define a function to extract face embeddings from an image

7. Define a function to compare an image with images in a folder based on face embeddings

8. Define a function to display matched images side by side

9. Define a function to compare images in two folders and update a student database based on matches

10. In the main function:

11. Set paths for student images and default no mask images folders

12. Load images from both folders

13. Initialize an empty dictionary to store student IDs and their match counts

14. Compare each image in the default no mask images folder with each image in the student images folder

15. For each match, extract the student ID from the image name

16. Update the match count and the default mask value for the student ID in the student database

17. Print the updated default mask value for each student ID

18. Print the number of matched images for each student ID

19. Call the main function if the script is executed as the main program

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics.pairwise import cosine\_similarity

from facenet\_pytorch import MTCNN, InceptionResnetV1

import torch

from PIL import ImageEnhance

from PIL import Image

import json

# Initialize MTCNN for face detection and alignment

mtcnn = MTCNN(keep\_all=True)

# Initialize InceptionResnetV1 for face recognition

resnet = InceptionResnetV1(pretrained='vggface2').eval()

def load\_images\_from\_folder(folder\_path):

images = []

for filename in os.listdir(folder\_path):

img\_path = os.path.join(folder\_path, filename)

if os.path.isfile(img\_path):

img = cv2.imread(img\_path)

if img is not None:

images.append((filename, img))

return images

def enhance\_image(image):

# Convert image to PIL format

img\_pil = Image.fromarray(cv2.cvtColor(image, cv2.COLOR\_BGR2RGB))

# Enhance brightness

enhancer = ImageEnhance.Brightness(img\_pil)

img\_pil = enhancer.enhance(1.5)

# Convert back to OpenCV format

enhanced\_image = cv2.cvtColor(np.array(img\_pil), cv2.COLOR\_RGB2BGR)

return enhanced\_image

def extract\_face\_embeddings(img):

# Enhance image to improve performance in low-light conditions

img = enhance\_image(img)

# Convert image to RGB format (MTCNN requires RGB images)

img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)

# Detect faces in the image

faces = mtcnn(img\_rgb)

# Check if faces were detected

if faces is None:

return []

# Extract embeddings for each detected face

embeddings = []

for face in faces:

if face is not None:

# Resize face to the required input size of the face recognition model

resized\_face = cv2.resize(face.permute(1, 2, 0).numpy(), (96, 96))

# Convert resized face to tensor

resized\_face\_tensor = torch.tensor(resized\_face.transpose(2, 0, 1), dtype=torch.float32)

# Calculate embeddings using InceptionResnetV1

embedding = resnet(resized\_face\_tensor.unsqueeze(0))

# Append the embedding to the list

embeddings.append(embedding.detach().numpy())

return embeddings

def compare\_image\_with\_folder(image\_path, folder\_path, threshold=0.6):

# Load input image

img = cv2.imread(image\_path)

# Extract face embeddings from input image

input\_embeddings = extract\_face\_embeddings(img)

# Load images from the folder

images = load\_images\_from\_folder(folder\_path)

max\_similarity = -1

matched\_image\_path = None

for filename, img in images:

# Extract face embeddings from folder image

folder\_embeddings = extract\_face\_embeddings(img)

# Compare input embeddings with folder embeddings

for input\_embedding in input\_embeddings:

for folder\_embedding in folder\_embeddings:

similarity = cosine\_similarity(input\_embedding, folder\_embedding)

if similarity >= threshold and similarity > max\_similarity:

max\_similarity = similarity

matched\_image\_path = os.path.join(folder\_path, filename)

if matched\_image\_path is not None:

print(f"Match found with highest similarity: {max\_similarity[0][0]}")

display\_matched\_images(image\_path, matched\_image\_path)

else:

print("No match found.")

def display\_matched\_images(image\_path1, image\_path2):

img1 = cv2.imread(image\_path1)

img2 = cv2.imread(image\_path2)

# Resize images to have the same height

height = min(img1.shape[0], img2.shape[0])

img1 = cv2.resize(img1, (int(img1.shape[1] \* height / img1.shape[0]), height))

img2 = cv2.resize(img2, (int(img2.shape[1] \* height / img2.shape[0]), height))

# Concatenate images horizontally

concatenated\_img = np.concatenate((img1, img2), axis=1)

