MASK IT

Prepared by: Team PySquad (Preeti Das(201800190) and Kritika Berry(201800500))

Under the guidance of Prof. Bijoyeta Roy.

INDEX

1. Introduction

As part of the #ShowYourSkill contest organized by Manipal Coursera, team **PySquad- Kritika Berry (201800500) and Preeti Das(201800190)** have participated under the guidance of **Prof.(Dr.) Bijoyeta Roy, Sikkim Manipal Institute of Technology**.
The objective states “How to improvise face-mask detector for mask identification?”.
The web application, **“Mask It”**, developed employs the use of **OpenCV, Flask, HTML, CSS and JavaScript**. With the skillset learned from Manipal Coursera platform, team PySquad have used it intensively in the development of the use-case.
It can be easily used with the basic provision of a web-camera.

2. Applications

The **improvised Face-Mask detector** detects the surgical mask on a face, while also persuading the user to wear a mask. Not only this, it also contains a normal face detector. **As per the problem statement, we have shown the difference between the normal and improvised face detector with masks**.

The most commonly used and available masks are the **surgical masks**. Hence, the dataset used to train the model consists of human faces with surgical masks.

“Mask It” also provides useful links to the covid-19 live tracker and then daily news updates. Hence, the user can use “Mask It” as a portal for useful links regarding covid-19 along with a face-mask detector.

As the current scenario is evident, it’s use can be employed in public places like markets and malls. It can also be used to ensure that all employees in corporate spaces are wearing masks at all times.

3. Tools/Frameworks used

a. OpenCV
b. Flask
c. Python
d. Tensorflow
e. Keras
f. HTML
g. CSS
h. JavaScript
i. Sublime/Atom text editors
j. Collaborating over Github

4. Templates

As mentioned above, “Mask It” uses Flask to provide backend functionalities.
Templates are files that contain **static data as well as placeholders for dynamic data**. A template is rendered with specific data to produce a final document. Flask uses the **Jinja template library** to render templates.

Flask reads the html files only from templates folder. Hence, in the templates folder, we incorporated the html file along with internal JavaScript and CSS (cascading style sheets).
As is shown in the screenshots, the frontend interface is implemented.

5. Flask

Flask is a web framework that provides libraries to build lightweight web applications in python. With the help of modules like **cv2, requests, werkzeug, tensorflow and keras, the functionality of using a web-camera** integrated with the face-mask detector was implemented.

Functions with the help of flask:

1. Automatic usage of webcamera
2. Integrating face-mask detector module with the web application
3. Application Routing
4. Supports all modules like tensorflow, requests, OpenCV, werkzeug.utils, tensorflow and keras.

Flask also renders the images, css and JavaScript files from static folder and index.html from the templates folder.

