



Online attendance system based on facial recognition with face mask detection

Muhammad Haikal Mohd Kamil¹ · Norliza Zaini¹ · Lucyantie Mazalan¹ · Afiq Harith Ahamad¹

Received: 29 December 2021 / Revised: 15 September 2022 / Accepted: 6 February 2023 /
Published online: 7 March 2023

© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract

This paper presents an online system for recording attendance based on facial recognition incorporating facial mask detection. The main objective of this project is to develop an effective attendance system based on face recognition and face mask detection, and to provide this service online through a browser interface. This would allow any user to use this system without the need to install special software. They simply need to open the interface of this system in a browser through any terminal. Recording attendance information online allows data to be easily recorded in a centralized online database. Since faces are used as biometric signatures in this project, all users registered in the system will have their profiles loaded with their face-images samples. Initially, before face recognition can be done, the model training phase based on SVM will be carried out, mainly to develop a trained model that can perform face recognition. A set of synthetic data will also be used to train the same model so that it can perform identification for users wearing face masks. The server application is coded in Python and uses the Open-Source Computer Vision (OpenCV) library for image processing. For web interfaces and the database, PHP and MySQL are used. With the integration of Python and PHP scripting programs, the developed system will be able to perform processing on online servers, while being accessible to users through a browser from any terminal. According to the results and analysis, an accuracy of about 81.8% can be achieved based on a pre-trained model for face recognition and 80% for face mask detection.

Keywords Attendance system · Online · Face recognition · Face mask detection · OpenCV · Python

✉ Norliza Zaini
nzaini815@gmail.com

¹ School of Electrical Engineering, College of Engineering, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia

1 Introduction

In any learning program, the full attendance of participants is vital to ensuring that the objectives of the program are achieved. This is closely related to the teaching process in educational institutions, where full attendance is very important to ensure that no student is left behind. Per Jacksi et al., [6], a minimum percentage of class attendance is required in most institutions. However, such policies are often not complied with due to the various challenges faced by current attendance recording methods. The conventional process in recording student attendance is to collect each student's signature manually. Besides wasting time in getting signatures, as well as in the process of preparing the form hardcopy, other visible disadvantages include potential loss or damage to the sheets. Numerous systems have been developed for attendance measuring purpose. The use of clickers and swipe identification cards are among the methods used in tracking student attendance.

In the same context, many projects have been carried out. Among them is the RFID-Based Student Attendance Management System [1]. RFID can be used to record student attendance, the main purpose of which is to reduce the time wasted in recording attendance. There is also an Android-based smart student attendance system [4], which has proposed as a mobile online solution to record attendance faster, cheaper, and with automated attendance reports. Another RFID system integrated with face recognition [5] has also been proposed to track approved and counted students as they enter and leave the classroom. The system as mentioned above is very useful, taking into account the current trend in which Android is increasingly deployed. Other Android-based systems have also been introduced by Sunaryono et al., [14], in which long lines can be avoided in recording previous automated presence processes.

Referring to the validity of the data collected, the methods described above still have loopholes which allow abuse to occur. Since a person's actual presence is very important, an important method that can be used to detect the actual presence is face recognition. Usually, this face recognition process [13] is subject to biometric verification when used for security purposes. Biometric data is unique to each individual and can thus be used to identify each individual. Face recognition several advantages when compared to traditional card recognition, fingertips, or iris recognition. Among them is being able to detect authenticity, user-friendliness, and no need for physical contact. From security to education, every field has begun to adopt biometric technology for identity tracking applications.

Many face recognition applications have been developed, but most must be downloaded and installed first to the user's device before they can be used. Therefore, to avoid the burden of users like this, in this paper we recommend a face recognition system that can be accessed directly without having to download and install new applications. The idea is to develop an online application that runs on a web server and is easily accessible through a web browser from any terminal.

Referring to the latest scenario with the emergence of the Covid-19 global pandemic declared by the WHO [7, 9, 12], there is a serious need for security measures, with face masks being one of the main efforts to overcome this pandemic. To ensure the safety of the public within the scope of our focus, plans or strategies must be made so that we can monitor participants' compliance with these basic safety principles. As this project is carried out due to concerns about the methods used by the organizers or lecturers to keep track of attendance to the program, this proposed system will be implemented to integrate two features, namely face recognition for attendance recording and face mask detection [8] to check for safety

compliance. Concerning features for face mask detection, one existing system identified is the mask detection system [2], which was built based on YOLOv3. This work is very relevant to what we propose in this study considering the norm of wearing a face mask, which is needed during this pandemic season to prevent the spread of the Covid-19 virus. In this study, after taking into account the need to recognize faces and detect masks, as well as the need to optimize time in recording attendance, the face mask detection process will use the same face-recognition mechanism, but with some improvements in terms of data pre-processing.

Based on the issues discussed above, as well as some of the aims we want to achieve, the overall focus of this project is on developing an online attendance recording system based on face recognition and face mask detection. The face recognition process is required to identify the user’s identity for recording attendance, while the detection of the face mask is to identify safety compliance by the user. Incorporating these two features into one system allows the system to detect a person’s identity, regardless of whether they are exposing their entire face or wearing a face mask. The system is developed as a web-based application, which can offer easy access to users via a web browser from any terminal to record their attendance without having to download or install a specific application.

2 Methodology

In this section, the overall design of the system will be presented, together with the flow of activities that take place in the system. Overall, the designed system consists of two main components, as illustrated in Fig. 1, namely the server application and client-side application. The server application will be presented first in the next subsection, followed by the description of the client-side application in the following subsection.