# Display the concatenated image using Matplotlib

plt.imshow(cv2.cvtColor(concatenated\_img, cv2.COLOR\_BGR2RGB))

plt.title("Matched Images")

plt.axis('off')

plt.show()

def main(threshold=0.6):

# Set paths for the two image folders

student\_images\_path = r"D:\Web Development\Face\_mask\_detection-master\student\_images"

default\_no\_mask\_images\_path = r"D:\Web Development\Face\_mask\_detection-master\results\without\_mask"

# Load images from both folders

student\_images = load\_images\_from\_folder(student\_images\_path)

default\_no\_mask\_images = load\_images\_from\_folder(default\_no\_mask\_images\_path)

# Initialize an empty dictionary to store student IDs and their match counts

match\_counts = {}

# Compare each image in the default\_no\_mask\_images folder with each image in the student\_images folder

for default\_image\_name, default\_image in default\_no\_mask\_images:

for student\_image\_name, student\_image in student\_images:

# Extract embeddings for the default image

default\_embeddings = extract\_face\_embeddings(default\_image)

# Extract embeddings for the student image

student\_embeddings = extract\_face\_embeddings(student\_image)

# Compare embeddings

for default\_embedding in default\_embeddings:

for student\_embedding in student\_embeddings:

similarity = cosine\_similarity(default\_embedding, student\_embedding)

if similarity >= threshold:

# Extract student ID from the image name

student\_id = int(student\_image\_name.split("\_")[0])

# Update match count for this student ID

match\_counts[student\_id] = match\_counts.get(student\_id, 0) + 1

# Increment the default\_mask value for this student ID by 1

# (Ensure it only increases once)

if match\_counts[student\_id] == 1:

# Load student database

student\_database\_path = "student\_database.json"

with open(student\_database\_path, "r") as file:

student\_database = json.load(file)

# Update default\_mask value

student\_database[str(student\_id)]["default\_mask"] = student\_database.get(str(student\_id), {"default\_mask": 0})["default\_mask"] + 1

# Save updated student database

with open(student\_database\_path, "w") as file:

json.dump(student\_database, file, indent=4)

# Print message indicating database update

print(f"Default mask for Student ID {student\_id} updated to {student\_database[str(student\_id)]['default\_mask']}")

# Print matched images and match accuracy

for student\_id, match\_count in match\_counts.items():

print(f"Student ID {student\_id}: {match\_count} image(s) matched")

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

main()

**Sending Mail to the Repeat Offenders**

1. Import necessary libraries

2. Define a function to load the student database from a file

3. Define a function to check the default\_mask count and send an email if it exceeds 10

4. Define a function to send an email

5. Define the main function to check and send emails

6. Load the student database

7. Get the current month

8. Iterate over each student in the database

9. Check if the default\_mask count exceeds 10

10. If the count exceeds 10, construct an email message

11. Send the email

12. Print a success message if the email is sent successfully

13. If no emails are sent, print a message indicating no emails were sent

14. If there is an error sending the email, print an error message

15. Call the main function if the script is executed as the main program

import json

import smtplib

from email.mime.text import MIMEText

from datetime import datetime

# Function to load the student database from a file

def load\_database():

try:

with open("student\_database.json", "r") as file:

return json.load(file)

except FileNotFoundError:

return {}

except Exception as e:

print(f"An error occurred: {e}")

return {}

# Function to check default\_mask and send email if exceeds 10

def check\_and\_send\_email():

student\_database = load\_database()

current\_month = datetime.now().strftime("%B")

emails\_sent = False

for student\_id, student\_info in student\_database.items():

default\_mask\_count = student\_info.get("default\_mask", 0)

if default\_mask\_count > 10:

student\_name = student\_info.get("name", "Unknown")

student\_email = student\_info.get("email", "")

message = f"Dear {student\_name},\n\nYour default\_mask count has exceeded 10 for the month of {current\_month}. Please pay the fine of RS.579 to abcd@okaybank UPI ID if you would like to give the upcoming semester exam.\n\nRegards,\nUtkal University"

send\_email(student\_email, "Notification: Default Mask Exceeds Limit", message)

print(f"Email sent successfully to {student\_email} for {student\_name}")

emails\_sent = True

if not emails\_sent:

print("No emails were sent because all default\_mask counts are within the limit.")

