COVID-19 HELPER USING ALEXA AND RASPBERRY-PI

BY

SIDDHARTH M NAIR 18BCE1238

VARSHA R 18BCE1344

RAM GUNASEKARAN 18BCE1234

C.K.M.GANESAN 18BCE1266

ABHISHEK MOHAN 18BCE1192

A project report submitted to

Dr. Bala Murugan M S

SCHOOL OF ELECTRONICS ENGINEERING

in partial fulfilment of the requirements for the course of

ECM1003 – Cyber Physical System

in

B.Tech. ELECTRONICS AND COMPUTER ENGINEERING

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

VIT CHENNAI

Vandalur – Kelambakkam Road

Chennai – 600127

NOVEMBER 2020

BONAFIDE CERTIFICATE

Certified that this project report entitled "COVID-19 HELPER USING ALEXA AND RASPBERRY PI**"** is a bonafide work of **SIDDHARTH M NAIR 18BCE1238,VARSHA R 18BCE1344,RAM GUNASEKARAN A 18BCE1234, C.K.M.GANESAN 18BCE1266, ABISHEK MOHAN 18BCE1192** who carried out the Project work under my supervision and guidance for **ECM1003 – Cyber Physical System.**

Dr. Bala Murugan M S

Associate Professor

School of Electronics Engineering (SENSE),

VIT Chennai

Chennai – 600 127.

ABSTRACT

After the outbreak of COVID-19 virus there is high necessity to ensure social distancing, face mask and it is imperative to take necessary preventive measures. The proposed project is a perfect application for this scenario. The system is automatically triggered when someone ring the door bell. It uses the mic and camera attached at the door to ensure whether the person is wearing a mask and the person inside the home can ask their alexa about the person outside. The system triggers the alexa skills whenever someone rings the door bell.A complete description of how the person is looking whether he is wearing a mask,or is he coughing ,if more than two people the system can say whether they are six feet apart. Apart from these, the custom skill trained alexa can get our location and can give a detailed report about the number of cases in the device location.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Prof. Bala Murugan M S,** Associate Professor, School of Electronics Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr Sivasubramanian A**, Dean of the School of Electronic Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We express our thanks to our Head of the Department **Dr.P.VETRIVELAN** for his support throughout the course of this project.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

We thank our parents, family, and friends for bearing with us throughout the course of our project and for the opportunity they provided us in undergoing this course in such a prestigious institution.

SIDDHARTH M NAIR VARSHA R RAM GUNASEKARAN C.K.M.GANESAN ABISHEK MOHAN

TABLE OF CONTENTS

1. INTRODUCTION

1.1 OBJECTIVES AND GOALS

- Develop a deep learning model to detect face mask
- Develop a deep learning model to ensure social distancing
- Develop an alexa skill which responds in natural language when someone asks a question about face mask and social distancing.
- Design a system with raspberry-pi to support all of the above mentioned points

1.2 BENEFITS

At the current scenario it is now an necessity for everyone to wear a face mask and maintain social distancing.But it is not enough only for us to stick to the morns.It is also important for the ones we interact to also wear a mask and maintain social distancing.When someone knocks our door it is an imperative to check whether they follow all the preventive measures.With the help of this project we can make sure that the ones who are waiting outside follow all the norms with a simple question to ALEXA

Apart from this it is important to be aware of the conditions of the area we live in.So with a single question you can get all the updated number of cases in your locality.In this way our project proves to be of great help to the user

1.3 FEATURES

- Detect whether the person is wearing face mask or not
- Detect whether people adhere to social distancing protocols
- Alexa skill set for face mask and social distancing integrated to your Amazon echodot.
- Microcontroller used is RaspberryPi
- Python language is used to train and test the deep learning models
- Get the updated number of cases and transfer it to the cloud in real time
- The cloud service used is Amazon Web Services
- Alexa developer Console is used to build the alexa skill

2 COVID-19 HELPER USING ALEXA AND RASPBERRY-PI

2.1 LITRATURE REVIEW

According to Maneesh Ayi, Ajay Kamal Ganti , Maheswari Adimulam , Badiganti Karthik , Manisha Banam , G Vimala Kumar **Face Tracking and Recognition Using MATLAB and Arduino(2017)** they made use of Viola Jones algorithm for the detection and tracking of the face and PCA and Eigen face approach for face recognition. The aim of their paper with the face detection was meant for real time applications like security systems and video surveillance.[1]

In continuation, the work done by VENKATA SASANK PAMULAPATI , YEKULA SUMITH ROHAN , VEMULA SAI KIRAN , SARANU SANDEEP , MARAM SRINIVASA RAO **Real-time Face Tracking using MATLAB and Arduino(2018)**,made use of the Viola-Jones algorithm which they had used to identify a face which was present in every frame present in the video and they had used the still imagesend signals to the Arduino board to control the movement of the camera using two servo motors.[2]

In the paper **Incorporating skin color for improved face detection and tracking system in** the year 2016 written by B.Dahal , Abeer Alsadoon , P.W.C. Prasad and Amr Elchouemi, they detect the skin colour and do gender classification. To achieve this goal, they use a hybrid method where they try out various combinations using the RGB colour codes. They perform morphological operations on this and get the final result as a detected dace with the required features identified.[3]

Also, M. N. Mohammed , Halim Syamsudin , S. Al-Zubaidi , Sairah A.K. , Rusyaizila Ramli and Eddy Yusuf in the paper "**NOVEL COVID-19 DETECTION AND DIAGNOSIS SYSTEM USING IOT BASED SMART HELMET"** paid close attention to the fact that the one of the most common symptoms of corona virus is fever and the lack of availability of doctors to check each and every patient for every symptom is extremely difficult. Hence, this paper focuses on using smart helmet with Mounted Thermal Imaging System. A thermal camera is placed in the helmet and when it is combined with the technologies of IOT, it can be immediately be put into action and it can detect a pedestrians temperature, this is one of the best use in the present age as it can help to prevent the spread of the coronavirus spreading wider.[4]

Harshal V. Khodaskar , Shashank Mane in the paper **Human Face Detection & Recognition Using Raspberry Pi (2017)**" stated the importance of face detection in this present situation and paid attention to how he implemented it. They made use of Raspberry pi which was low cost and dismisses noise and blurs the additional effects making it efficient. They made a conclusion that the system they proposed can recognise faces even if the images have poop quality .[5]

Aydin Teyhouee, Nathaniel D. Osgood in the paper **Cough Detection Using Hidden Markov Models** investigated the ability to which a simple machine learning approach in the form of Hidden Markov Models.(HMMs) could be used to classify different states of coughing using univariate (with a single energy band as the input feature) and multivariate (with a multiple energy band as the input features) binned time series using both of cough data. They further used the model to distinguish cough events from other events and environmental noise. Their Hidden Markov algorithm achieved

92% AUR (Area Under Receiver Operating Characteristic Curve) in classifying coughing events in noisy environments. Moreover, comparison of univariate with multivariate HMMs suggest a high accuracy of multivariate HMMs for cough event classifications.[6]

Cough detection using deep neural networks by Jia-Ming Liu , Mingyu You , Zheng Wang ,Guo-Zheng Li , Xianghuai Xu ,Zhongmin Qiu gives an idea abou how A two step cough detection system is built based on deep neural networks(DNN) and hidden markov model(HMM). The experimental data set contains audio recordings from 20 patients with each recording lasting for about 24 hours. The performances of the newly proposed system were evaluated via sensitivity, specificity, F1 measure and macro average of recall. Different configurations of deep neural networks are evaluated. Experimental results show that many of the DNN configurations outperform Gaussian Mixture Model (GMM) on sensitivity, specificity and F1 measure respectively. On macro average of recall, 13.38% and 22.0% relative error reduction are achieved. The newly proposed system provides better performance and potential capacity for modeling big audio data on the cough detection task[7]

Raphael Leong in the paper **Analyzing the Privacy Attack Landscape for Amazon Alexa Devices** investigates the potential privacy issues that may emerge in the context of always-on speakers that are now prevalent in people’s homes. They analyzed all the potential security and privacy issues surrounding Alexa skills, Echo firmware and third-party hardware that acts as alternatives to the Echo device. Our core plan initially was to evaluate and statically analyze all the available Alexa skill repositories that were made public by developers on Github. They then branched out to look at issues regarding Echo firmware which is based off the Android OS and also review the state of third-party Alexa devices.[8]

Amazon S3 for Science Grids: a Viable Solution? by Mayur Palankar , Adriana Iamnitchi , Matei Ripeanu , Simson Garfinkel evaluates S3's ability to provide storage support to large-scale science projects from a cost, availability, and performance perspective.They also identify a set of additional functionalities that storage services targeting data-intensive science applications should support. Finally they propose unbundling the success metrics for storage utility performance as a solution, to reduce storage costs.[9]

Serverless computing provides on-demand high performance computing for biomedical research by Dimitar Kumanov1, Ling-Hong Hung1, Wes Lloyd1 , Ka Yee Yeung1 demonstrate how serverless computing provides low cost access to hundreds of CPUs, on demand, with little or no setup. In particular, they illustrate that the all-against-all pairwise comparison among all unique human proteins can be accomplished in approximately 2 minutes, at a cost of less than $1, using Amazon Web Services Lambda. This is a 250x speedup compared to running the same task on a typical laptop compute.[10]

Face Recognition Using Raspberry PI by Shruti Ambre,Mamata Masurekar,Shreya Gaikwad explores a real-time face recognition system using easily-attainable components and libraries, such as Raspberry PI and Dlib, Face Recognition library and Open Source Computer Vision Library (OpenCV). It also covers various face recognition machine learning algorithms. The results show that in real-time applications, the system runs at 2 frames per second and recognizes faces despite Raspberry PI’s limitations such as low CPU and GPU processing power.[11]

2.2 BLOCK DIAGRAM

The four main features of the basic block diagram (given below) are

- The camera connected to the raspberry pi
- Simple storage services bucket for transfer of data-
- Amazon lambda as an endpoint for AWS
- IOT Alexa skill for alexa skill development.

Each of the above units must be programmed properly for the proper working as a whole system.

