A Project Report on

Using ML for Facial Mask Detection

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Engineering

in

Information Technology

by

Harsh Saraiya(18104006)

Under the Guidance of

Prof. Kiran Deshpande
Prof. Kaushiki Upadhyaya
Prof. Nahid Shaikh

Department of Information Technology
NBA Accredited
A.P. Shah Institute of Technology
G.B.Road,Kasarvadavli, Thane(W), Mumbai-400615
UNIVERSITY OF MUMBAI

Academic Year 2021-2022

Approval Sheet

This Project Report entitled ***"Using ML for Facial Mask Detection"*** Submitted by ***"Harsh Saraiya"(18104006)*** is approved for the partial fulfillment of the requirement for the award of the degree of ***Bachelor of Engineering*** in ***Information Technology*** from ***University of Mumbai***.

(Prof. Kaushiki Upadhyaya, Prof. Nahid Shaikh)
Co-Guide

(Prof. Kiran Deshpande)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Place: A.P.Shah Institute of Technology, Thane
Date:

CERTIFICATE

This is to certify that the project entitled ***"Using ML for Facial Mask Detection"*** submitted by ***"Harsh Saraiya" (18104006)*** for the partial fulfillment of the requirement for award of a degree ***Bachelor of Engineering*** in ***Information Technology***,to the University of Mumbai,is a bonafide work carried out during academic year 2021-2022.

(Prof. Kaushiki Upadhyaya, Prof. Nahid Shaikh)
Co-Guide

(Prof. Kiran Deshpande)
Guide

Prof. Kiran Deshpande
Head Department of Information Technology

Dr. Uttam D.Kolekar
Principal

External Examiner(s)

1.

2.

Place:A.P.Shah Institute of Technology, Thane
Date:

Acknowledgement

We have great pleasure in presenting the report on **Using ML for Facial Mask Detection.** We take this opportunity to express our sincere thanks towards our guide **Prof. Kiran Deshpande** & Co-Guides **Prof. Kaushiki Upadhyaya, Prof. Nahid Shaikh** Department of IT, APSIT thane for providing the technical guidelines and suggestions regarding line of work. We would like to express our gratitude towards his constant encouragement, support and guidance through the development of project.

We thank **Prof. Kiran B. Deshpande** Head of Department,IT, APSIT for his encouragement during progress meeting and providing guidelines to write this report.

We thank **Prof. Vishal S. Badgujar** BE project co-ordinator, Department of IT, APSIT for being encouraging throughout the course and for guidance.

We also thank the entire staff of APSIT for their invaluable help rendered during the course of this work. We wish to express our deep gratitude towards all our colleagues of APSIT for their encouragement.

Student Name: Harsh Saraiya
Student ID: 18104006

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, We have adequately cited and referenced the original sources. We also declare that We have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Harsh Saraiya and 18104006)

Date:

Abstract

The COVID-19 pandemic has severely impacted our daily lives and continues to spread over the globe. Wearing a protective face mask to prevent the spread of infection has become the accepted norm. Since people have become a bit careless towards wearing facial masks outdoors, this system detects the human individual through videos not wearing a mask and informs the authorities about it. This paper proposes a simplified approach to accomplishing this goal utilising TensorFlow, Keras, and OpenCV, as well as some fundamental Machine Learning packages. We have used the Res10 SSD Caffe model for face detection and Image classification is done through MobileNetV2 architecture for the implementation of our system. By attaining high precision, real-time detection, and classification, the suggested methodology established its effectiveness in identifying facial masks. In this pandemic situation, where whole world is dreaming to return to normal routine, this system will play effective role in monitoring the use of face masks at workplaces. This system also contributes to public healthcare, as it helps in keeping environment healthy.

Contents

List of Figures

List of Abbreviations

ML:	Machine Learning
OpenCV:	Open Source Computer Vision
CNN:	Convolutional Neural Network
MTCNN:	Multi-Task Cascaded Convolutional Neural Network
KNN:	K-Nearest Neighbour
RCNN:	Region-based Convolutional Neural Network
DNN:	Deep Neural Networks
YOLO:	You Only Look Once
VGG:	Visual Geometry Group
ROI:	Region Of Interest
API:	Application Programming Interface
RTDB:	Real Time Data Base
CCTV:	Closed-Circuit Television.
FCM:	Firebase Cloud Messaging
SDK:	Software Development Kit

Chapter 1

Introduction

Due to COVID-19 Pandemic, many individuals have died as a result of this illness, and thousands more are affected every day. The 2019 Coronavirus Infection (COVID-19) has infected more than 20 million people worldwide, resulting in over 0.7 million deaths, according to the official Situation Report of the World Health Organization - 205. Coronavirus has infected 213 countries, including all industrialized countries such as the United States, the United Kingdom, Russia, China, Japan, Italy, and many others. People's carelessness and lack of consciousness were the main causes of the infected virus. Every day, people go to work or live in other apartments without wearing a mask. Surveillance is extremely difficult and time-consuming all of the time.

This study is primarily aimed at resolving this issue and assisting people in selfdefense. It's critical, especially in COVID-19, to safeguard ourselves from other individuals. To prevent the transmission of Coronavirus, it is strongly advised that the general public wear face masks. Furthermore, with the lifting of the COVID19 quarantine, government and public health agencies are suggesting face masks as vital precautions to take when going out in public. To make the usage of facemasks mandatory, some measures must be devised that require people to put on a mask before entering public spaces.

Face mask detection is the process of determining whether or not someone is wearing a mask. In actuality, the issue is reverse engineering of face detection, in which the face is detected using various machine learning algorithms for security, authentication, and surveillance purposes. In the domain of Computer Vision and Pattern Recognition, face detection is crucial. Deep convolutional neural networks (CNN)-based face recognition algorithms have become increasingly popular in recent years as a way to improve detection performance. Furthermore, detecting faces with or without a mask in public is difficult due to the little dataset available for detecting masks on human faces, making the model difficult to train. As a result, for a similar face identification task, the notion of transfer learning is used to transfer learned kernels from networks trained on a big dataset. Detecting mask faces can be done in a variety of ways, including from various perspectives. The goal of this research is to create a system for detecting mask faces using CNN's classifiers. Working with CNN enhances the accuracy of identifying the mask face in a certain region, and it's much more responsive when the face enters the webcam's field of vision, resulting in speedier detection. In some ways, the community is safe while more flu stops are dispersed in the air and create entry barriers into the human body.

Chapter 2

Literature Review

In this section, we take a look at some of the similar efforts that have been done in this field. Despite the fact that face detection research has been ongoing for decades and has yielded significant results, only a few approaches and algorithms are explicitly developed for face mask recognition.

The proposed framework in [1] uses the MTCNN facial recognition model to identify the faces in the video frame and their relevant facial landmarks. These images and cues are then examined by a neoteric classifier that detects masked regions using the MobileNetV2 architecture as an object detector.

The purpose of Literature [2] is to give a general review of many methods and algorithms for human recognition using a face mask. The Haar cascade, Adaboost, VGG-16 CNN Model, and other techniques are described. A study of various strategies is conducted to evaluate which technique is feasible.

The author of the paper [3] developed a deep learning architecture based on a dataset of images of people wearing and not wearing masks obtained from various sources. For previously unreported test data, the trained architecture distinguished facial mask wearers and non-mask wearers with 98.7% accuracy.

Two distinct approaches to detect real-time masking and unmasking of faces are proposed in this study [4]. An object detection model is used in the first technique to discover and distinguish between masked and uncovered faces. A YOLO face detector recognises faces, whether masked or not, and then categories them into masked and unmasked groups using a dataset in the second method.

With a minimal number of training samples, Mask R-CNN, an established object detector, has been trained for face detection as well as instance segmentation and object bounding box detection in this study [5]. The results show that the trained Mask R-CNN outperforms the baseline detector in terms of detection rates.

On the basis of the SSD algorithm, literature [7] proposes a method for detecting face masks. To increase the model's ability to exhibit essential features, SSD-Mask incorporates a channel attention technique. At the same time, the loss function is optimised and the information from different feature levels is fully exploited.

The effectiveness of three algorithms: KNN, SVM and MobileNet was compared in the literature [6] to discover the best algorithm for detecting someone is wearing a mask in a real-time context. MobileNet has the maximum accuracy for both input photos and input videos from a camera, according to the results.