# Function to send email

def send\_email(receiver\_email, subject, message):

try:

sender\_email = "samsubhtushmca@outlook.com" # Your Outlook email address

sender\_password = "Subhashis@2001" # Your Outlook email password

# Create a MIMEText object

msg = MIMEText(message)

msg["Subject"] = subject

msg["From"] = sender\_email

msg["To"] = receiver\_email

# Establish a connection with the SMTP server

server = smtplib.SMTP("smtp.office365.com", 587) # Outlook SMTP server and port

server.starttls()

# Login to the email account

server.login(sender\_email, sender\_password)

# Send the email

server.sendmail(sender\_email, receiver\_email, msg.as\_string())

# Close the connection

server.quit()

print(f"Email sent successfully to {receiver\_email}")

except Exception as e:

print(f"An error occurred while sending email to {receiver\_email}: {e}")

# Main function to check and send emails

def main():

check\_and\_send\_email()

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

main()

# TESTING PROCESS

## Data Splitting

Divide your labeled dataset of images with and without masks into training, validation, and test sets.

Training Set: Used to train the mask detection model.

Validation Set: Used to fine-tune hyperparameters and monitor training progress.

Test Set: Used to evaluate the final model's performance on unseen data.

## 4.1.1 Training

Train your mask detection model on the training dataset. Common approaches include deep learning architectures like convolutional neural networks (CNNs). Monitor metrics like accuracy and loss during training to ensure the model is learning effectively.

## 4.1.2 Validation

Evaluate the trained model on the validation set using metrics like accuracy, precision, recall. Based on the results, you might:

Fine-tune hyperparameters (e.g., learning rate, number of filters in CNN layers).

Adjust the model architecture (e.g., add or remove layers).

Perform data augmentation (e.g., introducing variations like brightness changes) to improve model robustness.

## Testing

Assess the final model's performance on the test set to evaluate its generalization ability. Here are key testing aspects:

### Unit Testing:

Test individual program components in isolation:

Image pre-processing (e.g., resizing, normalization).

Mask detection algorithm.

Student image comparison with facial recognition.

Verify each component works as expected and produces correct outputs.

### Integration Testing:

Test how different modules work together:

\* Data flow from camera capture to processing and storage.

\* Ensure mask detection results correctly trigger text/audio messages and image capture.

\* Verify student image comparison and `default\_mask` parameter update functionality.

### Functional Testing:

Verify the program functions correctly in various scenarios:

Test mask detection accuracy with different lighting conditions, face orientations, and mask types.

Ensure student image comparison works with variations in image quality and poses.

Verify the email reminder generation process based on the `default\_mask` parameter.

### Performance Testing:

Evaluate processing speed and resource utilization:

Measure the time taken to process video frames and detect masks in real-time.

Assess system performance on different hardware configurations.

### Robustness Testing:

Test the program's resilience to noise and variations:

Introduce simulated noise to video frames and evaluate detection accuracy.

Test with blurry or partially obscured faces to assess robustness.

## Cross-Validation

Split the dataset further into folds. Train the model on k-1 folds and validate on the remaining fold. Repeat this process k times, using a different fold for validation each time. Average the performance metrics across folds for a more robust estimate of model generalizability..

## Evalution Metrics

Accuracy: Overall percentage of correctly classified images (with/without masks).

Precision: Proportion of images identified with masks that actually have masks.

Recall: Proportion of images with masks that are correctly identified.

## Real-World Testing

Conduct real-world testing in the college environment:

Deploy the program with actual camera setup and student traffic.

Monitor performance and address any issues related to lighting, camera angles, or unexpected scenarios (e.g., students wearing glasses, covering their faces partially).

Gather data on mask detection accuracy and student compliance in a real-world setting.

## Edge Cases Testing

Real-World Test: Deploy the program in the college with real students and cameras. Monitor for issues like lighting, angles, and unexpected scenarios (glasses, masks). This ensures it works in the actual environment.

Performance Monitoring: Track detection accuracy and email reminders after deployment. Identify drops in accuracy or adjust reminder thresholds if needed. This keeps the program reliable over time

## Performance Monitoring

Continuously monitor the program's performance after deployment:

Track mask detection accuracy metrics over time to identify potential degradation.

Monitor email reminders sent and adjust thresholds for the `default\_mask` parameter if necessary.

Implement logging and alerting mechanisms to detect and address issues promptly (e.g., system crashes, low accuracy detection rates).