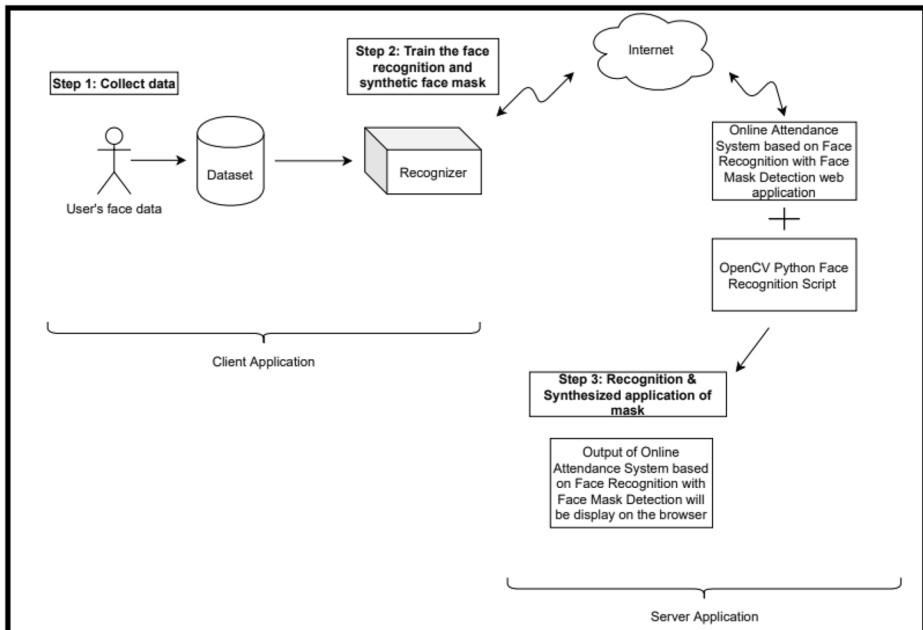


Fig. 1 Overall System Architecture

2.1 Server application based on Python

The first step in the development of this server application is to train a model able to recognize faces. For this purpose, facial images of all users need to be collected and saved for model training purposes. Since this system has a special feature for face recognition behind face masks, two datasets are required. One dataset will contain all the original user faces without wearing face masks, while another dataset is synthetically processed by applying a face mask for each sample image available. Thus, for each individual, there will be two sets of sample images that will be used, namely original face sample images and original images that are synthetically applied with face masks, respectively (refer to Table 1).

10 to 30 original face images were collected for each individual. All these images are saved in a specific folder for each individual. This folder is named according to the user's identity. For example, a folder named 'Haikal' will contain all sample images of an individual named Haikal. As for the synthetic dataset, another folder called Haikal_Mask will contain original Haikal's images that have been synthesized with the application of virtual masks.



The objective of collecting sample images in these two different categories is to train the model for identifying the individuals' faces with and without face masks. This way, if a user performs identification using a face mask, then both identity recognition and detection of mask-wearing will be carried out.

In regard to recognizer model training, a special Python program will be run to carry out the model training process until it can perform face recognition based on the existing dataset. This trained model will be invoked by the Python program which will perform face recognition and provide output in the form of a recognized user's identity or a label. For each user, two possible labels can be identified by the model, i.e. Haikal or Haikal_Mask. If the identification is made in the name of Haikal only, it means that the identified user (Haikal) is not wearing a face mask, while if the label given is Haikal_Mask, this means that the identified user (Haikal) is wearing a mask.

2.2 Client-side application

To use this system, all the user has to do is open the URL of this system through a browser. On the front page of this site, the user will be able to activate the camera to take a selfie. This web

Table 1 Comparison between an original data sample and the generated synthetic data for facial identification and face mask detection

Original Data Sample	Synthetic Data Sample
	

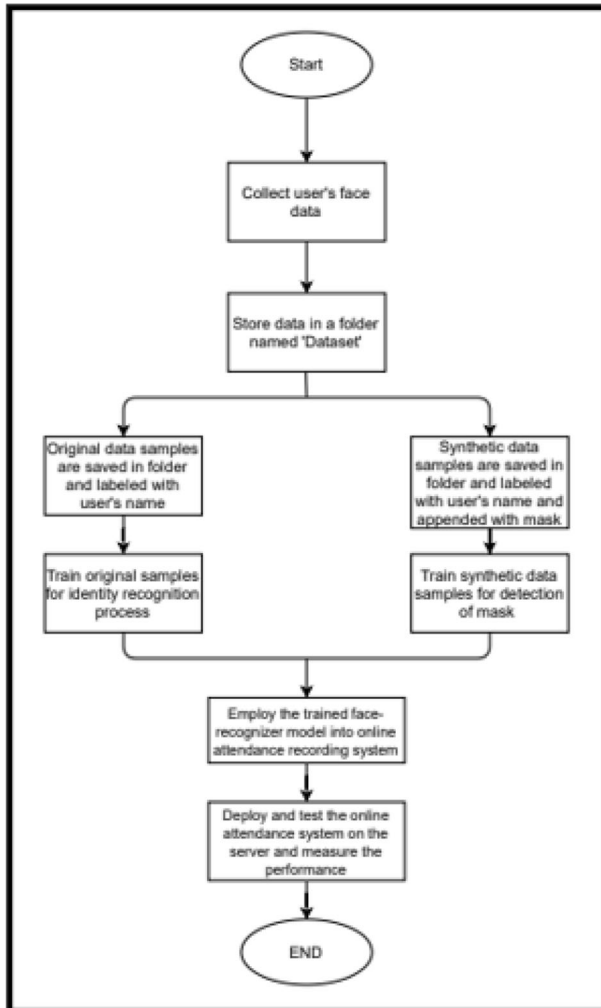
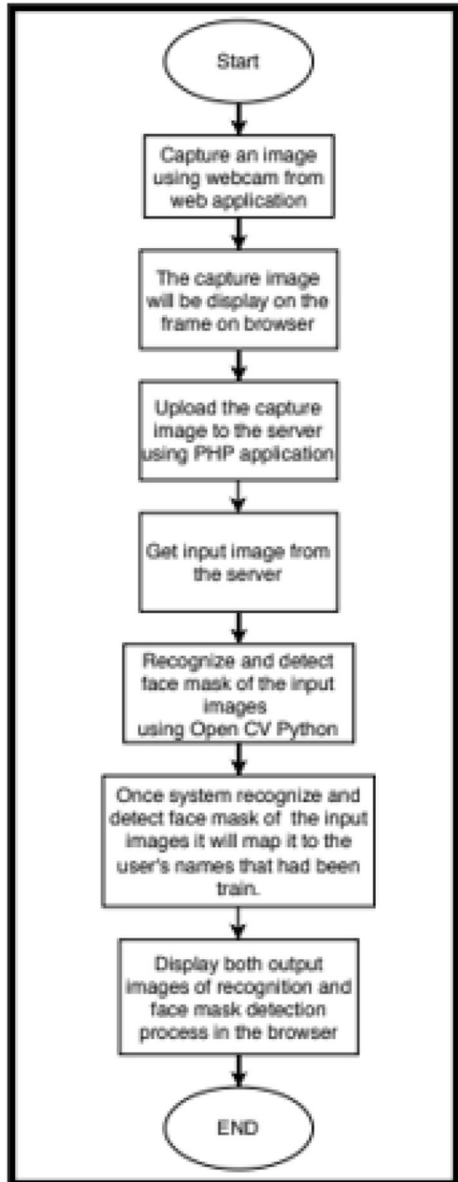


