International Institute of Information Technology, Bangalore

Software Production Engineering project

MASK Detection live

Under the Guidance of
Prof. B. Thangaraju

Pavan Sudeesh Peruru   Chebrolu Suchith kumar

IMT2018517   IMT2018019

Table of Contents

1.Abstract:

The automatic door opening systems are used in commercial buildings, shopping malls, theaters, etc. These systems are used to open the door when a person comes near to the entrance of the door and closes it after he moves away from the door or after entering into the door. There are various kinds of sensors are available in the market to make such types of systems. such as Radar sensors, PIR sensors, Infrared sensors, and Laser sensors, etc. This automatic door opening system uses a PIR sensor to open or close the door automatically which senses the infrared energy produced by the human body. When someone approaches the door, the IR energy sensed by the PIR sensor changes and activates the sensor to open and close the door automatically. Further, the signal sent to the micro controller to control the door.

Due to the unexpected pandemic all around the world everyone wants a live mask detection systems for their offices. The purpose of developing a mask detection system is to build a vision based automatic door entry system, to allow door opening only if the human is wearing a mask. In this report, we will be looking at some procedures to achieve this, For training purposes, we have taken data set from https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

The architecture of our project demands two layers.

- Front end.
- Back end.

The front end of the project is handled by “html” and the back end are swiftly handled by “Python”, using “Flask” for interaction between ends. There are 3 main steps in human mask detection: Human detection, face detection and mask detection. In this project Human detection was done using YOLO algorithm. Whereas Face detection was done by VIOLA JONES and mask detection was done by CNN.

2 Introduction

2.1. OVERVIEW

‘Mask detector live’ provides solution to all malls and commercial centers around the world, who want to implement the automatic door opening system that allows entry only to the people wearing masks. It needs no extra effort from outside and can take care of detecting masks and opening door on its own.

2.2 FEATURES

Live Mask Detector.

2.3. WHY DEVOPS?

Our whole approach of the project was modular, we wanted to make different sets of development modules and wanted to deploy them with every new release without any hindrance. Wanted to test the changed code with continuous integration and then continuously deploying it. So, what all fills all these blanks was a culture, a philosophy DevOps. Devops provides all the tools to increase the capability to complete above set goals within minimum time and less trouble for developers. DevOps tools consist of configuration management, test and build systems, application deployment, version control and monitoring tools. Continuous integration, continuous delivery and continuous deployment require different tools.

2.3.1. Devops Features

- Improve deployment frequency
- Achieve faster time to market with lower failure rate
- More stable operating environments
- Improve communication and collaboration among teams

3. System Configuration

3.1. Operating system

Ubuntu 18.04.04 and it should have access to system camera or web cam.

3.2. CPU and RAM

4 core processor and RAM 8 GB

3.3 KERNEL VERSION

Linux machine 5.8.0-44-generic

3.4. Language

python3, Flask framework.

3.5. DevOps Tools

- Source Control Management - GitHub
- Continuous Integration – Github actions
- Containerization - Docker
- Continuous deployment – Github actions, Heroku.
- Monitoring - ELK Stack (Elastic Search, Logstash, Kibana)

4. Software Development life cycle

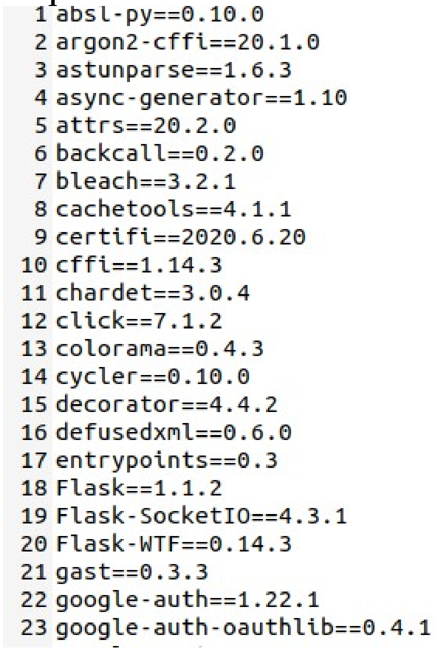
4.1. INSTALLATION

4.1.1 Python Setup

The entire ML code is written in python and the web server is written using flask, a library in python. To install all those libraries required we use pip.

python3 -m pip freeze > requirements.txt- This command then lists all the version of libraries installed and directs them to file “requirements.txt”

python3 -m pip install -r requirements.txt- This command then installs the version of libraries into the environment using requirements.txt



```
absl-py==0.10.0
argon2-cffi==20.1.0
astunparse==1.6.3
async-generator==1.10
attrs==20.2.0
backcall==0.2.0
bleach==3.2.1
cachetools==4.1.1
certifi==2020.6.20
cffi==1.14.3
chardet==3.0.4
click==7.1.2
colorama==0.4.3
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
entrypoints==0.3
Flask==1.1.2
Flask-SocketIO==4.3.1
Flask-WTF==0.14.3
gast==0.3.3
google-auth==1.22.1
google-auth-oauthlib==0.4.1
```

fig1 : requirements.txt looks like this.

4.2. SOURCE CONTROL MANAGEMENT

A Source Code Management (SCM) is a software tool used by programmers to manage the source codes. For our project every team member would clone the repository from github, create a different branch locally on their system and then merge it with master. The cycle of pulling the latest code from git, resolving any conflicts and then pushing the changes to git keeps happening for every update.

- **git clone** - This command copies the entire data on the git url
- **git checkout -b <branch_name>** This command creates a new branch with the name as in 'branch_name'
- **git add** - This command adds changes in the working directory to the staging area
- **git commit -m "message while committing"**-This command is used to save your changes to the local repository with -m used to provide a concise description that helps your teammates (and yourself) understand what happened.
- **git checkout master**- This command switches to master branch
- git pull-This command is used to update the local version of a repository from a remote.
- **git merge -**This command is used to integrate changes from another branch.
- **git push-**This command will push all the latest code to the repository.

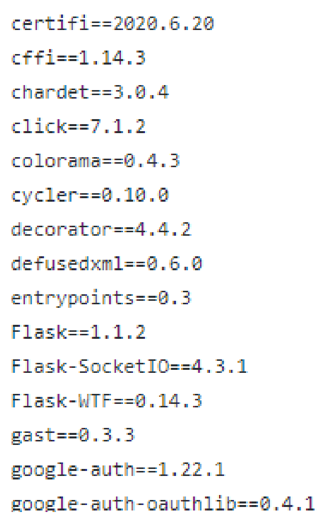
For Mask Detector live GitHub link used was
https://github.com/pavansudeesh/spe_final_project

4.3 BUILDING

PYTHON:
To build the python project we use requirements.txt file, and using pip we install all the dependent libraries for the project. Here's the link to complete requirements.txt

https://github.com/pavansudeesh/spe_final_project/blob/master/requirements.txt

pip install -r requirements.txt
the above command will install all necessary requirements



```
certifi==2020.6.20
cffi==1.14.3
chardet==3.0.4
click==7.1.2
colorama==0.4.3
cycler==0.10.0
decorator==4.4.2
defusedxml==0.6.0
entrypoints==0.3
Flask==1.1.2
Flask-SocketIO==4.3.1
Flask-WTF==0.14.3
gast==0.3.3
google-auth==1.22.1
google-auth-oauthlib==0.4.1
```

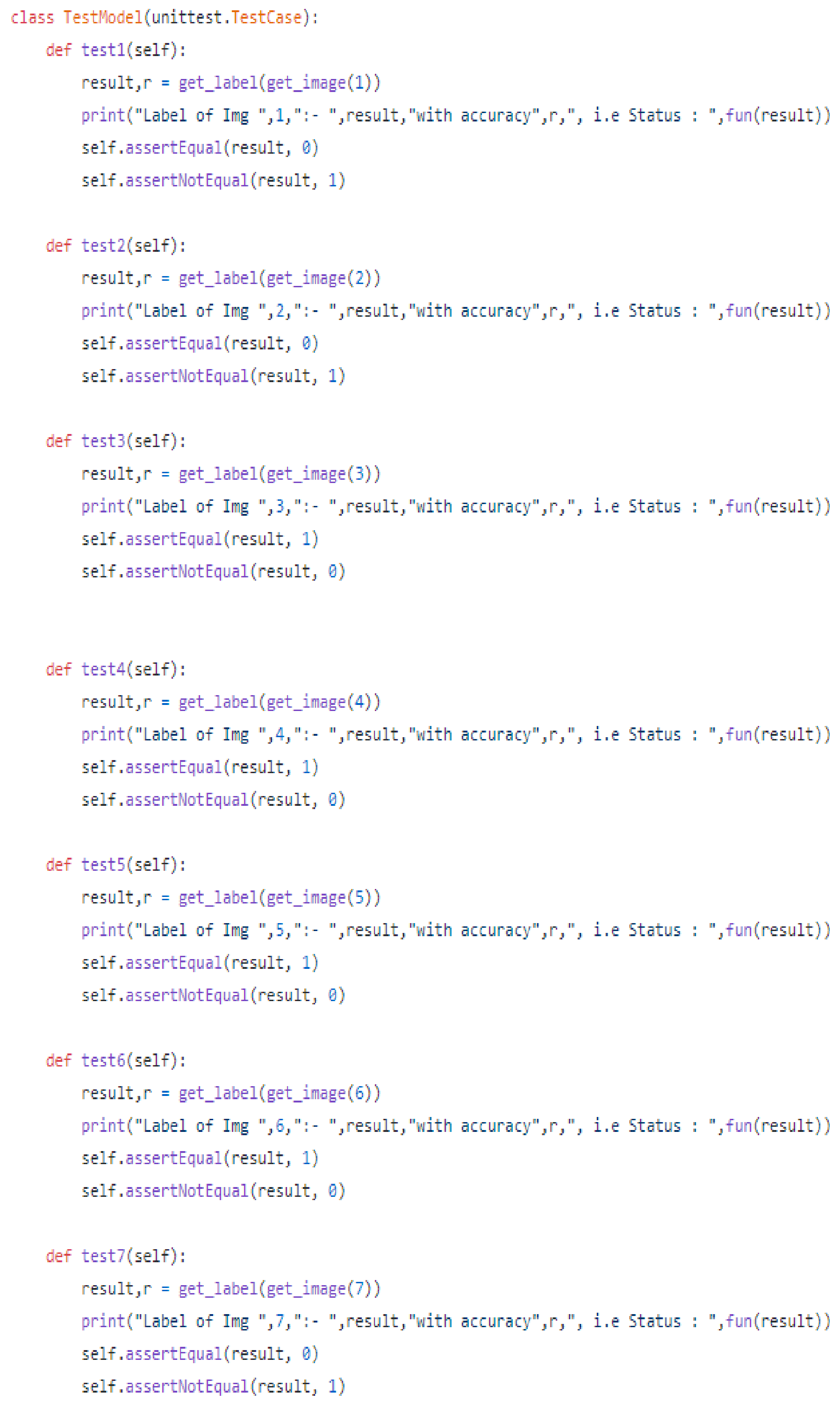
fig2: Requirements text file format.

4.4 TEST

For testing the Mask Detector live, we use unit test. Test cases are written in keeping a view to cover all possible cases.