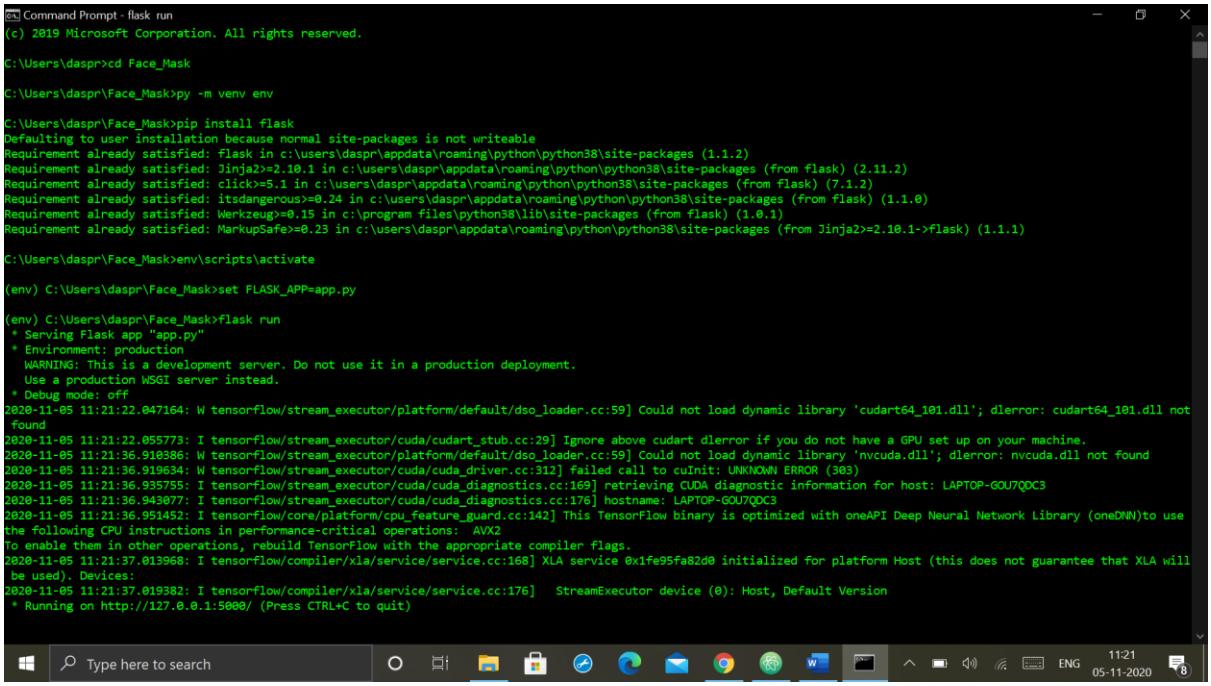


Fig: Flask server

6. Open CV

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, **video capture** and analysis including features like **face detection and object detection.**

FUNCTION NAME	**FUNCTION MEANING**
load_model	Model gets saved and loaded.
CascadeClassifier	Takes the trained classifier model.
VideoCapture	This function either takes a filename as its argument(to detect within the video) or takes 0 (when the user wants to capture video from webcam).
Release	Releases the camera device source
read	This function is used for reading the frame. It returns two variables. The first one is a flag which indicates whether the frame is read correctly or not. The second variable is the frame itself (here named as the variable frame).
cvtColor	Converts each frame to gray scale image.

detectMultiScale	The result obtained after using the trained cascade classifier is now put here and we call an opencv method detectMultiScale. This method takes in 3 arguments. • First argument: image(gray-scale image) • Second argument: scaleFactor parameter specifies how much the image size is reduced at each image scale • Third argument: minNeighbors parameter specifies how many neighbours each candidate rectangle should have to retain it.
resize	This function resizes the image.
reshape	This function reshapes the image.
predict	Predicts a label and associated confidence (e.g. distance) for a given input image.
argmax	Returns the array of indices of the maximum values along an axis.
Rectangle	Draws a rectangle with the given coordinates
putText	Displays the text in the image at the top of the rectangle
imencode	Compresses the image and stores it in the memory buffer that is resized to fit the result.
tobytes	Return image as a bytes object.

7. CourseraApplication

As a team PySquad, we have ensured that all the skillsets acquired from Manipal Coursera are focused in the development domain.