Fig. 2 Face-recognition model training flow chart

page is created using HTML, JavaScript, and CSS that can capture the selfie image and subsequently send the image to the server for processing after a click of a button. The server application that receives the image will activate a Python script that will perform a face recognition process to determine the identity of the face owner. The two main outputs produced by this Python script are the name of the identified face owner and also the status of whether the person is wearing a mask or not. The results will be displayed on the web page in the form of a processed selfie image printed with output labels.

Figure 2 illustrates, in general, the process of how a face recognizer model is trained. Initially, a set of data samples was collected for each individual, which included 10–30 facial images without a face mask. Then, a new collection of synthetic data samples is generated for the same individual by virtually applying a mask on each of the original images. The original data samples are saved in a folder and labeled with the individual's name, while another folder storing the synthetic data samples is named with the individual's name and appended with a

Fig. 3 The overall flow of activities



mask label at the end. Both of the data samples for each individual will be used to train the model to recognize the individual's face and whether he or she is wearing a mask or not. Once ready, this trained model will be used by a Python script that performs facial recognition to produce the outputs in the form of the identified individual name and status (wearing a mask or not).

Figure 3 illustrates the flow of activities that occur when a user begins to open the system's main URL in a browser. To record attendance, the user will initially take a selfie via webcam

and ensure that his or her face is taken in full. Then the user will click a button to upload the captured image to the server for processing.

The application server that gets the image will then activate a Python script that can recognize the face from the image. The Python script will return the output in the form of an individual name identified through the images, as well as mask-wearing status. This information will be used to record attendance in the database. The final result on the webpage that will be displayed to the user is in the form of a selfie picture that has been labeled with face recognition information and mask-wearing status.

2.3 Python's codes for model training and facial recognition

This online attendance system performs face recognition with face mask detection driven by a Python program. For the overall process, three stages of Python codes [10] will be run so that the recognition process can take place. The extract embeddings code will be run first followed by the train model code and lastly the recognition code. Each of these codes will be elaborated on in the subsection below.

2.3.1 Extract embeddings Python code

Figure 4 shows the codes used for creating a 128-D vector representing a face using a deep learning function extractor [10]. To construct embeddings, all faces of a user in the dataset will be transferred through the neural network. The 'imagePaths' variable contains the path to each image in the dataset. Then, the embeddings and corresponding names will be held in two lists which are known as 'knownEmbeddings' and 'knownNames'. The variable called 'total' explains how many faces had been processed.

```
# grab the paths to the input images in our dataset
print("Load image dataset..")
imagePaths = list(paths.list_images(args["dataset"]))

# initialize our lists of extracted facial embeddings and
# corresponding people names
knownEmbeddings = []
knownNames = []

# initialize the total number of faces processed
total = 0

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
    # extract the person name from the image path
    name = imagePath.split(os.path.sep)[-2]

    # load the image, resize it to have a width of 600 pixels (while
    # maintaining the aspect ratio), and then grab the image
    # dimensions
    image = cv2.imread(imagePath)
    image = imutils.resize(image, width=600)
    (h, w) = image.shape[:2]
```

Fig. 4 A snippet of codes to extract embeddings [10]

```

# load the face embeddings
print("Loading face embeddings of the dataset")
data = pickle.loads(open(args["embeddings"], "rb").read())

# encode the labels
print("Encoding image labels")
le = LabelEncoder()
labels = le.fit_transform(data["names"])

# train the model used to accept the 128-d embeddings of the face and
# then produce the actual face recognition
print("Training the model usng SVM...")
recognizer = SVC(C=1.0, kernel="linear", probability=True)
recognizer.fit(data["embeddings"], labels)

# write the actual face recognition model to disk
f = open(args["recognizer"], "wb")
f.write(pickle.dumps(recognizer))
f.close()

# write the label encoder to disk
f = open(args["le"], "wb")
f.write(pickle.dumps(le))
f.close()

```

Fig. 5 Codes used in training face recognizer model [10]

