Project Report

1.INTRODUCTION

Team ID : NM2023TMID22339

Project Name : Go No Queue -Rush Estimator for Corporate Cafeteria

College : Agni College of Technology

Member : KISHORE KUMAR M

1.1 Project Overview:

Go No Queue -Rush Estimator for Corporate Cafeteria

The Cafeteria Crowd Estimation System is designed to address the issue of food shortage in corporate cafeterias during peak hours. By implementing computer vision techniques and utilizing entry and exit devices, the system accurately estimates the number of people entering, leaving, and present in the cafeteria. The gathered data is stored in the cloud and made available to cafeteria authorities through a web application, enabling them to monitor crowd estimation in real-time and make informed decisions regarding food preparation.

Key Components:

1. Entry and Exit Devices: Devices are installed at the entry and exit points of the cafeteria to track the movement of people. These devices can be in the form of sensors, cameras, or other suitable technologies capable of detecting individuals.

2. Computer Vision Techniques: Computer vision algorithms and techniques are employed to process the video feed from the entry and exit devices. These techniques involve object detection, tracking, and counting to identify individuals entering and leaving the cafeteria.

3. Data Processing and Crowd Estimation: The system processes the collected data from the entry and exit devices to estimate the number of people present in the cafeteria at any given time. By analyzing the entries and exits, the system keeps a real-time count of the crowd.

4. Cloud Storage: The collected data, including entry and exit counts and crowd estimation, is securely stored in the cloud. Cloud storage ensures data reliability, accessibility, and scalability.

5. Web Application: A web application is developed for cafeteria authorities to access and monitor the crowd estimation data. The web app provides an intuitive interface to view real-time crowd numbers, historical trends, and other relevant analytics. It enables cafeteria management to make data-driven decisions regarding food preparation and resource allocation.

1.2 Purpose:

Purpose of the Cafeteria Crowd Estimation System:

To address the challenges faced by corporate cafeterias in managing food shortage during peak hours. The system aims to provide a solution that enables accurate estimation of the number of people entering, leaving, and present in the cafeteria. The main purposes of implementing this system are:

1. Optimize Food Preparation: By accurately estimating the crowd in the cafeteria, the system allows cafeteria authorities to optimize food preparation. They can ensure that an adequate amount of food is available based on the number of people present, reducing the risk of food shortage or wastage.

2. Improve Customer Satisfaction: By avoiding situations where customers face food shortage or long waiting times, the system enhances customer satisfaction. Customers can enjoy a smooth dining experience without the frustration of limited food availability.

3. Enhance Operational Efficiency: The system enables cafeteria management to make data-driven decisions regarding staffing, resource allocation, and menu planning. This improves overall operational efficiency by aligning resources with the actual demand, resulting in cost savings and improved productivity.

4. Real-time Monitoring and Insights: The system provides real-time monitoring of crowd estimation data through a web application. Cafeteria authorities can access this data to gain insights into crowd trends, peak hours, and historical patterns. These insights facilitate proactive decision-making and allow for adjustments in staffing and food preparation schedules.

5. Scalable Solution: The system is designed to be scalable and adaptable to different cafeteria environments. It can accommodate multiple entry and exit points, making it suitable for cafeterias of varying sizes and layouts.

Overall, the purpose of the Cafeteria Crowd Estimation System is to enhance the efficiency and profitability of corporate cafeterias by accurately estimating crowd numbers, optimizing food preparation, improving customer satisfaction, and enabling data-driven decision-making.

2. IDEATION & PROPOSED SOLUTION

2.1 Problem Statement Definition:

Customer Problem Statement :

As a corporate cafeteria owner, I am facing challenges with food shortages during peak hours due to a large number of people visiting the cafeteria. This issue not only affects the profitability of the cafeteria but also leads to customer dissatisfaction. To overcome this problem, I am seeking a solution that can accurately estimate the number of people entering and leaving the cafeteria and provide real-time information on the crowd size. I envision a system that utilizes computer vision techniques to track and count individuals as they enter and exit the cafeteria. By analyzing this data, I want to be able to estimate the number of people present in the cafeteria at any given time. Storing this information securely in the cloud will enable me to access it remotely. To make informed decisions about food preparation, I require a userfriendly web application that provides real-time crowd estimation and historical trends. This application should alert me when there is a risk of food shortage based on the current crowd size. Additionally, the system should have the capability to analyze the collected data and provide predictive insights on future crowd estimation. By implementing this solution, I aim to optimize the cafeteria's operations, improve customer satisfaction, and ensure a sufficient food supply during peak hours

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
PS-1	Cafeteria Owner	Selling Food items	Food Shortages During Peak Hours	A large number of people visiting the cafeteria	Affects the profitability of the cafeteria customer dissatisfaction
PS-2	Customer	Buying Food	The Queue is too Long	A large number of people visiting the cafeteria	Frustrated

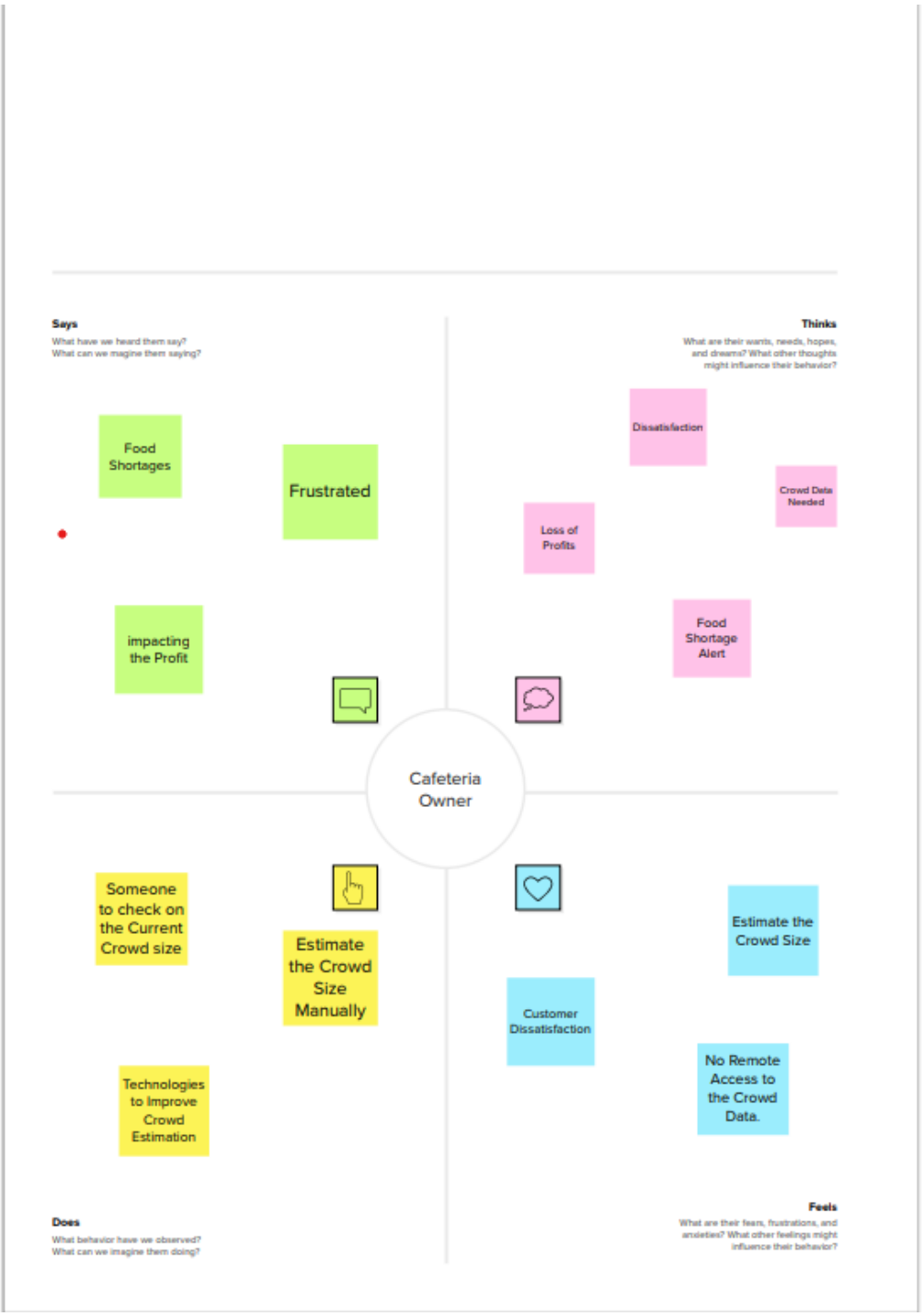
2.2 Empathy Map Canvas:

Template

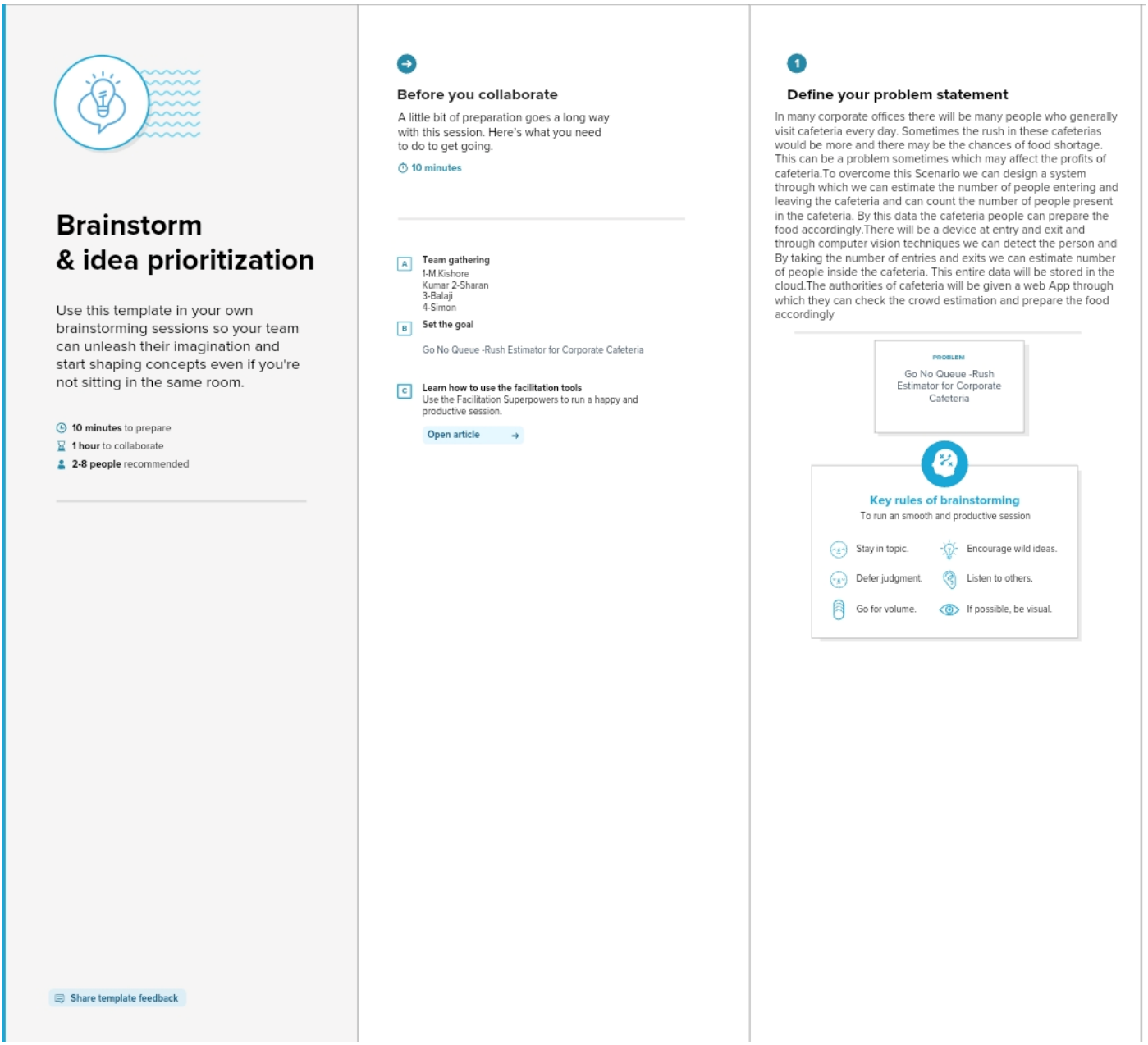
Empathy map

Use this framework to develop a deep, shared understanding and empathy for other people. An empathy map helps describe the aspects of a user's experience, needs and pain points, to quickly understand your users' experience and mindset.

Share template feedback



2.3 Ideation & Brainstorming:



2

Brainstorm

Write down any ideas that come to mind that address your problem statement.

10 minutes

M.Kishore kumar		Simon		Balaji		Sharan	
Implement a computer vision	Entry and Exit points	cloud-based database	collect data on cafeteria occupancy.	machine learning	predict future crowd	notification system	web application
cloud-based database	LOW Accuracy	people entering and leaving	Increase Food Stock	analyze data	cloud-based database	crowd estimation accuracy	cloud-based database

3

Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you and break it up into smaller sub-groups.