Across the full latency spectrum, the MobileNetV2 models are faster for the same accuracy. MobileNetV2 is a powerful feature extractor for detecting and segmenting objects. For example, when paired with the recently released SSDLite, the new model is around 35 percent faster than MobileNetV1 while maintaining the same accuracy. As we've shown, MobileNetV2 is a powerful mobile-oriented model that may be utilised to solve a variety of visual identification problems.

Chapter 3

Objectives

- To automate the process of face mask detection using a CCTV camera.
- To Classify people into the masked and unmasked category.
- Image recognition of unmasked people and notification alert to the Authority about it.
- To ensure a safe working environment by creating an atmosphere of awareness preparedness in the locality.
- To enforce the mandate for wearing masks in public places following the COVID-19 pandemic.
- To develop an efficient computer-vision based system on the real-time automated monitoring of people to detect face masks in public places.

Chapter 4

Project Design

4.1 Existing System

In the existing system, there are two distinct phases such as train face mask detector phase and apply face mask detector phase. Python script is used to train a face mask detector on the dataset using Keras and TensorFlow. There are two more additional Python scripts used to detect COVID-19 face masks in images and to detect face masks in real-time video streams.

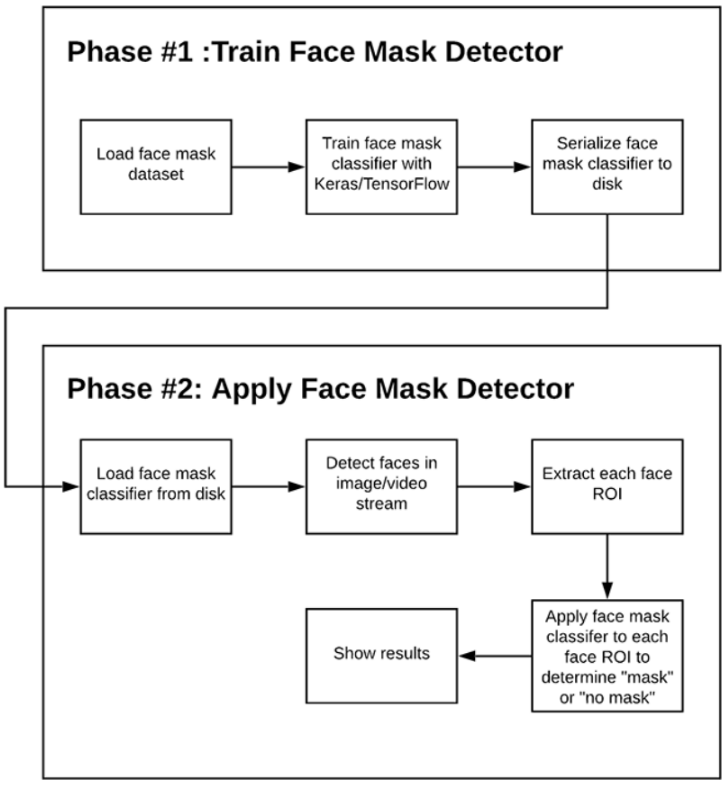


Figure 4.1: Phases and individual steps for building a COVID-19 face mask detector

In order to train a custom face mask detector, we need to break our system into two distinct phases, each with its own respective sub-steps as shown by above figure. The training phase focuses on loading our face mask detection dataset from disk, training a model using Keras/TensorFlow on the dataset and then serializing the face mask detector to disk. In deployment phase, once the face mask detector is trained, we can then move on to loading the mask detector, performing face detection, and then classifying each face as with_mask or without_mask. The dataset used which consists of 1,376 images belonging to two classes: with_mask class having 690 images and without_mask class having 686 images. After classification, it determines masked and unmasked faces and show results such as displaying red coloured bounding box for unmasked people and green coloured bounding box for masked people.

4.2 Proposed System

In this proposed system we have three phases, each with its own respective sub steps. First phase is training the face mask detector model. The second phase is Detection and Recognition. The third phase is Deployment on cloud. We have used the MobileNetV2 classifier model, a highly efficient architecture in order to identify a face mask.

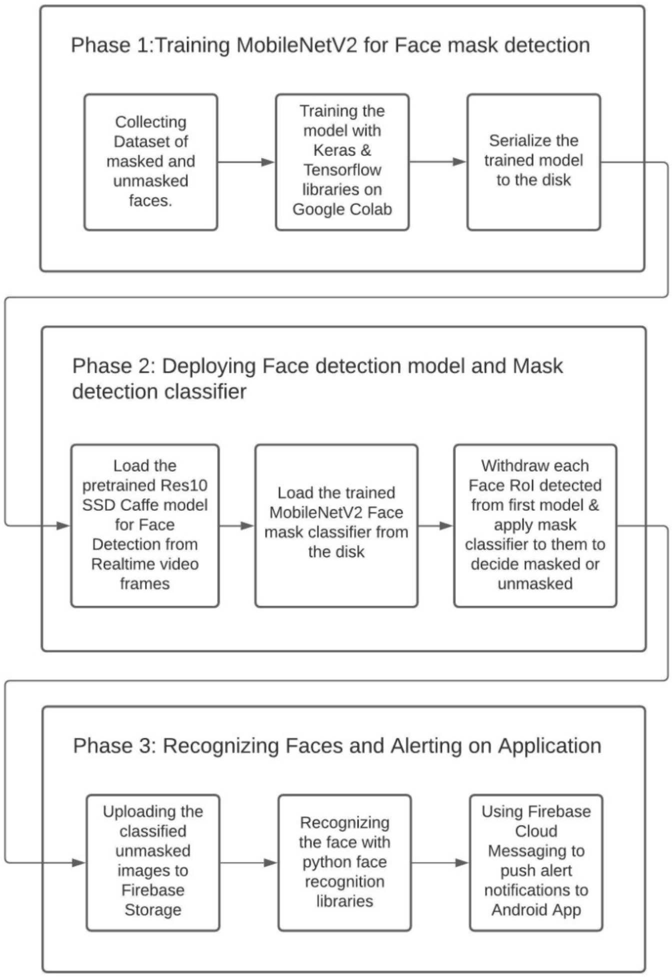


Figure 4.2: Block Diagram of Facial Mask Detection System

4.2.1 Training the Face Mask Model

We have collected 1396 images of faces[9] out of which1000 are faces with masks and 396 images are without masks.These images are then loaded with a target size of 224x224and then preprocessed using Keras mobilenetv2 library. We have split the dataset for the training and testing phase. We trained the model on 80% of the data and tested it on 2%of the data. For Data Augmentation, we created a training picture generator. After some trial and error, we found that 1e-4 offered us the best outcomes and the quickest reduction in losses. With a batch size of 32, the model was trained for 20epochs. A classification report was created during the training procedure and the trained model was preserved.

4.2.2 Detection & Recognition

Face detection from the video frames is carried out by using a pre-trained Res10 SSD Caffe model with the OpenCV DNN module. Initially both the models are loaded from the disk and Video Stream from the required input is started using OpenCV and frames are read and resized. The Facial ROI is detected from the blob of the image and these are passed through the Caffe model. The faces and their corresponding location on the frame are then stored into a list. Then using the (startX, startY, endX, endY) coordinates of the Facial ROI, and converting it from BGR to RGB channel, ordering it, resizing them to 224x224 which was desired for the MobileNetV2 model and preprocessing the image using Keras libraries. Passing this list into the MobileNetV2 model will return the predictions made on the Facial images whether masked or unmasked. Then looping over the coordinates and predictions list, a bounding box is appended over the real time frames of the Video Stream. If prediction = "Mask" bounding box color is assigned to Green and if prediction = "Unmask" bounding box color is assigned to Red.

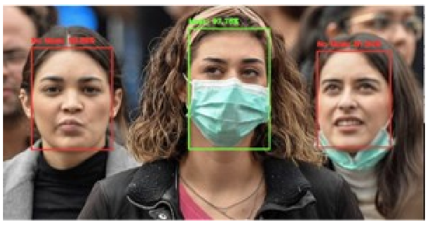


Figure 4.3: Image of masked and unmasked people from real time video footage

From the above classification, the unmasked faces will be used for facial recognition. We used the Python face recognition 1.3.0 package for face recognition, which is based on dlib's state-of-the-art face recognition developed using deep learning[8]. This model will be trained on a known faces dataset of a company, a hospital staff, college campus and so on. The unmasked faces will be classified by this deep learning library and the authority will get to know the details of the unmasked person viz. Name, Unique Identification number, Department and so on.