2.3.2 Train model Python code

Upon completion, based on the extracted face embeddings from the dataset, the face recognition model training will commence by running the Python codes [10] as shown in Fig. 5. The training process starts where the SVM model will be initialized and the model training will begin. This model is then exported and the encoder will be labeled to disks as pickle files after training the model. The entire process will allow the trained model to be applied in recognizer codes to process input images for facial recognition.

```

# load the image, resize it to have a width of 600 pixels (while
# maintaining the aspect ratio), and then grab the image dimensions
image = cv2.imread(args["image"])
image2 = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
image3 = cv2.cvtColor(image2, cv2.COLOR_GRAY2RGB)
image = imutils.resize(image, width=600)
image3 = imutils.resize(image3, width=600)
(h, w) = image.shape[:2]
# construct a blob from the image
imageBlob = cv2.dnn.blobFromImage(
    cv2.resize(image, (300, 300)), 1.0, (300, 300),
    (104.0, 177.0, 123.0), swapRB=False, crop=False)
# apply OpenCV's deep learning-based face detector to localize
# faces in the input image
detector.setInput(imageBlob)
detections = detector.forward()

```

Fig. 6 A snippet of codes to recognize a face


```

# extract the face ROI, convert it from BGR to RGB channel
# ordering, resize it to 224x224, and preprocess it
face = image[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)
face = np.expand_dims(face, axis=0)

# pass the face through the model to determine if the face
# has a mask or not
(mask, withoutMask) = model.predict(face)[0]

# determine the class label and color we'll use to draw
# the bounding box and text
label = "Mask" if mask > withoutMask else "No Mask"
color = (0, 255, 0) if label == "Mask" else (0, 0, 255)

# include the probability in the label
label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)

```

Fig. 7 A snippet of codes to detect facial mask [11] (in the first approach)

2.3.3 Facial recognizer codes

Code for facial recognition [10] through input images has been employed. Generally, this code includes the process of detecting faces, extracting embeddings, and querying the SVM model so that it can determine who is in the image. Figure 6 shows a snippet of the modified code, where the input image is processed to perform the face recognition process. Since the original input image is in RGB format, it must first be converted to a grayscale form. This is to reduce information unnecessary for face recognition, such as color and light effects. In this context, an analysis has been made to compare when face data is trained in RGB and grayscale forms. For RGB image recognition, accuracy is around 10% to 50%, while for face recognition through grayscale images, accuracy is around 20% to 70%. This proves that the percentage of recognition accuracy in grayscale form is better than face recognition based on RGB image.

2.3.4 Facial mask detection

We have explored two methods for the additional process of detecting face masks. The first approach is to employ a separate code from face recognition, namely a “face mask detector” code from Rosebrock, [11]. A code snippet is shown in Fig. 7. In this first approach, the facial recognition and mask detection processes are carried out by two different programs.

For the second approach, we employ the same facial recognition program from Rosebrock, [10] to also detect face masks by generating an additional synthetic dataset before model training. This synthetic dataset is generated by adding a virtual face mask layer on every sample image of each individual as a new mask dataset for that individual. With the mask dataset for each individual, the trained model will also be able to recognize the owner’s face, even when wearing a face mask. Using face recognition, face mask detection can also be achieved.

These two methods are compared especially in the measurement of time required to produce both outputs, namely face recognition and face mask detection.

2.4 Online web scripting

2.4.1 Interface to capture input image through a browser

To allow input images to be captured from the camera through a browser, HTML and JavaScript and CSS have been coded as the front page of the system. This script will activate the locally attached camera every time the page is opened from the browser. Once the camera is activated, the user can view the live feed from the camera and when ready can press the ‘Take snapshot’ button to snap a picture. The user can first examine the picture taken and if it is acceptable, then the user can proceed to send this picture to the server for processing. Figure 8 shows the HTML and JavaScript written for the front page of this system.

```

<h1 style="color:LightGreen" class="text-center">
  Online Attendance System Based On Face Recognition with Face Mask Detection
</h1>

<form method="POST" action="storeImage.php">
  <div class="row">
    <div class="col-md-6">
      <div id="my_camera"></div>

      <br/>

      <input type="button" value="Take Snapshot" onClick="take_snapshot()">
      <input type="hidden" name="image" class="image-tag">
    </div>

    <div class="col-md-6">
      <div id="results">Your captured image will appear here...</div>
    </div>

    <div class="col-md-12 text-center">
      <br/>
      <button class="btn btn-success">Submit</button>
    </div>
  </div>
</form>
</div>

<!-- Configure a few settings and attach camera -->

<script language="JavaScript">

  Webcam.set({
    width: 490,
    height: 390,
    image_format: 'jpeg',
    jpeg_quality: 90
  });

  Webcam.attach( '#my_camera' );

  function take_snapshot() {
    Webcam.snap( function(data_uri) {
      $(".image-tag").val(data_uri);
      document.getElementById('results').innerHTML = '';
    } );
  }
</script>

```

Fig. 8 HTML and JavaScript of the front page

```

$img = $_POST['image'];
$folderPath = "upload/";

$image_parts = explode(";",base64,"", $img);
$image_type_aux = explode("image/", $image_parts[0]);
$image_type = $image_type_aux[1];

$image_base64 = base64_decode($image_parts[1]);
$file_name = uniqid() . '.png';

$file = $folderPath . $file_name;
file_put_contents($file, $image_base64);

print('Successfully save the image into the server ->');
print_r($file_name);
chmod($folderPath . $file_name, 0755);

if(isset($_POST['image']))
{
    exec("python recognize.py --image " . $folderPath . $file_name);
}

```

Fig. 9 PHP script to handle uploaded image

2.4.2 PHP script for processing images and displaying output

A PHP script has been written to process pictures sent by users through browsers. First of all, this script will save the received picture file by remembering the name of the file. Then, this script will activate the Python program to run the face recognition process. The name of the received image file will be specified for the Python program as the image input to be processed. Figure 9 shows the PHP script written for this purpose.