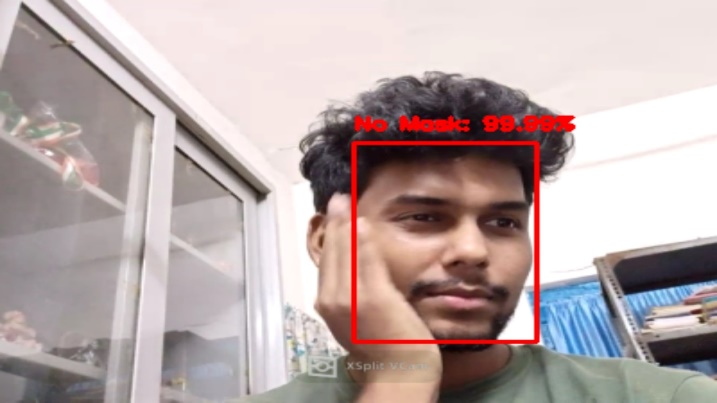
## List Of Test Cases

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.No**. | **Test Name** | **Input** | **Expected Output** | **Actual Output** | **Tast Case Result** |
| 1 | Loading the dataset | Dataset | Dataset loaded | Dataset read | PASS |
| 2 | Split the dataset | Training and testing dataset | Splitting of the data into training and testing | Dividing of the data into training and testing | PASS |
| 3 | Training the model | Training set, parameters | Trained with the provided set | Trained model | PASS |
| 4 | Validation of the model | No of entries from testing data | Validation of the model with best fit | Model generated | PASS |
| 5 | Predicting the accuracy | Accuracy matrices | Predicted accuracy | Accuracy predicted | PASS |
| 6 | Test data | Test column in data | Prediction for the given test case | Predicted result | PASS |