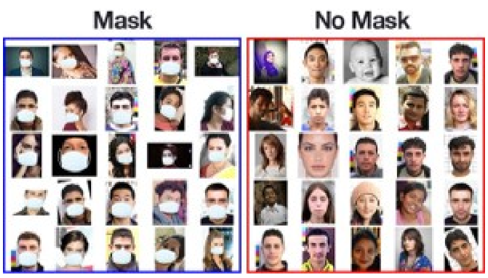


Figure 4.4: Image of masked and unmasked people from real time video footage

4.2.3 Deployment on Cloud For cloud storage

Firebase is a useful platform developed by Google which we will implement for application development. It is the greatest mobile backend as a service since it allows mobile apps to connect to cloud storage and APIs on the backend. In the database, Firebase maintains all data in real time. As a result, data transfer to and from the database is simple and speedy. Google integrated Firebase Cloud Storage with the Firebase SDK, letting users save the files on the cloud in a matter of seconds. We will be using these Firebase SDKs for storing video/images of the people without masks.

The identical files on the server will be accessed via Google Cloud Storage. Every file will be saved in a Google Cloud Storage Bucket and will be accessible from both Firebase and Google Cloud. This allows us to use Firebase SDKs to retrieve and upload files from mobile clients, as well as do serverside computing using Google Cloud. The Firebase Real time Database (Firebase RTDB) is a NoSQL database management system maintained by Firebase over the cloud. The images of unmasked people which were collected earlier for the mask detection will be uploaded to the cloud storage. FCM (Firebase Cloud Messaging) connects the server and clients such that receiving and sending alert notifications/messages is simple and dependable. We'll use FCM to alert a relevant authority using the Firebase admin SDK. A notification will be sent to the application using the Firebase admin SDK. A custom notification will be sent to the concerned authority recognizing the details of the person without mask. When a person without a mask is identified, the application will be notified. The application will be developed in Flutter as it provides better user experience, robust performance and enormous time and effort saving.

4.2.4 Mask Check Application

After face classification into masked and unmasked faces, the unmasked faces are then passed to the python face-recognition library. This library converts the unmasked faces into face encodings and compares it with the faces registered in the known dataset. From this the unmasked face is recognized and the name of the person is catched. This parameter is used to fetch the data of the person who was caught unmasked from the firebase firestore database using firebase python admin sdk. The Firebase Admin SDK is a set of server libraries that lets you interact with Firebase from privileged environments. The Admin SDK for Python provides APIs for authentication, user management, Realtime Database, Firestore, Firebase Cloud Messaging (FCM) and more.

Mask Check App is developed using Flutter framework. This app will be used by the authority or the admin who wants the notification alerts of the unmasked people caught in the video footage. Mask Check app starts with the Registration/Sign up Screen for the authority. If already signed up, you can login via Login Screen. Email Password authentication is done via FirebaseAuth. After authentication, the user is redirected to the MainScreen where all the alert notification history is displayed. The AppBar has the option to add data entries. Via add entries, authority/admin can create the user profiles of his community, colleagues, staff etc wherever this project is being deployed. Add user profiles screen consist of a form consisting of a profile picture, name, email id, phone, moodle id (unique id), designation. This data is then stored in the Firebase Firestore database in the users collection.

When any person is caught unmasked in the area covered in the CCTV, he is recognized using python face-recognition library. After recognizing who that unmasked person is, the data of the user is fetched on the server from the firestore database using admin SDK. From here, a Push Notification Alert is generated using Firebase Cloud Messaging consisting of the data of the user, and the time when he/she was caught unmasked. This notification is sent to the App and is also saved in the firestore database in the ‘notifications’ collection. This push notification appears in the notification bar of the phone, if the app is minimized or not opened; and if the app is opened, then appears on the MainScreen of the appin the notification history. Notification contains all the details of the caught unmasked person, viz. Name, profile picture, email, phone, moodle id, designation and the time when he/she was caught. If you return back to the MainScreen, here all the notifications are displayed in a ListView widget using a StreamBuilder. The data Stream is the notification details that were saved into the firestore database using admin SDK. This stream is listened to on the MainScreen for displaying notification history. You can view the history of people caught unmasked when you were offline. You can tap on the ListTile to view the profile details of the person who was caught.

4.3 UML Diagrams

4.3.1 Use Case Diagram

Use case diagrams are used to gather the requirements of a system including internal and external influences. These requirements are mostly design requirements. Hence, when a system is analyzed to gather its functionalities, use cases are prepared and actors are identified. In this use case diagram, we have defined what use cases will be available for the primary actor like the user. And the use cases need a secondary actor like FirebaseAuth and Database in Fig 4.6. Some use cases have included use cases or extended use cases e.g. while login, App needs to verify the email and password details before directing the user into the application. So, the login use case must have verify but it may not be necessary to have a login error use case every time. Some use cases need to be performed on the secondary actor side like for verification app sends the request to firebase for authentication. The user can log in, Register, Add or Delete user entries, View notifications and update user profile.

Figure 4.5: Use Case Diagram of Face Mask Detection

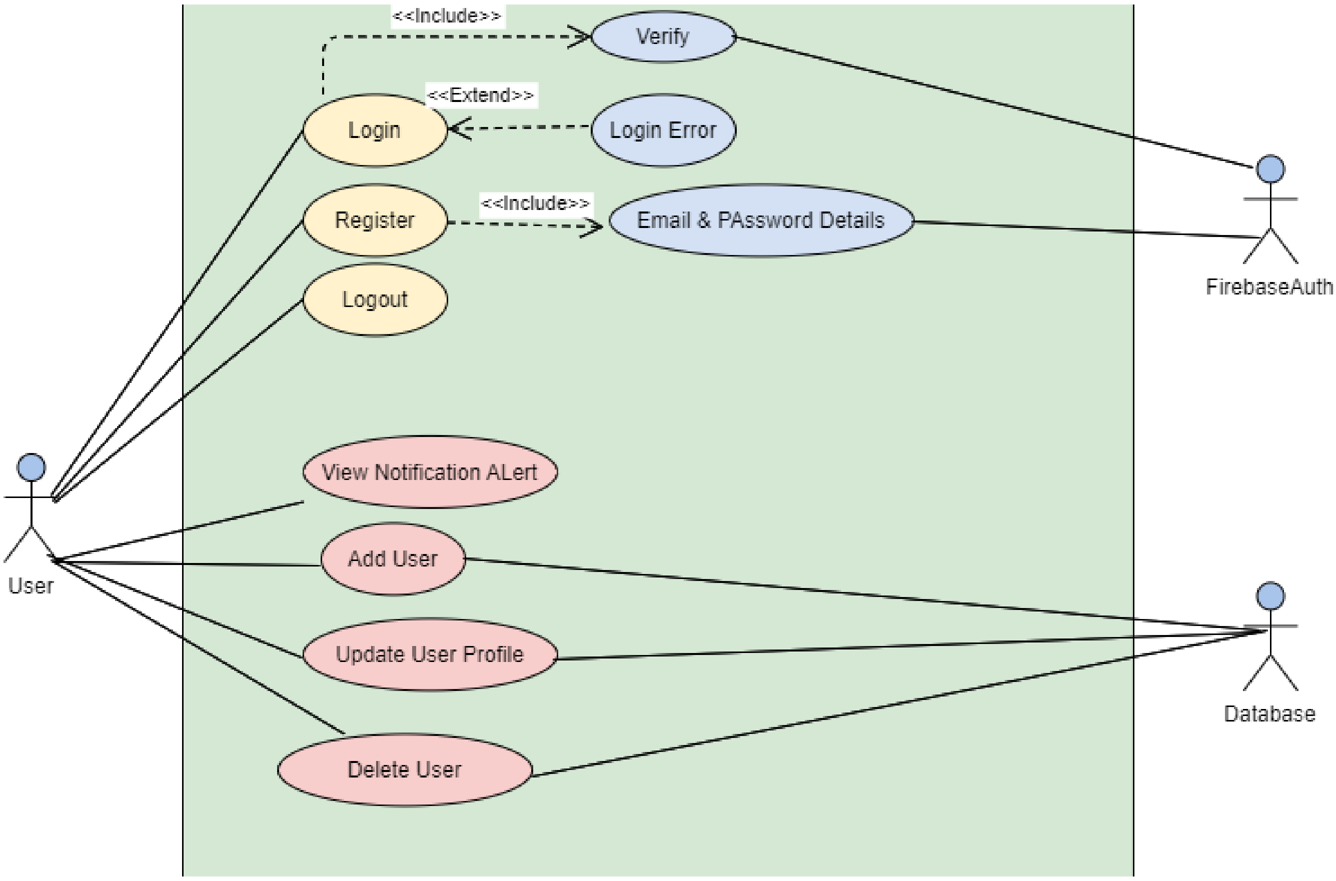


Figure 4.6: Use Case Diagram of Mask Check Application

4.3.2 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc. In this activity diagram, we will discuss the face mask detection system flow. It starts with capturing real time video using webcam/CCTV. In the next step, the system will capture images and detect faces. If the face is unmasked then it will recognize the face, get the data of that recognized person from firestore and push alert notification on application.