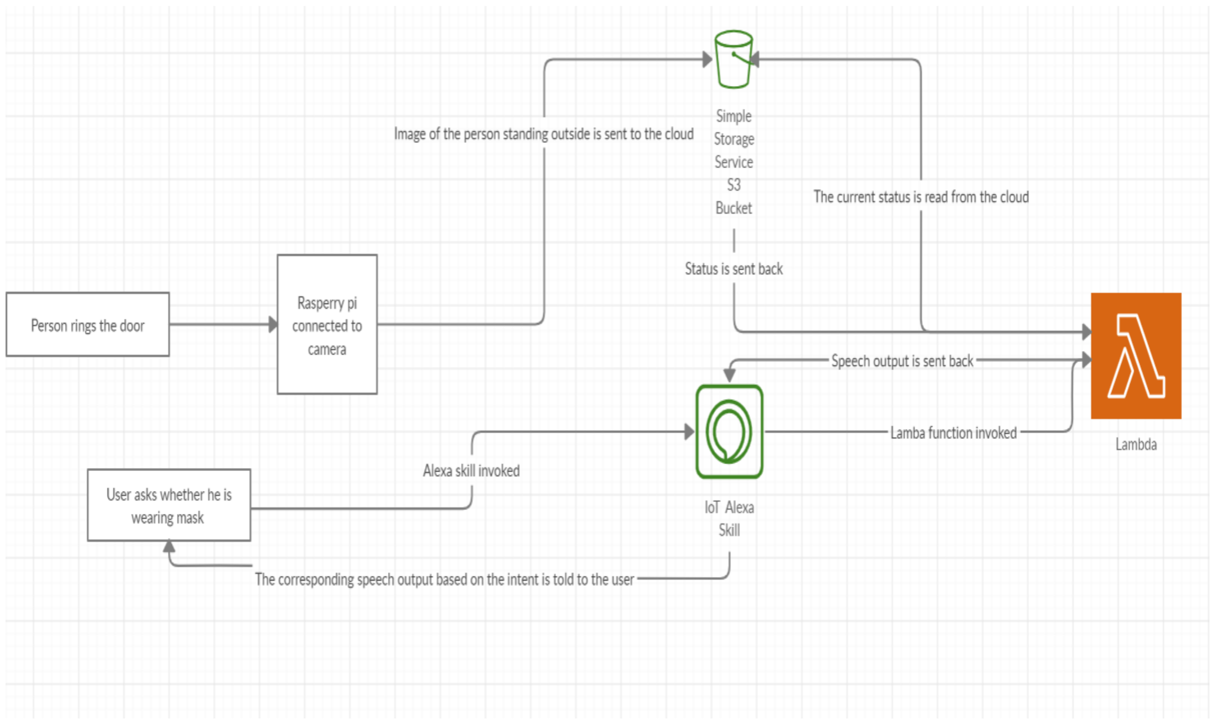

Simple Storage Service S3 Bucket
Image of the person standing outside is sent to the cloud
The current status is read from the cloud
Status is sent back
Person rings the door
Rasperry pi connected to camera
Speech output is sent back
Lambda
Lamba function invoked
Alexa skill invoked
User asks whether he is wearing mask
IoT Alexa Skill
The corresponding speech output based on the intent is told to the user

Figure 1. Overall architecture of the system.

Figure 1 shows an idea of how the system look like . The secondary level block diagram is shows how the face mask and social distancing looks like.

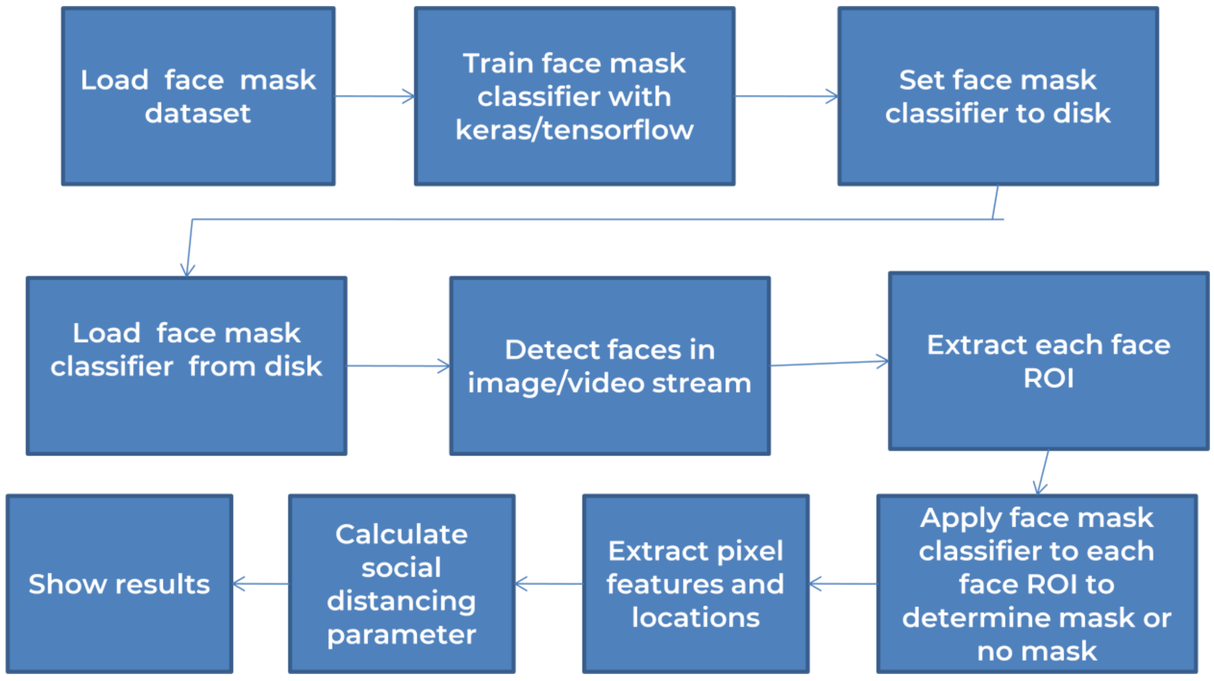

Load face mask dataset
Train face mask classifier with keras/tensorflow
Set face mask classifier to disk
Load face mask classifier from disk
Detect faces in image/video stream
Extract each face ROI
Apply face mask classifier to each face ROI to determine mask or no mask
Extract pixel features and locations
Calculate social distancing parameter
Show results

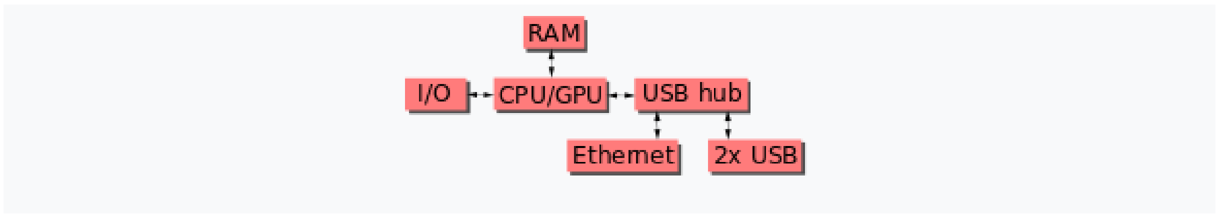
Figure 2: Face mask and social distancing

Figure 2 gives a clear idea of how the face mask and social distancing models are built.It gives the overall algorithm of face mask detection and social distancing.

2.3 HARDWARE AND SOFTWARE ANALYSIS

2.3.1 Raspberry Pi

The Raspberry Pi hardware has evolved through several versions that feature variations in the type of the central processing unit, amount of memory capacity, networking support, and peripheral-device support.



This block diagram describes Model B and B+; Model A, A+, and the Pi Zero are similar, but lack the Ethernet and USB hub components. The Ethernet adapter is internally connected to an additional USB port. In Model A, A+, and the Pi Zero, the USB port is connected directly to the system on a chip (SoC). On the Pi 1 Model B+ and later models the USB/Ethernet chip contains a five-port USB hub, of which four ports are available, while the Pi 1 Model B only provides two. On the Pi Zero, the USB port is also connected directly to the SoC, but it uses a micro USB (OTG) port. Unlike all other Pi models, the 40 pin GPIO connector is omitted on the Pi Zero with solderable through holes only in the pin locations. The Pi Zero WH remedies this.

Processor speed ranges from 700 MHz to 1.4 GHz for the Pi 3 Model B+ or 1.5 GHz for the Pi 4; on-board memory ranges from 256 MiB to 1 GiB random-access memory (RAM), with up to 8 GiB available on the Pi 4. Secure Digital (SD) cards in MicroSDHC form factor (SDHC on early models) are used to store the operating system and program memory. The boards have one to five USB ports. For video output, HDMI and composite video are supported, with a standard 3.5 mm tip-ring-sleeve jack for audio output. Lower-level output is provided by a number of GPIO pins, which support common protocols like I^2C. The B-models have an 8P8C Ethernet port and the Pi 3, Pi 4 and Pi Zero W have on-board Wi-Fi 802.11n and Bluetooth.

2.3.2 Amazon echo

Amazon Echo (shortened to Echo) is a brand of smart speakers developed by Amazon. Echo devices connect to the voice-controlled intelligent personal assistant service Alexa, which will respond when you say "Alexa". Users may change this wake word to "Amazon", "Echo" or "Computer". The features of the device include: voice interaction, music playback, making to-do lists, setting alarms, streaming podcasts, and playing audiobooks, in addition to providing weather, traffic and other real-time information. It can also control several smart devices, acting as a home automation hub. The smart speaker needs to use Wi-Fi to connect to Internet, there is no Ethernet port. In the default mode, the device continuously listens to all speech, monitoring for the wake word to be spoken, which is primarily set up as "Alexa" (derived from Alexa Internet, the Amazon-owned Internet indexing company). Echo's microphones can be manually disabled by pressing a mute button to turn off the audio processing circuit.Echo devices require a wireless Internet connection in order to work. Echo's voice recognition capability is based on Amazon Web Services and the voice platform Amazon acquired from Yap, Evi, and IVONA.The smart speakers perform well with a "good" (low-latency) Internet connection, which minimizes processing time due to minimal communication round trips, streaming responses and geo-distributed service endpoints. While the application is free, an Amazon account is required, and setup is not possible without one.

