"COVID-19 SAFETY PROTOCOL MONITORING SYSTEM"

A PROJECT WORK REPORT SUBMITTED TO

THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU
(AUTONOMOUS UNDER VTU)

In fulfillment of the requirements for project work,
Eighth Semester
Bachelor of Engineering in
Computer Science and Engineering

Submitted by

M VENKATESHWAR REDDY (4NI17CS032)
MUKUND N ACHARYA (4NI17CS043)
NIKHIL KUMAR (4NI17CS047)
RAHUL K P (4NI17CS064)

Under the guidance of

Dr. JAYASRI B S
Professor
Department of CS&E
NIE, Mysuru-570008.

Department of Computer Science and Engineering
The National Institute of Engineering
(Autonomous under VTU)
Manandavadi Road, Mysuru - 570 008

2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

THE NATIONAL INSTITUTE OF ENGINEERING

CERTIFICATE

This is to certify that the project work entitled "**COVID-19 SAFETY PROTOCOL MONITORING SYSTEM"** is a bona fide work carried out by **M VENKATESHWAR REDDY (4NI17CS032), MUKUND N ACHARYA (4NI17CS043), NIKHIL KUMAR (4NI17CS047) and RAHUL K P (4NI17CS064)** in complete fulfillment of Project Work Phase – II, Eighth Semester, for award of degree of **Bachelor of Engineering** in **Computer Science and Engineering** during the academic year 2020-2021. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The Project Work Phase – II report has been approved in complete fulfillment as per academic regulations of **The National Institute of Engineering, Mysuru.**

………………………	………………………….	………………………
Signature of the Guide	Signature of the Co-Guide	Signature of the HOD
Dr. Jayasri B S	**Mr. Lokesh S**	**Dr. Annapurna V K**
Professor	Associate Professor	Professor and Head
Dept. of CSE,	Dept. of CSE,	Dept of CSE,
NIE, Mysuru	NIE, Mysuru	NIE, Mysuru

……………………………..

Signature of the Principal

Dr. N V Raghavendra
Principal
NIE, Mysuru

ACKNOWLEDGEMENTS

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people whose ceaseless cooperation made it possible whose constant guidance and encouragement crown all efforts.

First and foremost, we would like to thank our beloved principal **Dr. N.V Raghavendra** for being the patron and the beacon light for this project.

We would like to express our sincere gratitude to our HOD **Dr. Annapurna V K,** Department of CSE, NIE for her relentless support and encouragement.

It gives us immense pleasure to thank our guide **Dr. Jayasri B S,** Professor, Department of CSE and our co-guide **Mr. Lokesh S** Associate Professor, Department of CSE, NIE for their valuable suggestions and guidance during the process of the project and for having permitted us to pursue work on the subject.

-M VENKATESHWAR REDDY

-MUKUND N ACHARYA

-NIKHIL KUMAR

-RAHUL K P

ABSTRACT

The entire globe is going through a severe health crisis due to the rapid spread of Covid-19 Disease, which in turn is creating a deep impact on the human lives and their day-to-day livelihood. Only hope of preventing it from further spread of disease is by following all the precautionary measures provided by the WHO. Therefore, face mask detection and safe social distance monitoring has become a crucial computer vision task to help the global society.

We have proposed to develop a system that can monitor whether a person is wearing a facemask correctly/not in real time. This will help to reduce the rapid spread of the disease in public places and various other organizations. We have proposed a solution that uses Artificial Intelligence and has a capability to detect the violation of wearing a face mask in real-time using Image Recognition and Video Processing Techniques.

The main intention of the proposed work is to provide a Web based front-end software for administrators to monitor the violation. The system will capture the violated instance and store the data. Our system will also include a user registration form where the users have to register their information along with their face images. This will be an input to the face recognition model, which will train and alert the user whenever the violation occurs.

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

INTRODUCTION

S. Wang Chenet al. [1] have stated that the entire globe is going through a severe health crisis due to the rapid spread of Covid-19 Disease, which in turn is creating a deep impact on the human lives and their day-to-day livelihood. Only hope of preventing it from further spread of disease is by following all the precautionary measures provided by the WHO. Therefore, face mask detection and safe social distance monitoring has become a crucial computer vision task to help the global society.

One of the most important safety measures is to wear face mask and maintain social distancing. Hence, we have proposed to develop a system that can monitor whether a person is wearing a facemask correctly/not in real time. This will help to reduce the rapid spread of the disease in public places and various other organizations. We have proposed a solution that uses Artificial Intelligence and has a capability to detect the violation of wearing a face mask in real-time using Image Recognition and Video Processing Techniques. In accordance with Matrajt L et al. [2], WHO recommends that people should wear face masks to avoid the risk of virus transmission and also recommends that a social distance of at least 6 feet to be maintained between individuals to prevent person-to-person spread of disease.

Reopening of the schools will be a major issue as countries which have reopened schools, have witnessed a sudden spike in the COVID cases. Therefore, face mask detection and safe social distance monitoring has become a crucial computer vision task to help the global society.

The main intention of the proposed work is to provide a Web based front-end software for administrators to monitor the violation. The system will capture the violated instance and store the data. Our system will also include a user registration form where the users have to register their information along with their face images. This will be an input to the face recognition model, like which will train and alert the user whenever the violation occurs.

Chapter 2

LITERATURE SURVEY

Object detection in a real-time environment depends majorly on machine learning. For a model to detect objects in live video streams, it has to be trained with labeled data. The dataset consists of labeled data, from which a part is used as training images, while the remaining is used to test the model. The accuracy of the model is dependent on the size and the quality of the images in the dataset. More data in the dataset results in more training data, and more training data leads to the better training of the model, which yields better results in the real world. It learns through trial and error, constantly trying to predict the best outcome. This process is called Supervised Learning.

In the past decade, as the hardware became stronger, Deep-learning based object detection models have caught the attention of many scientists and developers. Zhong-Qiu Zhao et al. [3] explained that a **Convolutional Neural Networks** (CNN) takes an image as an input, allocates learnable weights and biases to various characteristics of the image, which helps it to differentiate one image from another. While in primitive methods filters are hand-engineered, with enough training, Convolutional Neural Network has the ability to learn these filters/characteristics. We will be using Convolutional neural networks because of the fact that CNN's require fewer pre-processing of data in comparison of other similar classification algorithms.

The architecture of a Convolutional Neural Network is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A schematic diagram of a Convolutional Neural Network sequence to classify hand written digits has been illustrated by Sumit Saha [4] below.

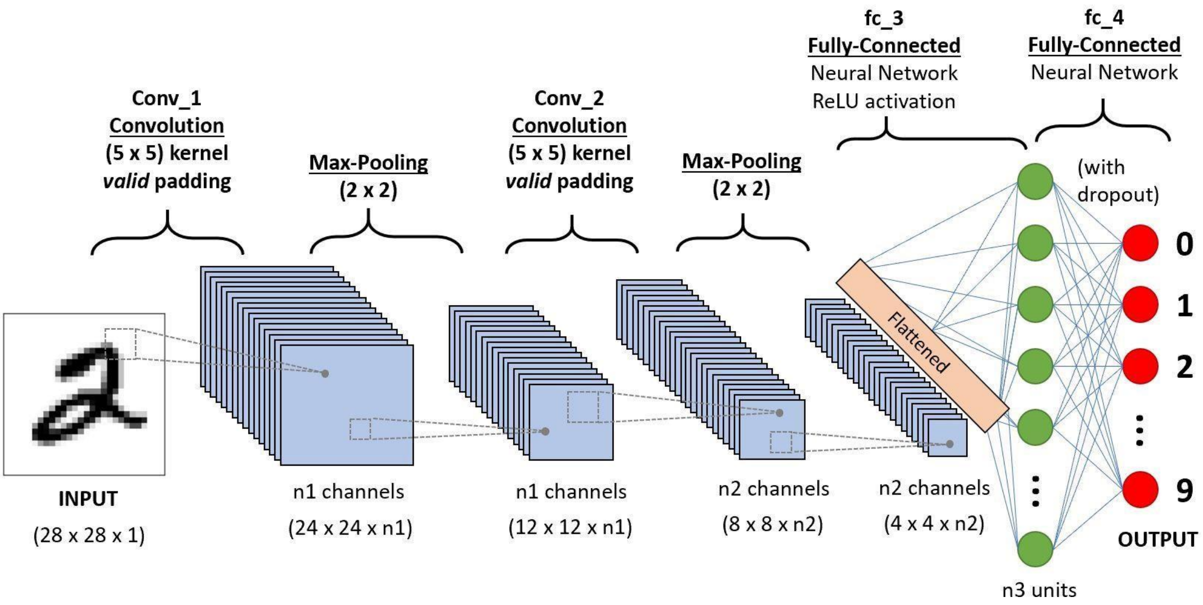

Fig 2.1: A Convolutional Neural Network sequence to classify handwritten digits

The advantages of a Convolutional Neural Network for face mask detection are:

- The architecture of CNN provides an opportunity to jointly optimize several related tasks together.
- Compared with traditional shallow models, a deeper architecture provides an exponentially increased expressive capability.
- Benefitting from the large learning capacity of deep CNNs, we can get better results for computer-vision based object detection problems.
- Hierarchical feature representation of input can be learned from data automatically and hidden factors of input data can be separated.

Convolutional Neural Networks have many versions of pre-trained and well-architected networks like AlexNet, ResNet, Inception, LeNet, MobileNet and so on. In our case we have chosen the MobileNetV2 due to its lightweight and very efficient mobile-oriented model.

MobileNetV2 is a Convolutional neural network architecture that seeks to perform well on mobile devices. Sandler et al. [5] elucidated that it is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.

The MobileNetV2 introduces two new features to the architecture as compared to its predecessor MobileNetV1:

- Linear Bottlenecks present between the layers
- Timesaving connections between the bottlenecks

The basic structure of MobileNetV2 has been illustrated by Andrew Howard et al. [6] below.