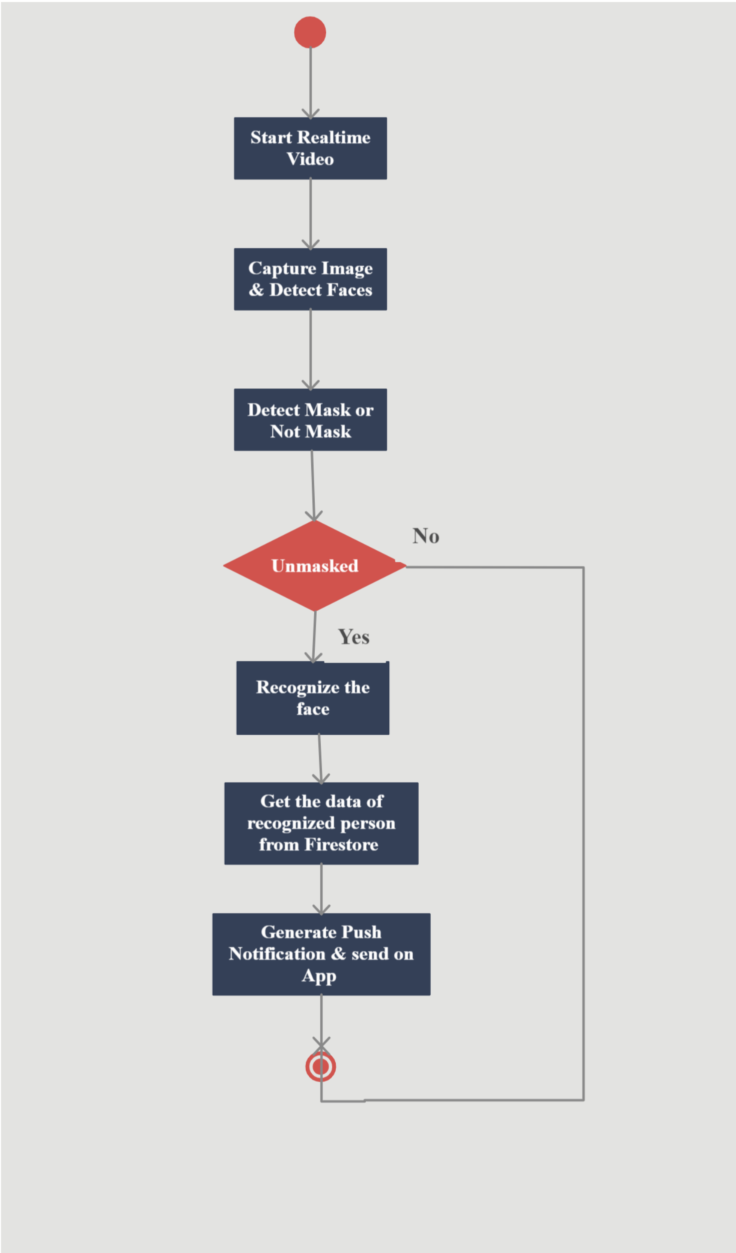


Figure 4.7: Activity Diagram of Face Mask Detection System

Chapter 5

Project Implementation

```python
import ...

INIT_LR = 1e-4
EPOCHS = 20
BS = 32

DIRECTORY = r"/content/drive/MyDrive/dataset/dataset"
CATEGORIES = ["with_mask", "without_mask"]

# grab the list of images in our dataset directory, then initialize
# the list of data (i.e., images) and class images
print("[INFO] loading images...")

data = []
labels = []

for category in CATEGORIES:...

# perform one-hot encoding on the labels
lb = LabelBinarizer()
labels = lb.fit_transform(labels)
labels = to_categorical(labels)

data = np.array(data, dtype="float32")
labels = np.array(labels)

(trainX, testX, trainY, testY) = train_test_split(data, labels,
    test_size=0.20, stratify=labels, random_state=42)

# construct the training image generator for data augmentation
aug = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.15,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.15,
    horizontal_flip=True,
    fill_mode="nearest")

# load the MobileNetV2 network, ensuring the head FC layer sets are
# left off
baseModel = MobileNetV2(weights="imagenet", include_top=False,
    input_tensor=Input(shape=(224, 224, 3)))

# construct the head of the model that will be placed on top of the
# the base model
```

Figure 5.1: Code to train MobilNetV2 model

```python
headModel = baseModel.output
headModel = AveragePooling2D(pool_size=(7, 7))(headModel)
headModel = Flatten(name="flatten")(headModel)
headModel = Dense(128, activation="relu")(headModel)
headModel = Dropout(0.5)(headModel)
headModel = Dense(2, activation="softmax")(headModel)

# place the head FC model on top of the base model (this will become
# the actual model we will train)
model = Model(inputs=baseModel.input, outputs=headModel)

# loop over all layers in the base model and freeze them so they will
# *not* be updated during the first training process
for layer in baseModel.layers:
    layer.trainable = False

# compile our model
print("[INFO] compiling model...")
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss="binary_crossentropy", optimizer=opt,
    metrics=["accuracy"])

# train the head of the network
print("[INFO] training head...")
H = model.fit(
    aug.flow(trainX, trainY, batch_size=BS),
    steps_per_epoch=len(trainX) // BS,
    validation_data=(testX, testY),
    validation_steps=len(testX) // BS,
    epochs=EPOCHS)

# make predictions on the testing set
print("[INFO] evaluating network...")
predIdxs = model.predict(testX, batch_size=BS)

# for each image in the testing set we need to find the index of the
# label with corresponding largest predicted probability
predIdxs = np.argmax(predIdxs, axis=1)

# show a nicely formatted classification report
print(classification_report(testY.argmax(axis=1), predIdxs,
    target_names=lb.classes_))

# serialize the model to disk
print("[INFO] saving mask detector model...")
model.save("mask_detector.model", save_format="h5")
```

Figure 5.2: Code to train MobileNetV2 model

```python
import ...

token = [
    "cVM3iyH7QmK2zQxJczHomf:APA91bGZ5eOMkYeGyoIw36RsDG6L__Ouv68VakxWm7KiF0FVUWKUyhX3ziA-0o0D7Bulc_DWeZ_rQnMwnDZCn6MxaR9ahnggj0EnNBfRYqcOPKG4dwsYTgLcF_hx9mHqpw0d7bCuUdDk",
    "f28_DI-STnmMiGHzWDd6P8:APA91bEMT-2j0tdVajWHtu3WgKlTD7WMB-43I80W85qrDoeRmxuFz-N0_gEpgPVAtdfXi8RfKMRHgJGPLDzbsDlXqMMWK3Jtcg0clFdc3FUUznis2mkUZKUA-ilsLODY6Du6GXcrMiHb"]

cred = credentials.Certificate("C:/Users/harsh/Downloads/mask-check-d53b8-firebase-adminsdk-rhycc-851b989cb1.json")
firebase_admin.initialize_app(cred)
db = firestore.client()
users_ref = db.collection('users')
notifications_ref = db.collection('notifications')
notification_data = {
    'name': 'name',
    'profilepicurl': 'url',
    'email': 'email',
    'moodle id': 'moodle id',
    'phone': 'phone',
    'designation': 'designation',
    u'time': datetime.datetime.now(tz=datetime.timezone.utc),
    'text': 'was seen without a mask in college premises on'
}


def sendPush(title, msg, img, registration_token, dataObject):...


def sendPushAndUpdate(title, msg, img, registration_token, dataObject, notification_data):...


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, (224, 224),
                                 (104.0, 177.0, 123.0))

    # pass the blob through the network and obtain the face detections
    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    # initialize our list of faces, their corresponding locations, and the list of predictions from our face mask
    # network
    faces = []
    locs = []
    preds = []
```