The Amazon Echo is able to connect to many different smart home devices. Thermostats, humidifiers, lightbulbs, plugs, dog and cat feeder, door locks, cameras, thermostats, security systems, speakers, WiFi, televisions, vacuums, microwaves, printers, and other smart home devices can now all be controlled through Alexa. Alexa can be used to activate and deactivate all of these smart home appliances, as well as change their settings depending on the device. For example, Alexa can be used to change the temperature in a house through a thermostat, turn off the lights with smart lights, put out dog or cat food via a smart pet feeder, and activate the security systems via a smart security system. The user is able organize these smart home devices by putting them into groups. For example, a user can make a "Music Group" on the Amazon Echo. The Amazon Echo will be able to play music and other media in multiple rooms in a house through other Echos and speakers that are in the "Music Group". Along with multiple groups, an Amazon Echo can hold multiple profiles. Switching between the profiles can allow users to play their music, access their calendars, and use their accounts for shopping, instead of just using one person's.

2.3.3 Amazon S3

Amazon S3 or **Amazon Simple Storage Service** is a service offered by Amazon Web Services (AWS) that provides object storage through a web service interface. Amazon S3 uses the same scalable storage infrastructure that Amazon.com uses to run its global e-commerce network. Amazon S3 can be employed to store any type of object which allows for uses like storage for Internet applications, backup and recovery, disaster recovery, data archives, data lakes for analytics, and hybrid cloud storage. Although Amazon Web Services (AWS) does not publicly provide the details of S3's technical design, Amazon S3 manages data with an object storage architecture which aims to provide scalability, high availability, and low latency with 99.999999999% durability and between 99.95% to 99.99% availability (though there is no service-level agreement for durability).The basic storage units of Amazon S3 are objects which are organized into buckets. Each object is identified by a unique, user-assigned key.Buckets can be managed using either the console provided by Amazon S3, programmatically using the AWS SDK, or with the Amazon S3 REST application programming interface (API).

Objects can be managed using the AWS SDK or with the Amazon S3 REST
API and can be up to five terabytes in size with two kilobytes of metadata. Additionally, objects can be downloaded using the HTTP GET interface and the BitTorrent protocol.Requests are authorized using an access control list associated with each object bucket and support versioning which is disabled by default. Amazon S3 can be used to replace significant existing (static) web-hosting infrastructure with HTTP client accessible objects.The Amazon AWS authentication mechanism allows the bucket owner to create an authenticated URL which is valid for a specified amount of time.Every item in a bucket can also be served as a BitTorrent feed. The Amazon S3 store can act as a seed host for a torrent and any BitTorrent client can retrieve the file. This can drastically reduce the bandwidth cost for the download of popular objects. While the use of BitTorrent does reduce bandwidth, AWS does not provide native bandwidth limiting and, as such, users have no access to automated cost control. This can lead to users on the free-tier of Amazon S3, or small hobby users, amassing dramatic bills. A bucket can be configured to save HTTP log information to a sibling bucket; this can be used in data mining operations.There are various User Mode File System (FUSE)-based file systems for Unix-like operating systems (Linux, etc.) that can be used to mount an S3 bucket as a file system such as S3QL. The semantics of the Amazon S3 file system are not that of a POSIX file system, so the file system may not behave entirely as expected

2.3.4 Amazon Lambda

AWS Lambda is an event-driven, serverless computing platform provided by Amazon as a part of Amazon Web Services. It is a computing service that runs code in response to events and automatically manages the computing resources required by that code. It was introduced in November 2014.The purpose of Lambda, as compared to AWS EC2, is to simplify building smaller, on-demand applications that are responsive to events and new information. AWS targets starting a Lambda instance within milliseconds of an event. Node.js, Python, Java, Go Rubyand C# (through .NET Core) are all officially supported as of 2018. In late 2018, custom runtime support was added to AWS Lambda, giving developers the ability to run a Lambda in the language of their choice.AWS Lambda supports securely running native Linux executables via calling out from a supported runtime such as Node.js. For example, Haskell code can be run on Lambda.AWS Lambda was designed for use cases such as image or object uploads to Amazon S3, updates to DynamoDB tables, responding to website clicks or reacting to sensor readings from an IoT connected device. AWS Lambda can also be used to automatically provision back-end services triggered by custom HTTP requests, and "spin down" such services when not in use, to save resources. These custom HTTP requests are configured in AWS API Gateway, which can also handle authentication and authorization in conjunction with AWS Cognito.

Unlike Amazon EC2, which is priced by the hour but metered by the second, AWS Lambda is metered in increments of 100 milliseconds. Usage amounts below a documented threshold fall within the AWS Lambda free tier - which does not expire 12 months after account signup, unlike the free tier for other AWS services.In 2019, at AWS' annual cloud computing conference (AWS re:Invent), the AWS Lambda team announced "Provisioned Concurrency", a feature that "keeps functions initialized and hyper-ready to respond in double-digit milliseconds." The Lambda team described Provisioned Concurrency as "ideal for implementing interactive services, such as web and mobile backends, latency-sensitive microservices, or synchronous APIs."Each AWS Lambda instance is a container created from Amazon Linux AMIs (a Linux distribution related to RHEL) with 128-3008 MB of RAM (in 64 MB increments), 512 MB of ephemeral storage (available in /tmp, the data lasts only for the duration of the instance, it gets discarded after all the tasks running in the instance complete) and a configurable execution time from 1 to 900 seconds. The instances are neither started nor controlled directly. Instead, a package containing the required tasks has to be created and uploaded (usually) to an S3 bucket and AWS is instructed (via Amazon Kinesis, DynamoDB or SQS) to run it when an event is triggered. Each such execution is run in a new environment so access to the execution context of previous and subsequent runs is not possible. This essentially makes the instances stateless, all the incoming and outgoing data needs to be stored by external means (usually via S3 or DynamoDB, inbound connections to the instances is disabled). The maximum compressed size of a Lambda package is 50 MB with the maximum uncompressed size being 250 MB.

2.3.5 Alexa Skill Set

Amazon allows developers to build and publish skills for Alexa using the Alexa Skills Kit known as Alexa Skills. These third-party-developed skills, once published, are available across Alexa-enabled devices. Users can enable these skills using the Alexa app.A "Smart Home Skill API" is available, meant to be used by hardware manufacturers to allow users to control smart home devices.Most skills run code almost entirely in the cloud, using Amazon's AWS Lambda service.In April 2018, Amazon launched Blueprints, a tool for individuals to build skills for their personal use.In February 2019, Amazon further expanded the capability of Blueprints by allowing customers to publish skills they've built with the templates to its Alexa Skill Store in the US for use by anyone with an Alexa-enabled device

Alexa provides a set of built-in capabilities, referred to as *skills*. For example, Alexa's abilities include playing music from multiple providers, answering questions, providing weather forecasts, and querying Wikipedia.

The Alexa Skills Kit lets you teach Alexa *new skills*. Customers can access these new abilities by asking Alexa questions or making requests. You can build skills that provide users with many different types of abilities. For example, a skill might do any one of the following:

- Look up answers to specific questions (*"Alexa, ask tide pooler for the high tide today in Seattle."*)
- Challenge the user with puzzles or games (*"Alexa, play Harry Potter quiz."*)
- Control lights and other devices in the home (*"Alexa, turn on the living room lights."*)
- Provide audio or text content for a customer's flash briefing (*"Alexa, give me my flash briefing"*)

2.4 HARDWARE AND SOFTWARE DESCRIPTION

2.4.1 FACE MASK DETECTION

- The mobilenet-ssd model is a Single-Shot multibox Detection (SSD) network intended to perform object detection. This model is implemented using the Caffe* framework.
- The model input is a blob that consists of a single image of 1x3x300x300 in BGR order, also like the densenet-121 model. The BGR mean values need to be subtracted as follows: [127.5, 127.5, 127.5] before passing the image blob into the network. In addition, values must be divided by 0.007843
- **SSD** is designed for object detection in real-time. Faster R-CNN uses a region proposal network to create boundary boxes and utilizes those boxes to classify objects. While it is considered the start-of-the-art in accuracy, the whole process runs at 7 frames per second.

2.4.1.1 DATASET

To train a neural network we must provide proper raw materials i.e data that we want train in our neural network. To classify whether a person is wearing a mask or not , we need to provide a good balanced dataset for both classes:

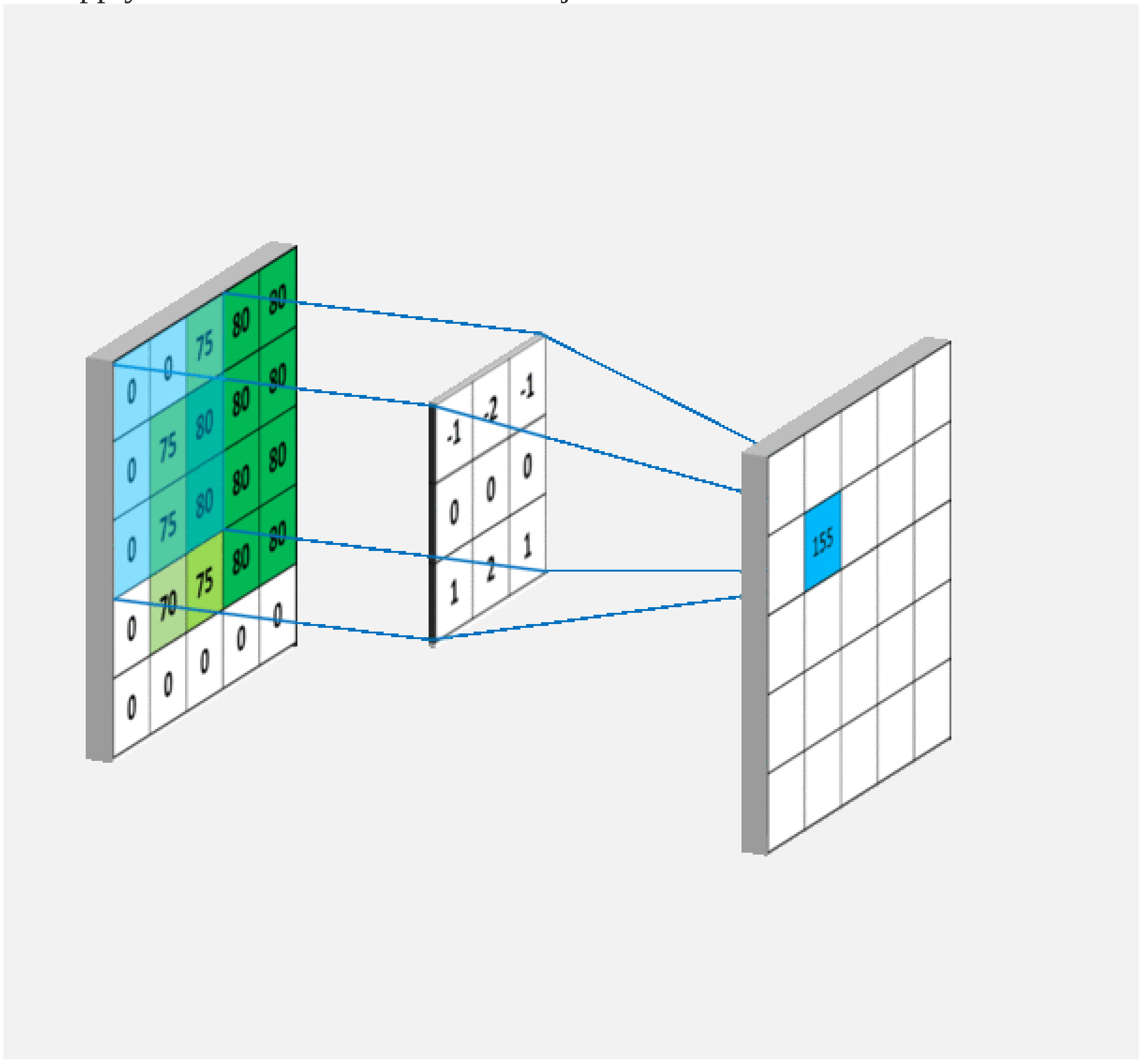
- Images with Mask
- Images without Mask
  Dataset must contains different kinds of masks with different shades of people all over the world and top of that data augmentation techniques is also applied over 1500 images for better training