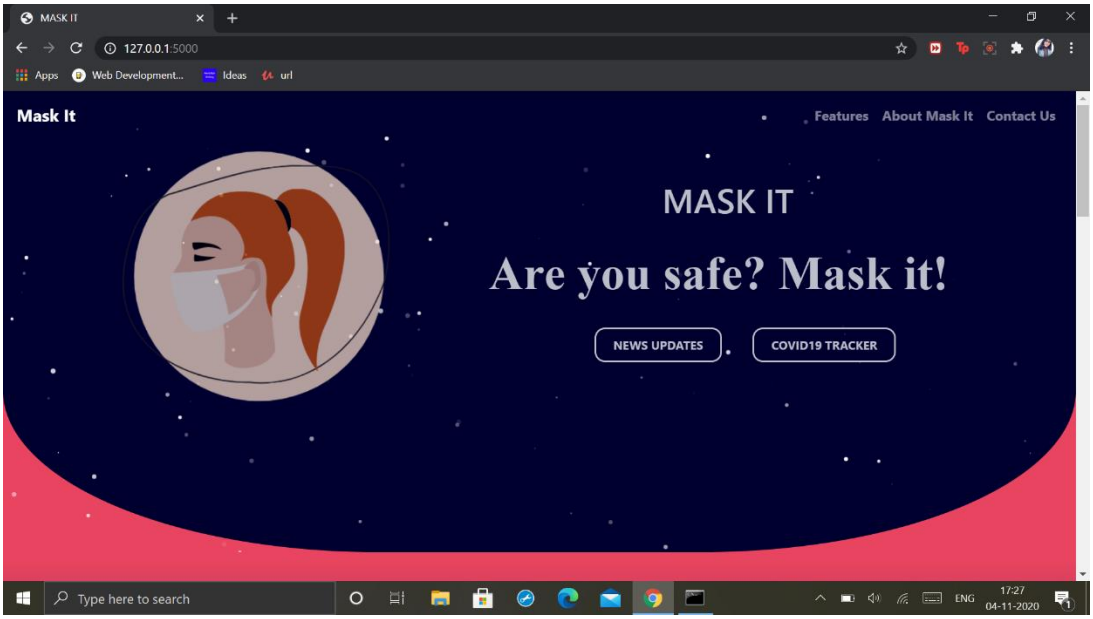
➔ **Face-Mask Detector**

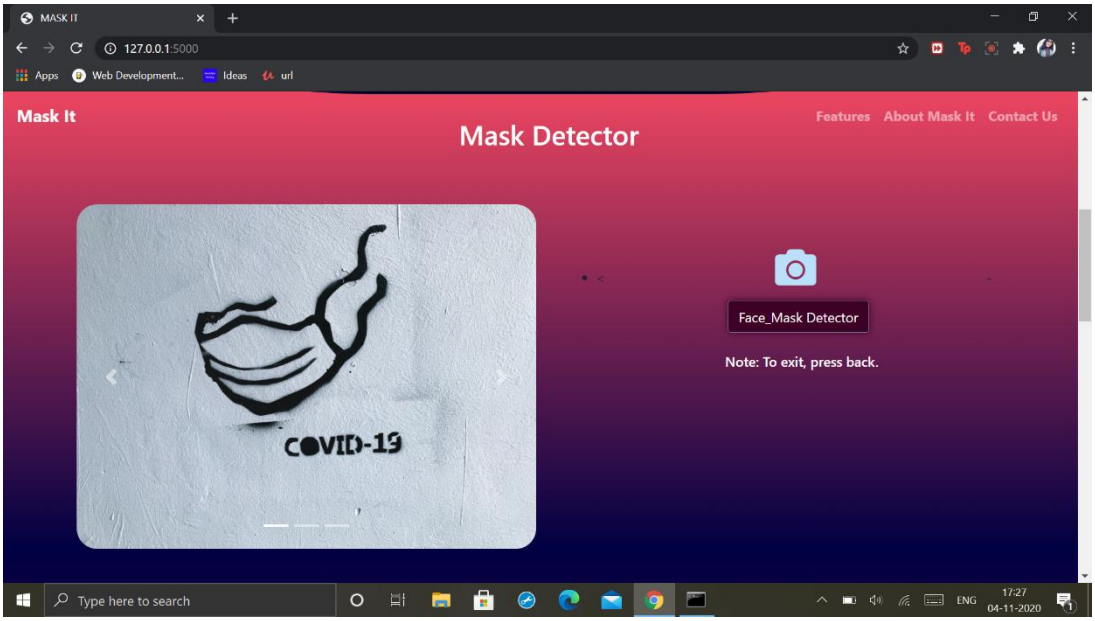
The detector employs the use of a pre-trained model and implements face-detection. Learning from the Python 3 Programming Specialization by University of Michigan, we used its in-depth knowledge of dictionaries, functions, classes (object-oriented programming) , modules and lists extensively. The

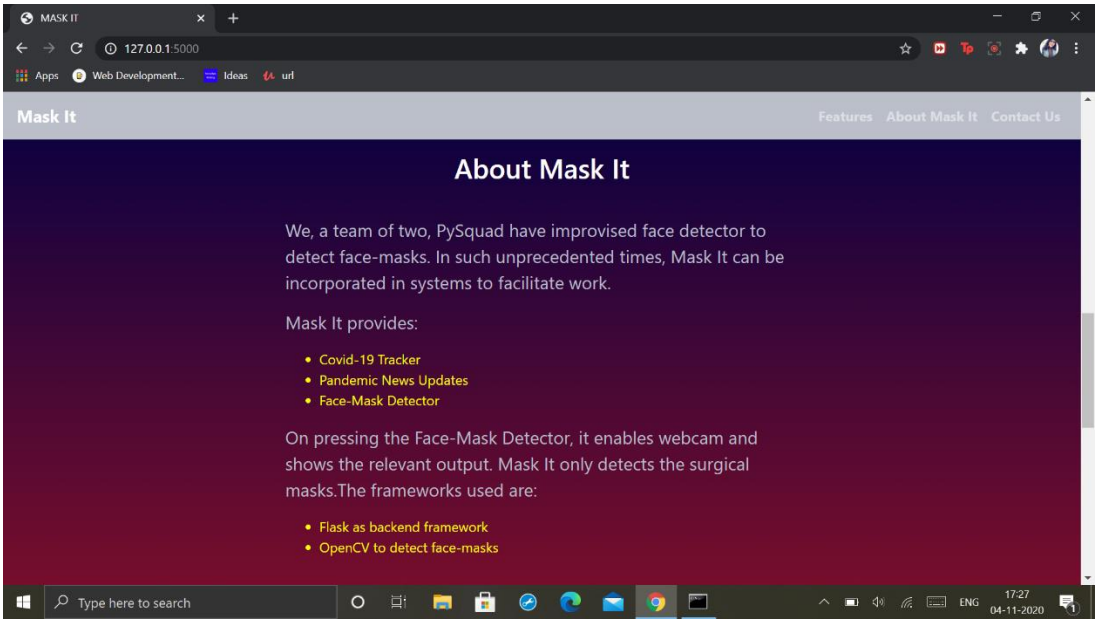
capstone project of the specialization involved OpenCV and thereby led team PySquad in developing the detector with the help of OpenCV and utilising its various in-built functions.
From extracting the face boundary (creating a rectangle boundary) , “Mask It” uses the pre-trained model and runs through the faces lists which have been converted to a gray image, identifying face-masks using convolutional neural networks and distinguishing between the correct and incorrect identification.