20 minutes

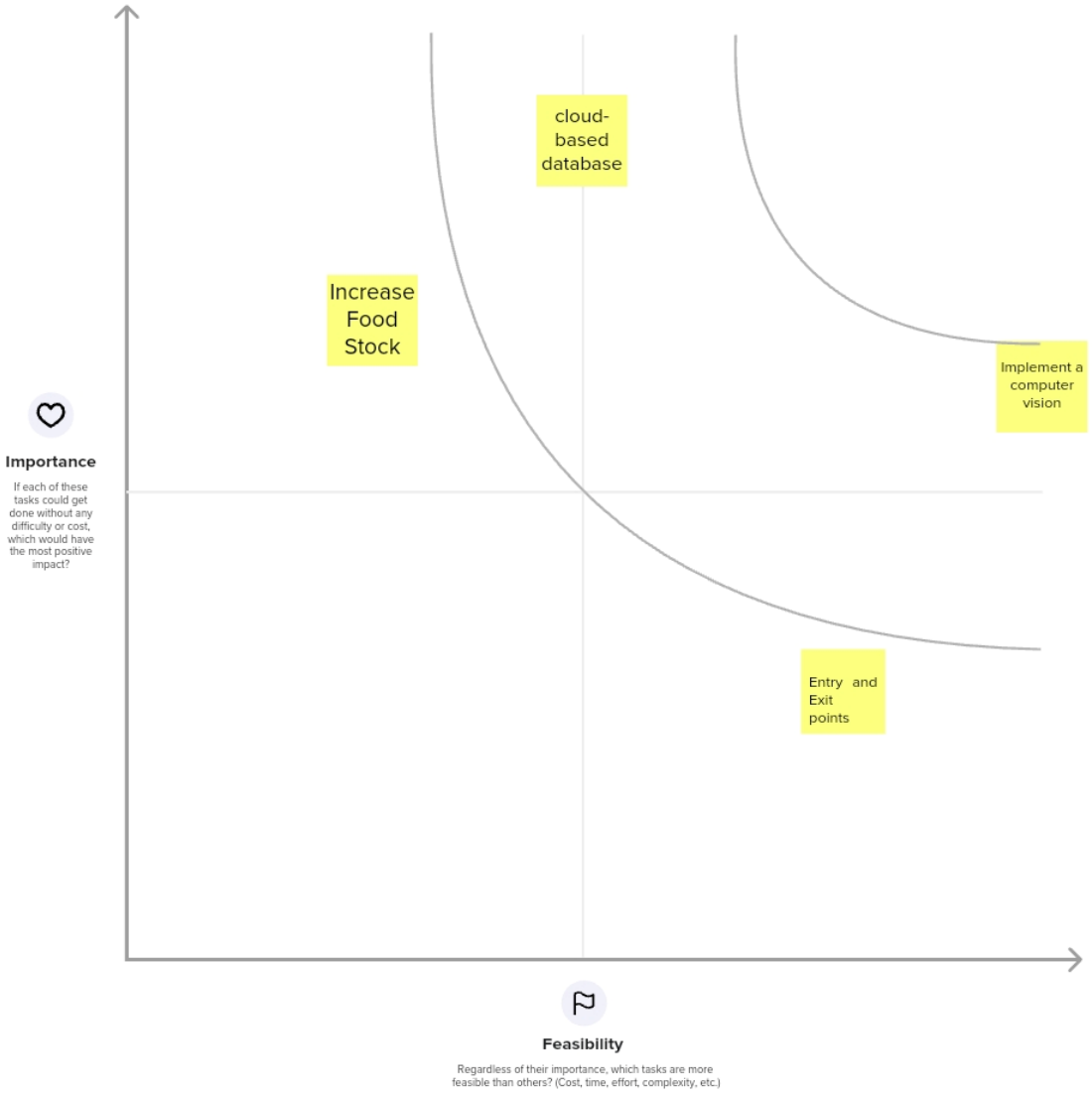
web application	Implement a computer vision	cloud-based database
Increase Food Stock	Entry and Exit points	people entering and leaving

4

Prioritize

Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

20 minutes



2.4 Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	The problem is the frequent rushes during peak hours in the cafeteria, leading to potential food shortage, dissatisfied customers, decreased profitability, and operational inefficiencies.
2.	Idea / Solution description	The solution is to implement a computer vision-based system at the entry and exit points of the cafeteria to estimate the number of people entering and leaving. This data is processed and aggregated to provide real-time crowd estimation. A web application is developed for cafeteria authorities to access and monitor this data, enabling them to make informed decisions regarding food preparation and staffing.
3.	Novelty / Uniqueness	The use of computer vision techniques to estimate crowd occupancy in a cafeteria is a novel approach. By leveraging video footage and implementing object detection and tracking algorithms, this solution provides real-time crowd estimation, enabling proactive management of food supply and reducing the risk of food shortage.
4.	Social Impact / Customer Satisfaction	The system improves customer satisfaction by ensuring an adequate supply of food during peak hours. It reduces wait times, minimizes the chances of food shortages, and enhances the overall dining experience. It also helps optimize cafeteria operations, leading to improved profitability and resource utilization
5.	Business Model (Revenue Model)	The revenue model can be based on a subscription or licensing model. The cafeteria management would pay a fee to access and use the web application and the cloud storage service. Alternatively, the system could be offered as a service, where the cafeteria management pays a monthly or yearly subscription fee based on their usage and the number of entry/exit points covered.

3. REQUIREMENT ANALYSIS

3.1 Functional requirement

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	Real-time Crowd Estimation	As a cafeteria manager, I want to be able to monitor the real-time crowd estimation in the cafeteria, so I can ensure an adequate supply of food during peak hours.
FR-2	Data Storage and Analytics	As a cafeteria supervisor, I want access to historical crowd estimation data and analytics, so I can analyze the cafeteria's performance, identify trends, and make informed decisions for future planning
FR-3	Integration and Compatibility	As a cafeteria administrator, I want the computer vision system to integrate seamlessly with existing access control systems or surveillance systems, ensuring compatibility and efficient data exchange.
FR-4	User-Friendly Web Application	As a cafeteria supervisor, I want the web application to have a user-friendly interface, providing clear visualizations and real-time updates of the crowd estimation, making it easy for me to monitor and manage the cafeteria's operations