2.4.1.2 HOW SSD WORKS

SSD is developed by Google researcher teams in 2016 to maintain the balance between the two object detection methods which are YOLO and RCNN.

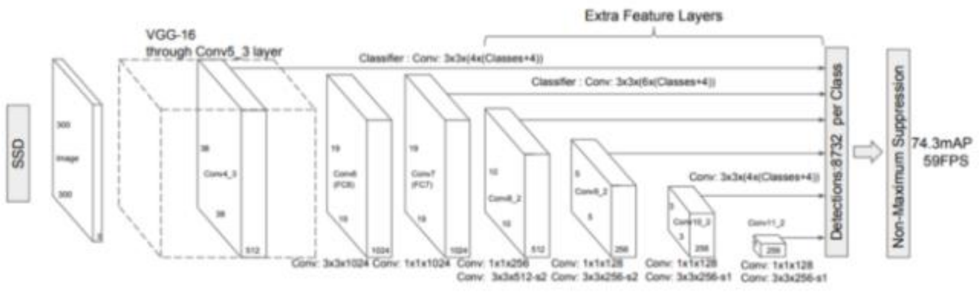
It composes of two parts:

- Extract feature maps, and
- Apply convolution filter to detect objects



2.4.1.3 WHY MOBILENETS IN SSD

The question arises why we use MobileNet, why we can't use Resnet, VGG or alexnet.

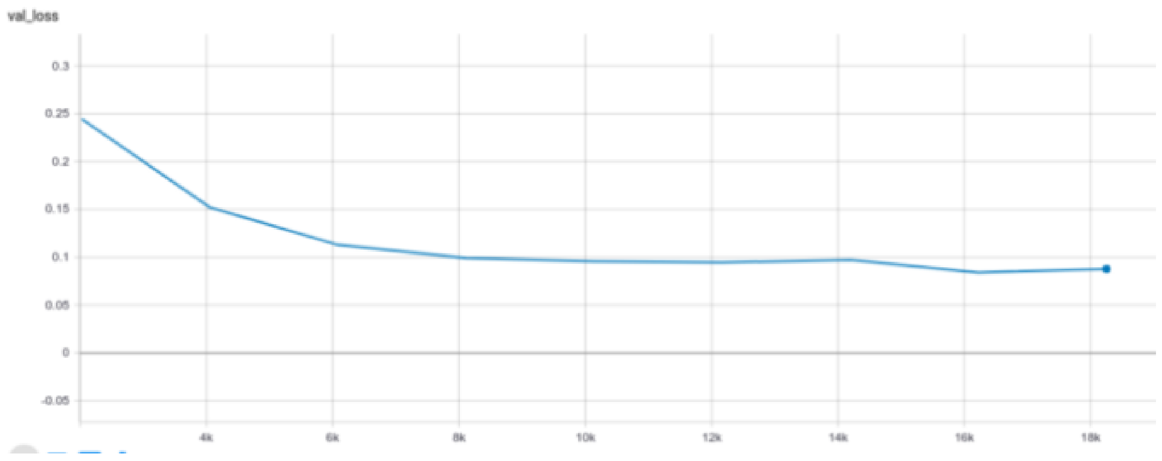


The answer is simple. Resnet or alexnet has a large network size and it increases the no of computation whereas in Mobilenet there is a simple architecture consisting of a *3×3* depthwise convolution followed by a *1×1* pointwise convolution.

2.4.1.4 Training Step

In the training step, we receive a batch of samples, pass them through our model via the forward pass, and compute the loss of that batch.We have trained our neural network in Google Colab — GPU for 20000 steps. After training for 20000 steps checkpoint files are generated in google drive with the weights of our neural network and now need to convert those .ckpt file into .pb file for testing.

2.4.1.5 Inference graph



Validation Loss

We can see that the validation loss is decreasing across epochs and the validation accuracy of our model reaches its highest peak to 96%.

2.4.2 SOCIAL DISTANCING DETECTION – YOLO DETECTION

WHAT IS YOLO:-

YOLO stands for You Only Look Once. It's a fast operating object detection system that can recognize various object types in a single frame more precisely than other detection systems.According to their documentation, it can predict up to 9000 classes. YOLO is built on a single CNN (convolutional neural network). The CNN separates an image into regions and calculates its bounding box and probabilities for each region. In YOLOv4, it uses CSPDarknet53 (CNN enhancement for increasing the learning capability), which is a lot faster than EfficientDet (YOLOv3 CNN enhancement), MaskRCNN, and RetinaNET. Because YOLO can be used with a conventional GPU, it provides widespread adoption, faster FPS, and more accuracy.

Analysis, in 5 Steps:-

Step 1:- Calculate Euclidean Distance of Two Points

We need to calculate the Euclidean distance in order to identify the distance between two bounding boxes. The function is_close gets two points, p1 and p2, as inputs for calculating the Euclidean distance and returns the calculated distance dst

```
def is_close(p1, p2):

    """
    # 1. Calculate Euclidean Distance of two points
    :param:
    p1, p2 = two points for calculating Euclidean Distance

    :return:
    dst = Euclidean Distance between two 2d points
    """
    dst = math.sqrt(p12 + p22)
         return dst
```

Step 2:- Converts Center Coordinates into Rectangle Coordinates

The function convertBack gets parameters x, y—the midpoint of the bounding box—and wand h—the width and height of the bounding box—as inputs. Then it will covert the center coordinates to rectangle coordinates and return the converted coordinates, xmin, ymin, xmax, and ymax.

def convertBack(x, y, w, h):

```
  """
    # 2. Converts center coordinates to rectangle
coordinates
    :param:
    x, y = midpoint of bounding box
    w, h = width, height of the bounding box

    :return:
    xmin, ymin, xmax, ymax
    """
    xmin = int(round(x - (w / 2)))
    xmax = int(round(x + (w / 2)))
    ymin = int(round(y - (h / 2)))
    ymax = int(round(y + (h / 2)))

    return xmin, ymin, xmax, ymax
```

Step 3:- Filtering the Person Class from Detections and Getting a Bounding Box Centroid for Each Person Detection

First, we check whether there is any detection in the image using a len(detections)>0 condition. If the condition succeed, we create a dictionary called centroid_dicT and a variable called objectId set to zero. Then it will run through another condition that filters out all the other detection types, except for person. We'll store the center points of all the person detections and append it to the bounding box for persons detected.

```
if len(detections) > 0:
     centroid_dict = dict()
    objectId = 0
    for detection in detections:
     name_tag = str(detection[0].decode())
          if name_tag == 'person':
               x, y, w, h = detection[2][0],\
                            detection[2][1],\
                            detection[2][2],\
                            detection[2][3]
                xmin, ymin, xmax, ymax =
convertBack(float(x), float(y), float(w), float(h))
                centroid_dict[objectId] = (int(x), int(y),
xmin, ymin, xmax, ymax)

                objectId += 1
```

Step 4:- Check which person bounding boxes are close to each other

First, we create a list that contains all object IDs of the under-threshold distance conditions. Through iteration, we can get all the combinations of close detection to calculate the Euclidean distance using the function from **Step 1**.