# RESULT AND DISCUSSIONS

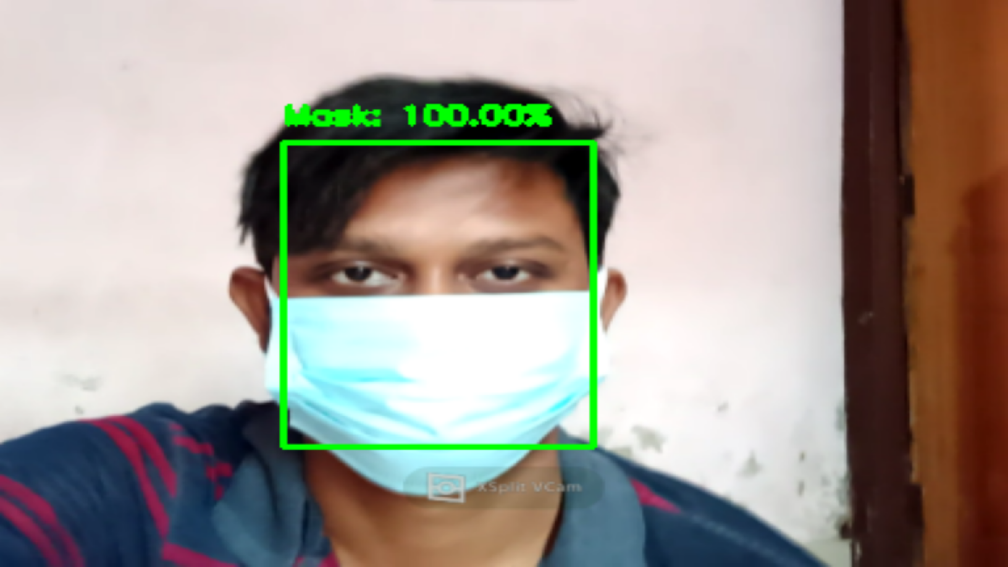
## Results and Outputs

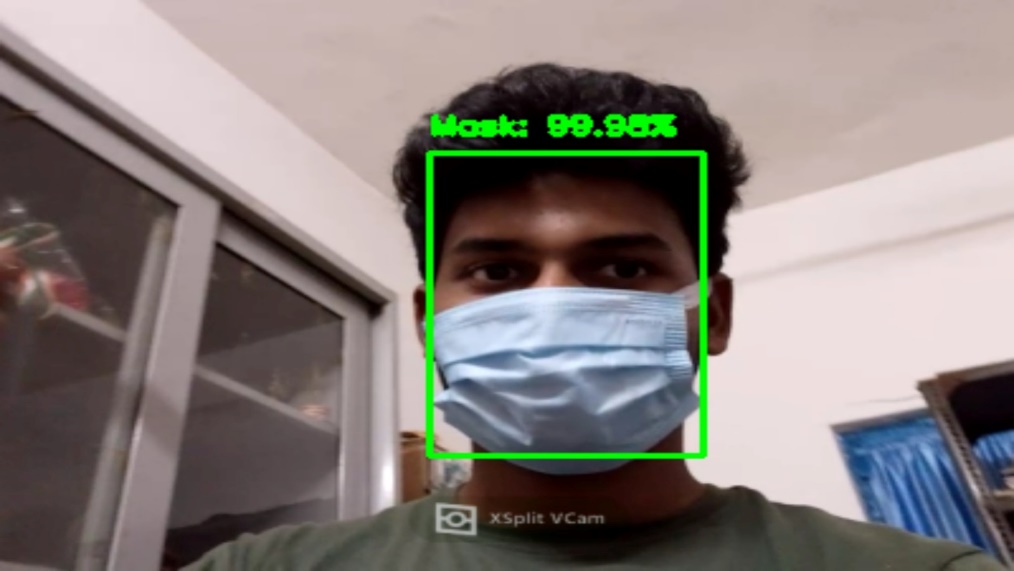
**No Mask Identification**



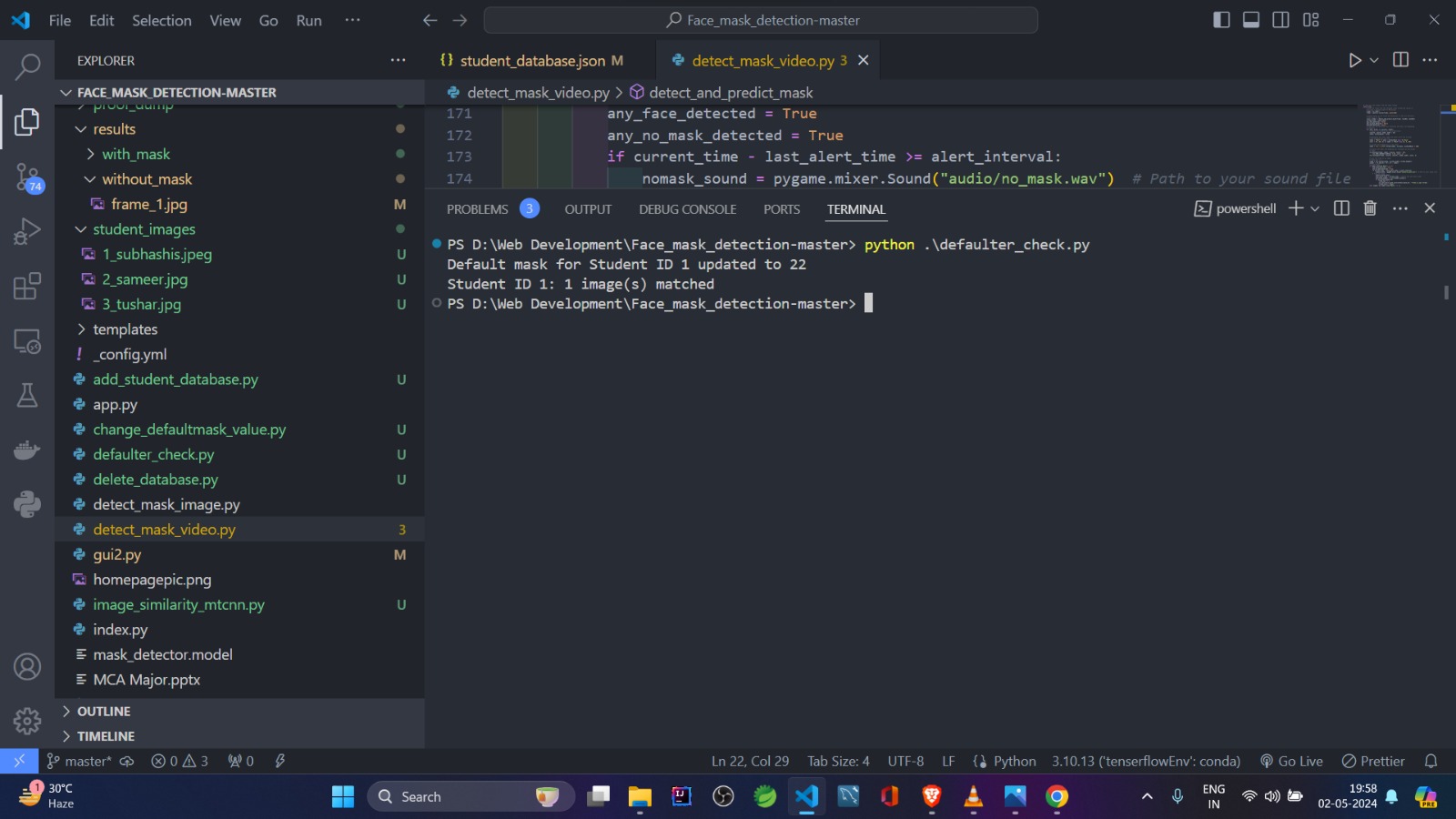


**Mask Identification**

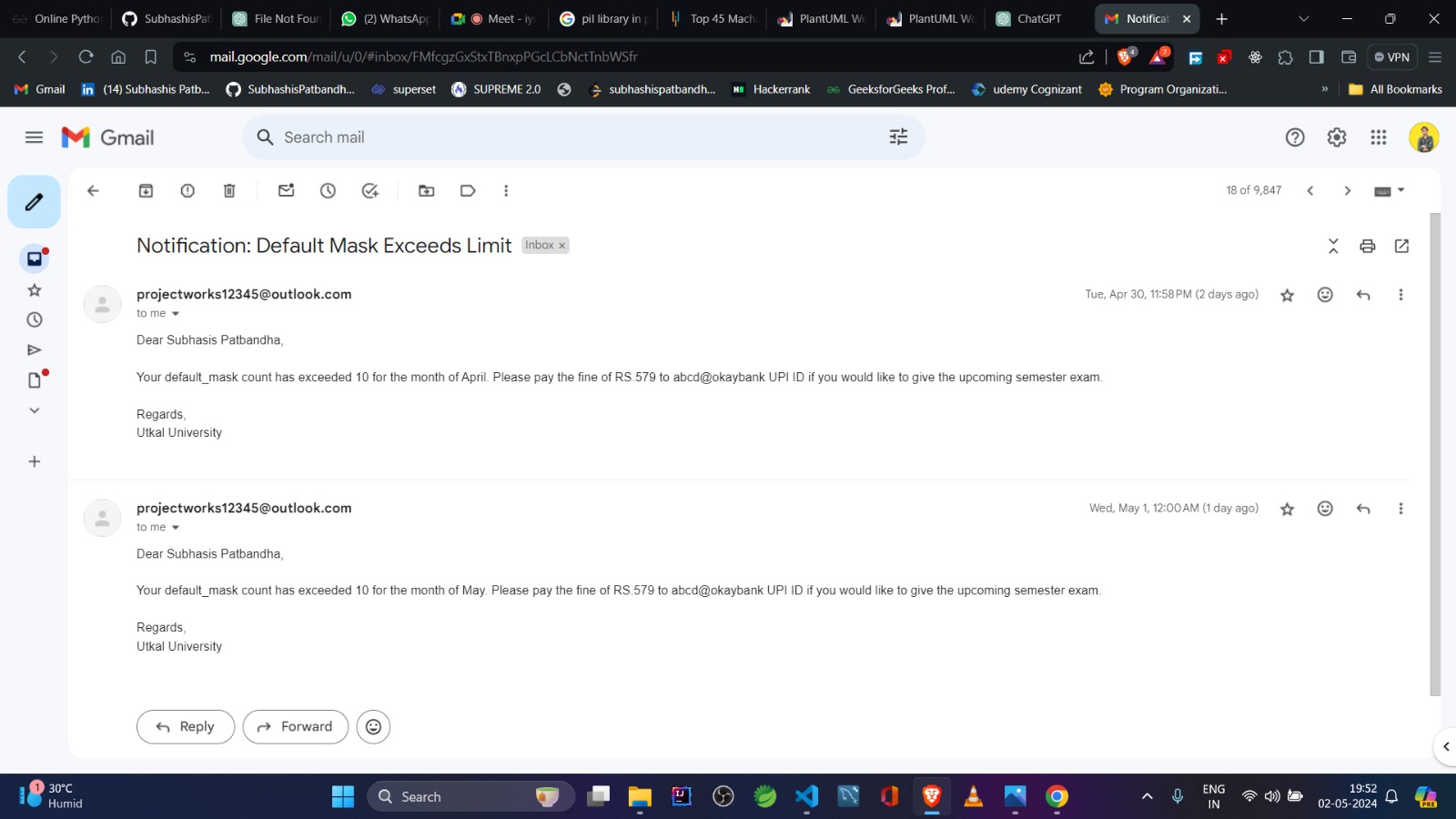




**Deploying Face Matching Algorithm to Update JSON Database**

****

**Sending Automated Emails to those Students Who Didn’t Wore a Mask for over 10 Days a Month**

****

# CONCLUSION AND FUTURE SCOPE

## Conclusion

This project presented a real-time mask detection and enforcement system designed to promote responsible mask usage and enforce mask-wearing regulations within an educational institution. The system leverages advancements in deep learning and facial recognition technologies to achieve its functionalities.

This project has the potential to significantly contribute to creating a safer learning environment within educational institutions during the COVID-19 pandemic and beyond. By promoting responsible mask usage and enforcing mask-wearing regulations, the system can help mitigate the spread of the virus and protect the health of students, faculty, and staff. The project also demonstrates the capability of technology to address public health challenges and promote responsible behaviour.

**Key Findings:**

* The system provides immediate feedback to individuals regarding their mask-wearing status, enhancing awareness and encouraging compliance.
* Automated data collection facilitates the identification of repeat offenders and allows for targeted enforcement actions.