Figure 5.3: Code for detection and recognition of a face mask

```python
    # 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 > 0.5:
            # 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]
            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=32)

    # return a 2-tuple of the face locations and their corresponding
    # locations
    return (locs, preds)


# load our serialized face detector model from disk
prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
```

Figure 5.4: Code for detection and recognition of a face mask

```python
maskNet = load_model("mask_detector.model")

image1 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/saloni.jpeg"))
image1_face_encoding = face_recognition.face_encodings(image1)[0]

image2 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/harsh.jpg"))
image2_face_encoding = face_recognition.face_encodings(image2)[0]

image3 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/praju.jpeg"))
image3_face_encoding = face_recognition.face_encodings(image3)[0]

image4 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/gitty.jpeg"))
image4_face_encoding = face_recognition.face_encodings(image4)[0]

image5 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/kiransir.jpg"))
image5_face_encoding = face_recognition.face_encodings(image5)[0]

image6 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/kaustubh.jpeg"))
image6_face_encoding = face_recognition.face_encodings(image6)[0]

image7 = face_recognition.load_image_file(
    os.path.abspath("C:/Users/harsh/PycharmProjects/Face-Mask-Detection/recognize/images/vishalsir.png"))
image7_face_encoding = face_recognition.face_encodings(image7)[0]

known_face_encodings = [
    image1_face_encoding,
    image2_face_encoding,
    image3_face_encoding,
    image4_face_encoding,
    image5_face_encoding,
    image6_face_encoding,
    image7_face_encoding
]
known_face_names = [
    "Saloni Rane",
    "Harsh Saraiya",
    "Prajakta Mhaske",
    "Saurav Hiwanj",
    "Prof Kiran Deshpande",
```

Figure 5.5: Code for detection and recognition of a face mask

```python
face_locations = []
face_encodings = []
face_names = []
process_this_frame = True
total_faces = []
total_faces_list = []
dictkey = ('Name')

print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()

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=800)

    # 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)

    # 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)

        if label == "No Mask":
            if process_this_frame:
                face_locations = face_recognition.face_locations(frame)
                # print(face_locations)
                face_encodings = face_recognition.face_encodings(frame, face_locations)
                # print(face_encodings)
                face_names = []
                for face_encoding in face_encodings:
                    matches = face_recognition.compare_faces(known_face_encodings, face_encoding, tolerance=0.6)
                    name = "Unknown"
                    face_distances = face_recognition.face_distance(known_face_encodings, face_encoding)
                    best_match_index = np.argmin(face_distances)
                    if matches[best_match_index]:
                        name = known_face_names[best_match_index]
                    face_names.append(name)
```

Figure 5.6: Code for detection and recognition of a face mask

```python
                face_names.append(name)
                print("Face detected -- {}".format(face_names))
                total_faces.extend(face_names)
                total_faces_list = list(dict.fromkeys(total_faces))
                # print(total_faces_list)

        process_this_frame = not process_this_frame

        # 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)

    for i in total_faces:
        result = users_ref.where("name", "==", i).get()
        for j in result:
            output = j.to_dict()
            print(output)
            notification_data.update({'name': output.get("name"), 'email': output.get("email"),
                                      'moodle id': output.get("moodle id"), 'phone': output.get("phone"),
                                      'designation': output.get("designation"),
                                      'profilepicurl': output.get("profilepicurl")})
            sendPushAndUpdate("ALERT", "{} was caught without a mask!".format(output.get("name")),
                              output.get("profilepicurl"),
                              token, output, notification_data)

    # show the output frame
    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1) & 0xFF

    # if the `q` key was pressed, break from the loop
    if key == ord("q"):
        break

# do a bit of cleanup
cv2.destroyAllWindows()
vs.stop()
```

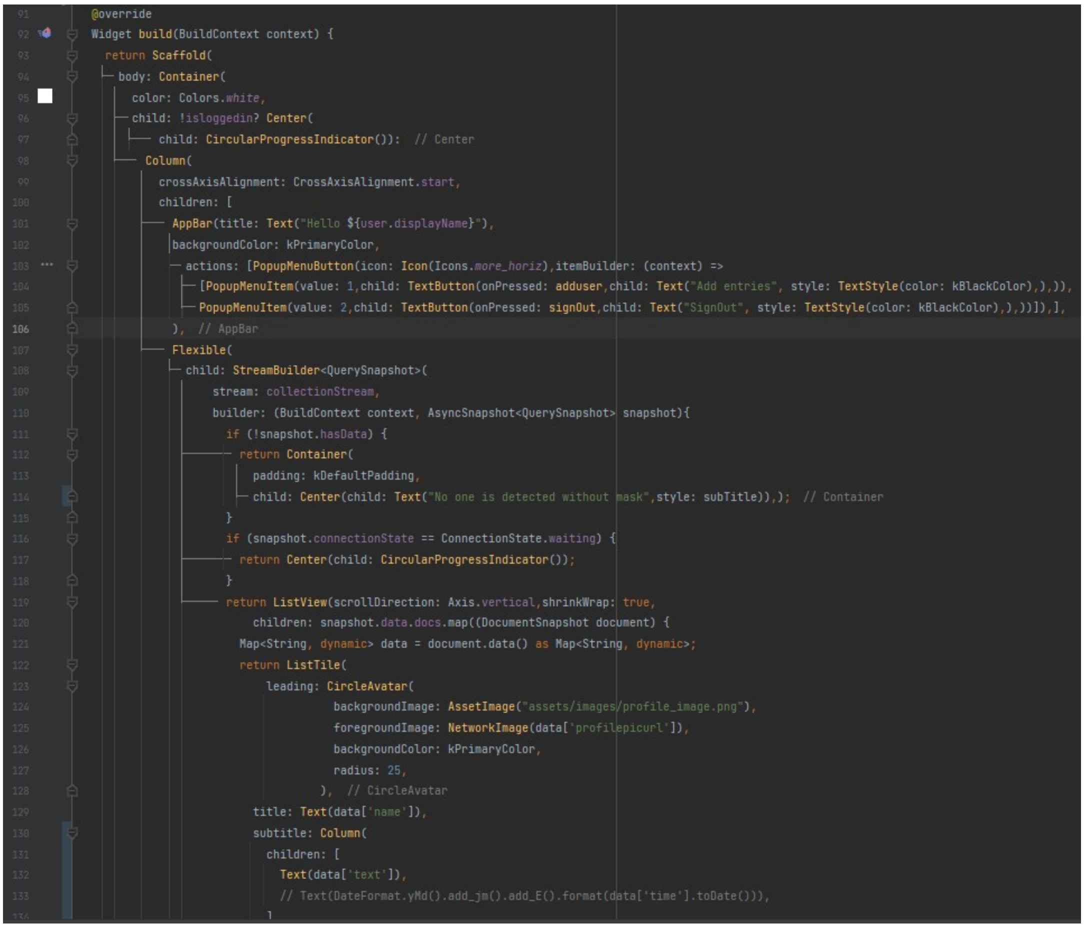
Figure 5.7: Code for detection and recognition of a face mask

```
import ...

class Mainpage extends StatefulWidget{
  @override
  _MainScreen createState() => _MainScreen();
}

class _MainScreen extends State<Mainpage>{

  final FirebaseAuth _auth = FirebaseAuth.instance;
  User user;
  bool isloggedin= false;
  Stream collectionStream = FirebaseFirestore.instance.collection('notifications').orderBy('time', descending: true).snapshots();

  checkAuthentication() async{

    _auth.authStateChanges().listen((user) {
      if (user == null) {
        Navigator.push(context, MaterialPageRoute(builder: (context)=> Login()));      }
    });
  }

  getUser() async {
    User firebaseUser = _auth.currentUser;
    await firebaseUser?.reload();
    firebaseUser = _auth.currentUser;

    if (firebaseUser != null) {
      setState(() {
        this.user = firebaseUser;
        this.isloggedin = true;
      });
    }
  }

  signOut() async {
    _auth.signOut();

    // final googleSignIn = GoogleSignIn();
    // await googleSignIn.signOut();
  }
```