Then we set the social distance threshold (75.0) (equivalent to 6 feet) and check whether it satisfies the condition distance < 75.0. Finally, we set colors for the bounding boxes. **Red** for an at-risk person and **Green** for a protected person

```
red_zone_list = []
red_line_list = []
for (id1, p1), (id2, p2) in
combinations(centroid_dict.items(), 2):
    dx, dy = p1[0] - p2[0], p1[1] - p2[1]
    distance = is_close(dx, dy)
    if distance < 75.0:
        if id1 not in red_zone_list:
            red_zone_list.append(id1)
            red_line_list.append(p1[0:2])
        if id2 not in red_zone_list:
            red_zone_list.append(id2)
            red_line_list.append(p2[0:2])

for idx, box in centroid_dict.items():
  if idx in red_zone_list:
     cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]),
(255, 0, 0), 2)
  else:
     cv2.rectangle(img, (box[2], box[3]), (box[4], box[5]),
(0, 255, 0), 2)
```

Step 5:- Display risk analytics and risk indicators

In this final module, we indicate the number of people at risk in a given frame. Starting from the text we want to display with a counted number of people at risk. For better graphical representation, Using **for** check **in** range, We tell the application to draw the lines between the nearby bounding boxes and iterate through the red_line_list

```
text = "No of at-risk people: %s" % str(len(red_zone_list))

location = (10,25)
cv2.putText(img, text, location, cv2.FONT_HERSHEY_SIMPLEX,
1, (246,86,86), 2, cv2.LINE_AA)

for check in range(0, len(red_line_list)-1):
    start_point = red_line_list[check]
    end_point = red_line_list[check+1]
    check_line_x = abs(end_point[0] - start_point[0])
    check_line_y = abs(end_point[1] - start_point[1])
    if (check_line_x < 75) and (check_line_y < 25):


        cv2.line(img, start_point, end_point, (255, 0, 0),
```

2.4.3 ALEXA INTEGRATION

- To build the cloud-based service for a custom Alexa skill is to use AWS Lambda, an Amazon Web Services offering that runs our code only when it's needed and scales automatically, so there is no need to provision or continuously run servers.
- We upload the code for our Alexa skill to a Lambda function and Lambda does the rest, executing it in response to Alexa voice interactions and automatically managing the compute resources for us
- The whole system is integrated to alexa with the help of AWS lambda function
- All the characteristics like face mask and social distancing will be added as an intent to the lambda function
- The lambda function will act as the end point for developed alexa skill
- Alexa developer console has been used for the whole process for the development process

3. SOFTWARE IMPLEMENTATION

3.1 IMPLEMENTATION

3.1.1 FACE MASK DETECTION

```
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing.image import img_to_array
from tensorflow.keras.models import load_model
from imutils.video import VideoStream
import numpy as np
import imutils
import time
import cv2
import os
import boto3
from boto.s3.key import Key
def upload_to_s3(i,labfinal):

    from botocore.client import Config

    ACCESS_KEY_ID = //Enter ID
    ACCESS_SECRET_KEY = //Enter Key
    BUCKET_NAME = 'face-mask-photos1'
    BUCKET_NAME1 = 'face-mask-status1'

    data = open('final.png', 'rb')
    f = open('status.txt', 'w')

    s3 = boto3.resource(
        's3',
        aws_access_key_id=ACCESS_KEY_ID,
        aws_secret_access_key=ACCESS_SECRET_KEY,
        config=Config(signature_version='s3v4')
    )
    s3.Bucket(BUCKET_NAME).put_object(Key='final.png', Body=data)
    f.write(labfinal)
    f.close()
    f=open("status.txt",'r+')
    s3.Bucket(BUCKET_NAME1).put_object(Key='status.txt', Body=f.read())
    print ("Done")

def detect_and_predict_mask(frame, faceNet, maskNet):
    (h, w) = frame.shape[:2]
    blob = cv2.dnn.blobFromImage(frame, 1.0, (224, 224),
        (104.0, 177.0, 123.0))

    faceNet.setInput(blob)
    detections = faceNet.forward()
    print(detections.shape)

    faces = []
    locs = []
    preds = []

    for i in range(0, detections.shape[2]):
        confidence = detections[0, 0, i, 2]


        if confidence > 0.5:
            box = detections[0, 0, i, 3:7] * np.array([w, h, w, h])
            (startX, startY, endX, endY) = box.astype("int")
            (startX, startY) = (max(0, startX), max(0, startY))
            (endX, endY) = (min(w - 1, endX), min(h - 1, endY))
            face = frame[startY:endY, startX:endX]
            face = cv2.cvtColor(face, cv2.COLOR_BGR2RGB)
            face = cv2.resize(face, (224, 224))
            face = img_to_array(face)
            face = preprocess_input(face)
            faces.append(face)
            locs.append((startX, startY, endX, endY))


    if len(faces) > 0:
        faces = np.array(faces, dtype="float32")
        preds = maskNet.predict(faces, batch_size=32)
    return (locs, preds)


prototxtPath = r"face_detector\deploy.prototxt"
weightsPath = r"face_detector\res10_300x300_ssd_iter_140000.caffemodel"
faceNet = cv2.dnn.readNet(prototxtPath, weightsPath)
maskNet = load_model("mask_detector.model")
print("[INFO] starting video stream...")
vs = VideoStream(src=0).start()
p=0
labfinal=''
while True:
    frame = vs.read()
    frame = imutils.resize(frame, width=400)
    (locs, preds) = detect_and_predict_mask(frame, faceNet, maskNet)
    for (box, pred) in zip(locs, preds):
        (startX, startY, endX, endY) = box
        (mask, withoutMask) = pred
        label = "odyMask" if mask > withoutMask else "odyNo Mask"
        labfinal=label
        if labfinal=="odyMask":
            label="Mask"
        else:
            label="No Mask"
        color = (0, 255, 0) if label == "Mask" else (0, 0, 255)
        label = "{}: {:.2f}%".format(label, max(mask, withoutMask) * 100)
        cv2.putText(frame, label, (startX, startY - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.45, color, 2)
        cv2.rectangle(frame, (startX, startY), (endX, endY), color, 2)
    cv2.imshow("Frame", frame)
    cv2.imwrite('final.png',frame)
    upload_to_s3(p,labfinal)
    p=p+1
    key = cv2.waitKey(1) & 0xFF
    if key == ord("q"):
        break
cv2.destroyAllWindows()
```

```
vs.stop()
```

3.1.2 SOCIAL DISTANCING DETECTION

```
import numpy as np
import time
import cv2
import math
import os
import boto3
from boto.s3.key import Key

def upload_to_s3(i,text):

    from botocore.client import Config

    ACCESS_KEY_ID = //Enter ID
    ACCESS_SECRET_KEY = //Enter Key
    BUCKET_NAME = 'social-distance-photos1'
    BUCKET_NAME1 = 'social-distance-status1'

    data = open('SDimage.png', 'rb')
    f = open('SDstatus.txt', 'w')

    s3 = boto3.resource(
        's3',
        aws_access_key_id=ACCESS_KEY_ID,
        aws_secret_access_key=ACCESS_SECRET_KEY,
        config=Config(signature_version='s3v4')
    )
    s3.Bucket(BUCKET_NAME).put_object(Key='SDimage.png', Body=data)
    f.write(text)
    f.close()
    f=open("SDstatus.txt",'r+')
    s3.Bucket(BUCKET_NAME1).put_object(Key='SDstatus.txt', Body=f.read())
    print ("Done")

labelsPath = "./coco.names"
LABELS = open(labelsPath).read().strip().split("\n")

np.random.seed(42)
COLORS = np.random.randint(0, 255, size=(len(LABELS), 3),
    dtype="uint8")

weightsPath = "./yolov3.weights"
configPath = "./yolov3.cfg"

net = cv2.dnn.readNetFromDarknet(configPath, weightsPath)


cap = cv2.VideoCapture(0)

while(cap.isOpened()):

    ret,image=cap.read()
    (H, W) = image.shape[:2]
```

```
ln = net.getLayerNames()
ln = [ln[i[0] - 1] for i in net.getUnconnectedOutLayers()]
blob = cv2.dnn.blobFromImage(image, 1 / 255.0, (416, 416),swapRB=True, crop=False)
net.setInput(blob)
start = time.time()
layerOutputs = net.forward(ln)
end = time.time()
print("Frame Prediction Time : {:.6f} seconds".format(end - start))
boxes = []
confidences = []
classIDs = []
for output in layerOutputs:
    for detection in output:
        scores = detection[5:]
        classID = np.argmax(scores)
        confidence = scores[classID]
        if confidence > 0.1 and classID == 0:
            box = detection[0:4] * np.array([W, H, W, H])
            (centerX, centerY, width, height) = box.astype("int")
            x = int(centerX - (width / 2))
            y = int(centerY - (height / 2))
            boxes.append([x, y, int(width), int(height)])
            confidences.append(float(confidence))
            classIDs.append(classID)

idxs = cv2.dnn.NMSBoxes(boxes, confidences, 0.5,0.3)
ind = []
for i in range(0,len(classIDs)):
    if(classIDs[i]==0):
        ind.append(i)
a = []
b = []

if len(idxs) > 0:
        for i in idxs.flatten():
            (x, y) = (boxes[i][0], boxes[i][1])
            (w, h) = (boxes[i][2], boxes[i][3])
            a.append(x)
            b.append(y)

distance=[]
nsd = []
for i in range(0,len(a)-1):
    for k in range(1,len(a)):
        if(k==i):
            break
        else:
            x_dist = (a[k] - a[i])
            y_dist = (b[k] - b[i])
            d = math.sqrt(x_dist * x_dist + y_dist * y_dist)
            distance.append(d)
            if(d <=250):
                nsd.append(i)
                nsd.append(k)
            nsd = list(dict.fromkeys(nsd))
            print(nsd)
color1 = (0, 0, 255)
color = (0, 255, 0)
```

```
    text=""
    p=0
    # for i in nsd:
    #     (x, y) = (boxes[i][0], boxes[i][1])
    #     (w, h) = (boxes[i][2], boxes[i][3])
    #     cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
    #     text = "Alert"
    #     cv2.putText(image, text, (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX,0.5, color1, 2)

    if len(idxs) > 0:
        for i in idxs.flatten():
            if (i in nsd):
                (x, y) = (boxes[i][0], boxes[i][1])
                (w, h) = (boxes[i][2], boxes[i][3])
                cv2.rectangle(image, (x, y), (x + w, y + h), color1, 2)
                text = "Alert"
                cv2.putText(image, text, (x, y -
 5), cv2.FONT_HERSHEY_SIMPLEX,0.5, color1, 2)
            else:
                (x, y) = (boxes[i][0], boxes[i][1])
                (w, h) = (boxes[i][2], boxes[i][3])
                cv2.rectangle(image, (x, y), (x + w, y + h), color, 2)
                text = 'OK'
                cv2.putText(image, text, (x, y -
 5), cv2.FONT_HERSHEY_SIMPLEX,0.5, color, 2)
            p=p+1

    cv2.imshow("Social Distancing Detector", image)
    cv2.imwrite('SDimage.png',image)
    print(text)
    upload_to_s3(p,text)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

3.1.3 ALEXA SKILL

```

SIDDHARTH NAIR
Fri, Oct 16, 12:22 PM
to me

from __future__ import print_function
import random
import boto3
# from insults import get_insult