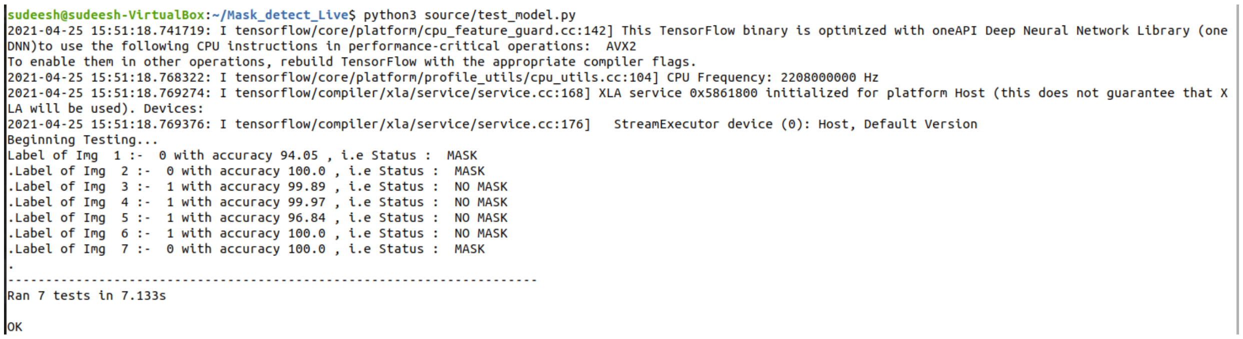
Steps to use unittest.

1. Import unittest from the standard library
2. Create a class called TestSum that inherits from the TestCase class
3. Convert the test functions into methods by adding self as the first argument
4. Change the assertions to use the self.assertEqual() method on the TestCase class
5. Change the command-line entry point to call unittest.main()



```
class TestModel(unittest.TestCase):
    def test1(self):
        result,r = get_label(get_image(1))
        print("Label of Img ",1,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 0)
        self.assertNotEqual(result, 1)

    def test2(self):
        result,r = get_label(get_image(2))
        print("Label of Img ",2,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 0)
        self.assertNotEqual(result, 1)

    def test3(self):
        result,r = get_label(get_image(3))
        print("Label of Img ",3,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 1)
        self.assertNotEqual(result, 0)


    def test4(self):
        result,r = get_label(get_image(4))
        print("Label of Img ",4,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 1)
        self.assertNotEqual(result, 0)

    def test5(self):
        result,r = get_label(get_image(5))
        print("Label of Img ",5,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 1)
        self.assertNotEqual(result, 0)

    def test6(self):
        result,r = get_label(get_image(6))
        print("Label of Img ",6,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 1)
        self.assertNotEqual(result, 0)

    def test7(self):
        result,r = get_label(get_image(7))
        print("Label of Img ",7,":- ",result,"with accuracy",r,", i.e Status : ",fun(result))
        self.assertEqual(result, 0)
        self.assertNotEqual(result, 1)
```

fig 3: unit test cases in mask detector live



```
sudeesh@sudeesh-VirtualBox:~/Mask_detect_Live$ python3 source/test_model.py
2021-04-25 15:51:18.741719: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (one
DNN)to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-04-25 15:51:18.768322: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency: 2208000000 Hz
2021-04-25 15:51:18.769274: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x5861800 initialized for platform Host (this does not guarantee that X
LA will be used). Devices:
2021-04-25 15:51:18.769376: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
Beginning Testing...
Label of Img  1 :-  0 with accuracy 94.05 , i.e Status :  MASK
.Label of Img  2 :-  0 with accuracy 100.0 , i.e Status :  MASK
.Label of Img  3 :-  1 with accuracy 99.89 , i.e Status :  NO MASK
.Label of Img  4 :-  1 with accuracy 99.97 , i.e Status :  NO MASK
.Label of Img  5 :-  1 with accuracy 96.84 , i.e Status :  NO MASK
.Label of Img  6 :-  1 with accuracy 100.0 , i.e Status :  NO MASK
.Label of Img  7 :-  0 with accuracy 100.0 , i.e Status :  MASK
.
----------------------------------------------------------------------
Ran 7 tests in 7.133s

OK
```

Fig 4: Running all our test cases and output shows mask, not mask along with their accuracy.

4.4.1 CI workflow(Github actions):

GitHub Actions help you automate tasks within your software development life cycle. GitHub Actions are event-driven, meaning that you can run a series of commands after a specified event has occurred. For example, every time someone creates a pull request for a repository, you can automatically run a command that executes a software testing script.

This diagram demonstrates how you can use GitHub Actions to automatically run your software testing scripts. An event automatically triggers the workflow, which contains a job. The job then uses steps to control the order in which actions are run. These actions are the commands that automate your software testing.

The components of GitHub actions:
Below is a list of the multiple GitHub Actions components that work together to run jobs. You can see how these components interact with each other.

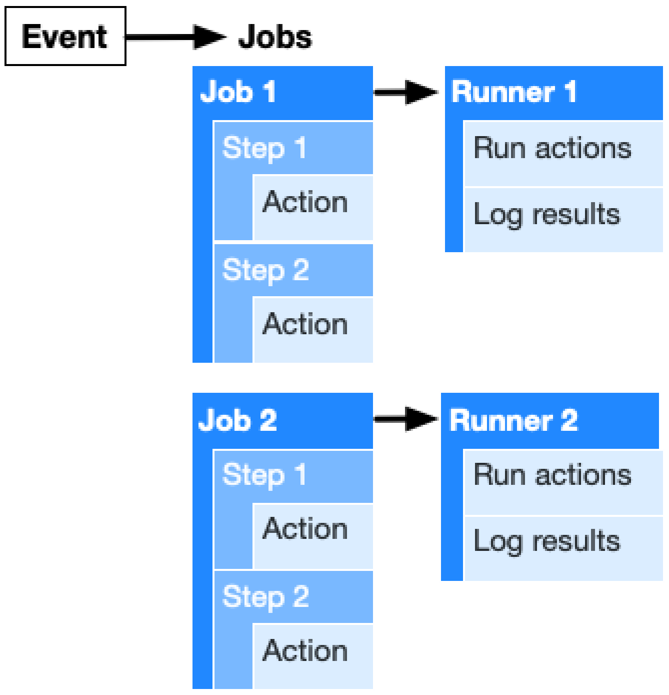


fig 5: Structure of GitHub actions components.

Workflows

The workflow is an automated procedure that you add to your repository. Workflows are made up of one or more jobs and can be scheduled or triggered by an event. The workflow can be used to build, test, package, release, or deploy a project on GitHub.

Events

An event is a specific activity that triggers a workflow. For example, activity can originate from GitHub when someone pushes a commit to a repository or when an issue or pull request is created.

Jobs

A job is a set of steps that execute on the same runner. By default, a workflow with multiple jobs will run those jobs in parallel. You can also configure a workflow to run jobs sequentially.

Steps

A step is an individual task that can run commands in a job. A step can be either an *action* or a shell command. Each step in a job executes on the same runner, allowing the actions in that job to share data with each other.

Actions

Actions are standalone commands that are combined into *steps* to create a *job*. Actions are the smallest portable building block of a workflow. You can create your own actions, or use actions created by the GitHub community. To use an action in a workflow, you must include it as a step.

Runners

A runner is a server that has the GitHub actions runner application installed. You can use a runner hosted by GitHub, or you can host your own. A runner listens for available jobs, runs one job at a time, and reports the progress, logs, and results back to GitHub.

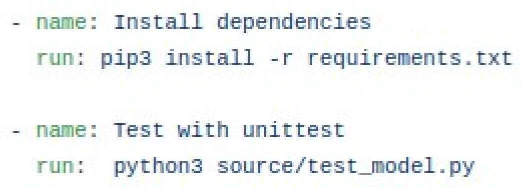
GitHub Actions uses YAML syntax to define the events, jobs, and steps. These YAML files are stored in your code repository, in a directory called .github/workflows.

Steps to be followed for creating a workflow:

1. In your repository, create the .github/workflows/ directory to store your workflow files.

2. In the .github/workflows/ directory create a new YAML file and add the code.
3. Commit those changes and push them to your GitHub repository.

In the Github actions workflow we write a script to first set up the environment to run the test cases written for mask_detector_live. And then we call to test various test cases. Reference figure



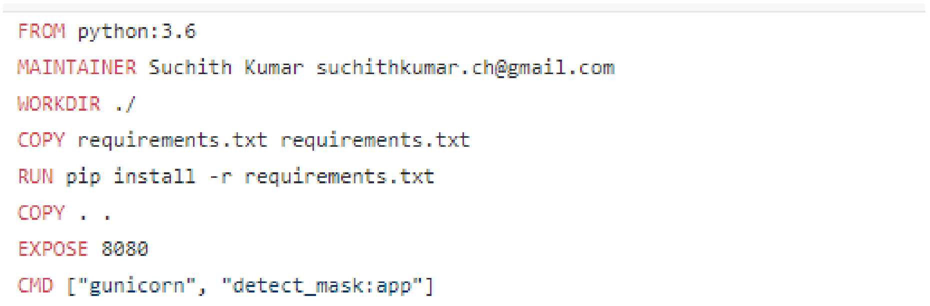
```
- name: Install dependencies
  run: pip3 install -r requirements.txt

- name: Test with unittest
  run:  python3 source/test_model.py
```

fig 6: installing requirements and calling unit test cases with mask_detector_live in github actions workflow.

4.5. DOCKER ARTIFACT

Docker is a software platform for building applications based on containers which are small and lightweight execution environments that make shared use of the operating system kernel but otherwise run-in isolation from one another. Docker images of the module is created and then pushed to docker hub, from where we can pull those images and run it at our end. Docker file is created for the project that will run when we build these docker images. So install docker on your deployment and development machines. Reference links and image

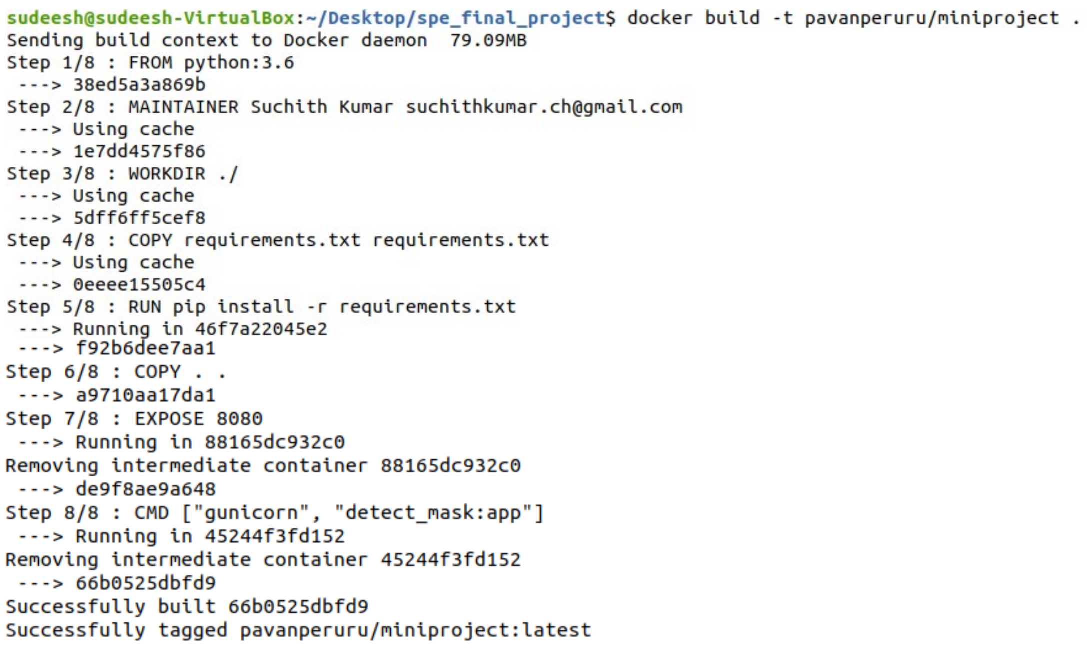
https://github.com/pavansudeesh/spe_final_project/blob/master/Dockerfile



```
FROM python:3.6
MAINTAINER Suchith Kumar suchithkumar.ch@gmail.com
WORKDIR ./
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
COPY . .
EXPOSE 8080
CMD ["gunicorn", "detect_mask:app"]
```

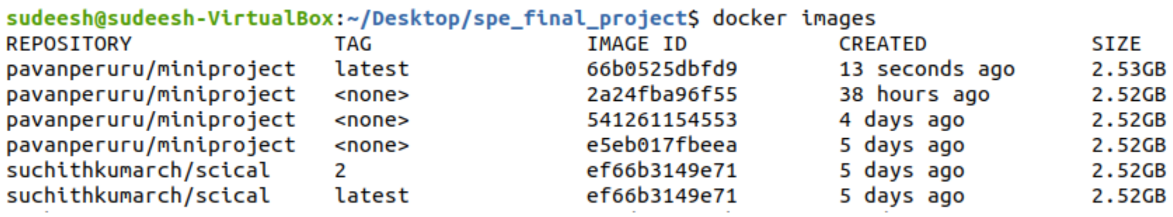
fig7: docker file for mask_detect_live, used to build docker image.