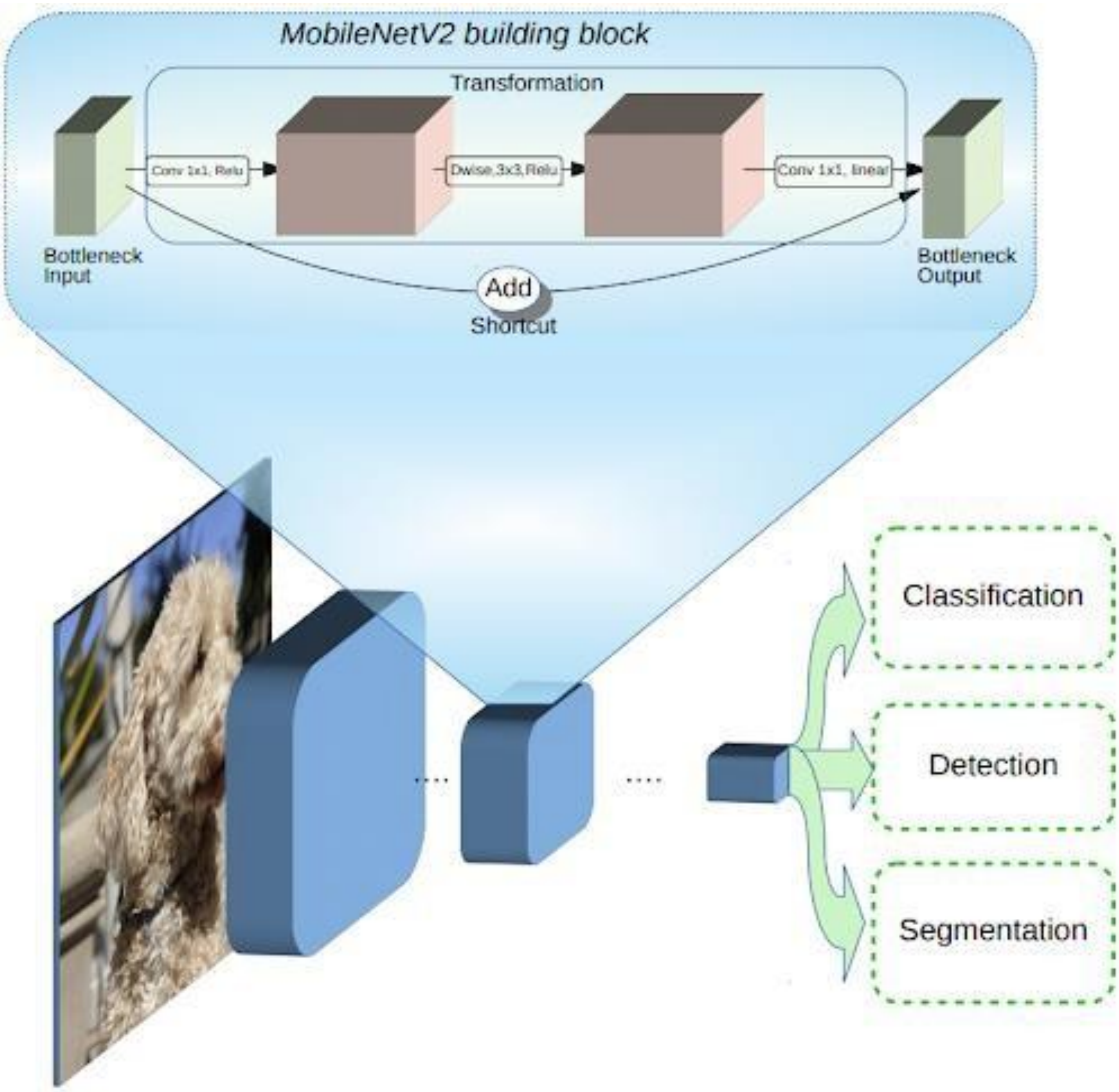

Fig 2.2: The basic structure of MobileNetV2

The typical MobilenetV2 architecture has as many layers listed in Fig 2.3; In PyTorch we can use the Models library in Torch-vision to create the MobileNetV2 model instead of defining/building our own model. The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output channels.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Fig 2.3: Architecture Layers in MobileNetV2

We have chosen MobileNetV2 because it performs smoothly on devices with lower computational power and it outperforms ShuffleNet and MobileNetV1 when tested on ImageNet dataset with equivalent model size and computational cost. When the MobileNetV2 framework detects a human face in an input image, the task is to recognize the violator by performing facial recognition.

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
ShuffleNet (1.5)	71.5	**3.4M**	292M	-
ShuffleNet (x2)	73.7	5.4M	524M	-
NasNet-A	74.0	5.3M	564M	183ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

Fig 2.4: Comparison of MobileNetV2 with other architectures

Similar to object detection, face detection and recognition of the violator needs to be performed in real- time. A human face is detected in real-time using the MobileNetV2 framework. Maintaining accuracy in real-time while improving the speed

Wesley L Passos, Igor M Quintanilla and Gabriel M Araujo [7] have suggested a real-time practical face recognition system that can be implemented to detect human faces in real time. The stages in real time face recognition are shown in the fig 2.5 below.

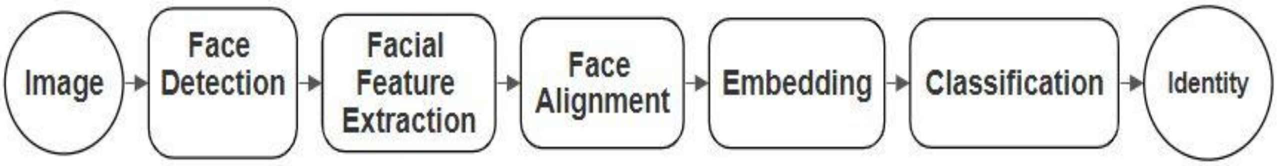

Fig 2.5: The flow diagram of real-time face recognition system

The various stages involved in real time face recognition system described are:

1. **Face Detection**: In this stage, we try to detect a face from the input image using real time systems. Fine tuning MobileNetV2 can be used for detection.
2. **Facial Features Extraction**: In this stage, important facial characteristics like eyes, nose, lips, eyebrows etc. are extracted using regression
3. **Face Alignment**: In this stage, the extracted facial characteristics are lined up with the image center to systemize the face representation. Facial recognition algorithm gets an advantage from this step at later stage.
4. **Embedding**: In this stage, the extracted and systemized characteristics are transformed into arrays and vectors. This is an important stage as facial recognition algorithms cannot run on image pixels.
5. **Classification**: As we know that two images of a same person will contain similar facial characteristics, and different people have different characteristics, we will create the feature vector of a person in advance and compare the calculated vector from the previous steps to the already stored facial vector, we can recognize who that person is.

Chapter 3

SYSTEM ANALYSIS

3.1 INTRODUCTION

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem-solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish the purpose.

3.2 EXISTING SYSTEM

According to Niall O' Mahony et al. [8], object detection techniques using deep models are potentially more capable than shallow models in handling complex tasks and they have achieved spectacular progress in computer vision. The need for a face mask detection system has been felt in the recent times due to the COVID-19 pandemic. Many systems have been developed to detect face masks, but many detectors fail to detect multiple faces in a live video feed. Furthermore, these systems are built just for the purpose of detecting a face mask, but a human has to be present to recognize the violator and send an alert to the violator, which is not fully automatic. Also, many models do not provide a simple User Interface for the administrators to use the system and takes a lot of training to get used to the interface.

Drawbacks of Existing System

- Present systems have performance issues when used with devices with low computational power in real-time systems.
- Present systems are capable of detecting face masks, but they lack the ability to recognize the face of the violator and alerting the violator.
- Present systems do not provide a simple User Interface for the administrators to use this system.

3.3 PROPOSEDSYSTEM

We propose an AI driven solution relying on Real Time Image recognition and Video Processing which will be capable of detecting the instances of violations of Face Mask wearing in the public. The advantage of our system will be that we will provide a Web based front-end software for administrators to monitor and capture the instances where violation occurs and store the data and users to register themselves to the application and will receive alerts whenever they violate the rule of wearing a face mask.

Advantages of the proposed system

- The system is capable of running on devices with low computational power which results good performance in real-time systems.
- The system will be capable of detecting multiple faces in one frame with a reasonable speed.
- The system will also be able to recognize the violators face and send an alert to the violator informing them about the breach of the rules.
- A simple Web application will be provided to the administrators in which they can monitor the violations and act on it.

Chapter 4

SYSTEM REQUIREMENTS

4.1 INTRODUCTION

Software Requirement Specification (SRS) is the starting point of software development activity. It includes an introduction that gives the purpose, scope and overview of the system. This needs requirement by talking to the people and understanding their needs. It also includes a general description of the product perspective, product function and certain user characteristics of the system. It also specifies the overall functional requirements, performance requirements and design constraints.

The SRS is a means of translating the idea in the mind of the clients (the input), into a formal document (the output of the requirement phase). The Software Requirement Specification document is organized in such a manner that it aids validation and system design.

4.2 HARDWAREREQUIREMENTS

- **Processor:** Intel Core i7 for server or laptop used for training the face mask detection model
- **Memory:** Minimum of 8GB for server or laptop
- **GPU**: Nvidia GPU with Nvidia CUDA installed on the laptop/server is recommended for better performance during the model training phase.
- **Video recording Device:** Any video recording device like a CCTV camera feed or a mobile phone live stream with Wi-Fi support.

4.3 SOFTWAREREQUIREMENTS

- **Operating System:** Windows 10 / Linux OS for laptop/server used for training the model
- **Programming Language:** Python, HTML, CSS
- **Drivers:** Nvidia CUDA, Nvidia cuDNN
- **Libraries:** OpenCV, TensorFlow, Scikit-learn, NumPy, Keras

- **Code Editors**: Sublime Text, Microsoft Visual Studio Code.
- **Algorithm Framework**: MobileNetV2, Face_recognition, FaceNet, CNN.

4.4 FUNCTIONALREQUIREMENTS

- Users should register to the portal with the help of the administrators with their details and face image.
- Face Mask detection and face recognition should take place without the need for the user or the administrator to interfere with the working of the app.
- When the algorithm detects a face with no face mask, the violators face is sent to the face recognition model, which recognizes the violators face and an alert is sent to the violator as well as the administrator.

4.5 NON-FUNCTIONALREQUIREMENTS

- **Usability:** This system will be useful to educational institutions for conducting in-person classes.
- **Reliability:** The system provides a reliable output with accuracy and will have an easy User Interface.
- **Re-usability:** The system may be used 24x7 and by the administrators.
- **Safety:** The system must be safe for the users and administrators.