# --------------- Helpers that build all of the responses ----------------------


def get_test_response(label):
    """ An example of a custom intent. Same structure as welcome message, just make sure to
 add this intent
    in your alexa skill in order for it to work.
    """
    session_attributes = {}
    card_title = "Test"
    op=str(label)
    # speech_output=""
    # print(op)
    if op=="b'odyMask'":
        speech_output="Person is wearing a mask"
    else:
        speech_output="Person is not wearing a mask. " "Please do so before going outside.
"

    # print(op)
    # speech_output = str(label)
    reprompt_text = "You never responded to the first test message. Sending another one."
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
card_title, speech_output, reprompt_text, should_end_session))


def get_SD_response(label):
    session_attributes = {}
    card_title = "Test"
    op=str(label)
    # speech_output=""
    # print(op)
    if op=="b'OK'":
        speech_output="Social Distancing Protocols are being maintained"
    else:
        speech_output="Social Distancing Protocols are not being maintained"

    # print(op)
    # speech_output = str(label)
    reprompt_text = "You never responded to the first test message. Sending another one."
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
card_title, speech_output, reprompt_text, should_end_session))


def build_speechlet_response(title, output, reprompt_text, should_end_session):
    return {
        'outputSpeech': {
            'type': 'PlainText',
            'text': output
        },
        'card': {
            'type': 'Simple',
            'title': "SessionSpeechlet - " + title,
            'content': "SessionSpeechlet - " + output
        },
        'reprompt': {
            'outputSpeech': {
                'type': 'PlainText',
                'text': reprompt_text
            }
        },
        'shouldEndSession': should_end_session
    }
```

```python
def build_response(session_attributes, speechlet_response):
    return {
        'version': '1.0',
        'sessionAttributes': session_attributes,
        'response': speechlet_response
    }


# --------------- Functions that control the skill's behavior ------------------
def get_compliment_response():
    """ An example of a custom intent. Same structure as welcome message, just make sure to
 add this intent
    in your alexa skill in order for it to work.
    """
    session_attributes = {}
    card_title = "Compliment"
    # compliments = [line for line in open('compliments.txt')]
    # print(compliments)
    # compliment = compliments[random.randint(0,len(compliments)-1)]
    speech_output = "This is test by siddharth"
    reprompt_text = "I said," + "This is test by siddharth"
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))

# def get_insult_response(intent, session):
#     """ An example of a custom intent. Same structure as welcome message, just make sure
to add this intent
#     in your alexa skill in order for it to work.
#     """
#     print("SESSION BELOW")
#     print(session)
#     session_attributes = {'insult_state': 1}
#     card_title = "Insult"
#     speech_output = "That's not very nice. I don't want to."
#     reprompt_text = "I would rather not do that."

#     if session.get('attributes', {}) and session['attributes']['insult_state'] == 1:
#         insult = get_insult()
#         speech_output = intent['slots']['name']['value'] + " is a " + insult
#         reprompt_text = "I said he is a " + insult

#     should_end_session = False
#     return build_response(session_attributes, build_speechlet_response(
#         card_title, speech_output, reprompt_text, should_end_session))

def get_welcome_response():
    """ If we wanted to initialize the session to have some attributes we could
    add those here
    """
    session_attributes = {}
    card_title = "Welcome"
    speech_output = "Your application has started!"
    # If the user either does not reply to the welcome message or says something
    # that is not understood, they will be prompted again with this text.
    reprompt_text = "I don't know if you heard me, welcome to our alexa application!"
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
        card_title, speech_output, reprompt_text, should_end_session))


def handle_session_end_request():
    card_title = "Session Ended"
    speech_output = "Thank you for trying our application. " "Have a nice day! "
    # Setting this to true ends the session and exits the skill.
    should_end_session = True
    return build_response({}, build_speechlet_response(
        card_title, speech_output, None, should_end_session))

# --------------- Events ------------------

def on_session_started(session_started_request, session):
    """ Called when the session starts.
        One possible use of this function is to initialize specific
        variables from a previous state stored in an external database
    """
    # Add additional code here as needed
    pass



def on_launch(launch_request, session):
    """ Called when the user launches the skill without specifying what they
    want
    """
    # Dispatch to your skill's launch message
    return get_welcome_response()


def on_intent(intent_request, session):
    """ Called when the user specifies an intent for this skill """

    intent = intent_request['intent']
    intent_name = intent_request['intent']['name']

    # Dispatch to your skill's intent handlers
    if intent_name == "HelloWorldIntent":
        return get_compliment_response()
    elif intent_name == "TextFileIntent":

        session = boto3.Session(aws_access_key_id='AKIAXNNZJOKYLR6UPOCU',aws_secret_access_
key='ySlX066MTdGM/w8c4rPlXRPsKHPtb9iu+1Mflg9y',)
        s1 = session.resource('s3')

    # Dispatch to your skill's intent handlers
        bucket = s1.Bucket('face-mask-sid')
        obj = bucket.Object(key='face-mask-storage.txt')
        response = obj.get()
        text = response["Body"].read()
        label=text
        return get_test_response(label)

    elif intent_name == "MaskIntent":
        session = boto3.Session(aws_access_key_id='AKIAXNNZJOKYLR6UPOCU',aws_secret_access_
key='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',)
```

```
        s1 = session.resource('s3')
        bucket = s1.Bucket('face-mask-status1')
        obj = bucket.Object(key='status.txt')
        response = obj.get()
        text = response["Body"].read()
        label=text
        return get_test_response(label)

    elif intent_name == "SDistancingIntent":
        session = boto3.Session(aws_access_key_id='AKIAXNNZJOKYLR6UPOCU',aws_secret_access_
key='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',)
        s1 = session.resource('s3')
        bucket = s1.Bucket('social-distance-status1')
        obj = bucket.Object(key='SDstatus.txt')
        response = obj.get()
        text = response["Body"].read()
        label=text
        return get_SD_response(label)



    # elif intent_name == "insult":
    #     return get_insult_response(intent, session)
    elif intent_name == "AMAZON.HelpIntent":
        return get_welcome_response()
    elif intent_name == "AMAZON.CancelIntent" or intent_name == "AMAZON.StopIntent":
        return handle_session_end_request()
    else:
        raise ValueError("Invalid intent")


def on_session_ended(session_ended_request, session):
    """ Called when the user ends the session.
    Is not called when the skill returns should_end_session=true
    """
    print("on_session_ended requestId=" + session_ended_request['requestId'] +
          ", sessionId=" + session['sessionId'])
    # add cleanup logic here


# --------------- Main handler ------------------

def lambda_handler(event, context):
    """ Route the incoming request based on type (LaunchRequest, IntentRequest,
    etc.) The JSON body of the request is provided in the event parameter.
    """
    print("Incoming request...")

    """
    Uncomment this if statement and populate with your skill's application ID to
    prevent someone else from configuring a skill that sends requests to this
    function.
    """
    # if (event['session']['application']['applicationId'] !=
    #         "amzn1.echo-sdk-ams.app.[unique-value-here]"):
    #     raise ValueError("Invalid Application ID")

    if event['session']['new']:
        on_session_started({'requestId': event['request']['requestId']},event['session'])