Command to build docker image in terminal:



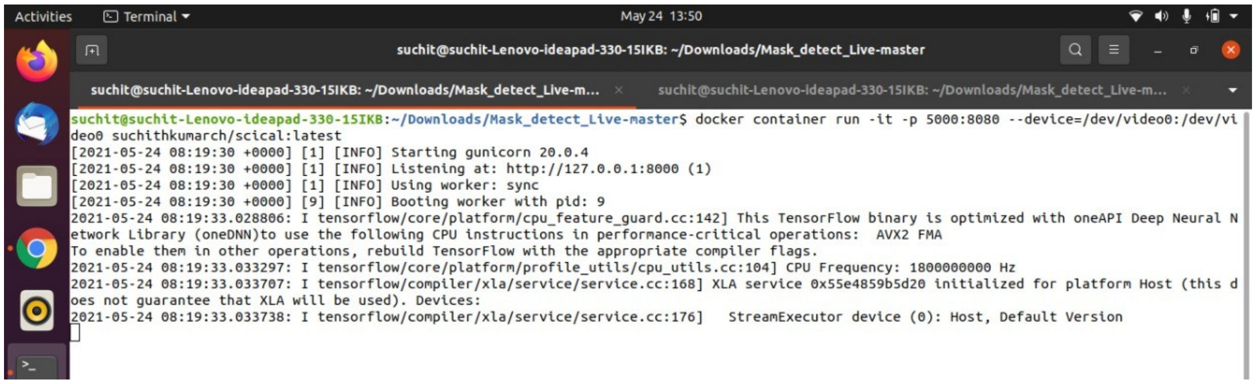
```
sudeesh@sudeesh-VirtualBox:~/Desktop/spe_final_project$ docker build -t pavanperuru/miniproject .
Sending build context to Docker daemon  79.09MB
Step 1/8 : FROM python:3.6
 ---> 38ed5a3a869b
Step 2/8 : MAINTAINER Suchith Kumar suchithkumar.ch@gmail.com
 ---> Using cache
 ---> 1e7dd4575f86
Step 3/8 : WORKDIR ./
 ---> Using cache
 ---> 5dff6ff5cef8
Step 4/8 : COPY requirements.txt requirements.txt
 ---> Using cache
 ---> 0eeee15505c4
Step 5/8 : RUN pip install -r requirements.txt
 ---> Running in 46f7a22045e2
 ---> f92b6dee7aa1
Step 6/8 : COPY . .
 ---> a9710aa17da1
Step 7/8 : EXPOSE 8080
 ---> Running in 88165dc932c0
Removing intermediate container 88165dc932c0
 ---> de9f8ae9a648
Step 8/8 : CMD ["gunicorn", "detect_mask:app"]
 ---> Running in 45244f3fd152
Removing intermediate container 45244f3fd152
 ---> 66b0525dbfd9
Successfully built 66b0525dbfd9
Successfully tagged pavanperuru/miniproject:latest
```

Fig 8: Building of docker image in local system.



```
sudeesh@sudeesh-VirtualBox:~/Desktop/spe_final_project$ docker images
REPOSITORY                TAG       IMAGE ID       CREATED          SIZE
pavanperuru/miniproject   latest    66b0525dbfd9   13 seconds ago   2.53GB
pavanperuru/miniproject   <none>    2a24fba96f55   38 hours ago     2.52GB
pavanperuru/miniproject   <none>    541261154553   4 days ago       2.52GB
pavanperuru/miniproject   <none>    e5eb017fbeea   5 days ago       2.52GB
suchithkumarch/scical     2         ef66b3149e71   5 days ago       2.52GB
suchithkumarch/scical     latest    ef66b3149e71   5 days ago       2.52GB
```

Fig 9: command to check docker images after building.


Activities
Terminal
May 24 13:50
suchit@suchit-Lenovo-ideapad-330-15IKB: ~/Downloads/Mask_detect_Live-master
suchit@suchit-Lenovo-ideapad-330-15IKB: ~/Downloads/Mask_detect_Live-m...
suchit@suchit-Lenovo-ideapad-330-15IKB: ~/Downloads/Mask_detect_Live-m...

```
suchit@suchit-Lenovo-ideapad-330-15IKB:~/Downloads/Mask_detect_Live-master$ docker container run -it -p 5000:8080 --device=/dev/video0:/dev/vi
deo0 suchithkumarch/scical:latest
[2021-05-24 08:19:30 +0000] [1] [INFO] Starting gunicorn 20.0.4
[2021-05-24 08:19:30 +0000] [1] [INFO] Listening at: http://127.0.0.1:8000 (1)
[2021-05-24 08:19:30 +0000] [1] [INFO] Using worker: sync
[2021-05-24 08:19:30 +0000] [9] [INFO] Booting worker with pid: 9
2021-05-24 08:19:33.028806: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural N
etwork Library (oneDNN)to use the following CPU instructions in performance-critical operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-05-24 08:19:33.033297: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency: 1800000000 Hz
2021-05-24 08:19:33.033707: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x55e4859b5d20 initialized for platform Host (this d
oes not guarantee that XLA will be used). Devices:
2021-05-24 08:19:33.033738: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
```

Fig 10: command access to external device like camera from the docker image.

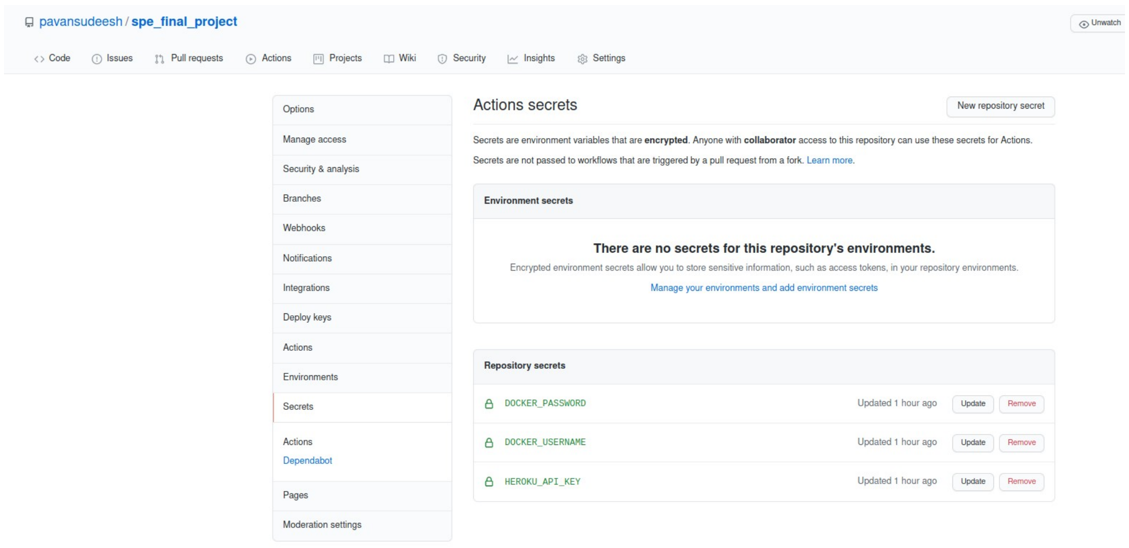
4.5.1 CI Workflow

First We need to add credentials for our docker hub and some others if necessary.

To do that

1.) go to settings in github.

2.) on the left side, bottom third one, select secrets.



Actions secrets

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

Environment secrets

There are no secrets for this repository's environments.

Encrypted environment secrets allow you to store sensitive information, such as access tokens, in your repository environments.

Manage your environments and add environment secrets

Repository secrets

DOCKER_PASSWORD	Updated 1 hour ago
DOCKER_USERNAME	Updated 1 hour ago
HEROKU_API_KEY	Updated 1 hour ago

Fig 11: way to add credentials.

3. Here select new repository secret and add the credentials.

Note: for every thing we need to give a name to it and then its value

eg: to add docker hub id, give something called as DOCKERHUB_USERNAME as name and our id pavanperuru as value. Same logic for every addition of credential

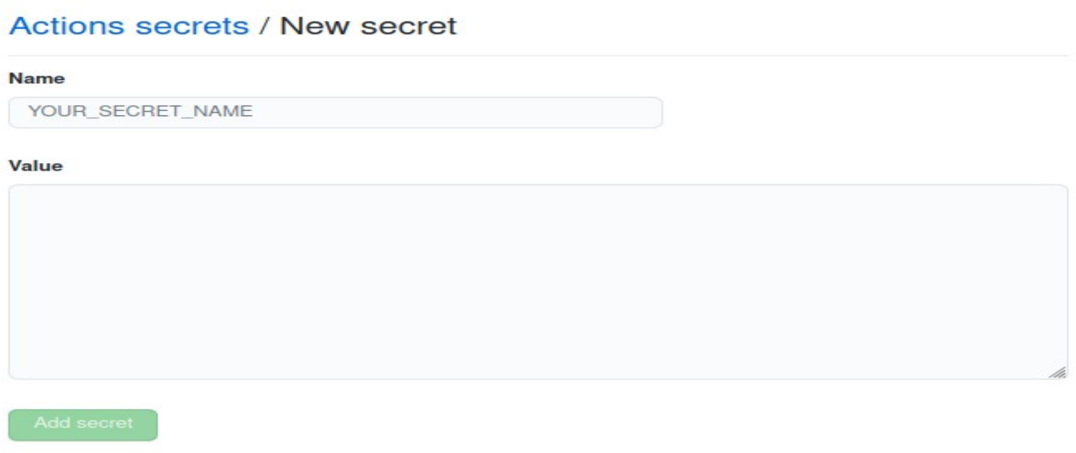
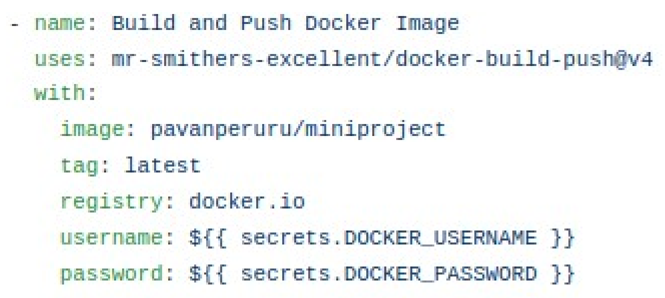


Fig 12: way to add credentials.