As soon as the called Python program finishes running, the PHP script will redirect the user to the next web page that will display the generated output from the Python's face recognition process, along with the output for mask detection (see Fig. 10).

2.5 Hardware requirement

The hardware required for this project is a computer used as a server to train the model and run the application server. Another computer is needed as a client device that needs to be equipped with a camera to take pictures for face detection. The last requirement is an internet connection to allow the user to access the server through a browser from the client terminal.

```

$outputfile = get_output($file_name);
$username = get_username($file_name);
$mask_status = get_mask_status($file_name);
$time_now = get_time_now();

print("<img src='$outputfile'><br/>");
print("<span style='background-color:yellow'>$username<br/>$mask_status<br/>");
print("Seen / Check in at $time_now</span>");
record_data($time_now, $username, $mask_status); // save to database

```

Fig. 10 PHP script to display the output

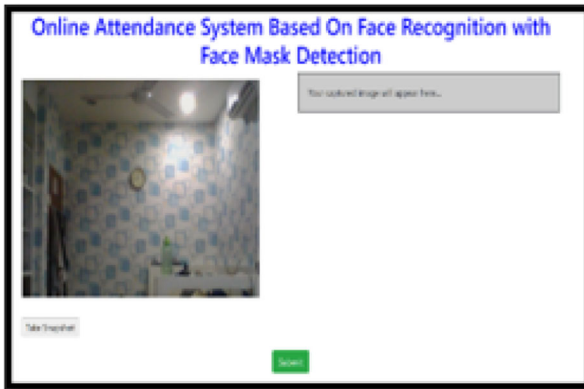


Fig. 11 The main front-end webpage accessible from the server

3 Results and discussion

This section will present the implementation of this project in the form of a server application, as well as the results obtained and an analysis of the results.

3.1 Main web Interface

Figure 11 shows the main web page of this system created based on HTML, JavaScript and PHP. Once opened, this page will automatically try to activate the camera on the client terminal. The user needs to click on the approval button for the use of the camera and then take selfies. This captured image will be used as an input image to be uploaded to the online attendance system server for facial recognition purposes, as well as face mask detection.

Figure 12 shows an example in which the user has successfully taken their selfie and the image is displayed on this web page. To capture a selfie photo, the user just needs to press the



Fig. 12 Output result once the user has taken a selfie image through a webcam

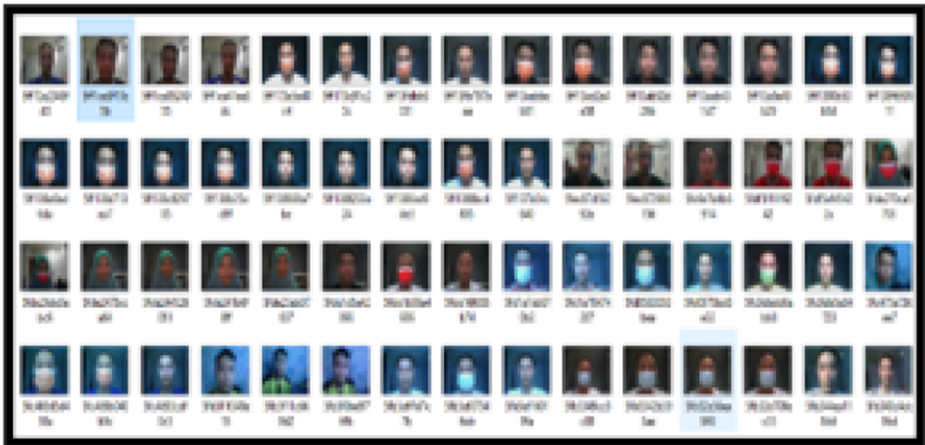


Fig. 13 All uploaded images are stored in one folder

‘Take Snapshot’ button and the captured image will appear in the right image frame on this page. The user is allowed to re-take pictures if the captured image is not acceptable. If the captured image is acceptable, then the user can press the submit button to upload it to the server for the face recognition process.

3.2 Image processing on the server

Figure 13 shows a temporary image repository on the server showing selfie images that have been successfully uploaded to the server after the submit button is pressed. All selfie images are saved as PNG files.

On the server, a PHP script will handle the process after the selfie image is uploaded. This PHP script will activate the python program that will run the process of face recognition and facial mask detection. As for the output generated by the Python program, the same PHP file will also display the results that have been obtained to the user through the browser.

Fig. 14 The output file in the form of an image annotated with the recognized identity



Table 2 Comparison of processing time for both methods

Approach	Processing Time Recorded for Method 1	Processing Time Recorded for Method 2
Capturing user without the face mask	24 seconds (see Fig. 15)	6 seconds (see Fig. 16)
Capturing user with face mask	21 seconds (see Fig. 17)	5 seconds (see Fig. 18)

Figure 14 shows the output file, which was generated after the Python recognition script was run. Such a file will be displayed to the user to show successful face recognition. A bounding box will be drawn to map with the face area, along with an identification label and recognition accuracy value written on the bounding box.

3.3 Comparison between two approaches for face mask detection

At the beginning of this study, the face recognition and facial masks detection processes were implemented by two different Python scripts. The first Python script [10] will be activated by PHP to detect the user's identity through the submitted facial image, while the second Python script [11] is activated specifically to detect whether the user is wearing a mask or not, also based on the same input image. However, the processing of these two scripts takes a long time on the server. Therefore, another approach has been proposed to achieve a more optimized processing time.

The second method is to rely on one Python script only; that is, the face recognition Python script based on the pre-trained model [10]. However, some improvements have been made to this method to allow it to also detect the use of face masks. One of the important improvements that has been made is to use synthetic data as a dataset that represents all user's faces with the application of virtual face masks. This synthetic dataset is also used in the model training process, so that the generated pre-trained model can recognize not only identity, but also to recognize whether the individual is wearing the face mask or not.