    if event['request']['type'] == "LaunchRequest":
        return on_launch(event['request'], event['session'])
    elif event['request']['type'] == "IntentRequest":
        return on_intent(event['request'], event['session'])
    elif event['request']['type'] == "SessionEndedRequest":
        return on_session_ended(event['request'], event['session'])
```

3.2 CODE ANALYSIS

Facemask detection:-

Number of Lines: 109
Number of Functions/Routines: 3

Social distancing:-

Number of Lines: 144
Number of Functions/Routines: 2

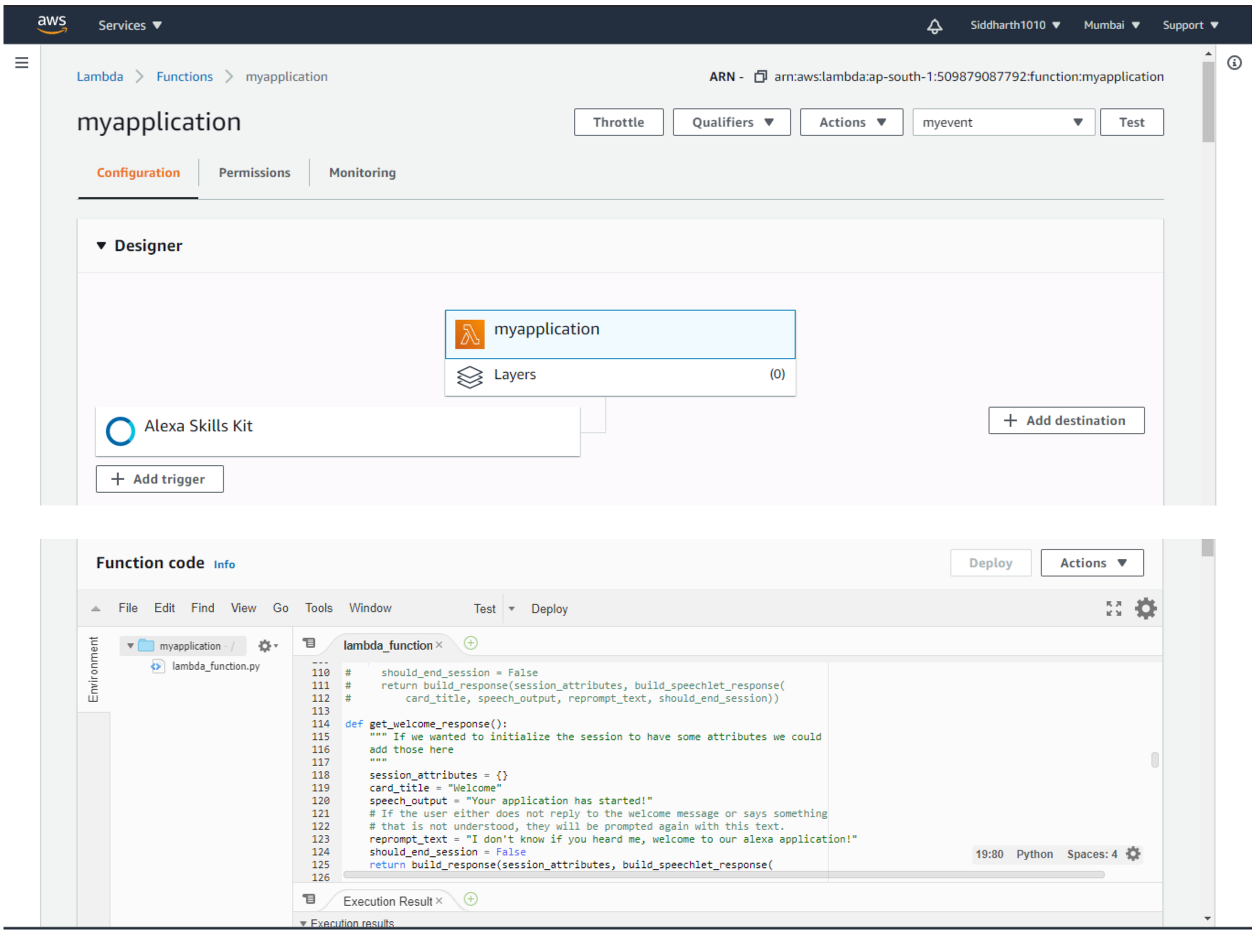
Alexa Skill:-

Number of lines: 250
Number of Functions/Routines: 12

4. RESULTS AND INFERENCES

<u>RESULTS:-</u>

This is the AWS Lambda console:-



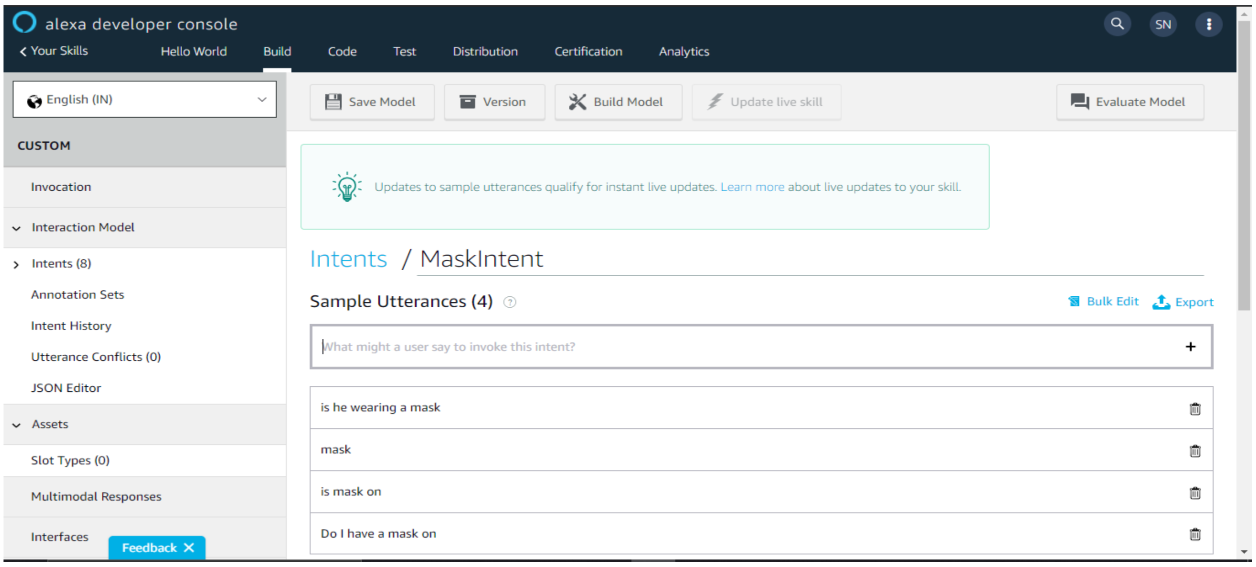
ARN - arn:aws:lambda:ap-south-1:509879087792:function:myapplication

myapplication

```
#     should_end_session = False
#     return build_response(session_attributes, build_speechlet_response(
#         card_title, speech_output, reprompt_text, should_end_session))

def get_welcome_response():
    """ If we wanted to initialize the session to have some attributes we could
    add those here
    """
    session_attributes = {}
    card_title = "Welcome"
    speech_output = "Your application has started!"
    # If the user either does not reply to the welcome message or says something
    # that is not understood, they will be prompted again with this text.
    reprompt_text = "I don't know if you heard me, welcome to our alexa application!"
    should_end_session = False
    return build_response(session_attributes, build_speechlet_response(
```

This is the mask Intent in the Alexa Developer console:-



Updates to sample utterances qualify for instant live updates. Learn more about live updates to your skill.

Intents / MaskIntent

Sample Utterances (4)

What might a user say to invoke this intent?

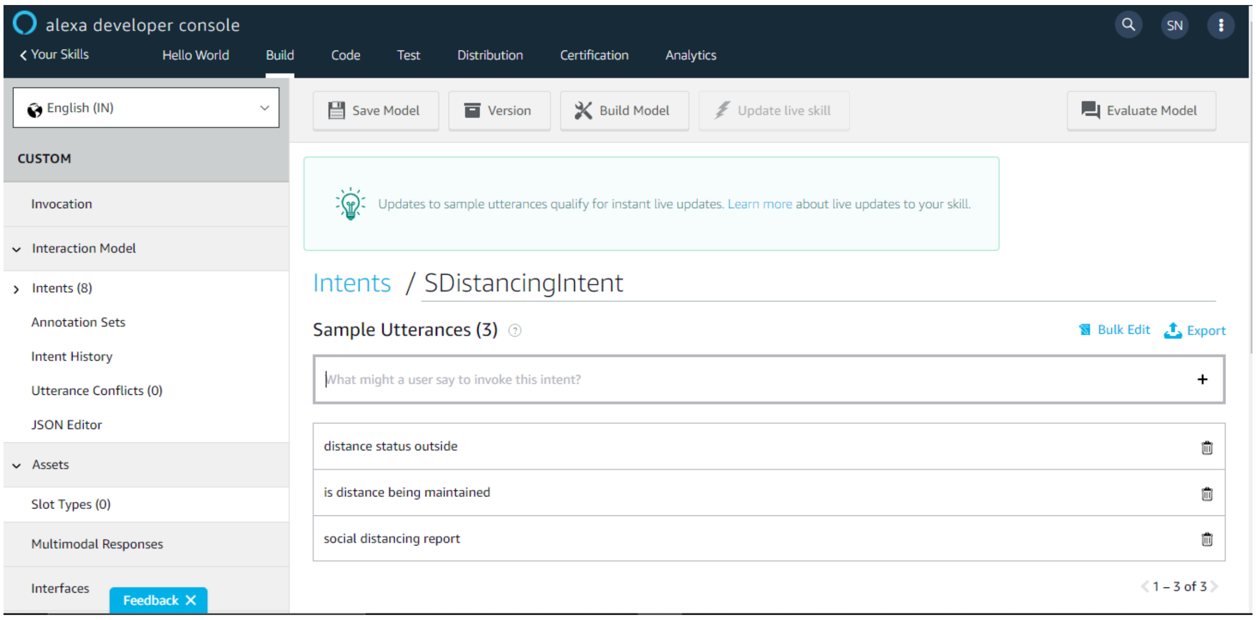
is he wearing a mask

mask

is mask on

Do I have a mask on

This is the social distancing intent on the Alexa Developer console:-



Updates to sample utterances qualify for instant live updates. Learn more about live updates to your skill.

Intents / SDistancingIntent

Sample Utterances (3)

What might a user say to invoke this intent?

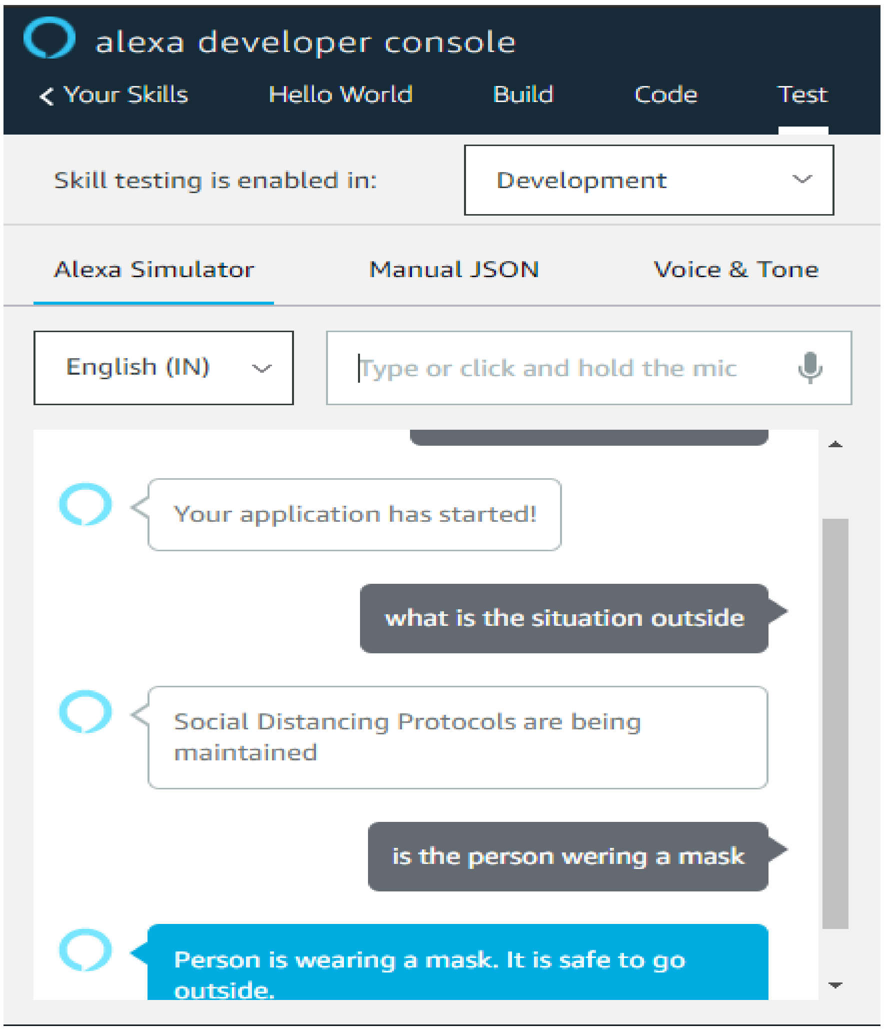
distance status outside

is distance being maintained

social distancing report

1 – 3 of 3

This is the output when tried with Alexa Simulator:-



Skill testing is enabled in: Development

Your application has started!

what is the situation outside

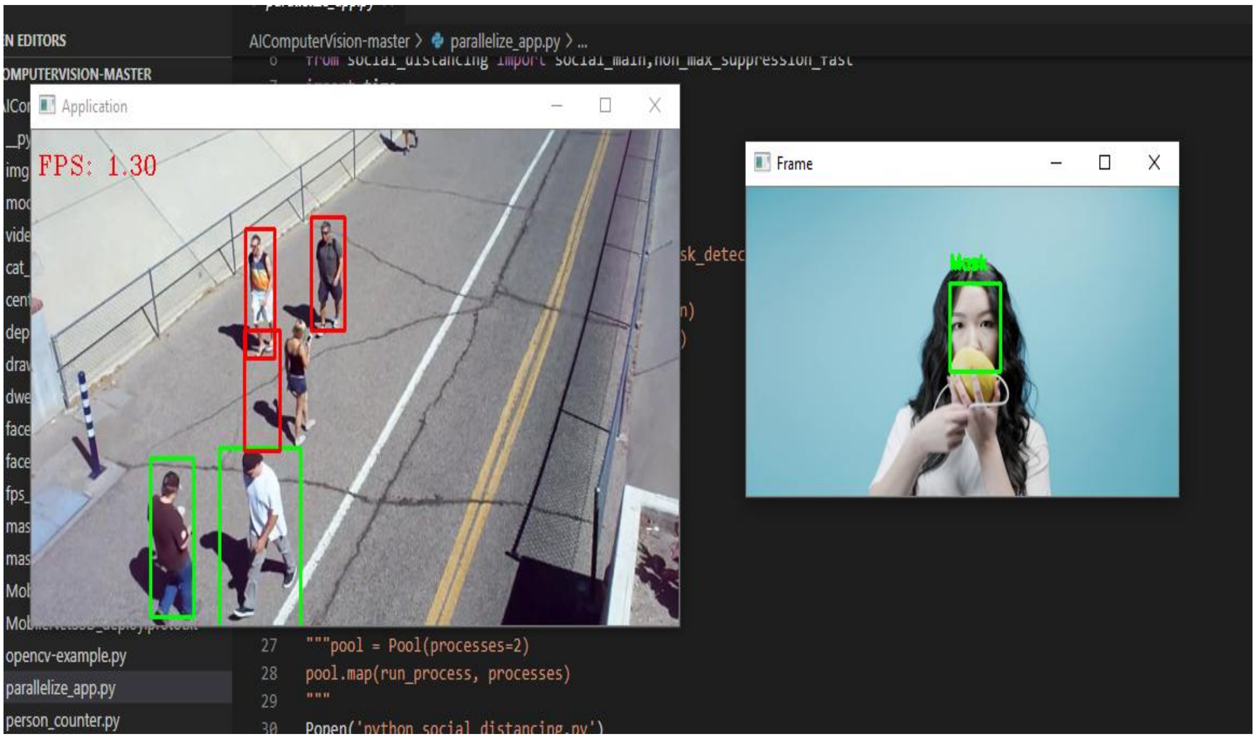
Social Distancing Protocols are being maintained

is the person wering a mask

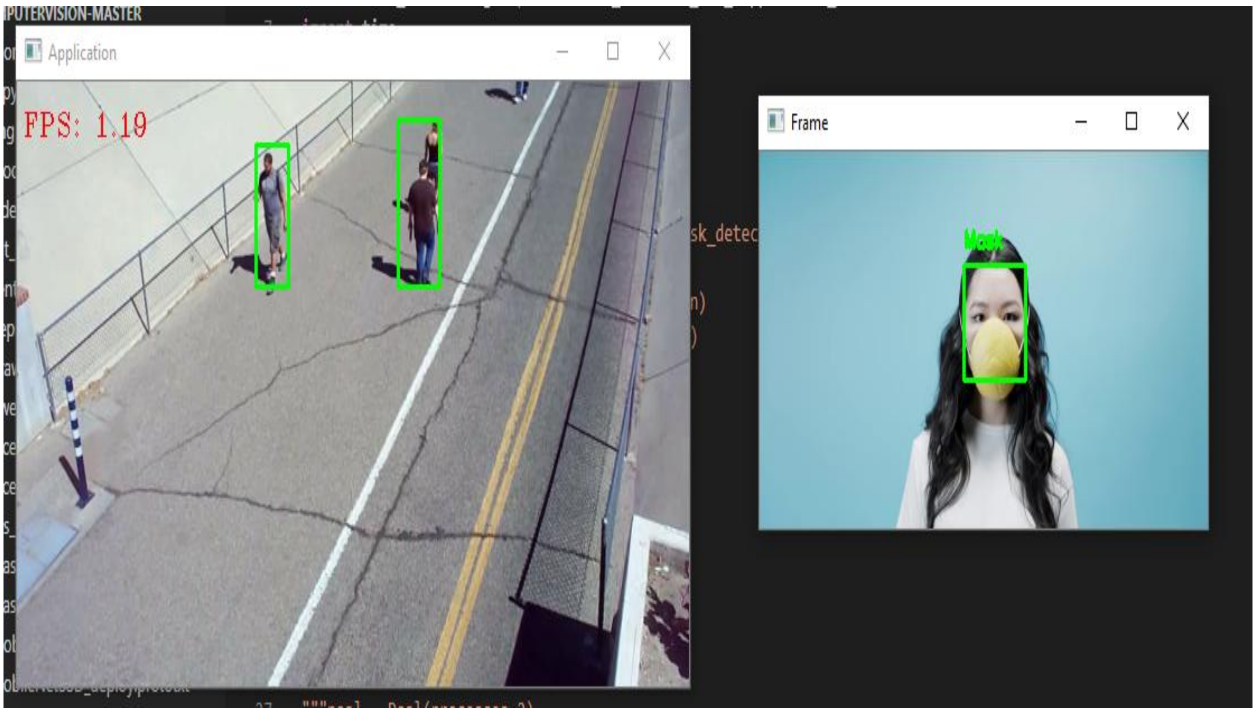
Person is wearing a mask. It is safe to go outside.

This is photo when I tried with actual amazon echo:-

These are the outputs of face mask and social distancing:-