Figure 5.8: Code for Home Screen

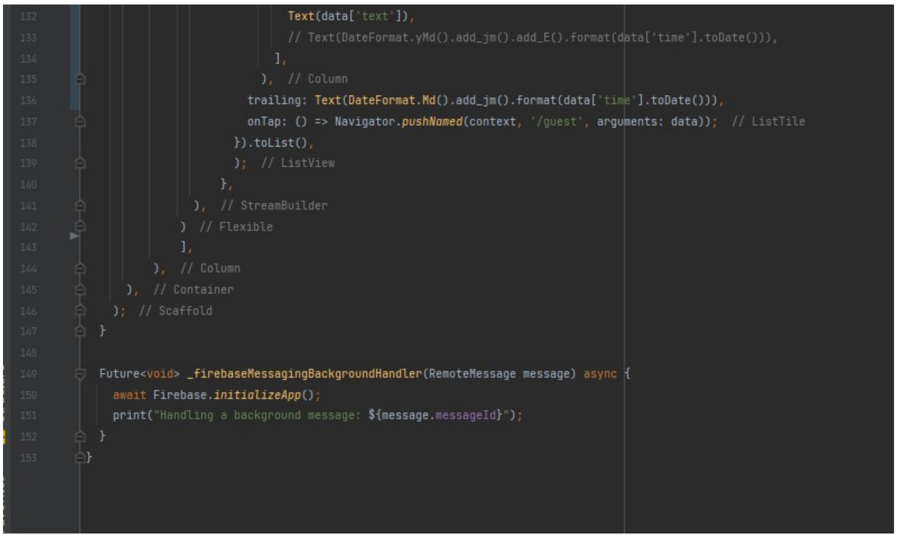
```
adduser() async {
  Navigator.push(context, MaterialPageRoute(builder: (context)=> AddUser()));
}

catchData() async {
  FirebaseMessaging.onMessage.listen((RemoteMessage message)
  {
    print("message received while on foreground");
    print('Message data: ${message.data}');
    print(message.notification.body);
  });

  FirebaseMessaging.onBackgroundMessage(_firebaseMessagingBackgroundHandler);
}

@override
void initState() {
  super.initState();
  this.checkAuthentication();
  this.getUser();
  FirebaseMessaging messaging = FirebaseMessaging.instance;
  messaging.getToken().then((value){
    print(value);
  });
  FirebaseMessaging.onMessage.listen((RemoteMessage message) {
    print("message received while on foreground");
    print('Message data: ${message.data}');
    print(message.notification.body);
  });

  FirebaseMessaging.onMessageOpenedApp.listen((message) async{
    var received = message.data;
    var time = message.sentTime;
    Navigator.pushNamed(context, '/guest', arguments: received);
  });
}
```

Figure 5.9: Code for Home Screen



```dart
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      color: Colors.white,
      child: !isloggedin? Center(
        child: CircularProgressIndicator()): // Center
      Column(
        crossAxisAlignment: CrossAxisAlignment.start,
        children: [
          AppBar(title: Text("Hello ${user.displayName}"),
          backgroundColor: kPrimaryColor,
          actions: [PopupMenuButton(icon: Icon(Icons.more_horiz),itemBuilder: (context) =>
            [PopupMenuItem(value: 1,child: TextButton(onPressed: adduser,child: Text("Add entries", style: TextStyle(color: kBlackColor),),)),
            PopupMenuItem(value: 2,child: TextButton(onPressed: signOut,child: Text("SignOut", style: TextStyle(color: kBlackColor),),))]),],
          ), // AppBar
          Flexible(
            child: StreamBuilder<QuerySnapshot>(
                stream: collectionStream,
                builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot){
                  if (!snapshot.hasData) {
                    return Container(
                      padding: kDefaultPadding,
                      child: Center(child: Text("No one is detected without mask",style: subTitle)),); // Container
                  }
                  if (snapshot.connectionState == ConnectionState.waiting) {
                    return Center(child: CircularProgressIndicator());
                  }
                  return ListView(scrollDirection: Axis.vertical,shrinkWrap: true,
                      children: snapshot.data.docs.map((DocumentSnapshot document) {
                    Map<String, dynamic> data = document.data() as Map<String, dynamic>;
                    return ListTile(
                        leading: CircleAvatar(
                              backgroundImage: AssetImage("assets/images/profile_image.png"),
                              foregroundImage: NetworkImage(data['profilepicurl']),
                              backgroundColor: kPrimaryColor,
                              radius: 25,
                            ), // CircleAvatar
                      title: Text(data['name']),
                      subtitle: Column(
                        children: [
                          Text(data['text']),
                          // Text(DateFormat.yMd().add_jm().add_E().format(data['time'].toDate())),
                        ]
```

Figure 5.10: Code for Home Screen



```dart
                          Text(data['text']),
                          // Text(DateFormat.yMd().add_jm().add_E().format(data['time'].toDate())),
                        ],
                      ), // Column
                    trailing: Text(DateFormat.Md().add_jm().format(data['time'].toDate())),
                    onTap: () => Navigator.pushNamed(context, '/guest', arguments: data)); // ListTile
                  }).toList(),
                  ); // ListView
                },
            ), // StreamBuilder
          ) // Flexible
        ],
      ), // Column
    ), // Container
  ); // Scaffold
}

Future<void> _firebaseMessagingBackgroundHandler(RemoteMessage message) async {
  await Firebase.initializeApp();
  print("Handling a background message: ${message.messageId}");
}
}
```

Figure 5.11: Code for Home Screen

```
import ...
class Guest extends StatefulWidget{
  @override
  GuestProfile createState() => GuestProfile();
}
class GuestProfile extends State<Guest> {
  void initState() {
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    Map<String, dynamic> rcvdData = ModalRoute.of(context).settings.arguments;
    return Scaffold(
      body: Container(
        padding: EdgeInsets.zero,
        child: Column(
          children: [
            AppBar(title: Text("Notification Alert"),
              backgroundColor: kPrimaryColor,),  // AppBar
            Flexible(
              child: Container(
                color: kPrimaryColor,
                padding: EdgeInsets.fromLTRB(125, 30, 125, 50),
                child: CircleAvatar(foregroundImage: NetworkImage(rcvdData['profilepicurl']),
                  radius: 75,
                ),  // CircleAvatar
              ),  // Container
            ),  // Flexible
            SizedBox(
              height: 20,
            ),  // SizedBox
            Padding(
              padding: kDefaultPadding,
```

Figure 5.12: Code for Notification Screen

```
        child: Text(
          '${rcvdData['name']}',
          style: primary,
        ), // Text
      ), // Padding
      Text("was caught without wearing a mask."),
      SizedBox(
        height: 20,
      ), // SizedBox
      Card(
        margin: EdgeInsets.fromLTRB(20, 0, 20, 0),
        shadowColor: Colors.black,
        elevation: 5,
        shape: RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)),
        child: Column(
          children: [
            ListTile(
              leading: Text("Designation", textAlign: TextAlign.left,),
              title: Text('${rcvdData['designation']}',textAlign: TextAlign.right,),
            ), // ListTile
            Divider(indent:10,endIndent: 10,),
            ListTile(
              leading: Text("Moodle ID", textAlign: TextAlign.left,),
              title: Text('${rcvdData['moodle id']}',textAlign: TextAlign.right,),
            ), // ListTile
            Divider(thickness: 1,indent:10,endIndent: 10,),
            ListTile(
              leading: Text("Email", textAlign: TextAlign.left,),
              title: Text('${rcvdData['email']}',textAlign: TextAlign.right,),
            ), // ListTile
            Divider(thickness: 1,indent:10,endIndent: 10,),
            ListTile(
              leading: Text("Phone", textAlign: TextAlign.left,),
```

Figure 5.13: Code for Notification Screen

```
                    leading: Text("Phone", textAlign: TextAlign.left,),
                    title: Text('${rcvdData['phone']}',textAlign: TextAlign.right,),
                  ), // ListTile
                ],
              ), // Column
            ), // Card
            SizedBox(
              height: 30,
            ), // SizedBox
          ],
        ), // Column
      ), // Container
    ); // Scaffold
  }
}
```

Figure 5.14: Code for Notification Screen

Chapter 6

Testing

6.1 Functional Testing

6.1.1 Unit Testing

Unit testing is the first level of testing and is often performed by the developers themselves. It is the process of ensuring individual components of a piece of software at the code level are functional and work as they were designed to. Developers in a test-driven environment will typically write and run the tests before the software or feature is passed over to the test team. Unit testing can be conducted manually. Unit testing will also make debugging easier because finding issues earlier means they take less time to fix than if they were discovered later in the testing process.