● Now add a related workflow script to build the docker image and push it to docker hub.
● Make sure the docker credential here in script matches your docker credential id you set earlier.



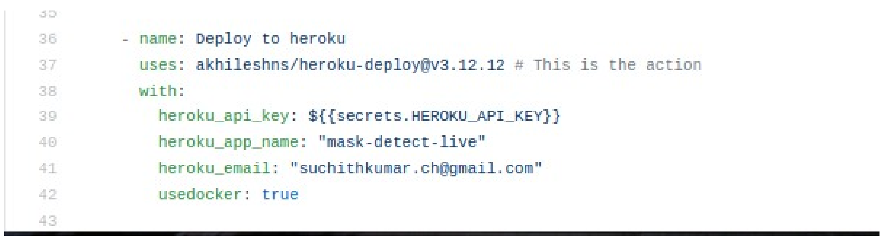
```
- name: Build and Push Docker Image
  uses: mr-smithers-excellent/docker-build-push@v4
  with:
    image: pavanperuru/miniproject
    tag: latest
    registry: docker.io
    username: ${{ secrets.DOCKER_USERNAME }}
    password: ${{ secrets.DOCKER_PASSWORD }}
```

Fig 13: Building and deploying mask_detector_live image on docker hub in github actions workflow.

● After successful deployment of docker images we can see the result on our docker-hub account. Reference links and image
https://hub.docker.com/repository/docker/pavanperuru/miniproject

4.6 Heroku:

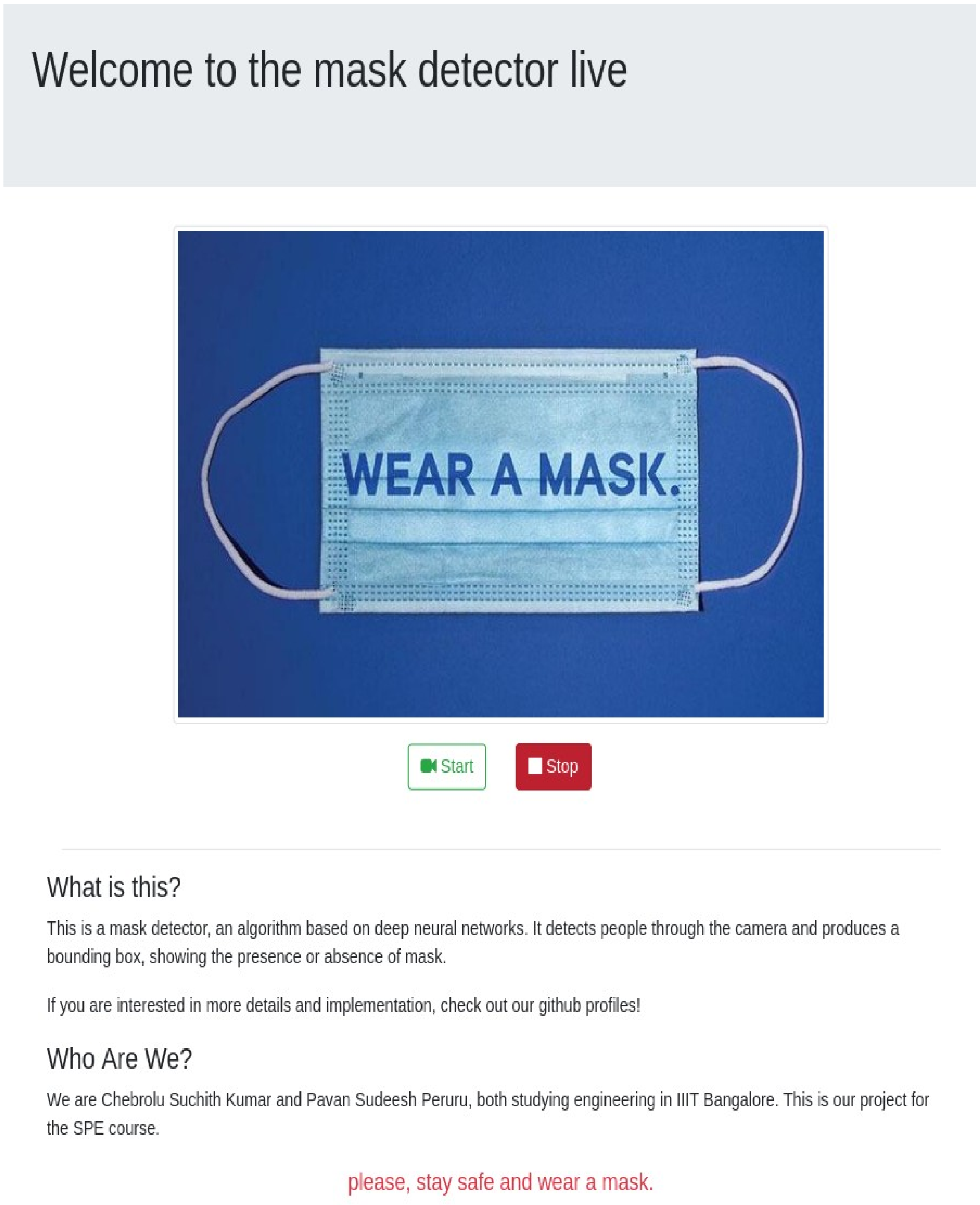
Heroku is a cloud Platform as a Service (PaaS) that we can use that to deploy, manage and to scale apps. For our mask_detector live we have deployed our code into heroku.



```
    - name: Deploy to heroku
      uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
      with:
        heroku_api_key: ${{secrets.HEROKU_API_KEY}}
        heroku_app_name: "mask-detect-live"
        heroku_email: "suchithkumar.ch@gmail.com"
        usedocker: true
```

Fig 14: deployment workflow of mask_detector_live.

After deployment if we use this following link we can run our application.
https://mask-detect-live.herokuapp.com/



Welcome to the mask detector live

What is this?

This is a mask detector, an algorithm based on deep neural networks. It detects people through the camera and produces a bounding box, showing the presence or absence of mask.

If you are interested in more details and implementation, check out our github profiles!

Who Are We?

We are Chebrolu Suchith Kumar and Pavan Sudeesh Peruru, both studying engineering in IIIT Bangalore. This is our project for the SPE course.

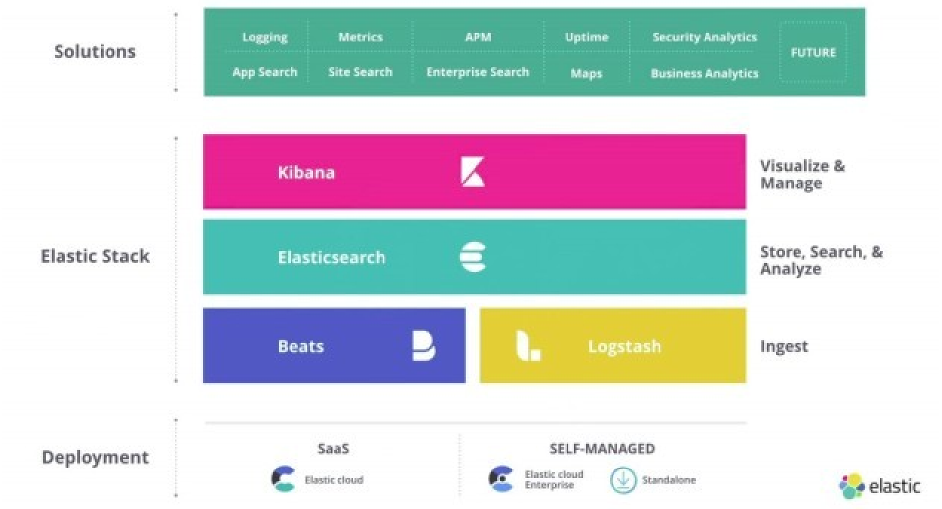
please, stay safe and wear a mask.

fig15: This is how the heroku interface looks.

4.7 Monitoring using ELK

Continuous monitoring provides near immediate feedback and insight into performance and interactions across the network.

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana .
● Elasticsearch is a search and analytics engine.
● Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch.
● Kibana lets users visualize data with charts and graphs in Elasticsearch.



4.7.1 Logstash, configuration file:

Downloading Logstash:
https://www.elastic.co/downloads/logstash
Download appropriate version for your OS. (Ex. LinuxX86_64)
After the tar.gz file is downloaded, use this command to extract

```
$tar -xvf path/to/file/tar.gz
```

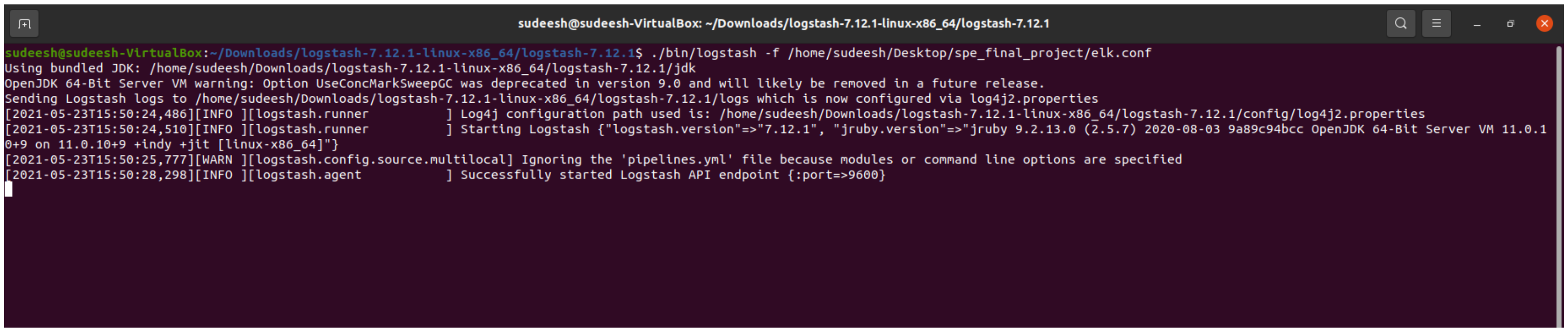
Configuration file:
The bare structure of the configuration file of logstash is

```
input {
plugin {
settings
}
}
filter {
plugin {
settings
}
}
output {
plugin{
settings
}
}
```

To run logstash, we have to input a configuration file on how to parse the data of a given log file.
Go into the extracted logstach directory and execute this command to run.

```
$./bin/logstash -f /path/to/configuration/file
```