Chapter 5

SYSTEM DESIGN

5.1 ARCHITECTURALDESIGN

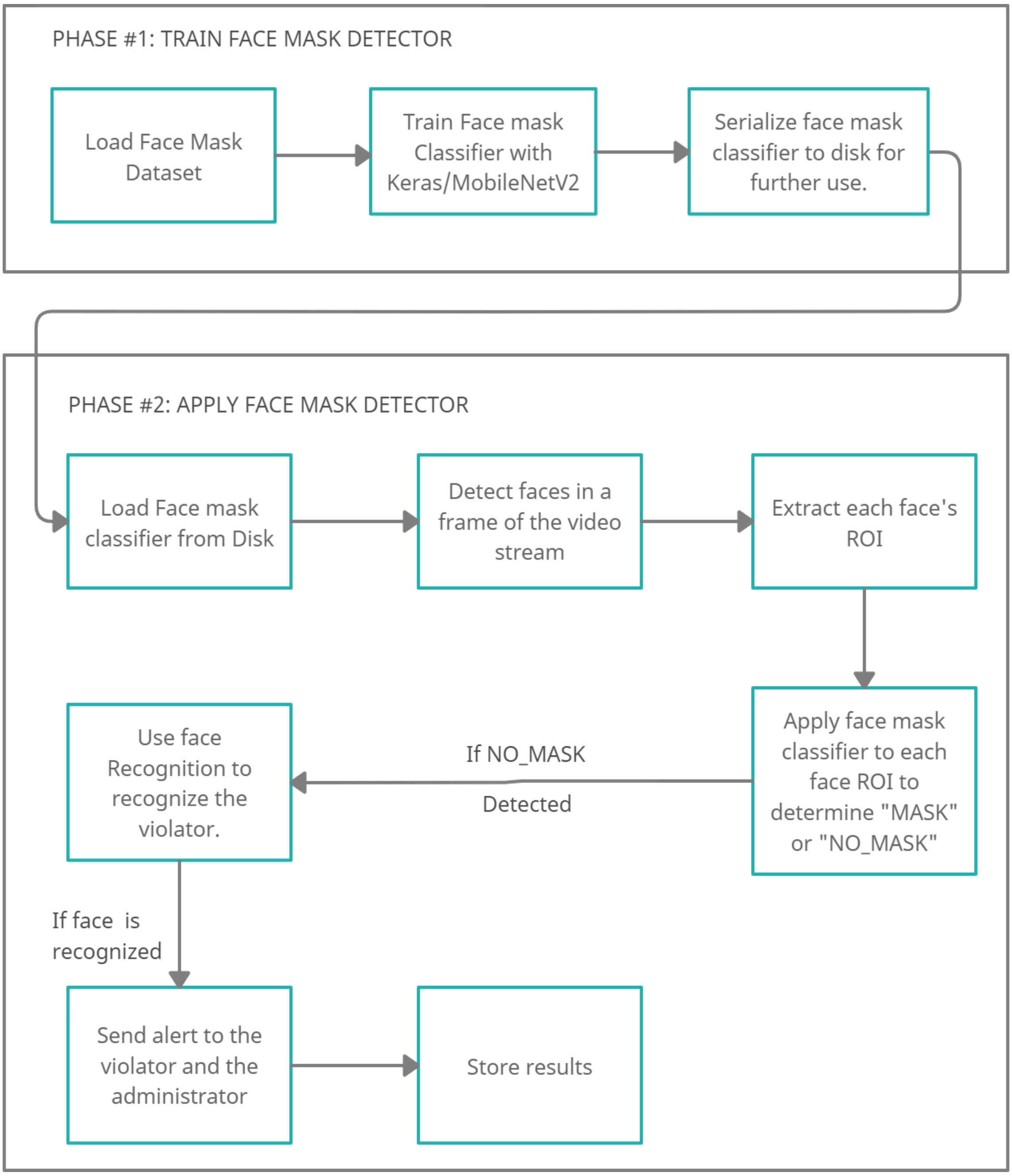

Fig 5.1: System Architecture of COVID-19 Safety Protocol Monitoring System

With reference to Fig 5.1, the system architecture is explained below. To start with, the face mask classifier has to be trained using a dataset which will contain numerous images belonging to two classes, images which contains a face wearing a face mask and the other will contain a face not wearing a face mask. More the number of images in the dataset, our classifier will be more accurate.

After the dataset is created and loaded, we will be using **Keras** and **TensorFlow** to train our classifier to automatically detect whether a person is wearing a mask or not. To accomplish this task, we'll be fine-tuning the **MobileNetV2** architecture, a highly efficient architecture that can be applied to embedded devices with limited computational capacity (ex., Raspberry Pi, Google Coral, NVIDIA Jetson Nano, etc.). After training the MobileNetV2 classifier, we will get our classifier which we can load when we want to detect face masks.

In the next phase of the system architecture, we can move on to loading the mask detector, performing face detection, and then classifying each face as **MASK or NO_MASK.** To start, we load the face mask classifier that we trained in the first phase for mask detection.

The next step involves the input of the live video stream from the IP camera devices. This is done by the library **OpenCV**. We can access the camera footage if the IP cameras are connected to the same network as the server/laptop in which this system will be running. OpenCV will then extract the frames from the video feed and these frames will be used as input to the face detector. When the face detector detects a face or multiple faces in the frame, it will extract the face regions of interest (ROI) from the image.

After this happens, the face mask classifier performs mask detection on each face ROI to predict the probabilities of **MASK** or **NO_MASK**. The result is decided by which variable has the higher value. If any one of the detected faces has a value of NO_MASK, the system passes the face ROI to the face recognition module, which recognizes the face and finds the details of the violator from the user database and an alert, will be sent to the violator and the administrator. The system keeps on running till it receives the input video stream.

The system will also include a user registration form where the users have to register their information along with their face images for the face recognition model to train and to alert the user when he/she will violate the face mask wearing rules.

Chapter 6

SYSTEM DETAILED DESIGN

6.1 Use Case Diagram

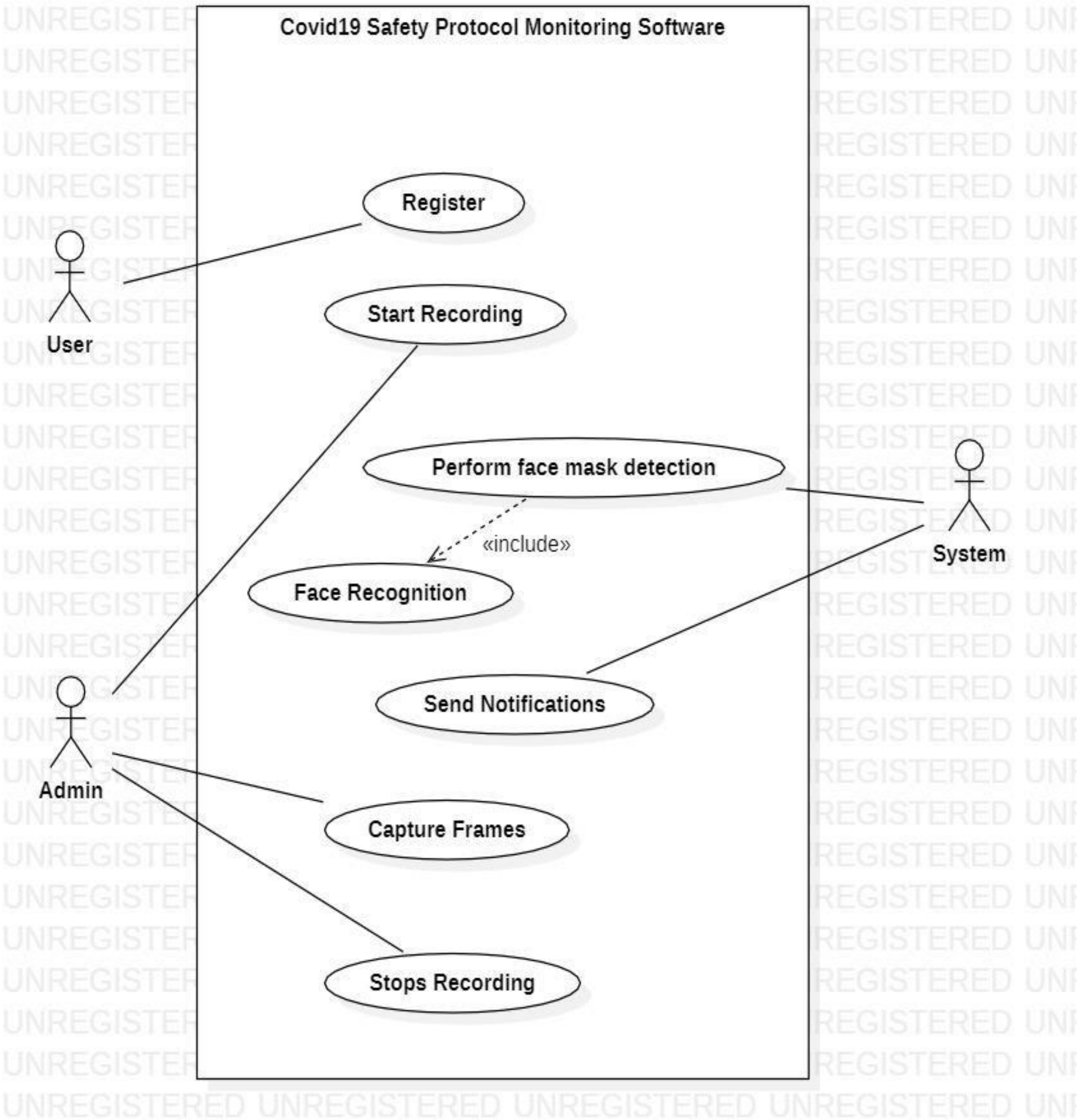

Fig 6.1: Use Case Diagram for COVID-19 Safety Protocol Monitoring System

The use case diagram is presented in Fig 5.2. Its purpose is to present the graphical view of the functionality provided by the system in terms of the actors, their goals and any dependencies between those use cases.

Actors:

- Users
- Admin
- Backend System

The use case diagram consists of three actors which are the Users, Admin and the system to perform the object detection.

Roles of the actors:

- The role of the Users is to register by providing their details along with their image. They will get notifications whenever they will be detected without mask.
- The role of the admin is to start and stop recordings, capture frames from the video. The admin can view the violators data and take necessary action.
- The role of the system is to perform object detection on the recorded video. If a face of a person without face mask is detected, then face recognition is performed. Finally, a notification is sent to both the user violating the rules.