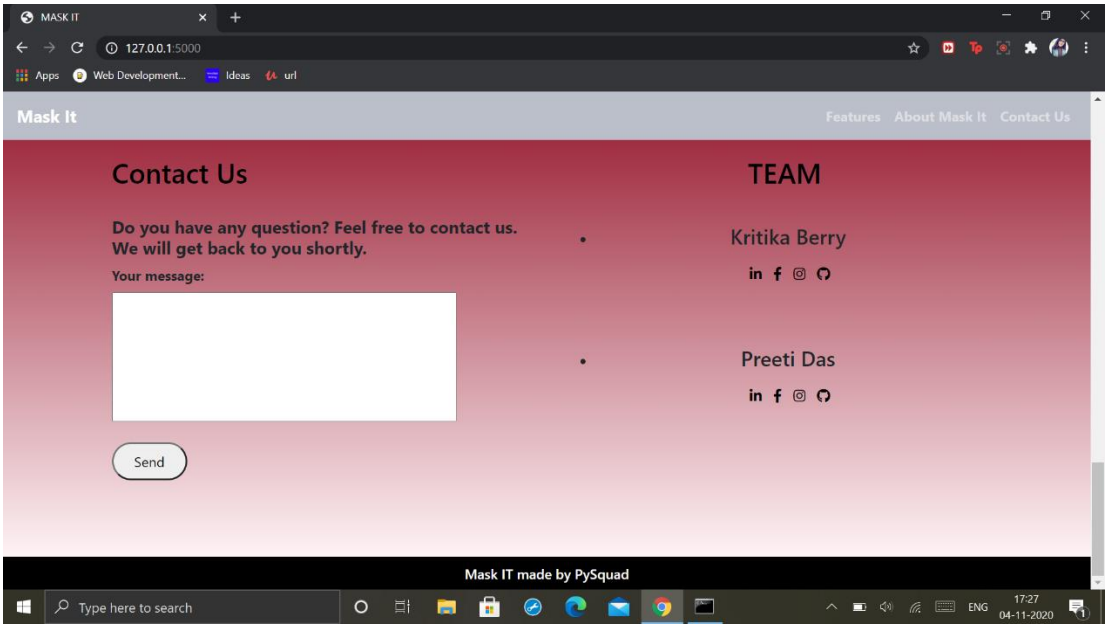
➔ **User Interface**

The frontend involving HTML, CSS and JavaScript was developed and integrated with Flask. With the help of Responsive Website Basics course by University of London, team PySquad developed a strong base and extended its use for the development of “Mask It”. Many features like image slider, external links to covid-19 tracker and news updates, styling, animated backgrounds, feedback provision through e-mail and the responsiveness of the website were incorporated in “Mask It”. Coursera gave us a learning edge in extending our knowledge from a concrete base and leading us to a successful web application.


MASK IT
127.0.0.1:5000
Apps
Web Development...
Ideas
url
Mask It
Features
About Mask It
Contact Us
MASK IT
Are you safe? Mask it!
NEWS UPDATES
COVID19 TRACKER
Type here to search
ENG
17:27
04-11-2020


MASK IT
127.0.0.1:5000
Apps
Web Development...
Ideas
url
Mask It
Mask Detector
Features
About Mask It
Contact Us
COVID-19
Face_Mask Detector
Note: To exit, press back.
Type here to search
ENG
17:27
04-11-2020





8. Acknowledgement

We would like to thank our teacher, Prof. Bijoyeta for guiding us throughout the project development and providing valuable insights.

We would also extend thanks to Manipal Coursera for providing us with an opportunity to utilize our time with the appropriate use of our skills and developing real-world useful projects. It also helped us in a smooth collaboration virtually and enriched our teamwork skills.

- ➔ Used Face-Mask Dataset by Prajna Bhandary * This dataset consists of 1,376 images belonging to two classes, *with mask* and *without mask*. (Surgical Mask only)
- ➔ Used and implemented Pre-Trained Model by Thakshila Dasun * (convolutional neural networks) The main focus of this model is to detect whether a person is wearing a mask or not.
- ➔ HaarCascade classifiers used to detect faces by OpenCV.

---

*Source:Internet