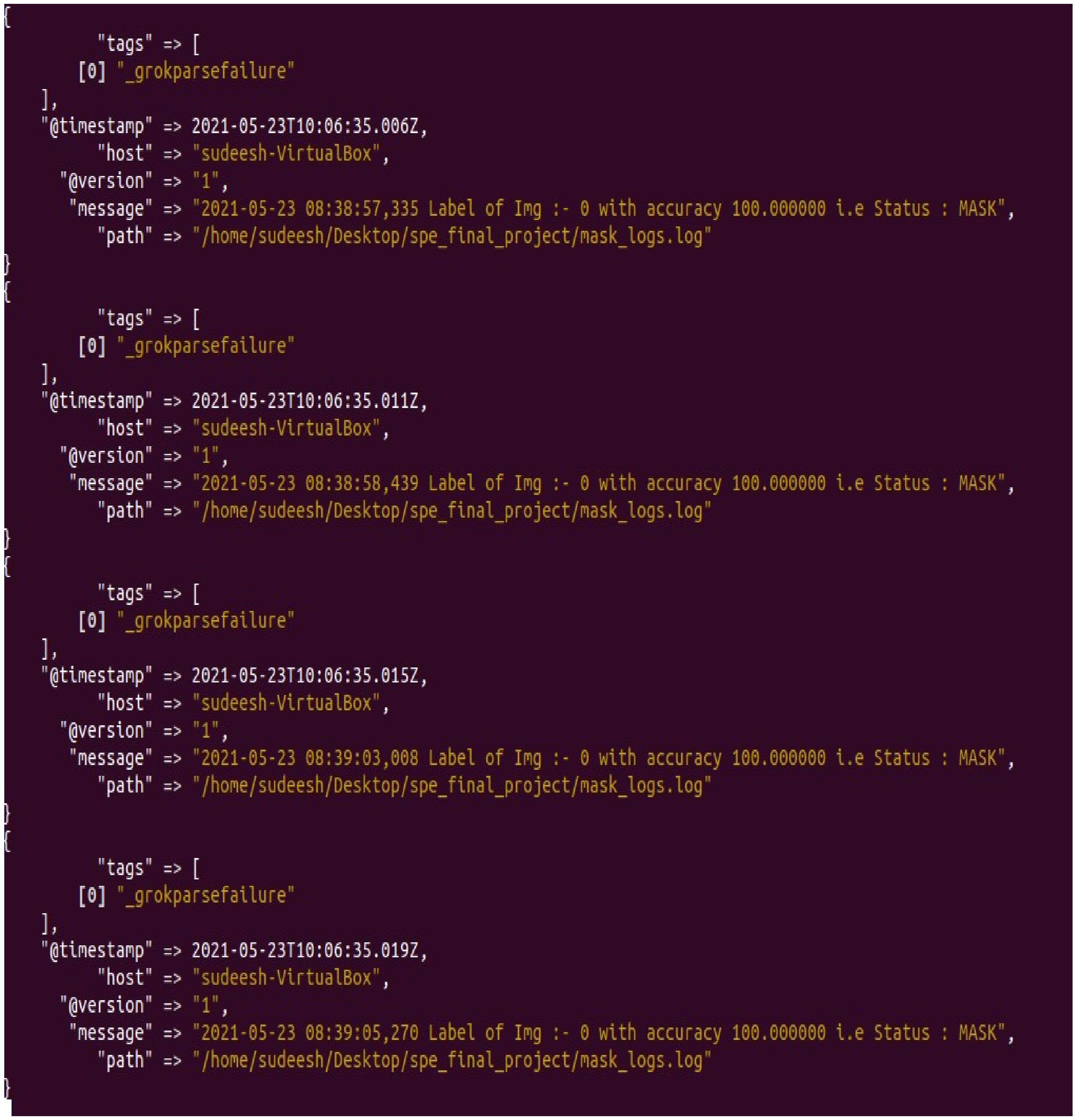
Command and the Output :



sudeesh@sudeesh-VirtualBox: ~/Downloads/logstash-7.12.1-linux-x86_64/logstash-7.12.1

```
sudeesh@sudeesh-VirtualBox:~/Downloads/logstash-7.12.1-linux-x86_64/logstash-7.12.1$ ./bin/logstash -f /home/sudeesh/Desktop/spe_final_project/elk.conf
Using bundled JDK: /home/sudeesh/Downloads/logstash-7.12.1-linux-x86_64/logstash-7.12.1/jdk
OpenJDK 64-Bit Server VM warning: Option UseConcMarkSweepGC was deprecated in version 9.0 and will likely be removed in a future release.
Sending Logstash logs to /home/sudeesh/Downloads/logstash-7.12.1-linux-x86_64/logstash-7.12.1/logs which is now configured via log4j2.properties
[2021-05-23T15:50:24,486][INFO ][logstash.runner          ] Log4j configuration path used is: /home/sudeesh/Downloads/logstash-7.12.1-linux-x86_64/logstash-7.12.1/config/log4j2.properties
[2021-05-23T15:50:24,510][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"=>"7.12.1", "jruby.version"=>"jruby 9.2.13.0 (2.5.7) 2020-08-03 9a89c94bcc OpenJDK 64-Bit Server VM 11.0.1
0+9 on 11.0.10+9 +indy +jit [linux-x86_64]"}
[2021-05-23T15:50:25,777][WARN ][logstash.config.source.multilocal] Ignoring the 'pipelines.yml' file because modules or command line options are specified
[2021-05-23T15:50:28,298][INFO ][logstash.agent           ] Successfully started Logstash API endpoint {:port=>9600}
```

fig 16:command to run the logstash.



```
{
         "tags" => [
        [0] "_grokparsefailure"
    ],
    "@timestamp" => 2021-05-23T10:06:35.006Z,
          "host" => "sudeesh-VirtualBox",
      "@version" => "1",
       "message" => "2021-05-23 08:38:57,335 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK",
          "path" => "/home/sudeesh/Desktop/spe_final_project/mask_logs.log"
}
{
         "tags" => [
        [0] "_grokparsefailure"
    ],
    "@timestamp" => 2021-05-23T10:06:35.011Z,
          "host" => "sudeesh-VirtualBox",
      "@version" => "1",
       "message" => "2021-05-23 08:38:58,439 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK",
          "path" => "/home/sudeesh/Desktop/spe_final_project/mask_logs.log"
}
{
         "tags" => [
        [0] "_grokparsefailure"
    ],
    "@timestamp" => 2021-05-23T10:06:35.015Z,
          "host" => "sudeesh-VirtualBox",
      "@version" => "1",
       "message" => "2021-05-23 08:39:03,008 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK",
          "path" => "/home/sudeesh/Desktop/spe_final_project/mask_logs.log"
}
{
         "tags" => [
        [0] "_grokparsefailure"
    ],
    "@timestamp" => 2021-05-23T10:06:35.019Z,
          "host" => "sudeesh-VirtualBox",
      "@version" => "1",
       "message" => "2021-05-23 08:39:05,270 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK",
          "path" => "/home/sudeesh/Desktop/spe_final_project/mask_logs.log"
}
```

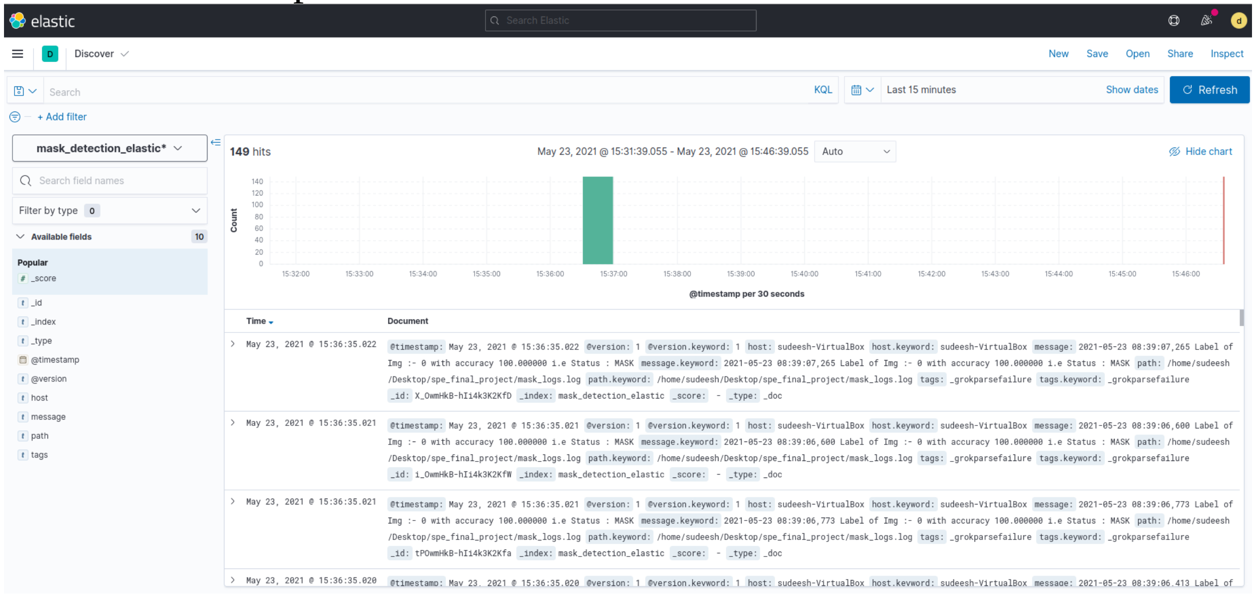
Fig 17: output of running logstashfile.

4.7.2 Elasticsearch, Kibana Visualizations:

Download the Elasticsearch archive for your OS https://www.elastic.co/downloads/elasticsearchDownload the Kibana archive for your OS, https://www.elastic.co/downloads/kibana Or use cloud service for the above. But the configuration file we have to be changed accordingly. Kibana is an open source browser based visualization tool mainly used to analyze large volumes of logs in the form of line graph, bar graph, pie charts, heat maps, region maps, coordinate maps, gauge, goals, timelion etc.

The visualization makes it easy to predict or to see the changes in trends of errors or other significant events of the input source.
With the index pattern we tell kibana what Elasticsearch index to analyze.

Follow the following steps to set up the index pattern:
In Kibana, go to Management → Stack Management Look for Kibana → Index Patterns → Create Index Pattern , set your index pattern based on the index pattern provided in logstash configuration file followed by *. And in the next step select @timestamp as your Time field. Hit Create index pattern , and you are ready to analyze the data. Explore discover and Dashboard options.