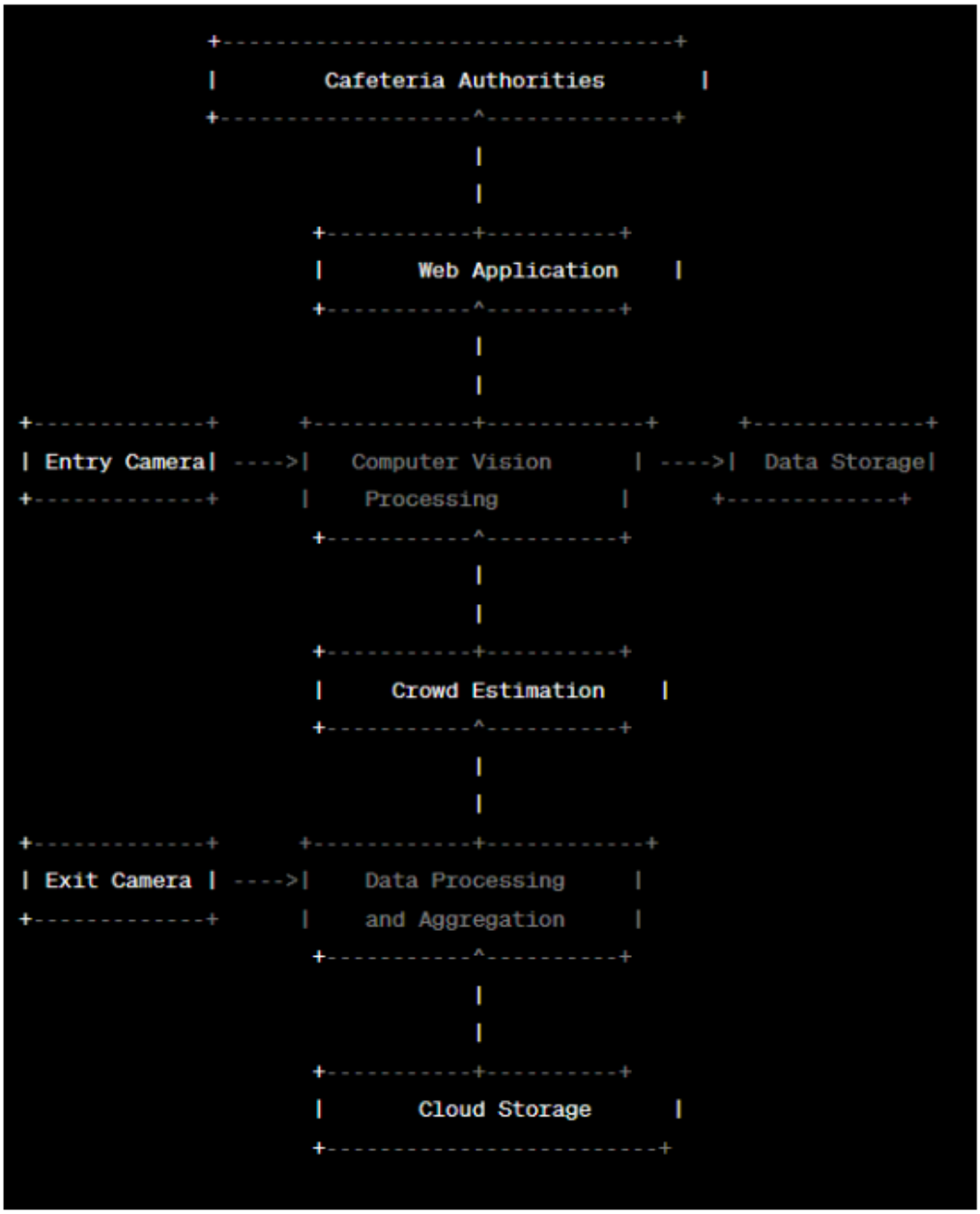
3.2 Non-Functional requirement

FR No.	Non-Functional Requirement	Description
NFR-1	Accuracy	The crowd estimation system should have a high degree of accuracy in counting and tracking individuals entering and exiting the cafeteria. The accuracy ensures reliable crowd estimation, which is essential for effective resource planning and preventing food shortages
NFR-2	Scalability	The solution should be scalable to accommodate varying cafeteria sizes and handle increasing data volume as the number of visitors grows. It should support the integration of additional entry and exit points without compromising the system's performance and accuracy.
NFR-3	Real-time Processing	The computer vision processing and data aggregation should be performed in real-time to provide up-to-date crowd estimation information. Real-time processing ensures that the cafeteria authorities can make timely decisions and take appropriate actions based on the current crowd levels.
NFR-4	Security and Privacy	The solution should prioritize the security and privacy of the collected data. It should employ robust security measures to protect the stored data from unauthorized access or breaches. Additionally, it should adhere to privacy regulations and ensure that the personal information of individuals is handled with utmost care
NFR-5	Reliability and Availability	The system should be highly reliable and available to ensure continuous operation. It should have mechanisms in place to handle potential failures, such as camera malfunctions or network disruptions, and recover seamlessly to prevent any disruption in crowd estimation or data storage.
NFR-6	Performance	The solution should demonstrate efficient performance, including fast response times for crowd estimation updates, quick data processing, and minimal latency in accessing historical data or generating analytics. It should be capable of handling the expected workload and maintaining optimal performance even during peak hours

4. PROJECT DESIGN

4.1 Data Flow Diagrams:

DFD Level 0 (Industry Standard)



4.2 Solution & Technical Architecture:

Solution & Technical Architecture:

The Cafeteria Crowd Estimation System utilizes computer vision techniques and cloud-based data storage to accurately estimate the number of people entering, leaving, and present in the cafeteria. Here is an overview of the solution and its technical architecture:

1. Hardware Setup:

- Entry and Exit Devices: Devices equipped with sensors or cameras are installed at the entry and exit points of the cafeteria to capture the movement of people.

2. Computer Vision:

- Image/Video Processing: The system uses computer vision algorithms to process the images or video streams captured by the entry and exit devices.
- Object Detection and Tracking: The system detects and tracks individuals in the images or video to determine their movement patterns.

3. Crowd Estimation:

- Counting Algorithm: By analyzing the detected individuals' movement patterns, the system counts the number of people entering and exiting the cafeteria.
- Presence Estimation: The system calculates the number of people present in the cafeteria by maintaining a count based on the entries and exits.

4. Cloud Storage and Data Analysis:

- Cloud Integration: The system is integrated with cloud storage services to store and manage the crowd estimation data.
- Data Processing: The system processes and analyzes the crowd estimation data to generate insights such as peak hours, historical patterns, and crowd trends.
- Web Application: A web-based application is developed to provide cafeteria authorities with access to real-time crowd estimation data, insights, and reports.

5. Communication and Reporting:

- IoT Connectivity: The entry and exit devices are connected to the cloud platform using IoT protocols for data transmission.
- Real-time Updates: The system provides real-time updates on crowd estimation to the web application, allowing cafeteria authorities to monitor the current crowd status.
- Reporting and Analytics: The web application generates reports and analytics based on the crowd estimation data, providing insights for decision-making.

6. Scalability and Flexibility:

- Multiple Entry/Exit Points: The system is designed to accommodate multiple entry and exit points, making it adaptable to different cafeteria layouts.
- Scalable Architecture: The solution architecture is scalable to handle varying crowd sizes and can be extended to additional cafeterias within an organization.

The combination of computer vision, cloud storage, data analysis, and web application development forms the technical architecture of the Cafeteria Crowd Estimation System. This architecture enables accurate crowd estimation, real-time monitoring, data-driven decision-making, and scalability to cater to the specific needs of corporate cafeterias.

4.3 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
Cafeteria Manager	Real-time Crowd Estimation	USN-1	As a cafeteria manager, I want to be able to monitor the real-time crowd estimation in the cafeteria, so I can ensure an adequate supply of food during peak hours	The web application should display the current number of people present in the cafeteria. The crowd estimation should be updated in real-time as people enter or leave the cafeteria.	High	M.kishore Kumar
Cafeteria Staff Member	Food Shortage Alerts	USN-2	As a cafeteria staff member, I want to receive alerts and notifications when the crowd estimation indicates a potential food shortage, so I can take immediate action to prevent any disruptions	The system should send real-time notifications to the staff members when the crowd estimation reaches a predefined threshold indicating a potential food shortage. - The notifications should be sent through a preferred communication channel (e.g., email, SMS)	High	Simon
Cafeteria Customer	Improved Dining Experience	USN-3	As a cafeteria customer, I want to experience shorter wait times and a seamless dining experience, knowing that	- The crowd estimation system should help the	Medium	Sharan

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Team Member
			the cafeteria has accurate crowd estimation to meet the demand	cafeteria staff manage the food supply efficiently, minimizing wait times for customers. - The system should ensure that an adequate supply of food is available during peak hours, reducing the likelihood of food shortages.		
Cafeteria Owner	Optimized Resource Allocation	USN-4	As a cafeteria owner, I want to optimize resource allocation, such as food preparation and staffing, based on the historical trends and predicted future crowd estimation, to improve profitability and operational efficiency	- The crowd estimation system should provide historical data and analytics to identify patterns and trends in customer footfall. - The system should generate accurate predictions of future crowd estimation based on various factors, such as time of day, day of the week, and special events.	High	Balaji