Before this second approach was adopted, a few analyses were performed to compare the processing time taken by both of these two different methods. As we can see in Table 2, the approach taken for method 2 is much faster which takes about 5 to 6 seconds to produce the face recognition and face mask detection outputs. The approach taken in method 1 requires

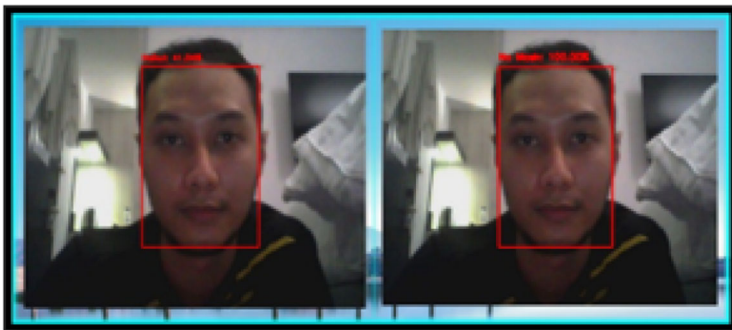
**Fig. 15** The first method capturing user without the mask

Fig. 16 The second method capturing user without the mask

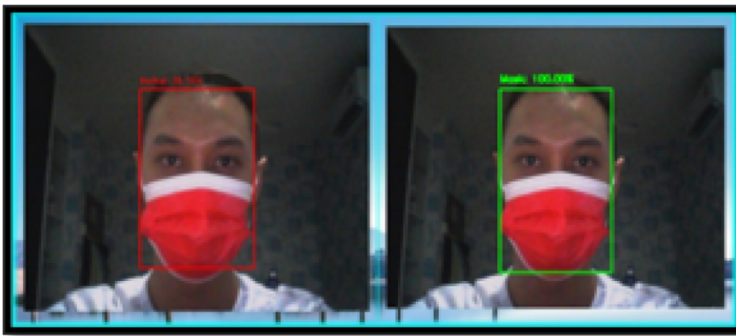
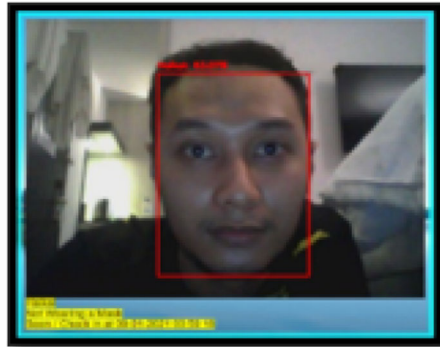
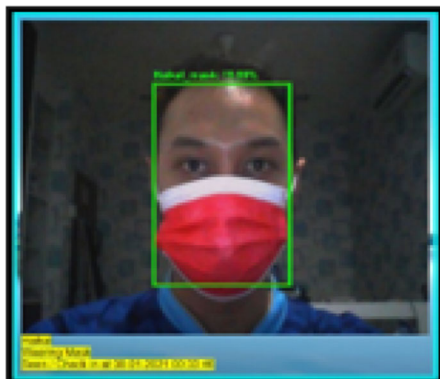


Fig. 17 The first method capturing user with a mask

Fig. 18 The second method capturing user with a mask



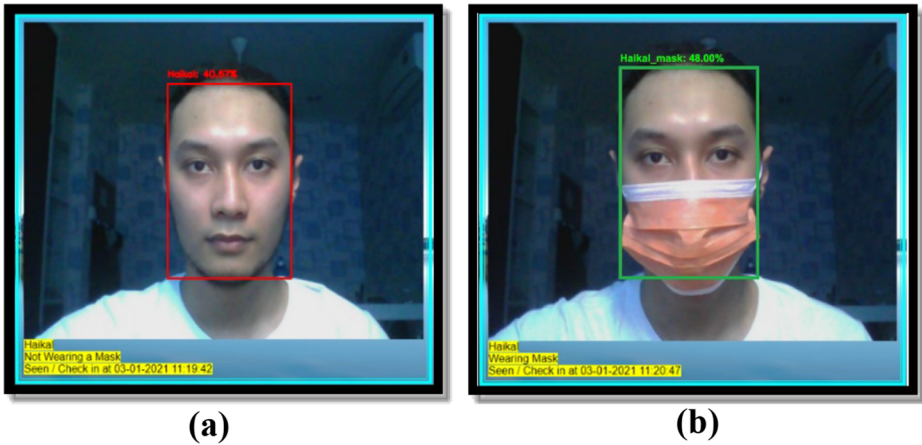


Fig. 19 a The result of the face recognition system with 40.67% similarity of the first subject not wearing a mask b The result of the face recognition system with 48% similarity of the first subject wearing a mask

more time (around 20 to 24 seconds) to produce the same outputs. From here, we can conclude that method 2 is a more optimized option in terms of time usage. Although this method relies on a single Python script, it has successfully performed both face recognition and face mask detection with a shorter processing time.

3.4 Analysis of the similarity of facial recognition and face mask detection results

This section will demonstrate the analysis used to observe the effectiveness of this program in performing face recognition and detection of face masks. Figures. 19, 20, 21 show the output obtained based on several facial samples based on the second approach in face recognition and mask detection.