Use Cases:

For Users:

- Register

For Admin:

- Start recording
- Stop recording
- Capture frames

For Backend System:

- Perform face mask detection
- Perform face recognition
- Send notifications

6.2 Activity Diagram

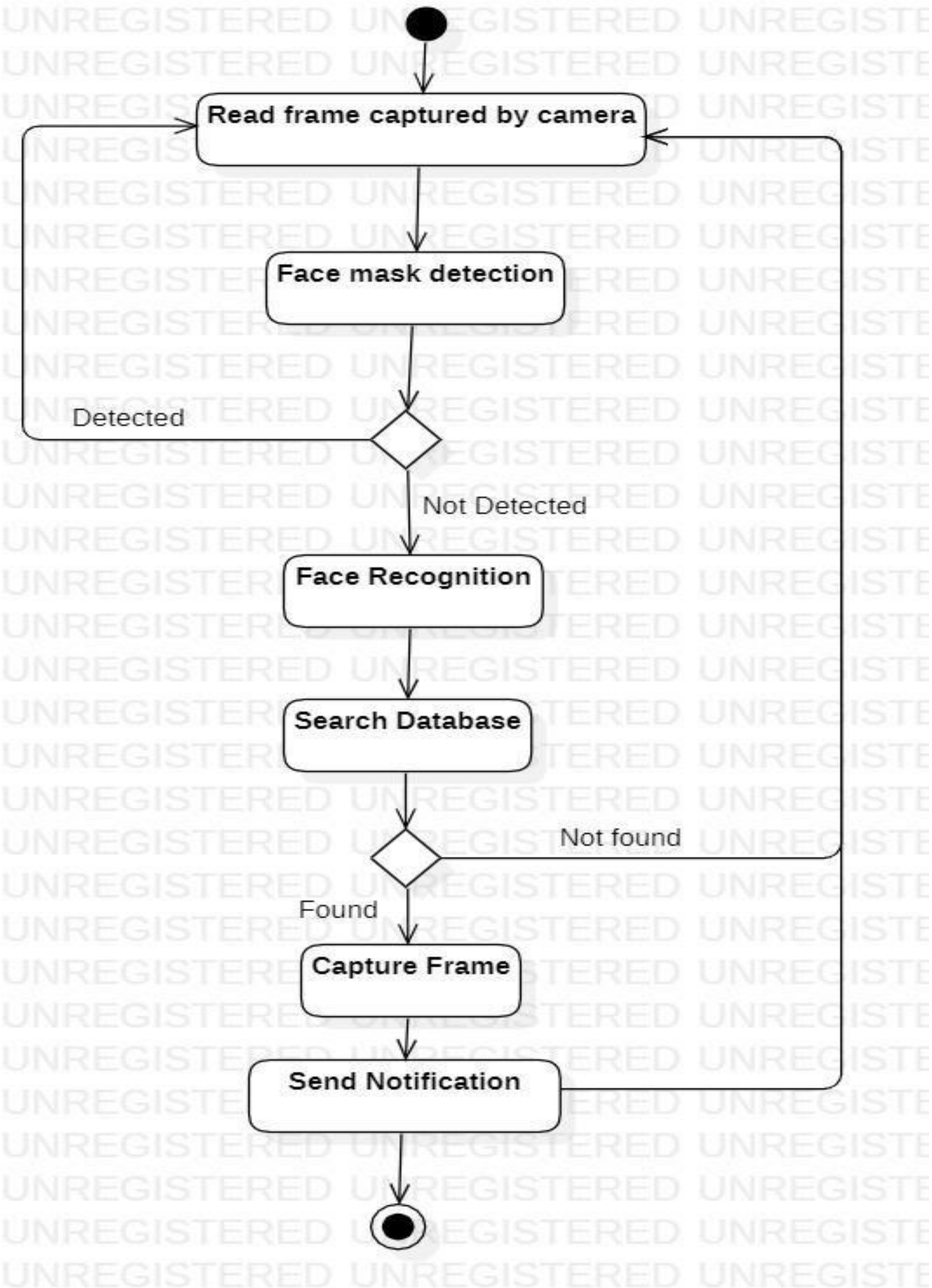

Fig 6.2: Activity Diagram for COVID-19 Safety Protocol Monitoring System

The activity diagram is presented in Fig 5.3. It represents the order in which a particular task of the system is performed to obtain the result.

When the system is started, the video recording gets started. The recorded video is then stored into the message queue for temporary storage. The backend system retrieves the stored video form the message queue for further processing. First, the video is broken down into its constituent frames. Next, object detection is performed on each frame by treating it as an image. If a human face is detected, then face mask detection is performed. In case if no face mask is detected, then face recognition is performed. Finally, a notification is sent to the user violating the COVID-19 norms and the data is stored in a database for the admin to view later.

6.3 SequenceDiagram

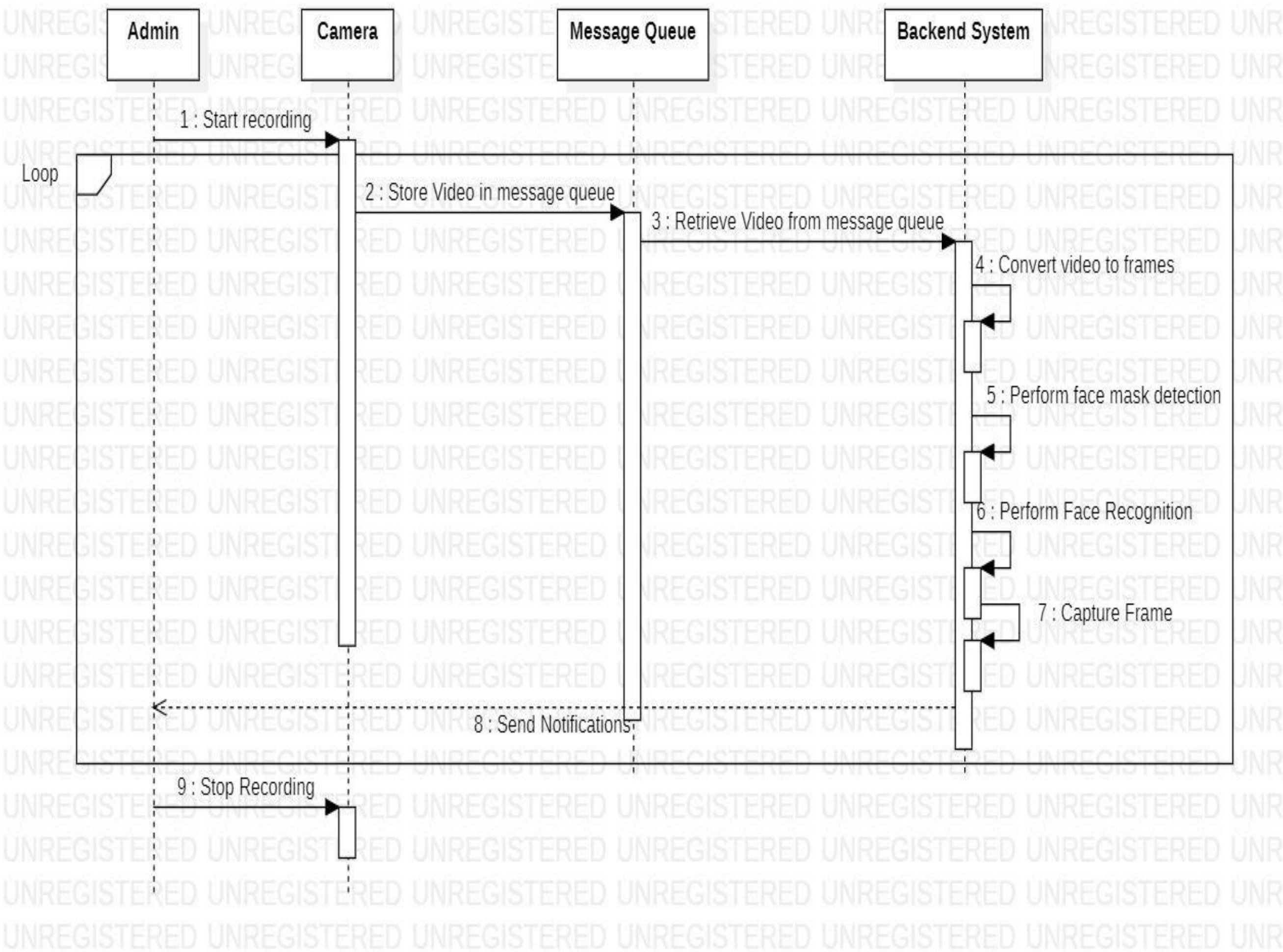

Fig 6.3: Sequence Diagram for COVID-19 Safety Protocol Monitoring System

The sequence diagram in Unified Modelling Language (UML), is a kind of interaction diagram that shows how the processes and the objects of the system interact with each other and in what order. The sequence diagram for the system is represented in Fig 5.4. It shows how each component and process interacts with each other.

Main Objects of the Sequence Diagram:

- Admin
- Camera
- Message Queue
- Backend System

The main sequence of events occurs as follows:

- The admin starts recording the video from the camera when the system is started.
- Now inside a looping construct, the other events will occur until the admin stops recording the video from the camera.
- Once the video is recorded, the message queue acts as a buffer to transfer the recorded video to the backend system.
- The backend system divides the video into frames and performs face mask detection on the frames. If face mask is not detected then face recognition is performed.
- The admin can capture the frames.
- A notification is sent to the person not wearing a mask.

Chapter 7

SYSTEM IMPLEMENTATION

7.1 INTRODUCTION

A crucial phase in the system lifecycle is the successful implementation of the new system design. Implementation simply means converting a new system design into operation. Implementation is a stage in which the design is converted into working system.

Implementation is the process of bringing the developed system into operational use and turning it to the user. This stage is considered to be the most crucial stage in the development of a successful system since a new system is developed and the users are given the confidence of its effectiveness.

7.2 MAJOR PHASES OF PROJECT IMPLEMENTATION

a) Phase I: In this phase, Python plays a key role as the whole system was built using it. A huge dataset of images was collected which contained images of two types, one in which the face contains a face mask and the other in which the face doesn't contain the face masks.

After collecting a total of 4412 images, 2206 images of each class, the model was trained on the dataset with 30 epochs using libraries like Keras and Tensorflow. After the model was trained, the model was found to be around 96% accurate on our test set.

b) Phase II: This was the second phase of implementation of the project where the UI part of the project was built using Python Django module. Since the project requires the use of the face mask detection and face recognition in the Django project, the face mask detection model and face recognition models were imported to the Django file structure.

Once the Django file structure was created, the UI part of the project was implemented using HTML5, CSS and JavaScript. This simple interface was created as per the likes of the end users of the application so that they don't face any issues while using the application in real life.

The initial versions of the application were tested for various test cases and parameters. The versions have also been tested in real time as a part of pilot runs by the developers of the application to further enhance the usability and functioning of the application. Despite these efforts some bugs may still persist and will be solved in the future enhancements of the application.

7.3 TOOLS AND TECHNOLOGIES USED

1) ***Python***: Python is a high level, interpreted programming language. There are a large number of libraries available in Python for the purpose of applications such as scikit-learn for machine learning, OpenCV for computer vision, TensorFlow for neural networks, etc.