5. CODING & SOLUTIONING

Code:

```
import numpy as np
import cv2
import Person
import time
import pyttsx3
import requests
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

organization = "iftbe4"
deviceType = "PeopleCounter"
deviceId = "123123"
authMethod = "token"
authToken = "12341234"
engine = pyttsx3.init()
engine.say('Hello')
engine.runAndWait()
#Contadores de entrada y salida
cnt_up   = 0
cnt_down = 0

#Fuente de video
#cap = cv2.VideoCapture(0)
#cap = cv2.VideoCapture('people.mp4')


#Propiedades del video
##cap.set(3,160) #Width
##cap.set(4,120) #Height

#Imprime las propiedades de captura a consola
cap = cv2.VideoCapture('people.mp4')
#cap = cv2.VideoCapture(0)
for i in range(19):
    print (i, cap.get(i))

w = cap.get(3)
h = cap.get(4)
frameArea = h*w
areaTH = frameArea/250
print ('Area Threshold', areaTH)

#Lineas de entrada/salida
line_up = int(2*(h/5))
```

```
line_down   = int(3*(h/5))

up_limit =   int(1*(h/5))
down_limit = int(4*(h/5))

print ("Red line y:",str(line_down))
print ("Blue line y:", str(line_up))
line_down_color = (255,0,0)
line_up_color = (0,0,255)
pt1 =  [0, line_down];
pt2 =  [w, line_down];
pts_L1 = np.array([pt1,pt2], np.int32)
pts_L1 = pts_L1.reshape((-1,1,2))
pt3 =  [0, line_up];
pt4 =  [w, line_up];
pts_L2 = np.array([pt3,pt4], np.int32)
pts_L2 = pts_L2.reshape((-1,1,2))

pt5 =  [0, up_limit];
pt6 =  [w, up_limit];
pts_L3 = np.array([pt5,pt6], np.int32)
pts_L3 = pts_L3.reshape((-1,1,2))
pt7 =  [0, down_limit];
pt8 =  [w, down_limit];
pts_L4 = np.array([pt7,pt8], np.int32)
pts_L4 = pts_L4.reshape((-1,1,2))

#Substractor de fondo
fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows = True)

#Elementos estructurantes para filtros morfoogicos
kernelOp = np.ones((3,3),np.uint8)
kernelOp2 = np.ones((5,5),np.uint8)
kernelCl = np.ones((11,11),np.uint8)

#Variables
font = cv2.FONT_HERSHEY_SIMPLEX
persons = []
max_p_age = 5
pid = 1
def ibmwork(cnt_up,cnt_down,deviceCli):
   data = { 'UP' : cnt_up, 'down': cnt_down}
      #print data
   def myOnPublishCallback():
      print ("Published Up People Count = %s" % str(cnt_up), "Down People Count
= %s " % str(cnt_down), "to IBM Watson")

   success = deviceCli.publishEvent("PeopleCounter", "json", data, qos=0,
on_publish=myOnPublishCallback)
   if not success:
```

```
        print("Not connected to IoTF")



    deviceCli.disconnect()


def ibmstart(cnt_up,cnt_down):

    try:
            deviceOptions = {"org": organization, "type": deviceType, "id": deviceId,
"auth-method": authMethod, "auth-token": authToken}
            deviceCli = ibmiotf.device.Client(deviceOptions)
            print(type(deviceCli))
            #.....................................

    except Exception as e:
            print("Caught exception connecting device: %s" % str(e))
            sys.exit()
    deviceCli.connect()
    ibmwork(cnt_up,cnt_down,deviceCli)


while(cap.isOpened()):
##for image in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    #Lee una imagen de la fuente de video
    ret, frame = cap.read()
##    frame = image.array

    for i in persons:
        i.age_one() #age every person one frame
    ############################
    #   PRE-PROCESAMIENTO   #
    ############################

    #Aplica substraccion de fondo
    fgmask = fgbg.apply(frame)
    fgmask2 = fgbg.apply(frame)

    #Binariazcion para eliminar sombras (color gris)
    try:
        ret,imBin= cv2.threshold(fgmask,200,255,cv2.THRESH_BINARY)
        ret,imBin2 = cv2.threshold(fgmask2,200,255,cv2.THRESH_BINARY)
        #Opening (erode->dilate) para quitar ruido.
        mask = cv2.morphologyEx(imBin, cv2.MORPH_OPEN, kernelOp)
        mask2 = cv2.morphologyEx(imBin2, cv2.MORPH_OPEN, kernelOp)
        #Closing (dilate -> erode) para juntar regiones blancas.
        mask =  cv2.morphologyEx(mask , cv2.MORPH_CLOSE, kernelCl)
        mask2 = cv2.morphologyEx(mask2, cv2.MORPH_CLOSE, kernelCl)
```

```
    except:
        print('EOF')
        print ('UP:',cnt_up)
        print ('DOWN:',cnt_down)
        break
    ##################
    #   CONTORNOS   #
    ##################

    # RETR_EXTERNAL returns only extreme outer flags. All child contours are left
behind.
    contours0, hierarchy =
cv2.findContours(mask2,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours0:
        area = cv2.contourArea(cnt)
        if area > areaTH:
            ##################
            #   TRACKING    #
            ##################

            #Falta agregar condiciones para multipersonas, salidas y entradas de pantalla.

            M = cv2.moments(cnt)
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            x,y,w,h = cv2.boundingRect(cnt)

            new = True
            if cy in range(up_limit,down_limit):
                for i in persons:
                    if abs(cx-i.getX()) <= w and abs(cy-i.getY()) <= h:
                        # el objeto esta cerca de uno que ya se detecto antes
                        new = False
                        i.updateCoords(cx,cy)   #actualiza coordenadas en el objeto and resets age
                        if i.going_UP(line_down,line_up) == True:
                            cnt_up += 1;
                            print ("ID:",i.getId(),'crossed going up at',time.strftime("%c"))
                            engine.say('A Person is Going UP ')
                            engine.runAndWait()
                        elif i.going_DOWN(line_down,line_up) == True:
                            cnt_down += 1;
                            print ("ID:",i.getId(),'crossed going down at',time.strftime("%c"))
                            engine.say('A Person is Going Down')
                            engine.runAndWait()
                        break
                    if i.getState() == '1':
                        if i.getDir() == 'down' and i.getY() > down_limit:
                            i.setDone()
                        elif i.getDir() == 'up' and i.getY() < up_limit:
                            i.setDone()
```

```
                if i.timedOut():
                    #sacar i de la lista persons
                    index = persons.index(i)
                    persons.pop(index)
                    del i     #liberar la memoria de i
                if new == True:
                    p = Person.MyPerson(pid,cx,cy, max_p_age)
                    persons.append(p)
                    pid += 1
            #################
            #   DIBUJOS     #
            #################
            cv2.circle(frame,(cx,cy), 5, (0,0,255), -1)
            img = cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
            #cv2.drawContours(frame, cnt, -1, (0,255,0), 3)

    #END for cnt in contours0

    ##########################
    # DIBUJAR TRAYECTORIAS  #
    ##########################
    for i in persons:
##        if len(i.getTracks()) >= 2:
##            pts = np.array(i.getTracks(), np.int32)
##            pts = pts.reshape((-1,1,2))
##            frame = cv2.polylines(frame,[pts],False,i.getRGB())
##        if i.getId() == 9:
##            print str(i.getX()), ',', str(i.getY())
        cv2.putText(frame,
str(i.getId()),(i.getX(),i.getY()),font,0.3,i.getRGB(),1,cv2.LINE_AA)

    #################
    #   IMAGANES    #
    #################
    str_up = 'UP: '+ str(cnt_up)
    str_down = 'DOWN: '+ str(cnt_down)
    print('----------------------------------------')
    print ('UP:',cnt_up)
    print ('DOWN:',cnt_down)

    #r1 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&f
ield1='+str(cnt_up))
   # r2 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&f
ield2='+str(cnt_down))
   # print(r1.status_code)
   # print(r2.status_code)
    frame = cv2.polylines(frame,[pts_L1],False,line_down_color,thickness=2)
    frame = cv2.polylines(frame,[pts_L2],False,line_up_color,thickness=2)
```

```
    frame = cv2.polylines(frame,[pts_L3],False,(255,255,255),thickness=1)
    frame = cv2.polylines(frame,[pts_L4],False,(255,255,255),thickness=1)
    cv2.putText(frame, str_up ,(10,40),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame, str_up ,(10,40),font,0.5,(0,0,255),1,cv2.LINE_AA)
    cv2.putText(frame, str_down ,(10,90),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame, str_down ,(10,90),font,0.5,(255,0,0),1,cv2.LINE_AA)

    cv2.imshow('Frame',frame)
    #cv2.imshow('Mask',mask)

    #preisonar ESC para salir

    ibmstart(cnt_up,cnt_down)




# Disconnect the device and application from the cloud

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
#END while(cap.isOpened())

##################
#   LIMPIEZA    #
##################

cap.release()
cv2.destroyAllWindows()
```