```
from social_distancing import social_main,non_max_suppression_fast
"""pool = Pool(processes=2)
pool.map(run_process, processes)
"""
Popen('python social_distancing.py')
```



INFERENCES:-

- From this project we inferred that Smart home devices like google home and amazon echo can be programmed for custom purposes and it can be deployed for user specific functions
- We also inferred that Raspberry pi can be integrated for any applications on the smart home area.
- We also inferred deep learning models can be integrated as a part of a cloud based system
- We also inferred how AWS and its services are been integrated with hardware so as to create an embedded system.

5. CONCLUSIONS AND FUTURE WORK

5.1 CONCLUSION

- The COVID-19 HELPER USING ALEXA AND RASPBERRY-PI system was built and implemented.
- The system is targeted for every individual for their safety.
- The prototype developed can be readily integrated as a part of home automation
- Till now the system has been tested with many angles and all the test results are satisfactory and promising.
- Many challenges were faced during the development stage but we were able to overcome it with sincere work.

5.2 FUTURE WORK

- This project can be further extended by adding features like Measuring a person’s body temperature with infrared thermometer and if it is higher than usual a alert can be sent to the user

- Also if a person is found to violate any safety criteria an alert can be sent to the user’s mobile**.**

- If any potentially diseased person comes in proximity an alert can be sent to the user.

- The system can be further extended to give the user live news on COVID-19

6. REFERENCES

1. Ayi, M., Ganti, A. K., Adimulam, M., Karthik, B., Banam, M., & Kumari, G. V. (2017). Face Tracking and Recognition Using MATLAB and Arduino. *International Journal for Research in Applied Science & Engineering Technology (IJRASET)*.
2. PAMULAPATI, V. S., ROHAN, Y. S., Kiran, V. S., SANDEEP, S., & RAO, M. S. (2018). Real-Time Face Tracking Using Matlab And Arduino. *Electronics And Communication Engineering, Vasireddy Venkatadri Institute Of Technology*.
3. Dahal, B., Alsadoon, A., Prasad, P. C., & Elchouemi, A. (2016, March). Incorporating skin color for improved face detection and tracking system. In *2016 IEEE Southwest Symposium on Image Analysis and Interpretation (SSIAI)* (pp. 173-176). IEEE.
4. Mohammed, M. N., Syamsudin, H., Al-Zubaidi, S., AKS, R. R., & Yusuf, E. (2020). Novel COVID-19 detection and diagnosis system using IOT based smart helmet. *International Journal of Psychosocial Rehabilitation*, *24*(7).
5. Khodaskar, H. V., & Mane, S. (2017). Human face detection & recognition using raspberry Pi. *International Journal of Advanced Engineering, Management and Science*, 1-2.
6. Teyhouee, A., & Osgood, N. D. (2019, July). Cough Detection Using Hidden Markov Models. In *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation* (pp. 266-276). Springer, Cham
7. Liu, J. M., You, M., Wang, Z., Li, G. Z., Xu, X., & Qiu, Z. (2014, November). Cough detection using deep neural networks. In *2014 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)* (pp. 560-563). IEEE.
8. Leong, R. (2018). Analyzing the Privacy Attack Landscape for Amazon Alexa Devices
9. Palankar, M. R., Iamnitchi, A., Ripeanu, M., & Garfinkel, S. (2008, June). Amazon S3 for science grids: a viable solution?. In *Proceedings of the 2008 international workshop on Data-aware distributed computing* (pp. 55-64).

10. Kumanov, D., Hung, L. H., Lloyd, W., & Yeung, K. Y. (2018). Serverless computing provides on-demand high performance computing for biomedical research. *arXiv preprint arXiv:1807.11659.*

11. Ambre, S., Masurekar, M., & Gaikwad, S. (2020). Face Recognition Using Raspberry PI. *Modern Approaches in Machine Learning and Cognitive Science: A Walkthrough*, 1-11.

7. WORKING PROTOTYPE

Youtube link:- https://youtu.be/A48kdt6N2LA

Github link:- https://github.com/Siddharth1010/Amazon-AWS-ALEXA-Smart-Covid19-Assistant