The Unit testing is best suited for our application development phase. In this phase, we started to code in units to create different modules. And test each module separately, like the login page, register page, home page, notification page etc. All these pages are tested and debugged before going further integrating. And check whether we are getting the desired output from each module as for the objectives.

6.1.2 Integration Testing

After each unit is thoroughly tested, it is integrated with other units to create modules or components that are designed to perform specific tasks or activities. These are then tested as group through integration testing to ensure whole segments of an application behave as expected (i.e, the interactions between units are seamless). These tests are often framed by user scenarios, such as logging into an application or opening files. Integrated tests can be conducted by either developers or independent testers and are usually comprised of a combination of automated functional and manual tests.

As we have discussed unit testing the next step is integration testing. All the units which we have tested and debugged are now ready to integrate into a whole single module. The integration part is crucial as we need to know which unit must interact without error, calling them in a different class accessing the instance of that class. So accordingly, modules are integrated and checked whether they behave as for the objectives.

6.2 Non Functional Testing

6.2.1 Usability Testing

Usability testing is a testing method that measures an application's ease of use from the end-user perspective and is often performed during the system or acceptance testing stages. The goal is to determine whether or not the visible design and aesthetics of an application meet the intended workflow for various processes, such as logging into an application. Usability testing is a great way for teams to review separate functions, or the system as a whole is intuitive to use.

6.2.2 Compatibility Testing

Compatibility testing is used to gauge how an application or piece of software will work in different environments. It is used to check that your product is compatible with multiple operating systems, platforms, browsers, or resolution configurations. The goal is to ensure that your software's functionality is consistently supported across any environment you expect your end-users to be using.

The framework we are using to develop our application is Flutter. It is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. We make sure that our application is compatible with Android operating systems. The features we developed are perfectly run in multiple operating systems without an error. For this reason, compatibility testing is best suited for our project.

6.3 Test Cases

Test Case	Description	Expected Output	Actual Output	Pass/Fail
User Login	User enter the correct login	Main page will be displayed	Main page displayed	Pass
User Login	User not enter a login	Message "Please enter the Email ID and Password" will be displayed	Message "Please enter the Email and Password" is displayed	Pass
User Login	User enter an incorrect login	Message "Invalid Email ID or password" will be displayed.	Message "Invalid Email ID or password" is displayed.	Pass
Sign Up	User enter Username, Email address and password	Main page of mask check application will be displayed.	Main page of mask check application is displayed.	Pass
Sign Up	User enter an invalid or incomplete information	Message "Please enter the correct information" will be displayed.	Message "Please enter the correct information" is displayed.	Pass
Home Page	Null	The system need to show the home page i.e. notification history to user.	Home page runs on the user screen	Pass
Add User Entries	Add correct details of user such as name, phone number, email and moodle Id.	Message "User added successfully" will be displayed.	Message "User added successfully" is displayed.	Pass
Add User Entries	User enter an invalid or incomplete information	Message "Please enter the correct information" will be displayed.	Message "Please enter the correct information" is displayed.	Pass
Notification Information	Null	By clicking on notification, unmasked person's information should display.	Displays unmasked person's information, by clicking on notification	Pass

Chapter 7

Result

7.1 Face Mask Detection

Facial mask detection system have been implemented successfully through live video streaming. Our system detects people who are wearing and not wearing a face mask in live video footage as shown in the below snapshots. This system can detect multiple masked and unmasked persons at a time.

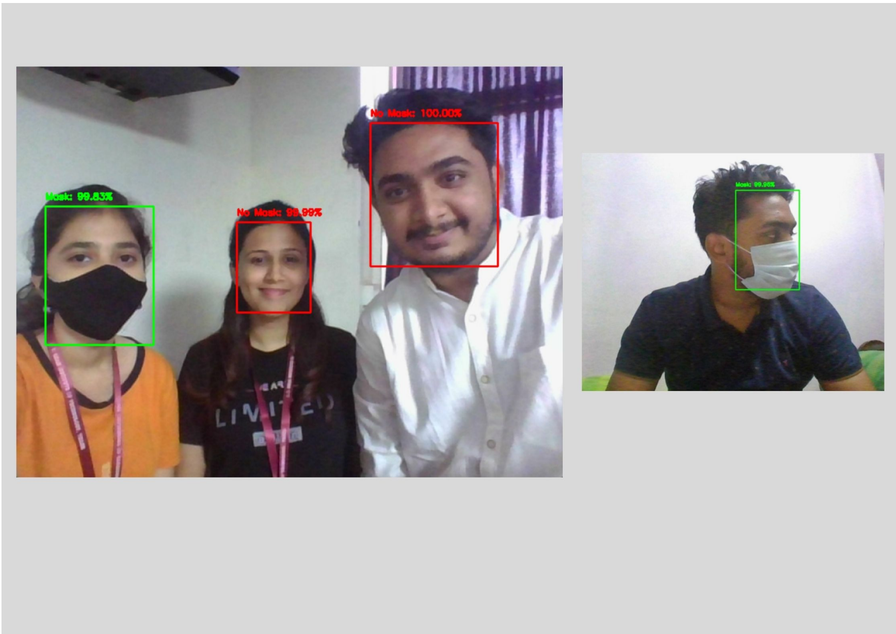


Figure 7.1: Snapshots of face mask detection

7.2 SignUP & Login Screen

We designed a simplistic UI, especially for admin/concerned authority, when a user interacts with an application for the first time, it is more attractive and user friendly. When a user interacts with the application for the first time can register and log in into the application. The user will be able to access the home page after logging in.

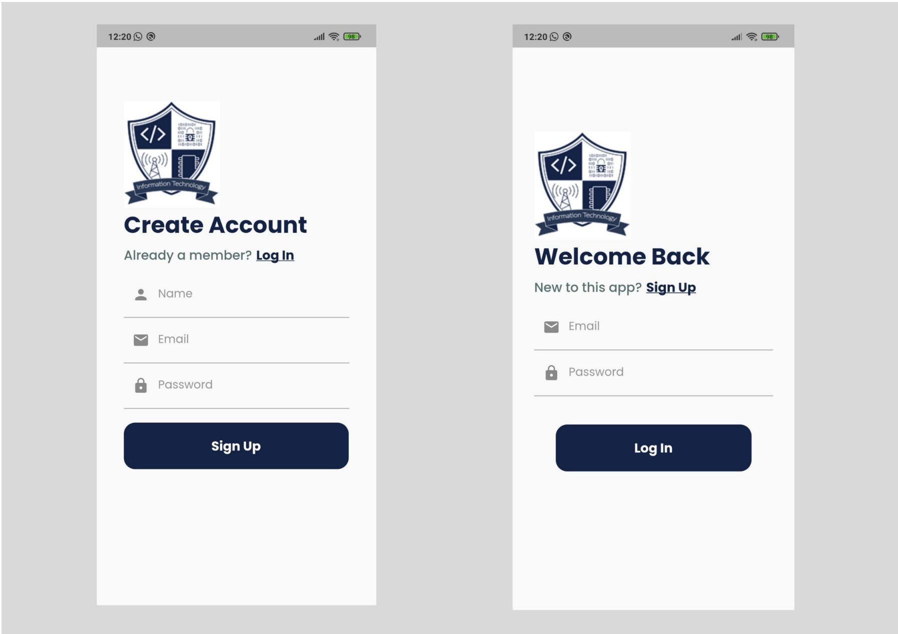


Figure 7.2: Snapshots of signup and login pages

7.3 Home Screen

After admin logs into the application, the home screen will be displayed as shown in the below snapshot. Here, the admin can add new user entries by filling required user information.