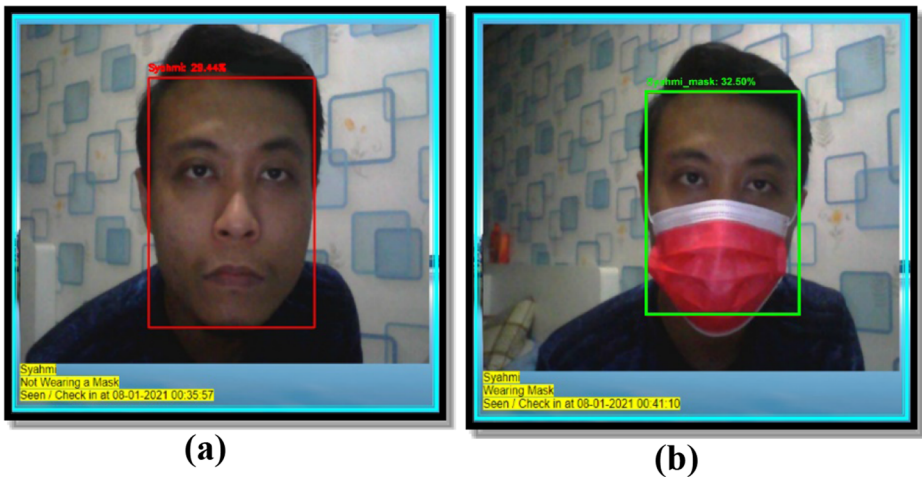


Fig. 20 a. The result of the face recognition system with 29.44% similarity of the second subject not wearing a mask b. The result of the face recognition system with 32.50% similarity of the second subject wearing a mask

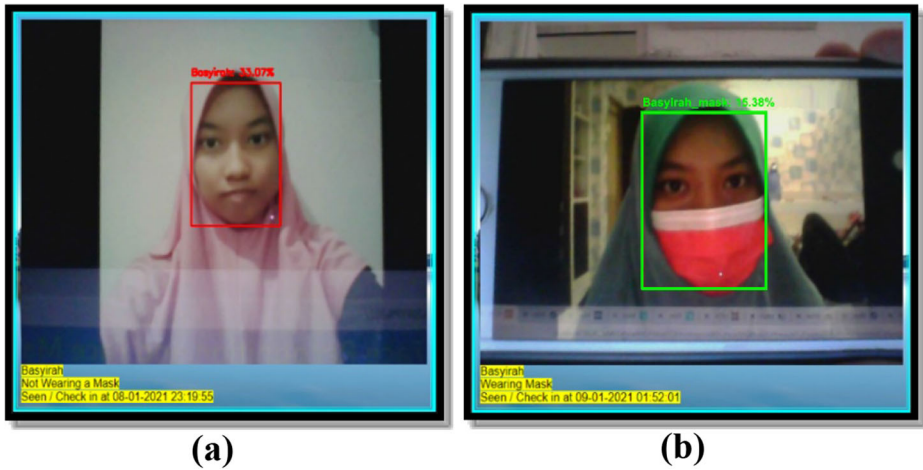


Fig. 21 a. The result of the face recognition system with 33.07% similarity of the third subject not wearing a mask b. The result of the face recognition system with 15.38% similarity of the third subject wearing a mask

3.4.1 The equation for accuracy, sensitivity and precision [3]

To measure the accuracy of facial recognition, the accuracy equation is used as in [1].

$$Accuracy = \frac{TP + TN}{(TP + TN + FP + FN)} \tag{1}$$

To measure the sensitivity of facial recognition, the sensitivity equation is used as in [2].

$$Sensitivity = \frac{TP}{(TP + FN)} \tag{2}$$

To measure the precision of facial recognition, the precision equation is used as in [3].

$$Precision = \frac{TP}{(TP + FP)} \tag{3}$$

Where TP is TRUE POSITIVES, FN is FALSE NEGATIVES, FP is FALSE POSITIVES and finally, TN represents TRUE NEGATIVES.

3.4.2 Facial recognition accuracy analysis

From the 10 datasets of facial image samples collected, the accuracy of the face recognition process has been calculated as shown in Tables 3 and 4 using Eq. (1).

Table 3 Analysis of facial recognition accuracy

Label	Description	Occurrence Count
TP	Correct user’s name based on the user’s samples	9
FN	Incorrect predicted user’s name based on train user’s samples	1
FP	Correct predicted user’s name based on untrain user’s samples	1
TN	Incorrect name and untrained user’s face samples	0

Table 4 Calculation on facial recognition analysis for Eq. (1) only

Accuracy	$\frac{9+0}{(9+1+1+0)} \times 100\%$	81.8%
----------	--------------------------------------	-------

3.4.3 Mask detection accuracy analysis

From the 24 samples of synthetic datasets of face mask detection samples collected, the accuracy of the mask detection process has been calculated as shown in Tables 5 and 6, based on Eqs. (1), (2) and (3).

Based on the testing and analysis, the accuracy of the system is calculated at 81.8% for face recognition and 80% for face mask detection same as the value in sensitivity for face mask detection. While the value for precision for face mask detection is 100%, which is a very decent level of accuracy, based on the calculation that had been done earlier. More data and synthetic data may be required to feed the recognizer to improve the accuracy, sensitivity and precision so that the face and face mask detection can be performed with greater accuracy.

3.5 Recording attendance information into the database

The main purpose of this system is to record the presence or attendance of users online through a browser by recognizing the faces captured through a webcam. For this record-keeping, a database was built to store all the data captured. This includes the time a user submitted the picture, the identified user's username and the status of whether the user is wearing a mask or not. Since this is an early prototype, the interface for this database was quickly developed (see Fig. 22).