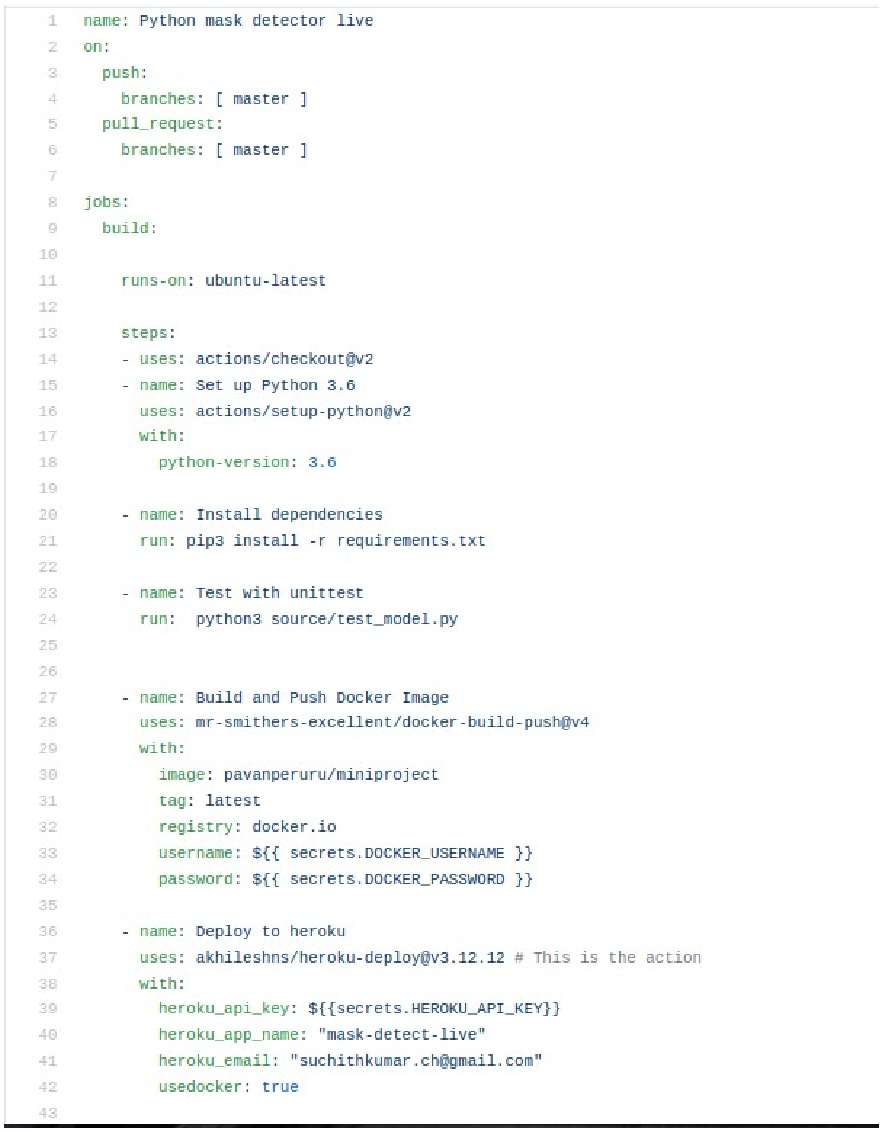


```
May 23, 2021 @ 15:36:35.022  @timestamp: May 23, 2021 @ 15:36:35.022 @version: 1 @version.keyword: 1 host: sudeesh-VirtualBox host.keyword: sudeesh-VirtualBox message: 2021-05-23 08:39:07,265 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK message.keyword: 2021-05-23 08:39:07,265 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK path: /home/sudeesh/Desktop/spe_final_project/mask_logs.log path.keyword: /home/sudeesh/Desktop/spe_final_project/mask_logs.log tags: _grokparsefailure tags.keyword: _grokparsefailure _id: X_OwmHkB-hIi4k3K2KfD _index: mask_detection_elastic _score: - _type: _doc

May 23, 2021 @ 15:36:35.021  @timestamp: May 23, 2021 @ 15:36:35.021 @version: 1 @version.keyword: 1 host: sudeesh-VirtualBox host.keyword: sudeesh-VirtualBox message: 2021-05-23 08:39:06,600 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK message.keyword: 2021-05-23 08:39:06,600 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK path: /home/sudeesh/Desktop/spe_final_project/mask_logs.log path.keyword: /home/sudeesh/Desktop/spe_final_project/mask_logs.log tags: _grokparsefailure tags.keyword: _grokparsefailure _id: i_OwmHkB-hIi4k3K2KfW _index: mask_detection_elastic _score: - _type: _doc

May 23, 2021 @ 15:36:35.021  @timestamp: May 23, 2021 @ 15:36:35.021 @version: 1 @version.keyword: 1 host: sudeesh-VirtualBox host.keyword: sudeesh-VirtualBox message: 2021-05-23 08:39:06,773 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK message.keyword: 2021-05-23 08:39:06,773 Label of Img :- 0 with accuracy 100.000000 i.e Status : MASK path: /home/sudeesh/Desktop/spe_final_project/mask_logs.log path.keyword: /home/sudeesh/Desktop/spe_final_project/mask_logs.log tags: _grokparsefailure tags.keyword: _grokparsefailure _id: tPOwmHkB-hIi4k3K2Kfa _index: mask_detection_elastic _score: - _type: _doc

May 23, 2021 @ 15:36:35.020  @timestamp: May 23, 2021 @ 15:36:35.020 @version: 1 @version.keyword: 1 host: sudeesh-VirtualBox host.keyword: sudeesh-VirtualBox message: 2021-05-23 08:39:06,413 Label of
```

Fig 18: visualizations in kibana.

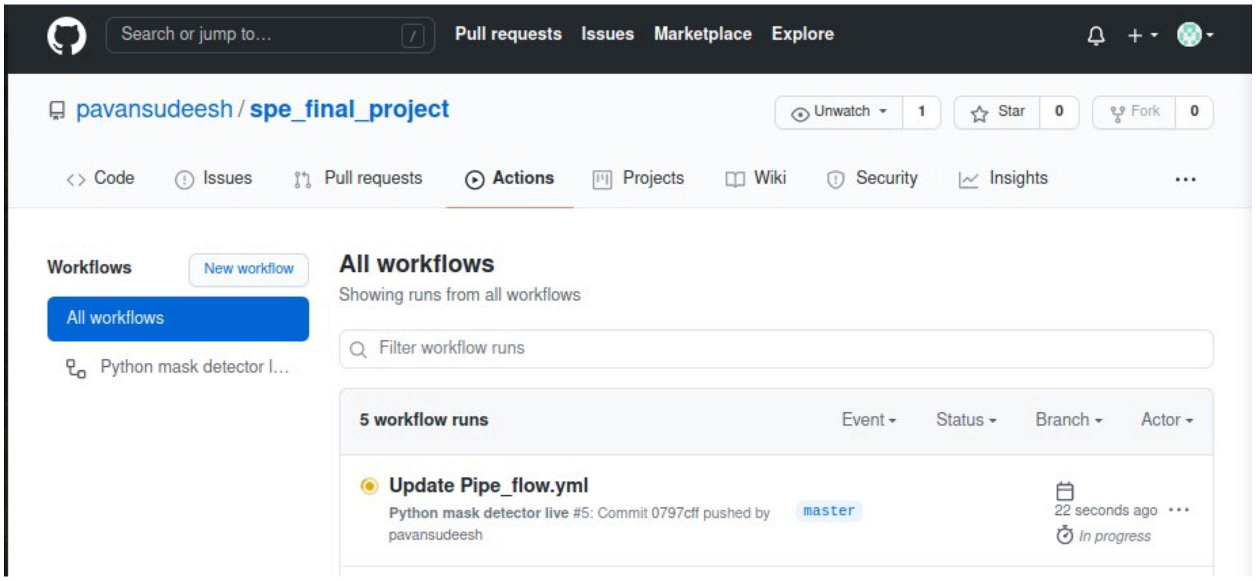
4.8. BUILDING WORKFLOW

From the above sections we have seen how we have integrated docker and other necessary tools with github actions. We were able to build docker images with github actions. Here’s the reference figure to the entire workflow and its final full stage view after a successful run of github workflow..



```yaml
name: Python mask detector live
on:
  push:
    branches: [ master ]
  pull_request:
    branches: [ master ]

jobs:
  build:

    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v2
    - name: Set up Python 3.6
      uses: actions/setup-python@v2
      with:
        python-version: 3.6

    - name: Install dependencies
      run: pip3 install -r requirements.txt

    - name: Test with unittest
      run:  python3 source/test_model.py


    - name: Build and Push Docker Image
      uses: mr-smithers-excellent/docker-build-push@v4
      with:
        image: pavanperuru/miniproject
        tag: latest
        registry: docker.io
        username: ${{ secrets.DOCKER_USERNAME }}
        password: ${{ secrets.DOCKER_PASSWORD }}

    - name: Deploy to heroku
      uses: akhileshns/heroku-deploy@v3.12.12 # This is the action
      with:
        heroku_api_key: ${{secrets.HEROKU_API_KEY}}
        heroku_app_name: "mask-detect-live"
        heroku_email: "suchithkumar.ch@gmail.com"
        usedocker: true

```

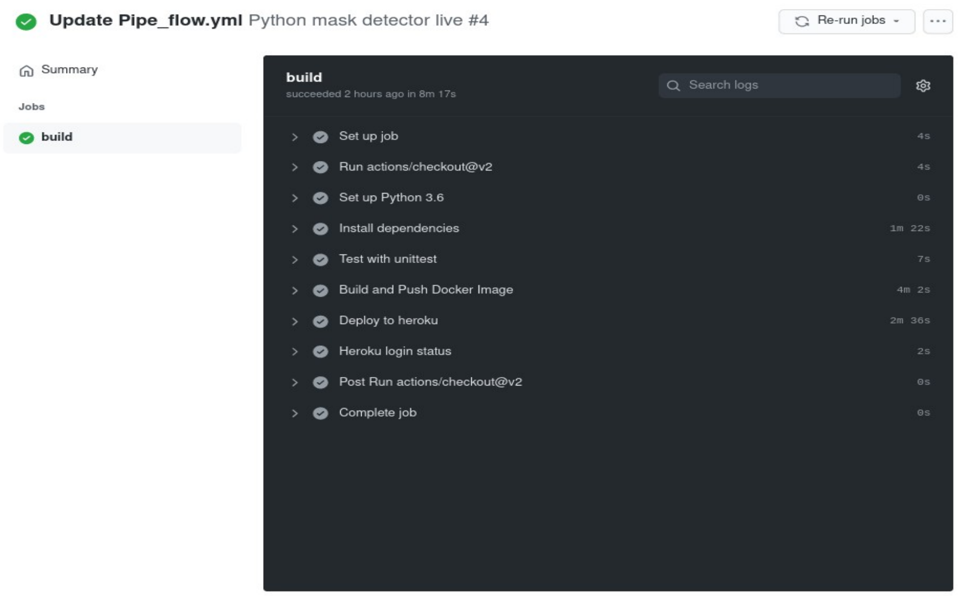
Fig 19: complete workflow of github actions.



```
Update Pipe_flow.yml
Python mask detector live #5: Commit 0797cff pushed by pavansudeesh    master
22 seconds ago
In progress
```

Fig 20: actions page

After completion of building workflow Click on the .yml file and then press build that's it after successful build the following image will show up. If its a green tick then it means that our build is correct.



```
build
succeeded 2 hours ago in 8m 17s

Set up job                          4s
Run actions/checkout@v2             4s
Set up Python 3.6                   0s
Install dependencies            1m 22s
Test with unittest                  7s
Build and Push Docker Image      4m 2s
Deploy to heroku                2m 36s
Heroku login status                 2s
Post Run actions/checkout@v2        0s
Complete job                        0s
```

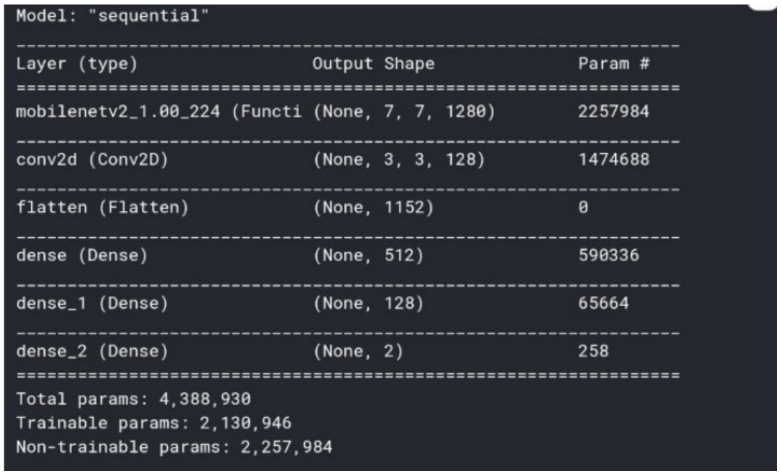
Fig 21: a successful build.

5. Model:

As the input is video, we read each frame and pass that to our mask detector and that will detect the mask accordingly. Mask was detected by using a CNN model.

1) Data set and Preprocessing: Data set has been taken from Kaggle which contains 12K images divided in 10K training, 800 validation images and 992 test images. We augmented the train dataset and then converted the shape of each image to 224x224x3. We also normalized the pixel values.