2) ***OpenCV:*** OpenCV is the library that we used for live video input stream for our application**.** The purpose of computer vision is to understand the content of the images. It extracts the description from the pictures, which may be an object, a text description, and three-dimension model, and so on. For example, cars can be facilitated with computer vision, which will be able to identify and different objects around the road, such as traffic lights, pedestrians, traffic signs, and so on, and acts accordingly.

3) **TensorFlow**: TensorFlow is a library and framework to serve machine learning tasks. It is used in machine learning and neural network applications. It has a flexible architecture and can be deployed on servers, laptops, mobile phones, edge devices, etc. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units. Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

4) ***Keras:*** Keras is a library that runs on top of TensorFlow that have we used to train our model. It is user-friendly and allows for the training of machine learning models with relative ease using neural networks. Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

***5) FaceNet*:** FaceNet is used for mapping face images and calculate face similarity which is used to extract high-quality aspects from faces that are used to train the face identification system in our application.

***6) Scikit-learn*:** Scikit-learn is a machine learning library that includes various classification, regression, and clustering algorithms for various machine learning applications and can be interoperated along with other Python libraries such as SciPy and NumPy. Scikit-learn also uses SVM (Support Vector machines) along with FaceNet during the training process to create accurate models for face recognition.

***7) MobileNetV2*:** MobileNetV2 is a Convolutional neural network architecture that seeks to perform well on mobile devices. It is based on an inverted residual structure where the residual connections are between the bottleneck layers. The intermediate expansion layer uses lightweight depth wise convolutions to filter features as a source of non-linearity.

***8) CNN*:** Convolutional Neural Network is a powerful recognition algorithm that is widely used in pattern recognition and image processing. CNN is used for the classification of images in our model. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage.

***9) Django*:** Django is a web framework based on Python that is used to build web applications and web APIs. Django's primary goal is to ease the creation of complex, database-driven websites. The framework emphasizes reusability of components, less code, low coupling, rapid development, and the principle of don't repeat yourself. . Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models.

***10) PostgreSQL*:** PostgreSQL, also known as Postgres, is a free and open-source relational database management system emphasizing extensibility and SQL compliance.

7.4 IMPLEMENTATION PROCESS FOR FACE MASK DETECTION

The dataset which is created for the classification of **MASK** and **NO_ MASK** faces will be trained, tested and saved as a .MODEL file. This process is shown in the figure.

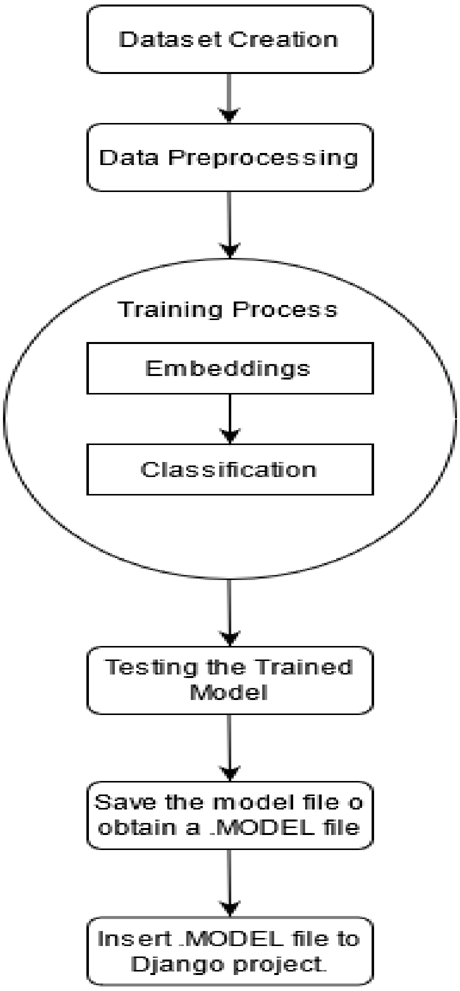

Fig.7.1: Implementation process for Face Mask Detection

7.5 PSEUDOCODE FOR FACE MASK DETECTION

Step 1: Import OpenCV.

Step 2: Import face mask detection model.

Step 3: Take frame from recording device.

Step 4: Detect all faces from the input frame

Step 5: If one or more faces found to have not wearing a face mask

Step 5a: Send the face embeddings to the Face Recognition and alerting phase.

Step 6: Take the next input frame from the recording device and continue from Step 4.

7.6 PSEUDOCODE FOR FACE RECOGNITION AND ALERTING

Step 1: Import face recognition model file

Step 2: Receive face embeddings from face mask detection phase.

Step 3: Run the face recognition model on the face embeddings.

Step 4: If face gets recognized

Step 4a: fetch email address from student database, send alert email and insert student data in Violators table.

Step 5: ELSE

Step 5a: Insert face data in the violators table as Unknown

Step 6: Receive next face embedding from the face mask detection phase.

Chapter 8

TESTING

8.1 INTRODUCTION

Testing is a process of executing a program to ensure that defined input will produce actual results that agree with the required outputs. In a software application, error can be initiated at any stage during the development. For each phase in software development cycle, there are different techniques for detecting and eliminating errors.

System testing is a critical element of software quality assurance and represents the ultimate review of specifications, design and coding. The testing phase involves the testing of system using various test data. Preparation of test data plays a vital role in the system testing. After preparing the test data, the system under study is tested. From those test data, errors were found and corrected by following testing steps and corrections are recorded for future references. Thus, series of testing are performed on the system before it is ready for implementation.

8.2 PURPOSE OF TESTING

Testing accomplishes a variety of things, but most importantly it measures the quality of the software that is being developed. This view presupposes there are defects in the software waiting to be discovered and this view is rarely disproved or even disputed. Several factors contribute to the importance of making testing a high priority of any software development effort. These include:

1. Reducing the cost of developing the program.

2. Ensuring that the application behaves exactly as we explain to the user for the vast majority of programs, unpredictability is the least desirable consequence of using an application.

3. Reducing the total cost of ownership. By providing software that looks and behaves as shown in the documentation, the customers require fewer hours of training and less support from the product experts.

8.3 TYPES OF TESTING

The various types of Testing performed on the system are:

1. UNIT TESTING

It focuses on the smallest unit of the software design. Smallest unit include the module which are integrated to produce the final project. The unit testing focuses on the internal logic and data structures within the boundaries of the component. Test considerations can be the data structures, boundary conditions, independent paths, error handling paths, etc. Unit testing was done on verifying the Face Mask Detection System.

1.1 FEW SAMPLE TEST CASES

Input	Expected Output	Obtained Output	Confidence Score	Status
Frontal face portion of a person properly wearing a mask	Face Mask to be detected	Face Mask Detected	0.98	Success
Side view of a person wearing a mask properly	Face Mask to be detected	Face Mask Detected	0.99	Success
Frontal face of a person who is not wearing a mask	Face Mask Violation should be detected	Face Mask Violation Detected	1.0	Success
Side view of a person not wearing a mask	Face Mask Violation should be detected	Face Mask Violation Detected	0.98	Success
Face of a person wearing a mask without covering the nose	Face Mask Violation should be detected	Face Mask Violation Detected	0.97	Success
Face of a person wearing a mask below his chin	Face Mask Violation should be detected	Face Mask Violation Detected	1.0	Success

Table 8.1: Test Cases for Face Mask Detection System

We have also performed Unit Testing on the second phase of our project, the Face

Recognition System, and below are the obtained test results:

Input	Expected Output	Obtained Output	Confidence Score	Status
Full Face of Mukund from a close distance	Face Recognized in the name of a person as Mukund	Mukund Recognized	1.0	Success
Full Face of Mukund from a distance of 12 inches	Face Recognized in the name of a person as Mukund	Mukund Recognized	1.0	Success

Table 8.2: Test Cases for Face Recognition System

2. INTEGRATION TESTING

Integration testing is a logical extension of unit testing. In its simplest form, two units that have already been tested are combined into a component and the interface between them is tested. A component, in this sense, refers to an integrated aggregate of more than one unit. The idea is to test combinations of pieces and eventually expand the process to test your modules with those of other groups. Eventually all the modules making up a process are tested together. Any errors discovered when combining units are likely related to the interface between units. This method reduces the number of possibilities to a far simpler level of analysis.

3. VALIDATION TESTING

The process of evaluating software during the development process or at the end of the development process to determine whether it satisfies specified business requirements.

Validation Testing ensures that the product actually meets the client's needs. It can also be defined as to demonstrate that the product fulfills its intended use when deployed on appropriate environment.

4. ACCEPTANCE TESTING

This is arguably the most important type of testing, as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirement. The QA team will have a set of pre-written scenarios and test cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors, or interface gaps, but also to point out any bugs in the application that will result in system crashes or major errors in the application.

By performing acceptance tests on an application, the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

5. REGRESSION TESTING

Whenever a change in a software application is made, it is quite possible that other areas within the application have been affected by this change. Regression testing is performed to verify that a fixed bug hasn't resulted in another functionality or business rule violation. The intent of regression testing is to ensure that a change, such as a bug fix should not result in another fault being uncovered in the application.

Regression testing is important because of the following reasons −

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the changes made did not affect any other area of the application.
- Mitigates risks when regression testing is performed on the application.

8.4 SAMPLE FAILURE TEST CASES

Despite the sheer amount of testing of the system, there are some use cases where our system has failed to produce the correct expected output. These problems will be solved in the future versions of the system.

Sample Input	Expected Output	Obtained Output	Confidence Score	Status
Only the left or right half of the face of Mukund	Face recognized with the name of person as Mukund	No Face Recognized.	1.0	Failure

Table 8.3: Sample Failure Test Cases for the Face recognition System

Chapter 9

SAMPLE OUTPUT AND SCREENSHOTS

9.1 TESTCASES OUTPUT

Below are the sample output images that were generated by the system during the testing phase with bounding boxes, labels and confidence values of the output. The system is tested with various angles of face as input to it with both wearing a mask by a person and not wearing a mask by a person.

Figure 9.1 shows when the frontal face portion of a person properly wearing a mask is given to system, it detected accurately with a confidence score of 0.98.