* The use of open-source libraries minimizes licensing costs, making the system economically feasible.
* Careful hardware selection and potential infrastructure upgrades can address technical requirements.
* Effective communication, transparency, and user education are crucial for achieving social acceptance of the system.

## Future Scope

This project lays a strong foundation for a comprehensive mask detection and enforcement system. Here are some exciting possibilities for future development:

* **Integration with Automatic Doors:** The system can be integrated with access control mechanisms like automatic doors. Entry will be granted only when an individual is wearing a mask, enhancing enforcement capabilities.
* **Accuracy Improvement:** Continuous improvement of the mask detection model's accuracy is an ongoing pursuit. Exploring advanced deep learning architectures, leveraging larger and more diverse datasets for training, and incorporating techniques like transfer learning can further refine the model's performance.
* **Enhanced Face Matching:** Facial recognition accuracy can be improved through techniques like data augmentation, incorporating pose variations, and utilizing more sophisticated facial recognition algorithms.
* **Data Storage and Analysis for Research:** The anonymized data collected by the system can be a valuable resource for research purposes. Insights can be gained into mask-wearing patterns, compliance trends, and potential areas for improvement in public health strategies.
* **Spreading Awareness:** The project's functionalities can be adapted for public awareness campaigns. Educational videos, interactive demonstrations, and data visualizations can be developed to promote responsible mask usage in various settings.
* **Public Place Implementation:** The core functionalities of the system can be adapted for implementation in public places like transportation hubs, shopping malls, and government buildings. This broader application can significantly contribute to public health and safety.
* **Privacy-Preserving Techniques:** As the system gathers data, implementing robust anonymization techniques and adhering to data privacy regulations are crucial. Transparency and user education regarding data collection practices will be essential for maintaining public trust.

By exploring these future directions, this project can evolve into a powerful tool for promoting mask usage, enforcing public health regulations, and ultimately contributing to a safer and healthier environment for everyone.

# REFERENCES

The following reference has been used to develop the project “Human Activity Recognition”:

* In Defense of Simpler Baseline for Deep Face Recognition" by I. J. Goodfellow, Y. Bengio, and A. Courville (2016). (<https://arxiv.org/pdf/1807.08169>)
* "Mask-RCNN" by K. He, G. Gkioxari, P. Dollár, and R. Girshick (2017). (<https://arxiv.org/pdf/1703.06870.pdf%5D%C2%A0>)
* "A Survey of Deep Learning Techniques for Face Anti-Spoofing" by Y. Guo, Z. Lei, and S. Z. Li (2020). (<https://arxiv.org/abs/2106.14948>)
* "OpenCV" (<https://opencv.org/>)
* "PyTorch" (<https://pytorch.org/>)
* "TensorFlow" (<https://www.tensorflow.org/>)