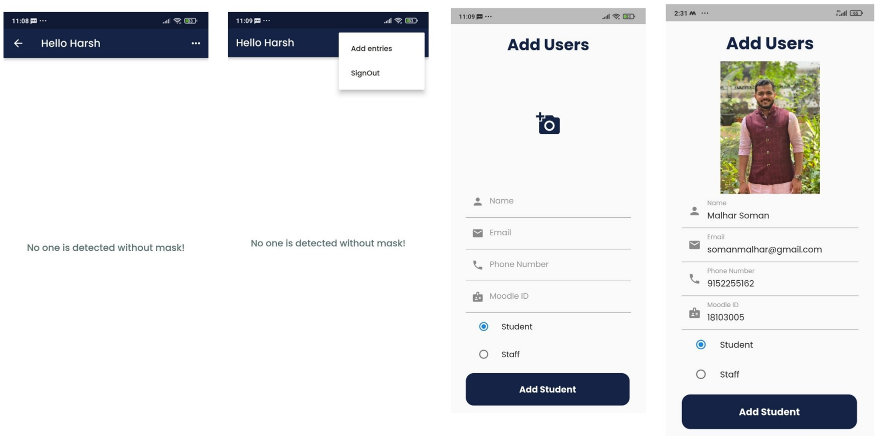


Figure 7.3: Snapshots of home screen

7.4 Notification Screen

When a person is caught unmasked then the admin will be notified on the application with unmasked person's details. Admin can also keep record of unmasked people caught throughout the day with time by viewing the notification history page.

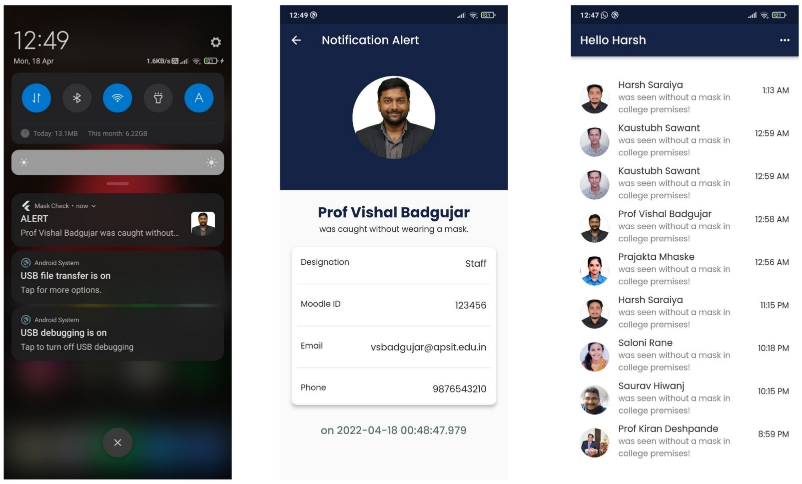


Figure 7.4: Snapshots of notification screen

Chapter 8

Conclusions and Future Scope

As the technology is developing with emerging trends, therefore the novel face mask detector using deep learning approach for detecting face masked people from real-time videos in public places is proposed using Convolution Neural Network MobileNetV2 classifier to curtail the community spread of Coronavirus. To select this base model, we evaluated metrics such as accuracy, precision, recovery and selected the MobileNetV2 architecture with the best performance with 96accuracy and 99% recall. A face mask detection architecture is included in the system, which employs a deep learning algorithm to recognise the mask on the user's face. The model is trained on a real dataset of masked and unmasked facial photos. This face mask detector will act as a valuable tool that can be used in many areas such as shopping malls, airports and other high-traffic locations to keep an eye on the public and prevent the spread of Virus by checking who is and who is not following the basic rules and informing the concerned authority. This collaborative technique not only aids in reaching high accuracy, but it also significantly improves detection speed.

Finally, the work opens interesting future directions for researchers. Firstly, the proposed technique can be integrated into an automatic door opening system in malls and offices. Secondly, it can be automated by using drones and robot technology to take action instantly.

Bibliography

[1] A. S. Joshi, S. S. Joshi, G. Kanahasabai, R. Kapil and S. Gupta, "Deep Learning Framework to Detect Face Masks from Video Footage," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), 2020, pp. 435-440, doi: 10.1109/CICN49253.2020.9242625.

[2] Mamata S.Kalas,"REAL TIME FACE DETECTION AND TRACKING USING OPENCV", International Journal of Soft Computing And Artificial Intelligence (IJSCAI) , pp. 41-44, Volume-2,Issue-1

[3] M. M. Rahman, M. M. H. Manik, M. M. Islam, S. Mahmud and J. -H. Kim, "An Automated System to Limit COVID-19 Using Facial Mask Detection in Smart City Network," 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), 2020, pp. 1-5, doi: 10.1109/IEMTRONICS51293.2020.9216386.

[4] S. Abbasi, H. Abdi and A. Ahmadi, "A Face-Mask Detection Approach based on YOLO Applied for a New Collected Dataset," 2021 26th International Computer Conference, Computer Society of Iran (CSICC), 2021, pp. 1-6, doi: 10.1109/CSICC52343.2021.9420599.

[5] O. Cakiroglu, C. Ozer and B. Gunsel, "Design of a Deep Face Detector by Mask R-CNN," 2019 27th Signal Processing and Communications Applications Conference (SIU), 2019, pp. 1-4, doi: 10.1109/SIU.2019.8806447.

[6] W. Vijitkunsawat and P. Chantngarm, "Study of the Performance of Machine Learning Algorithms for Face Mask Detection," 2020 - 5th International Conference on Information Technology (InCIT), 2020, pp. 39-43, doi: 10.1109/InCIT50588.2020.9310963.

[7] M. Xu, H. Wang, S. Yang and R. Li, "Mask wearing detection method based on SSD-Mask algorithm," 2020 International Conference on Computer Science and Management Technology (ICCSMT), 2020, pp. 138-143, doi: 10.1109/ICCSMT51754.2020.00034.

[8] face-recognition. (2020, February 20). PyPI. https://pypi.org/project/face-recognition/

[9] Bhandary, P. (n.d.). GitHub - prajnasb/observations. GitHub. https://github.com/prajnasb/observations

Appendices

Appendix-A: Installing Libraries

1. pip install tensorflow
Among the various open-source platforms, TensorFlow stands amongst the most widely and well-versed tools for machine learning and deep learning. With its various features, it retains the versatility to operate in different use case scenarios such as image recognition, voice recognition, video detection, etc.

2. pip install keras
Keras is a minimalist Python library for deep learning that can run on top of Theano or TensorFlow. It was developed to make implementing deep learning models as fast and easy as possible for research and development.

3. pip install imutils
Imutils is a package based on OpenCV, which can call the opencv interface more simply. It can easily realize a series of operations such as image translation, rotation, scaling, skeletonization and so on.

4. pip install computer-vision
A computer vision library is basically a set of pre-written code and data to build or optimize a computer program. The libraries are numerous tailored to specific needs or programming languages.

5. pip install numpy
NumPy can be used to perform a wide variety of mathematical operations on arrays. It adds powerful data structures to Python that guarantee efficient calculations with arrays and matrices and it supplies an enormous library of high-level mathematical functions that operate on these arrays and matrices.

6. pip install firebase_admin
The Firebase Admin SDKs provide developers with programmatic, second-party auth access to Firebase services from trusted environments. The Admin SDKs are meant to complement the existing Firebase web and mobile clients which provide third-party, end-user access to Firebase services on client devices.

Appendix-B: Importing Python Scripts

1. train_mask_detector.py
It accepts our input dataset and fine-tunes MobileNetV2 upon it to create our mask_detector model.

2. detect_mask_video.py
Using your webcam, this script applies face mask detection to every frame in the stream.

Appendix-C: Flutter Download and Installation

1. Download flutter sdk from
https://docs.flutter.dev/development/tools/sdk/releases?tab=windows

2. Place flutter sdk in your desired directory; like /user/Documents.

3. Go to environment variable set path for flutter.

4. Run flutter doctor

5. Download Android Studio from
https://developer.android.com/studio

6. Double click .exe file for installation.

Appendix-D: Packages for App implementation

1. https://pub.dev/packages/flutter_phone_direct_caller

2. https://pub.dev/packages/firebase_auth

Publication

Paper entitled **"Using ML for Facial Mask Detection"** is presented at **"International Conference on Computational Intelligence and Innovative Technologies (ICCIIT-2022)"** by **"Harsh Saraiya, Saloni Rane, Prajakta Mhaske, Kiran Deshpande, Kaushiki Upadhyaya, Nahid Shaikh".**