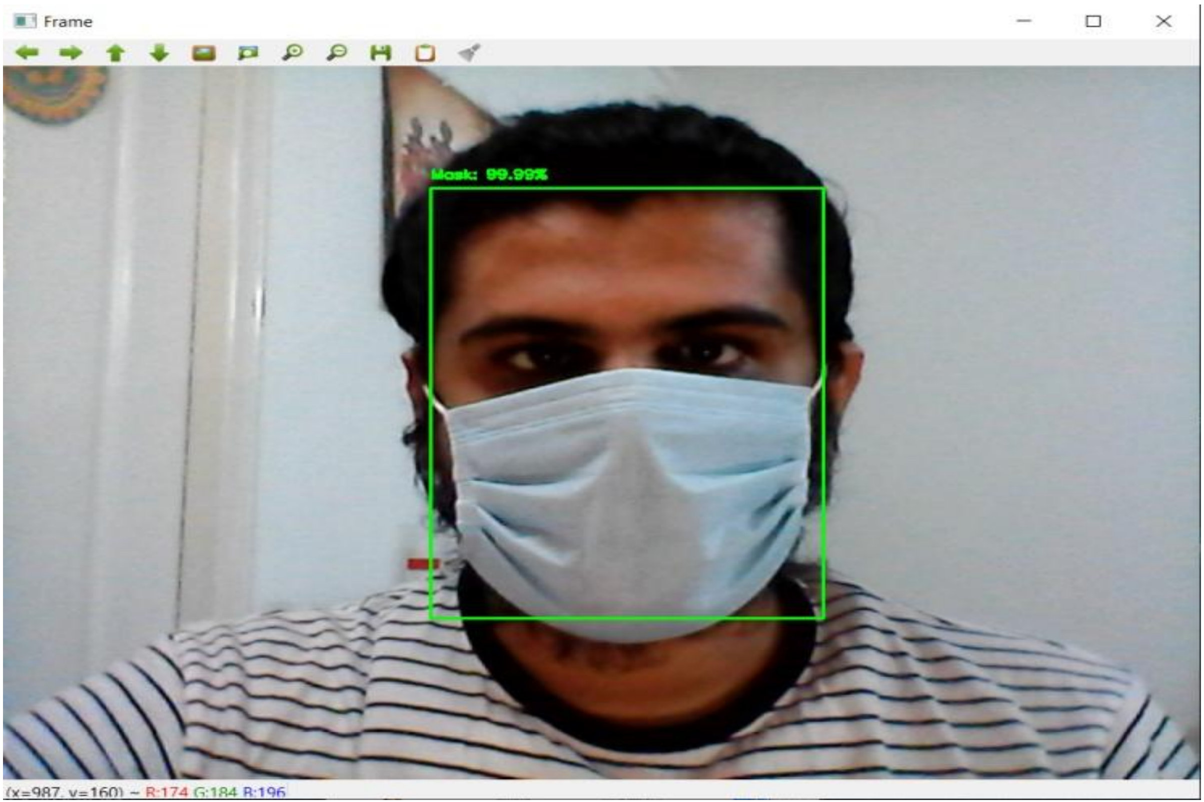

Fig 9.1: Sample output of front face covered by mask

Figure 9.2 shows when the side view of a person wearing a mask properly is given to the system, it detected accurately with a confidence score of 0.99.

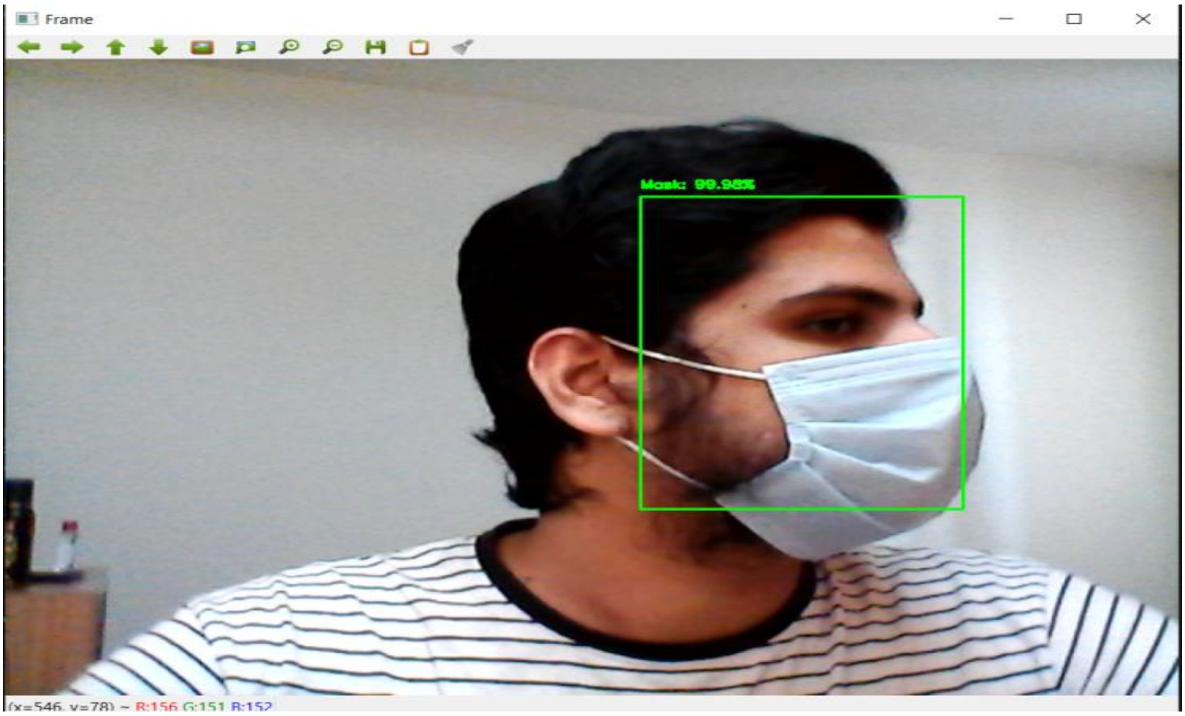

Fig 9.2: Side view of a person wearing a mask

Figure 9.3 shows when the Face of a person wearing a mask without covering the nose is given to the system, it detected accurately with a confidence score of 0.97.

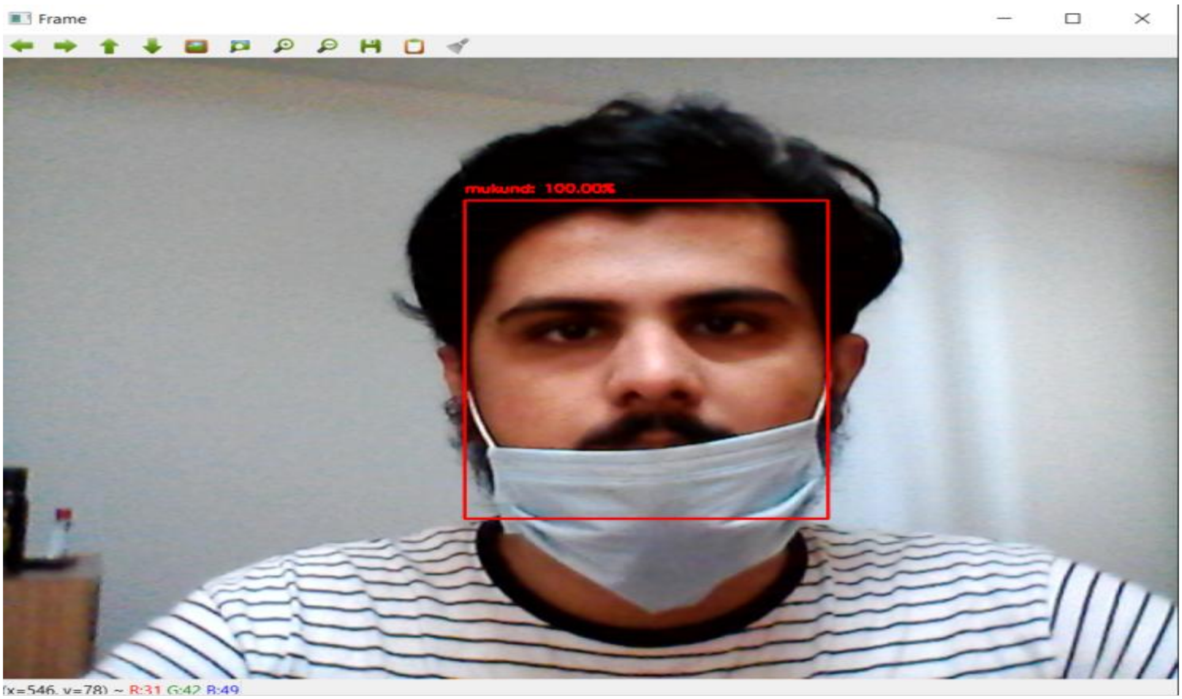

Fig 9.3: Wearing a mask without covering nose

Figure 9.4 shows when the Side view of a person not wearing a mask properly is given to the system, it detected the person not wearing a mask, but the face of the person is not recognized in the side view, with the confidence score of 1.0.

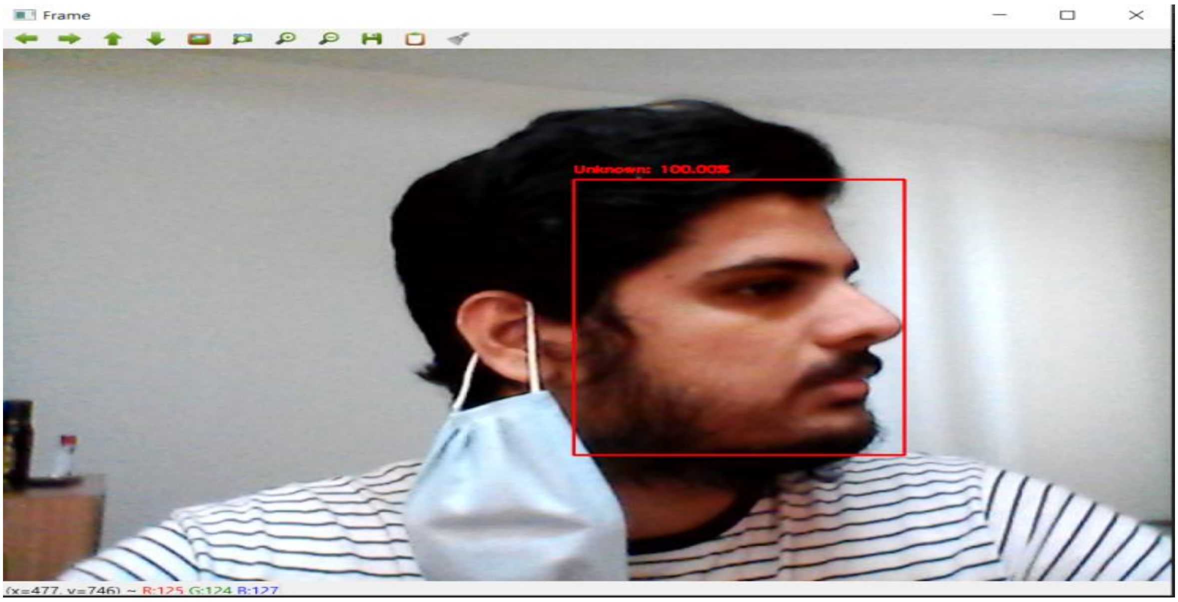

Fig 9.4: Side view of a person not wearing a mask properly

Figure 9.5 shows when the face of a person wearing a mask below his chin is given to the system, it detected accurately with a confidence score of 1.0 and the face of the person is recognized in this view.