5.1 Features:

1.Background Subtraction:

The code uses the `cv2.createBackgroundSubtractorMOG2` function to subtract the background from the current frame, resulting in a foreground mask.

2. Morphological Operations:

The code applies morphological operations such as opening (erosion followed by dilation) and closing (dilation followed by erosion) to remove noise and refine the foreground mask.

3. Contour Detection:

The code detects contours in the refined foreground mask using `cv2.findContours`. It filters out small contours and considers only those with an area above a certain threshold.

4. Object Tracking:

The code tracks individual persons by assigning a unique ID to each detected person. It updates the position of each person in subsequent frames and checks if they cross predefined lines in the frame.

5. Line Detection and Counting:

The code defines two lines in the frame, one for counting people going up and another for counting people going down. It tracks when a person crosses these lines and increments the corresponding count variables (`cnt_up` and `cnt_down`).

6. Visualization:

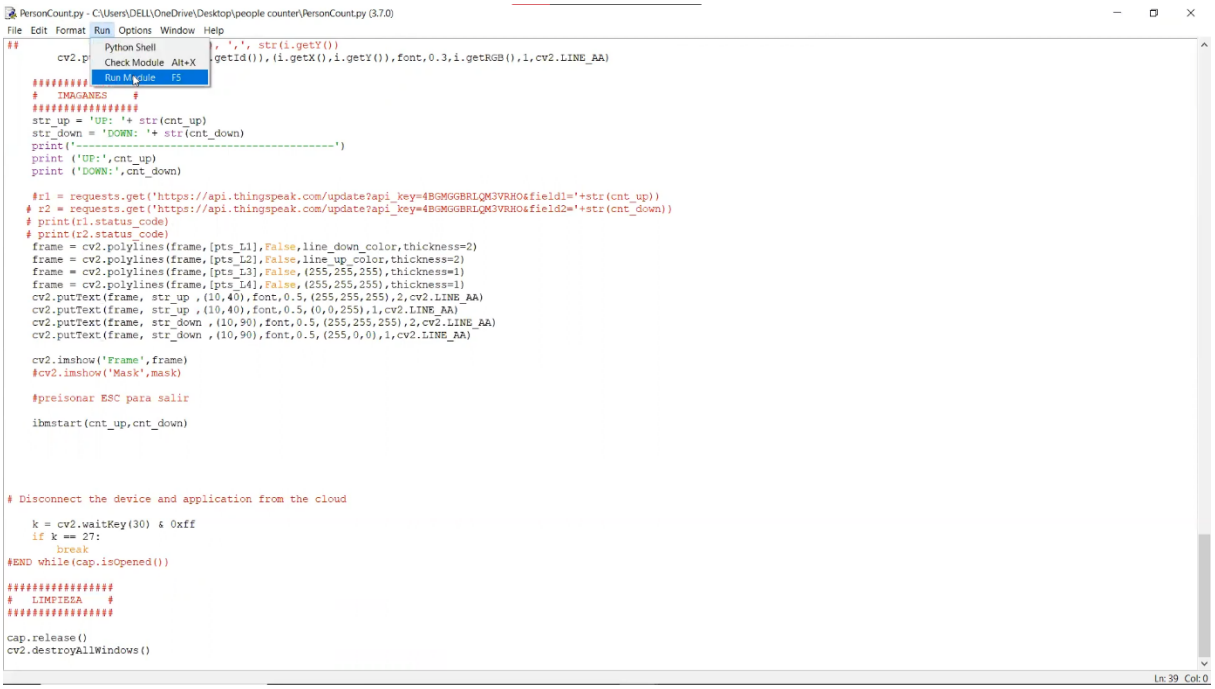
The code visualizes the tracked persons by drawing bounding boxes around them and displaying their IDs on the frame. It also displays the count of people going up and down in real-time.

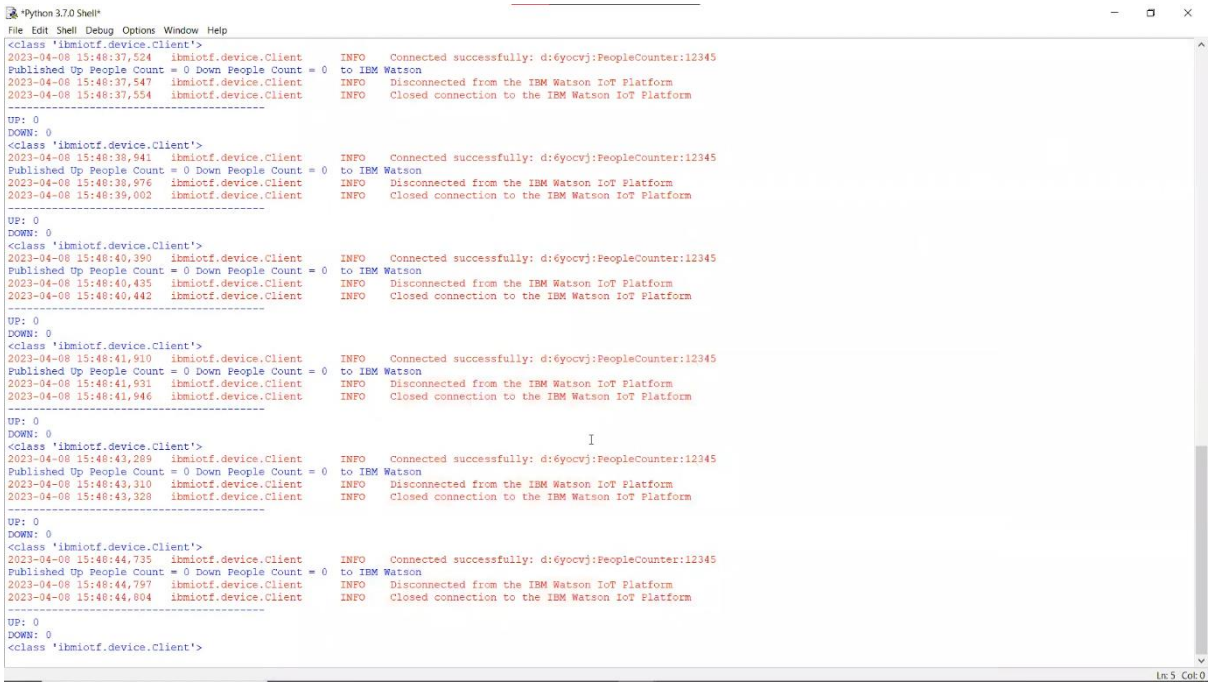
7. Integration with IBM Watson IoT:

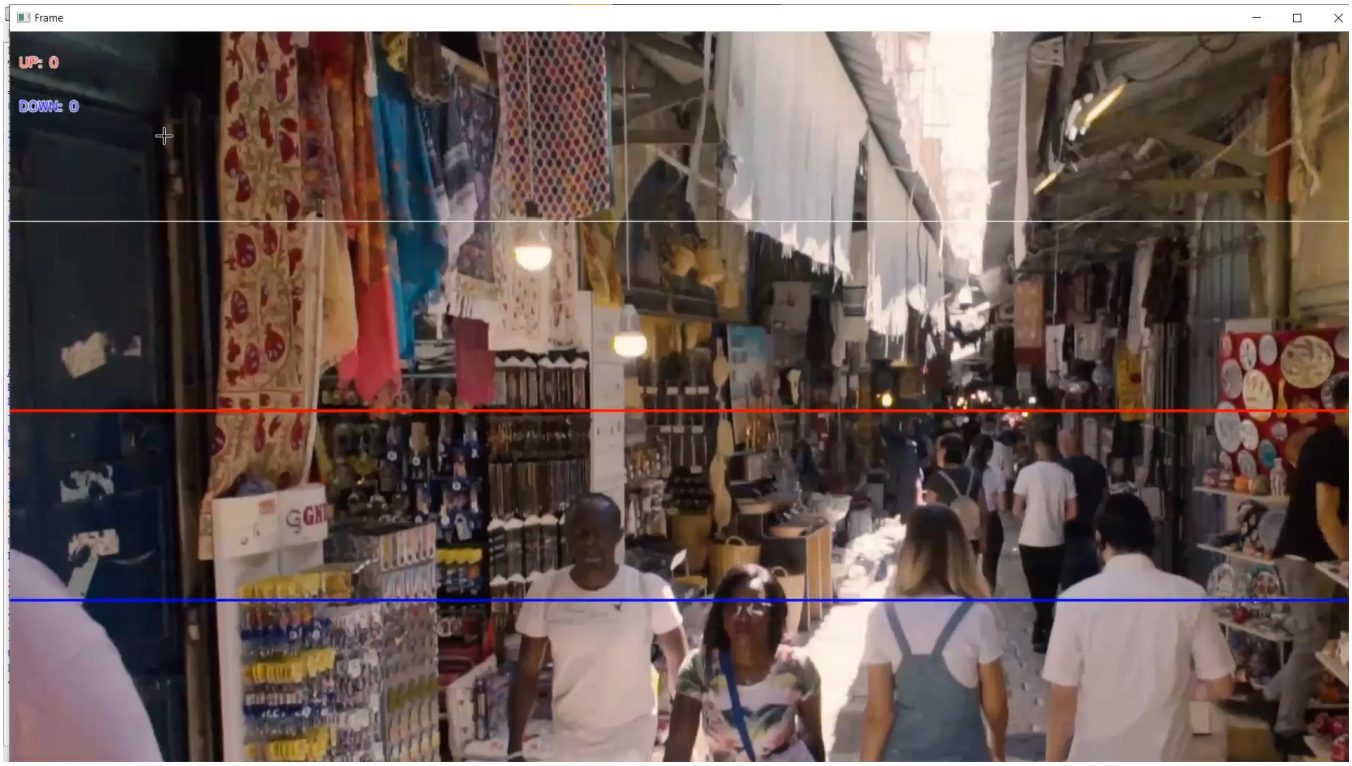
The code includes functions to connect to the IBM Watson IoT platform and publish the count data (number of people going up and down) as an event.

6. RESULTS

6.1 Performance Metrics




Python 3.7.0 Shell
File Edit Shell Debug Options Window Help
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:37,524 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:37,547 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:37,554 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:38,941 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:38,976 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:39,002 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:40,390 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:40,435 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:40,442 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:41,910 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:41,931 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:41,946 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:43,289 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:43,310 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:43,328 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
2023-04-08 15:48:44,735 ibmiotf.device.Client INFO Connected successfully: d:6yocvj:PeopleCounter:12345
Published Up People Count = 0 Down People Count = 0 to IBM Watson
2023-04-08 15:48:44,797 ibmiotf.device.Client INFO Disconnected from the IBM Watson IoT Platform
2023-04-08 15:48:44,804 ibmiotf.device.Client INFO Closed connection to the IBM Watson IoT Platform
UP: 0
DOWN: 0
<class 'ibmiotf.device.Client'>
Ln: 5 Col: 0


Frame
UP: 0
DOWN: 0

The Performance Metrics are

1. Counting Accuracy: The accuracy of the people counting algorithm is crucial. It can be measured by comparing the counted number of people with ground truth data, manually collected or obtained from another reliable source.

2. False Positives and False Negatives: False positives occur when the algorithm incorrectly counts an object or detects a person where there is none. False negatives occur when the algorithm fails to count a person or misses detection. Minimizing false positives and false negatives is important for accurate counting.

3. Tracking Robustness: The ability of the algorithm to track individuals accurately and consistently over time is another performance metric. It can be evaluated by measuring the tracking success rate and the number of instances where tracking fails or loses track of a person.

4. Processing Speed: The computational efficiency of the code is critical, especially when processing real-time video streams. The processing speed can be measured in frames per second (FPS) to ensure it meets the requirements of the application.

5. Scalability: The code's ability to handle different scenarios, such as crowded environments or varying lighting conditions, can be assessed. Evaluating its performance under different settings and scaling the system to handle larger video streams or multiple cameras can provide valuable insights.

6. Resource Utilization: Assessing the code's resource utilization, such as CPU and memory usage, is important to ensure it can run on the target hardware and handle multiple tasks concurrently without excessive resource consumption.

7. System Stability: The code's ability to run continuously without crashes or memory leaks is essential for long-term operation. Evaluating system stability and monitoring for any issues or errors is crucial to maintain reliable people counting.