Table 5 Analysis of face mask detection accuracy

Label	Description	Occurrence Count
TP	Correct face mask detection based on the synthetic dataset	24
FN	Incorrect face mask detection based on the synthetic dataset	6
FP	Untrained synthetic dataset	0
TN	Incorrect face mask detection and untrained synthetic dataset	0

Table 6 Calculation on face mask detection analysis for Eq. (1), (2) and (3)

Accuracy	$\frac{24+0}{(24+6+0+0)} \times 100\%$	80%
Sensitivity	$\frac{24}{(24+6)} \times 100\%$	80%
Precision	$\frac{24}{(24+0)} \times 100\%$	100%

DATE TIME	USERNAME	STATUS	ACTION
2021-01-15 00:18:57	Aqilah	Wearing Mask	DELETE
2021-01-15 00:18:23	Aqilah	Not Wearing a Mask	DELETE
2021-01-15 00:18:00	Aqilah	Not Wearing a Mask	DELETE
2021-01-15 00:17:11	Aqilah	Not Wearing a Mask	DELETE
2021-01-15 00:16:41	Haikal	Not Wearing a Mask	DELETE
2021-01-15 00:16:18	Haikal	Wearing Mask	DELETE
2021-01-15 00:14:09	Amirul	Not Wearing a Mask	DELETE
2021-01-15 00:13:30	Amirul	Wearing Mask	DELETE
2021-01-15 00:11:51	Basyirah	Not Wearing a Mask	DELETE
2021-01-15 00:11:02	Basyirah	Wearing Mask	DELETE
2021-01-15 00:10:07	Athanasius	Not Wearing a Mask	DELETE
2021-01-15 00:09:42	Athanasius	Not Wearing a Mask	DELETE
2021-01-15 00:08:49	Haikal	Wearing Mask	DELETE

Fig. 22 Attendance information is recorded in the MySQL database on the server

4 Conclusion

This project has achieved its targeted objectives. First of all, a prototype system was successfully built to capture online attendance records, in which user identities are recognized based on facial biometrics. This system is a web-based application and so users can access the system interface from any browser regardless of terminal. Thus, the user is not burdened by the need to install special applications for this purpose.

In addition to the server interface script, the server application consists of a Python program used for the face recognition process, which will process each selfie image uploaded by the user to identify the user identity. This Python program requires a trained model for face recognition. One of the purposes of this system is to allow users who wear or do not wear masks to be recognized. Therefore, in the process of training the face recognition model, the sample data set used consisted of original images of the user's face, as well as the synthetically generated images for the virtual face mask application. More than 200 user face data points were successfully obtained for testing and analysis purposes.

Some limitations of this system have been identified. For example, when the sample of user face data is insufficient, this will reduce the accuracy of face recognition and face mask detection process. Therefore, to get better accuracy, more samples of user face data are needed. Overall, the project successfully produced a system that could be used take attendance more easily and quickly, as well as monitor public safety by performing mask detection to stop the spread of the Covid-19 virus.

Acknowledgements We would like to extend our acknowledgment to those who have directly and indirectly contributed to our project. This research is funded by Universiti Teknologi MARA (UiTM) via the Lestari Covid-19 Research Grant registered as 600-RMC/LESTARI COVID/5/3 (008/2020).

Funding No fund received for this project.

Data Availability All the data is collected from the simulation reports of the software and tools used by the authors. Authors are working on implementing the same using real world data with appropriate permissions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Ethical Approval and Human Participation No ethics approval is required.

References

1. Arulogun OT, Olatunbosun A, Fakolujo OA, Olaniyi OM (2013) RFID-based students attendance management system. *Int J Eng Sci Res* 4(2)
2. Bhuiyan, MM, Khushbu, SA, Islam, MO (2020) A deep learning based assistive system to classify COVID-19 face mask for Human safety with YOLOv3. 2020 11th international conference on computing, communication and networking technologies (ICCCNT), 1–5
3. Confusion Matrix (2019) Data science and machine learning. <https://manisha-sirsat.blogspot.com/2019/04/confusion-matrix.html>. Accessed 29 Apr 2019
4. Hameed MAJ (2017) Android-based smart student attendance system. *Int Res J Eng Technol* 12:2395–2356
5. Hussain SS, Farooq SM, Ustun TS (2019) Implementation of blockchain technology for energy trading with smart meters. In: 2019 Innovations in Power and Advanced Computing Technologies (i-PACT). IEEE, vol 1, pp 1–5
6. Jacksi K, Ibrahim F, Zebari S (2018) Student Attendance Management System. *Scholars J Eng Technol* 6: 49–53. <https://doi.org/10.21276/sjet.2018.6.2.1>
7. Lai CC, Shih TP, Ko WC, Tang HJ, Hsueh PR (2020) Severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) and coronavirus disease-2019 (COVID-19): the epidemic and the challenges. *Int J Antimicrob Agents* 55(3):105924. <https://doi.org/10.1016/j.ijantimicag.2020.105924>
8. Lippert, P, Bergner, B, Ahmed, A, Ali, R, Adeel, S, Shahriar, MDH, Mojumder, MD (2020) Face Mask Detector. <https://doi.org/10.13140/RG.2.2.32147.50725>
9. Lu H, Stratton C, Tang Y (2020) Outbreak of pneumonia of unknown etiology in Wuhan, China: the mystery and the miracle. *J Med Virol* 92(4):401–402. <https://doi.org/10.1002/jmv.25678>
10. Rosebrock, A (2018) OpenCV face recognition. pyimagesearch. <https://www.pyimagesearch.com/2018/09/24/opencv-face-recognition/>. Accessed 18 Jun 2018
11. Rosebrock, A (2020) COVID-19: face mask detector with OpenCV, Keras/TensorFlow, and deep learning. PyImageSearch. <https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/>. Accessed 4 May 2020
12. Rothan HA, Byrareddy SN (2020) The epidemiology and pathogenesis of coronavirus disease (COVID-19) outbreak. *J Autoimmun* 109:102433. <https://doi.org/10.1016/j.jaut.2020.102433>

13. Sajid M, Hussain R, Usman M (2014) A conceptual model for automated attendance marking system using facial recognition. In: Ninth international conference on digital information management (ICDIM 2014). IEEE, pp 7–10
14. Sunaryono D, Siswanto J, Anggoro R (2021) An android based course attendance system using face recognition. *J King Saud Univ Comput Inf Sci* 33:304–312

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.