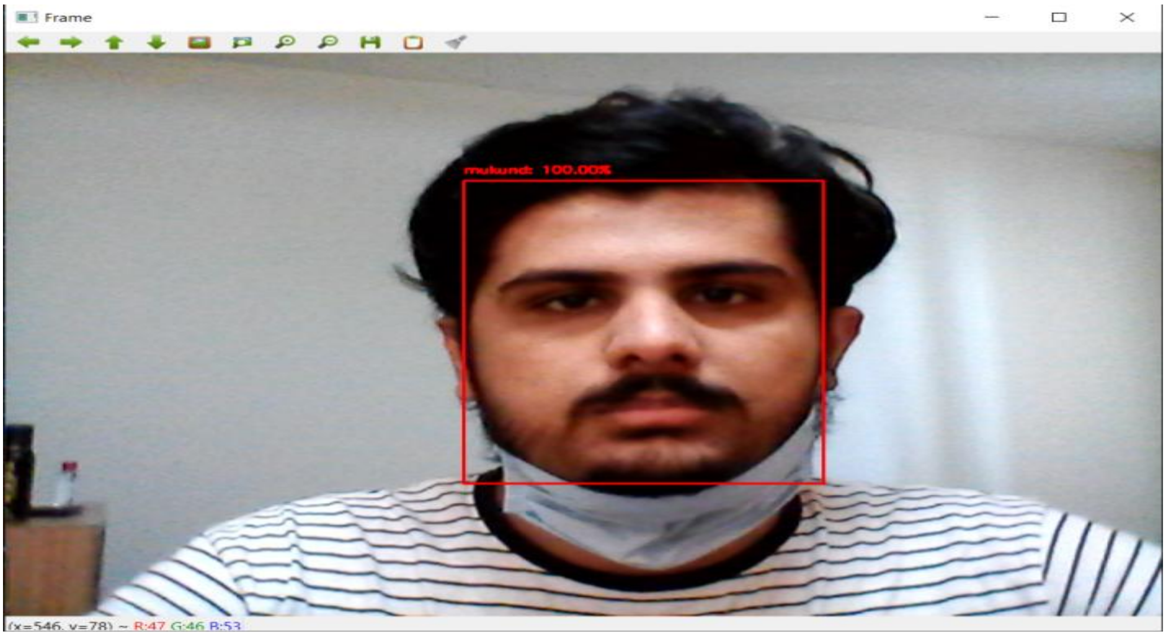

Fig 9.5: Wearing a face mask below the chin

9.1 SCREENSHOTS FROM THE APPLICATION

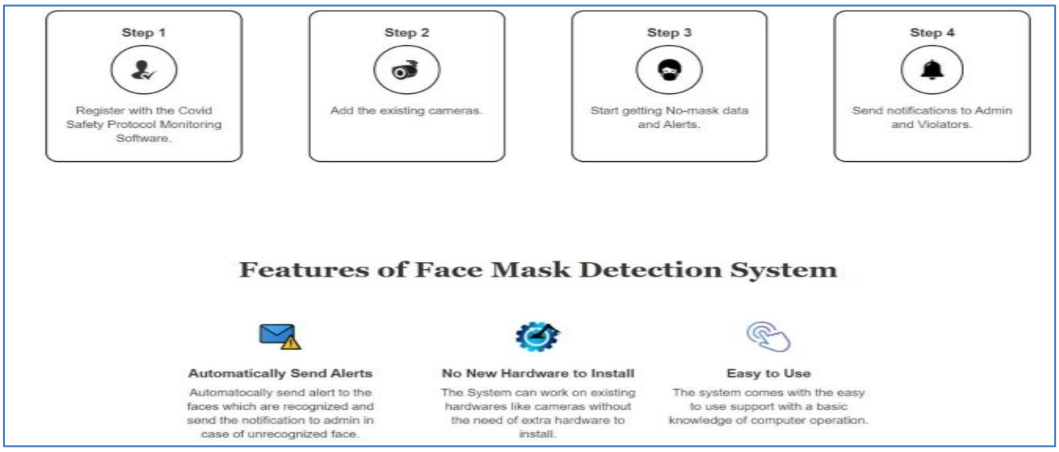

Fig 9.6: Home page of the application

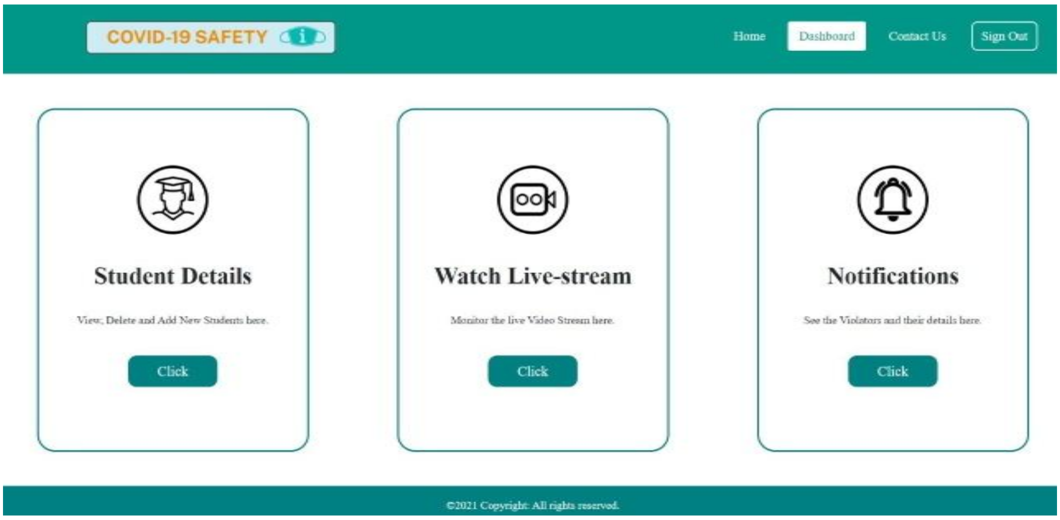

Fig 9.7: Admin Dashboard

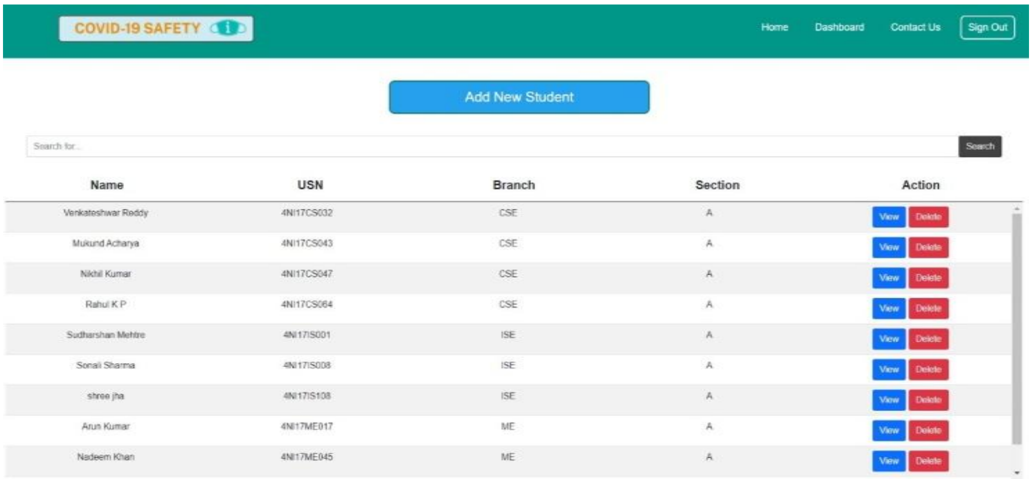

Fig 9.8: Student Details

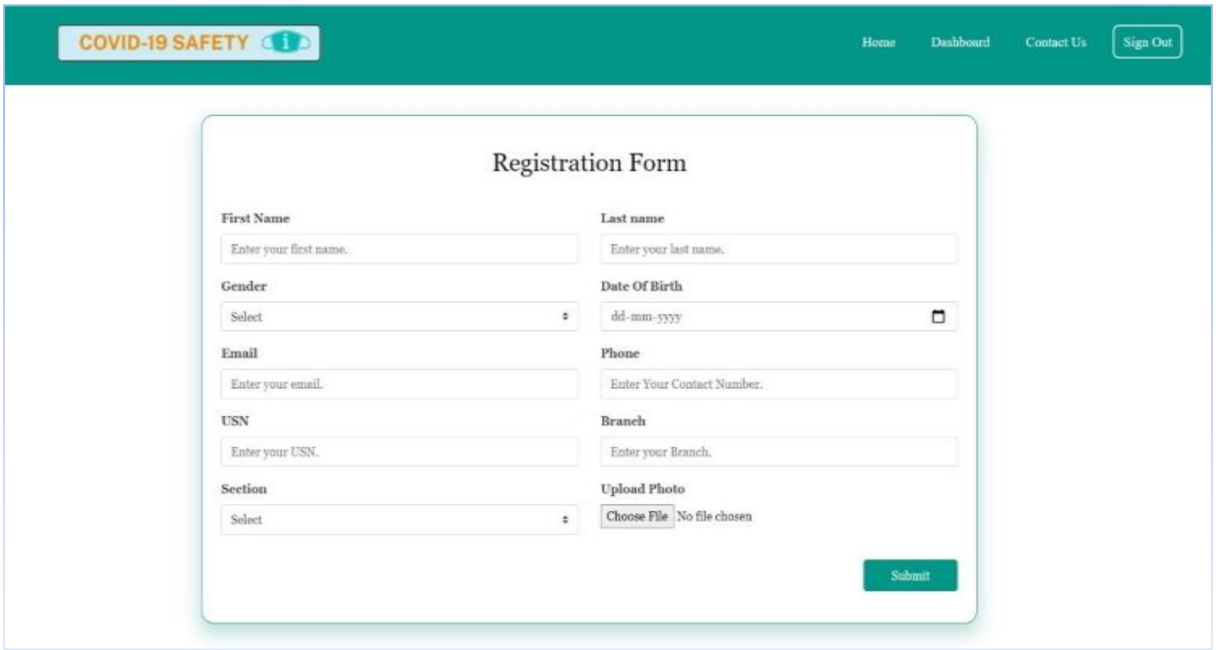

Fig 9.9: Register a new Student

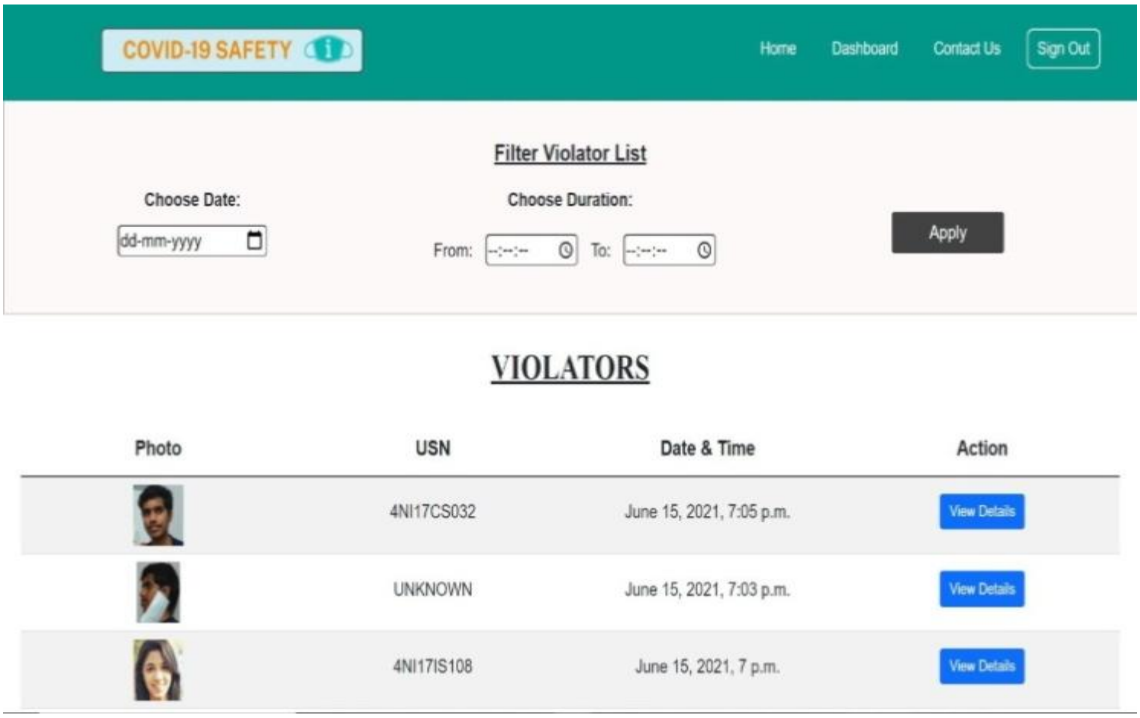

Fig 9.10: Violators List

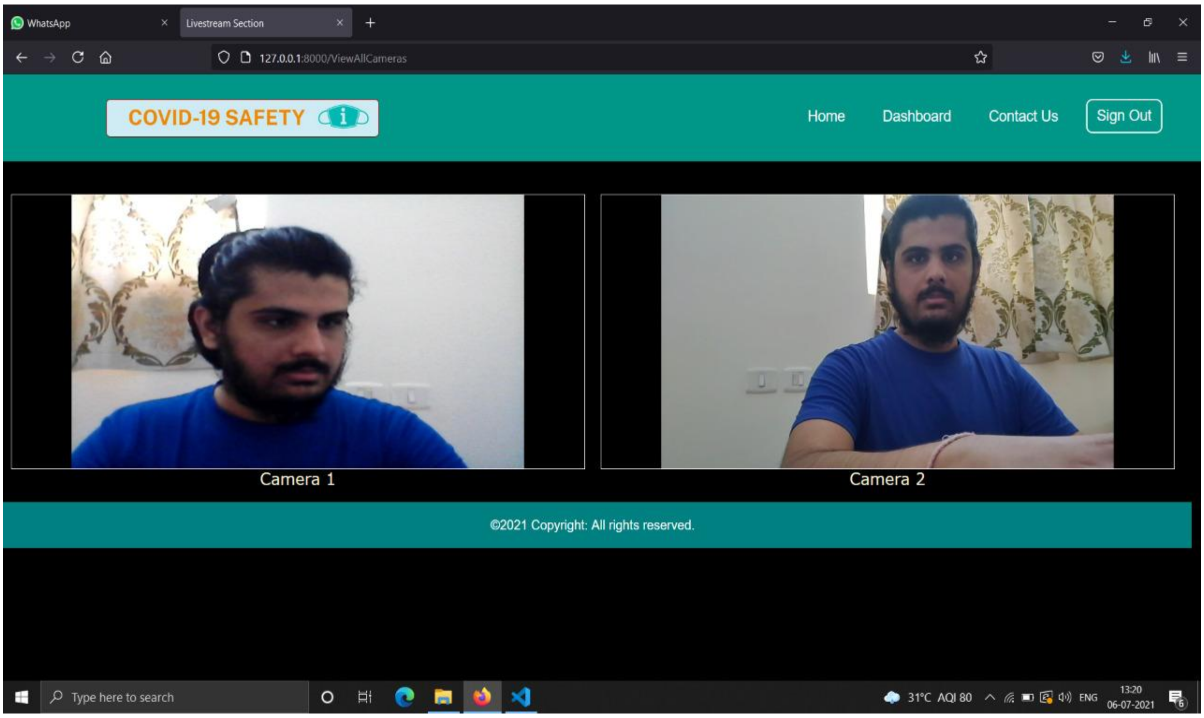

Fig 9.11: Live Stream page with multiple camera inputs

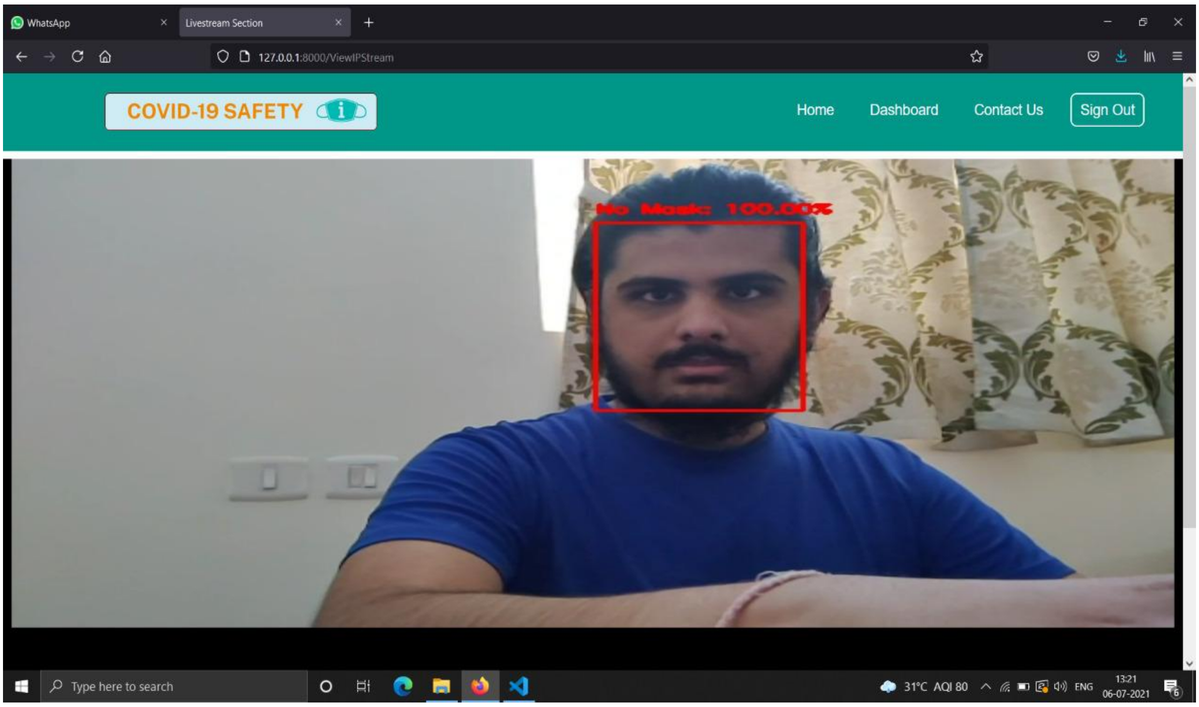

Fig 9.12: Running face mask detection on individual camera

CONCLUSION AND FUTURE ENHANCEMENT

Face Mask Detection is a major requirement in the recent times as COVID-19 is in its full effect in many parts of the world. While there are many solutions that have been able to achieve face mask detection, deploying the model in real time and using facial recognition to recognize and alert the violator needs high computing power to achieve.

Hence, the use of lightweight algorithms such as MobileNetV2 is used to decrease the dependency on high processing hardware. The use of much architecture such as Keras, Tensorflow and FaceNet has made this application a state of the art and deployable in the real world.

After successful deployment our project will be used for monitoring whether people are wearing face mask correctly and are following various other safety norms in schools and colleges. Our project checks for face masks and if violation is occurred, it sends an alert to the violator.