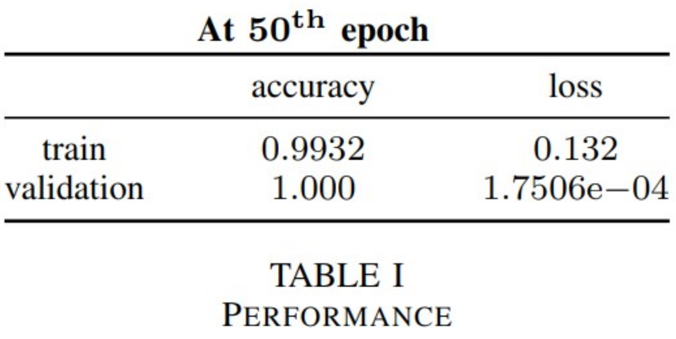
2) Architecture: MobileNet is a streamlined architecture that uses depth wise separable convolutions to construct lightweight deep convolution al neural networks and provides an efficient model for mobile and embedded vision applications. Depth wise separable convolution filters are composed of depth wise convolution filters and point convolution filters. The depth wise convolution filter performs a single convolution on each input channel, and the point convolution filter combines the output of depth wise convolution linearly with 1x1 convolutions. Pre trained weights of Image Net were loaded from TensorFlow. Base layers are frozen and new trainable layers are added. These layers are trained on the collected data set so that it can determine the features to classify a face wearing a mask from a face not wearing a mask. For trainable layers we added one convolutional layer and 3 fully connected layers as mentioned in Fig



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Functi (None, 7, 7, 1280)        2257984
_________________________________________________________________
conv2d (Conv2D)              (None, 3, 3, 128)         1474688
_________________________________________________________________
flatten (Flatten)            (None, 1152)              0
_________________________________________________________________
dense (Dense)                (None, 512)               590336
_________________________________________________________________
dense_1 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 258
=================================================================
Total params: 4,388,930
Trainable params: 2,130,946
Non-trainable params: 2,257,984
```

Fig 22: Model Architecture

We used Adam optimization algorithm as an optimizer and cross entropy loss as a loss function. We used 50 epochs to train the model and accuracy, loss as metrics for validating.



At 50th epoch

	accuracy	loss
train	0.9932	0.132
validation	1.000	1.7506e−04

TABLE I
PERFORMANCE

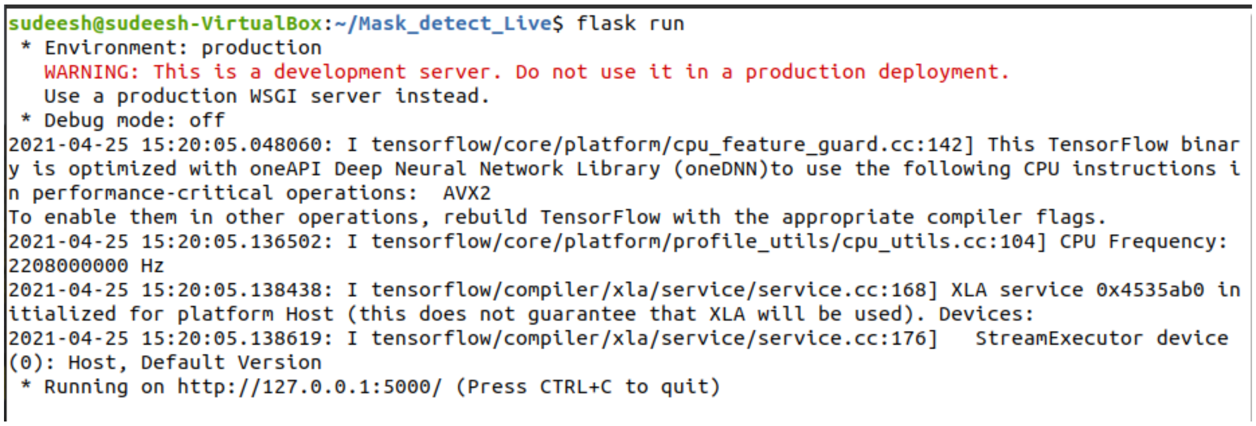
Fig 23: accuracy and loss at 50th epoch.

Time taken to train the model (with GPU) 6.46 minutes and test accuracy on the test data set is 99.79%.

6: Results and discussion:

To run it in the local server we should go the project directory and we should use the following command.

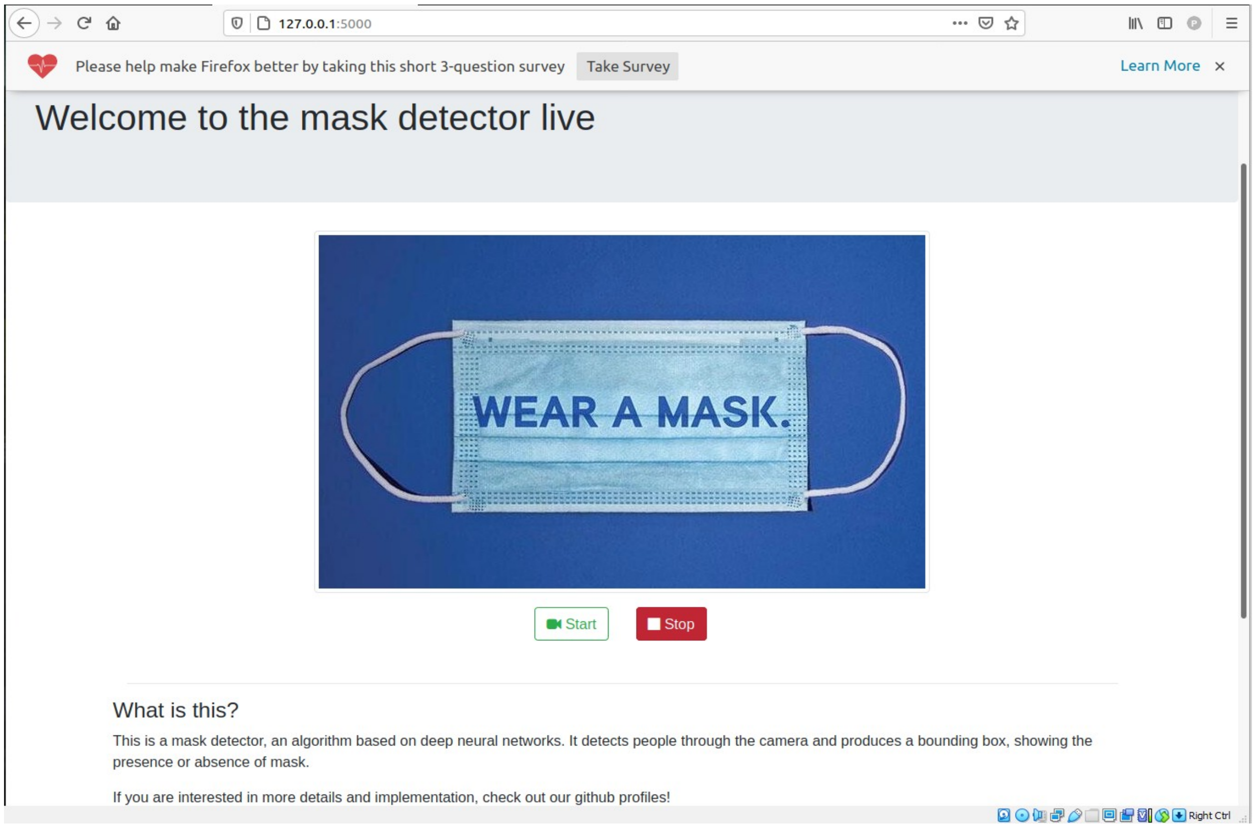
Flask run



```
sudeesh@sudeesh-VirtualBox:~/Mask_detect_Live$ flask run
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
2021-04-25 15:20:05.048060: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binar
y is optimized with oneAPI Deep Neural Network Library (oneDNN)to use the following CPU instructions i
n performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-04-25 15:20:05.136502: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency:
2208000000 Hz
2021-04-25 15:20:05.138438: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x4535ab0 in
itialized for platform Host (this does not guarantee that XLA will be used). Devices:
2021-04-25 15:20:05.138619: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device
(0): Host, Default Version
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Fig 24: Running the code in our local machine.

We can see that in the above figure there is a line saying “**Running on http://127.0.0.1:5000/**”. If we open that link our mask detector local page will be visible in our machines.


127.0.0.1:5000
Please help make Firefox better by taking this short 3-question survey
Take Survey
Learn More
Welcome to the mask detector live
WEAR A MASK.
Start
Stop

What is this?

This is a mask detector, an algorithm based on deep neural networks. It detects people through the camera and produces a bounding box, showing the presence or absence of mask.

If you are interested in more details and implementation, check out our github profiles!

Figure 25: This is how it looks when we open the link in our local machine.

•Then press START for live mask detection.

Around the human subject two boxes going to show up

1 The bigger box shows the category of the subject long with its accuracy. Eg: - Human 99.49%, cat 90.71%, car 99.9%… etc.
2 The smaller box shows mask or no mask along with accuracy value of its prediction. Eg:- mask 90.59%, no mask 99.9 %.

Reference images.

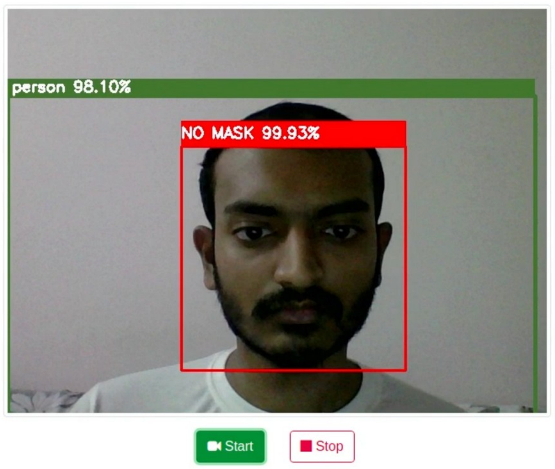


Figure 26: Outer box saying **PERSON** with 96.10% and inner box saying **NO MASK** with 99.93%.

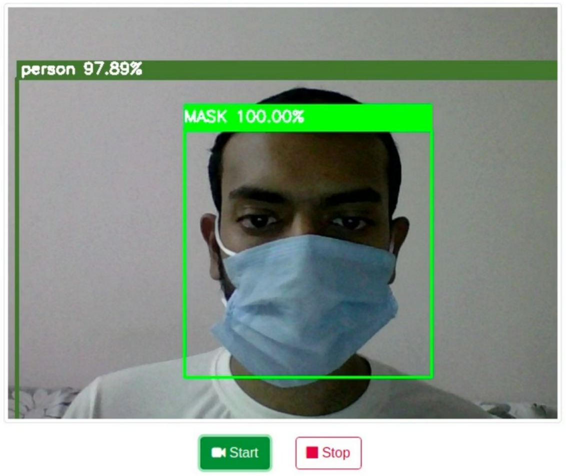
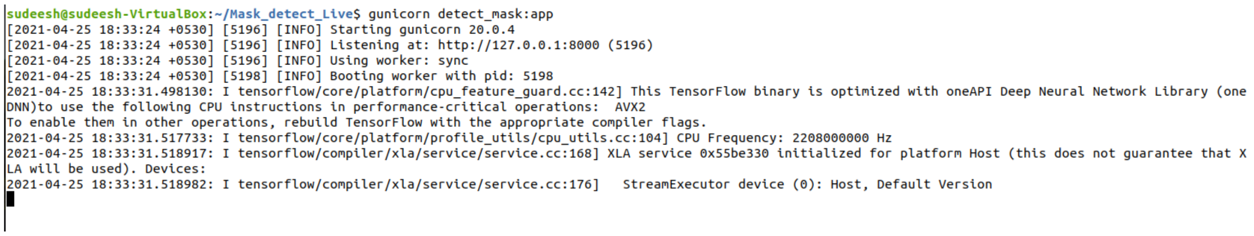


Figure 27: Outer box saying **PERSON** with 97.09% and inner box saying **MASK** with 100.00%.



```
sudeesh@sudeesh-VirtualBox:~/Mask_detect_Live$ gunicorn detect_mask:app
[2021-04-25 18:33:24 +0530] [5196] [INFO] Starting gunicorn 20.0.4
[2021-04-25 18:33:24 +0530] [5196] [INFO] Listening at: http://127.0.0.1:8000 (5196)
[2021-04-25 18:33:24 +0530] [5196] [INFO] Using worker: sync
[2021-04-25 18:33:24 +0530] [5198] [INFO] Booting worker with pid: 5198
2021-04-25 18:33:31.498130: I tensorflow/core/platform/cpu_feature_guard.cc:142] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library (one
DNN)to use the following CPU instructions in performance-critical operations:  AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
2021-04-25 18:33:31.517733: I tensorflow/core/platform/profile_utils/cpu_utils.cc:104] CPU Frequency: 2208000000 Hz
2021-04-25 18:33:31.518917: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x55be330 initialized for platform Host (this does not guarantee that X
LA will be used). Devices:
2021-04-25 18:33:31.518982: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): Host, Default Version
█
```

Flask run can’t be run on the deployment platform. Gunicorn command is used instead.