Future work in this field can be focused on implementing social distance using IoT-enabled sensors to detect the distance between two persons and alert the violators by collecting the real-time data and also to implement this in public places and other organisations.

REFERENCES

[1] S. Wang Chen, Horby Peter W, Hayden Frederick G, Gao George F. A novel corona virus epidemic of global concern for health it's the Lancet. 2020; 395(10223):470-473.10.1016/S0140-6736(20)30185-9.

[2] Matrajt L, Leung T. Evaluating the efficacy of social distancing strategies to postpone or flatten the curve of corona virus disease Emerg Infect Dis, man. 2020.

[3] Zhong-Qiu Zhao, Member, Peng Zheng, Shou-tao Xu, and Xindong Wu, "Object Detection with Deep Learning: A Review," IEEE Transactions on Neural Networks and Learning systems 2019.

[4] Sumit Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way", 2018 <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>

[5] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen, "Mobilenetv2: Inverted residues and linear bottlenecks," IEEE Conference on Computer Vision and Pattern Recognition, pp. 4510-4520. 2018.

[6] Mark Sandler and Andrew Howard, "MobileNetV2: The Next Generation of On-Device Computer Vision Networks", Tuesday, April 3, 2018, <https://ai.googleblog.com/2018/04/mobilenetv2-next-generation-of-on.html>.

[7] Wesley L. Passos, Igor M. Quintanilha and Gabriel M. Araujo., "Real-Time Deep Learning Based System for Facial Recognition," XXXVI Simpo´ sioBrasileriode.

[8] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Velasco Hernandez, Lenka Krpalkova, Daniel Riordan, Deep Learning vs. Traditional Computer Vision, Computer Vision Conference (CVC) 2019.