These are the plots for the mask detection model for 50 epochs:

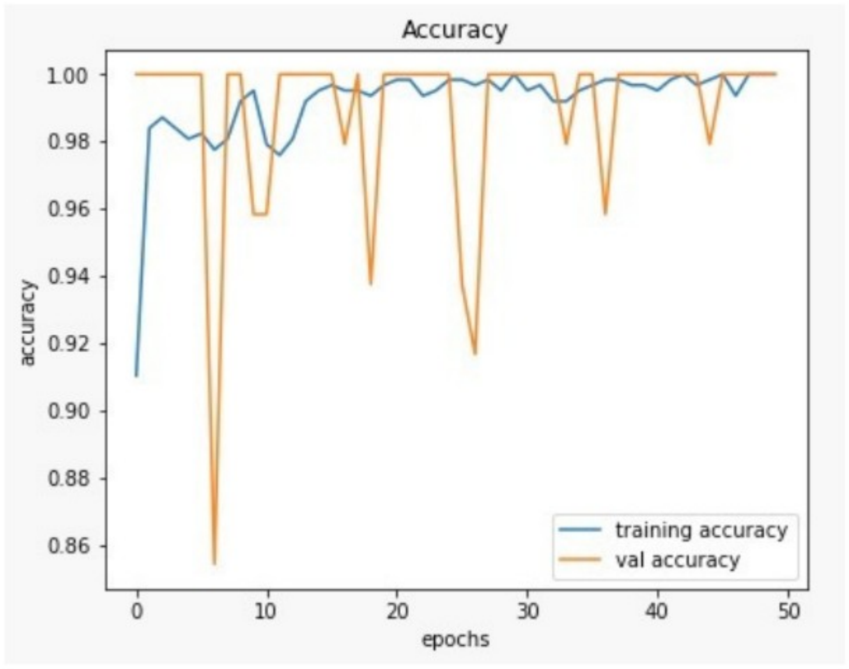


Fig 28: accuracy

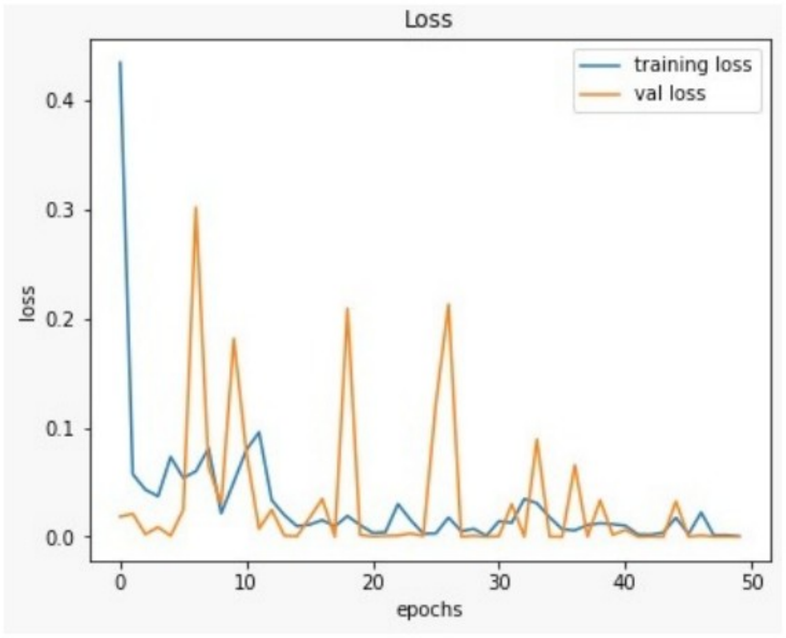


Fig 29: loss plot.

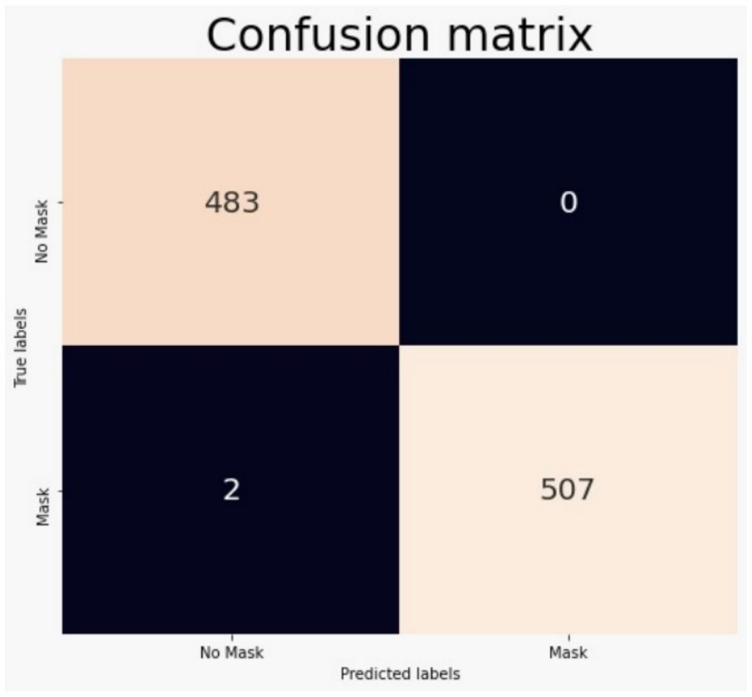
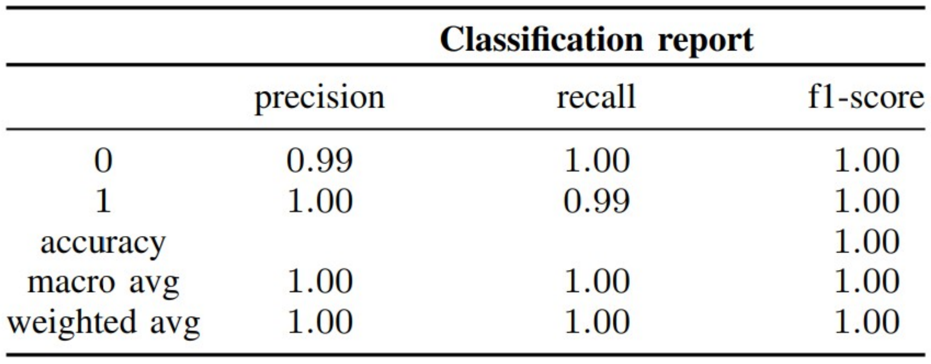


Fig 30: confusion matrix.



	Classification report		
	precision	recall	f1-score
0	0.99	1.00	1.00
1	1.00	0.99	1.00
accuracy			1.00
macro avg	1.00	1.00	1.00
weighted avg	1.00	1.00	1.00

TABLE II

Fig 31: classification report

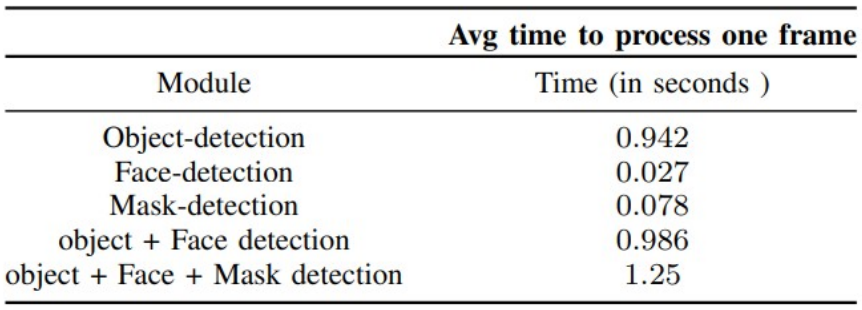
7 FUTURE SCOPE AND CHALLENGES

There is also a case where person wears a mask but not properly. We tried to detect that case by detecting nose and mouth similar to detecting face. The condition we kept here is if we detect mask and then if we detect nose or mouth then the person is not wearing mask properly. While performing this approach, we got some noise and it is taking more time to classify for one frame which affects the real time performance of the model. For future works we can tune the parameters for detecting nose and mouth with less noise and we can use GPU to solve real time performance in this case. We also want to try this in real time using IOT.

8 CONCLUSION

The Devops methodology and life cycle tools prove to be better than the Agile methodology in terms of technical, cultural and business benefits. By minimizing friction between independent teams, DevOps enables a collaborative approach for enterprise software development and delivery that reflects the needs of the entire application life cycle for today's modern enterprises. Thus, we can develop, test, deploy and monitor the application easily.

Accuracy Figure tells us that the model increases its test accuracy and converges after 50 epochs and Figure 4 tells us that the test loss keeps decreasing and converges after 50 epochs. Hence the final model has been taken after 50th epoch. Table II and Table III gives us the real-time performance and accuracy of each module. These values are obtained when run on a local machine with CPU.



	Avg time to process one frame
Module	Time (in seconds)
Object-detection	0.942
Face-detection	0.027
Mask-detection	0.078
object + Face detection	0.986
object + Face + Mask detection	1.25

TABLE III

Fig 32: times to process all the frames.

Overall the model can perform much better with high processing system and further reduce the processing time for each frame. After Detecting a person with no mask our model plays audio to alert the user as to wear the mask to enter the room. This is our complete model for vision based automatic door entry system which detects persons with or without masks.

9 Links:

https://github.com/pavansudeesh/spe_final_project

https://hub.docker.com/repository/docker/pavanperuru/miniproject

https://mask-detect-live.herokuapp.com/

https://www.kaggle.com/ashishjangra27/face-mask-12k-images-dataset

10 REFERENCES:

[1] https://towardsdatascience.com/yolo-object-detection-with-opencv-and-python-21e50ac599e9
[2] https://github.com/pjreddie/darknet/blob/master/cfg/yolov3.cfg
[3] https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c
[4] https://www.mygreatlearning.com/blog/viola-jones-algorithm/
[5] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7775036/

[6] https://www.pyimagesearch.com/2020/05/04/covid-19-face-mask-detector-with-opencv-keras-tensorflow-and-deep-learning/
[7] https://docs.github.com/en/actions/learn-github-actions/introduction-to-github-actions#:~:text=GitHub%20Actions%20are%20event%2Ddriven,executes%20a%20software%20testing%20script.