7. ADVANTAGES & DISADVANTAGES

Advantages:

1. **Improved food preparation:**
   By accurately estimating the number of people entering and leaving the cafeteria, the authorities can better plan and prepare the food accordingly. This reduces the chances of food shortage and ensures that customers are served efficiently.

2. **Cost optimization:**
   With a better understanding of the crowd estimation, the cafeteria can optimize their food production, minimizing waste and reducing costs. They can adjust the quantity of ingredients and minimize overstocking.

3. **Enhanced customer satisfaction:**
   By ensuring a sufficient food supply, customers are less likely to face situations where their desired items are unavailable. This improves customer satisfaction and enhances their overall dining experience.

4. **Real-time data analysis:**
   The system can provide real-time data on the number of people in the cafeteria, allowing the authorities to make timely decisions based on the current crowd situation. This helps in effective management and resource allocation.

Disadvantages:

1. **Implementation complexity:**
   Developing and implementing a system based on computer vision techniques for crowd estimation requires expertise in computer vision, hardware setup, and software development. It may involve challenges related to accuracy, scalability, and system integration.

2. **Privacy concerns:**
   The use of computer vision technology to count individuals in a cafeteria raises privacy concerns. The system needs to address privacy issues and ensure compliance with data protection regulations. Measures such as anonymization and data security should be implemented.

3. **Cost of implementation:**
   Setting up the required infrastructure, including entry/exit devices, computer vision cameras, and cloud storage, can involve significant upfront costs. Additionally, ongoing maintenance and support of the system may require additional resources.

4. **Technical limitations:**
   Computer vision techniques may have limitations in accurately counting people in certain scenarios, such as crowded or low-light environments. The system should account for potential challenges and provide reliable results under various conditions.

5. **User adoption and training:**
   Introducing a new system to cafeteria staff and users requires proper training and education. Staff members need to understand how to operate the system, while users may need to adapt to any changes in the cafeteria entry/exit process.

8. CONCLUSION

1. In conclusion, the proposed project aims to address the problem of food shortage in corporate cafeterias by implementing a system that estimates the number of people entering and leaving the cafeteria. By leveraging computer vision techniques and cloud storage, the system provides real-time data on crowd estimation, allowing cafeteria authorities to prepare food accordingly and optimize their operations.

2. The project offers several advantages, including improved food preparation, cost optimization, enhanced customer satisfaction, and real-time data analysis. These benefits can lead to better resource management, reduced food waste, and a more positive dining experience for customers.

3. However, it is important to consider the challenges and limitations associated with the proposed solution. These include implementation complexity, privacy concerns, cost of implementation, technical limitations, and the need for user adoption and training.

4. Before proceeding with the project, a thorough evaluation should be conducted to assess its feasibility, considering factors such as budget, resources, infrastructure requirements, and potential privacy implications. It is essential to address any privacy concerns and ensure compliance with relevant regulations.

5. Overall, by carefully considering the advantages, disadvantages, and potential challenges, the project can contribute to improving the efficiency and customer satisfaction of corporate cafeterias, ultimately benefiting both the cafeteria authorities and the individuals who rely on these facilities for their meals.

9. FUTURE SCOPE

1. Advanced analytics and prediction: The system can be further enhanced by implementing advanced analytics and prediction algorithms. By analyzing historical data and considering various factors such as day of the week, weather conditions, and events, the system can make accurate predictions about future crowd sizes. This can help cafeterias in proactive planning and minimizing food wastage.

2. Integration with mobile apps: Integrating the crowd estimation system with mobile applications can provide users with real-time information about the current crowd levels in the cafeteria. Users can check the occupancy status before visiting and make informed decisions about the best time to avoid crowded periods. Additionally, mobile apps can offer personalized food recommendations and pre-ordering options to enhance the overall dining experience.

3. Integration with cashless payment systems: By integrating the crowd estimation system with cashless payment systems, cafeteria authorities can streamline the ordering and payment process. Users can place orders in advance through the app or self-service kiosks, reducing waiting times and improving efficiency. The system can also provide data on popular food items and consumption patterns, aiding in menu planning and inventory management.

4. Integration with loyalty programs: Linking the crowd estimation system with loyalty programs can offer additional benefits to cafeteria customers. Rewards, discounts, and personalized offers can be tailored based on customer preferences and behavior. This can incentivize customer loyalty and encourage repeat visits.

5. Energy-efficient solutions: Implementing energy-efficient technologies, such as smart lighting and HVAC systems, can be integrated with the crowd estimation system. By adjusting lighting levels and temperature settings based on real-time occupancy data, cafeterias can optimize energy consumption and reduce costs.

6. Expansion to other sectors: The technology and concepts developed for crowd estimation in cafeterias can be extended to other sectors, such as restaurants, retail stores, and entertainment venues. This can help these establishments in managing crowd flow, improving customer experience, and optimizing their operations.

It is important to note that these future scope possibilities require careful planning, research, and consideration of various factors specific to each implementation scenario. However, they present opportunities for further innovation and improvement in managing crowd estimation and enhancing the overall customer experience

10. APPENDIX

Source Code:

1.Node-Red-Jason:

```
[
    {
        "id": "c5934f78f4190c79",
        "type": "tab",
        "label": "Flow 3",
        "disabled": false,
        "info": "",
        "env": []
    },
    {
        "id": "e5f73f2b952aa373",
        "type": "ibmiot in",
        "z": "c5934f78f4190c79",
        "authentication": "apiKey",
        "apiKey": "6d8be82fde335a13",
        "inputType": "evt",
        "logicalInterface": "",
        "ruleId": "",
        "deviceId": "12345",
        "applicationId": "",
        "deviceType": "PeopleCounter",
        "eventType": "+",
        "commandType": "",
        "format": "json",
        "name": "IBM IoT PeopleCounter",
        "service": "registered",
        "allDevices": "",
        "allApplications": "",
        "allDeviceTypes": "",
        "allLogicalInterfaces": "",
        "allEvents": true,
        "allCommands": "",
        "allFormats": "",
        "qos": 0,
        "x": 300,
        "y": 100,
        "wires": [
            [
                "a6dc594121aa7d9c",
                "e79b5aeebda95012",
                "c4f36429d4857ac1"
```

```
            ]
        ]
    },
    {
        "id": "a6dc594121aa7d9c",
        "type": "debug",
        "z": "c5934f78f4190c79",
        "name": "debug 4",
        "active": true,
        "tosidebar": true,
        "console": false,
        "tostatus": false,
        "complete": "payload",
        "targetType": "msg",
        "statusVal": "",
        "statusType": "auto",
        "x": 620,
        "y": 100,
        "wires": []
    },
    {
        "id": "e79b5aeebda95012",
        "type": "function",
        "z": "c5934f78f4190c79",
        "name": "UP",
        "func": "msg.payload=msg.payload.UP\n\n\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "initialize": "",
        "finalize": "",
        "libs": [],
        "x": 450,
        "y": 220,
        "wires": [
            [
                "581720e738110749"
            ]
        ]
    },
    {
        "id": "581720e738110749",
        "type": "ui_gauge",
        "z": "c5934f78f4190c79",
        "name": "",
        "group": "b7771ab55bf3a9e9",
        "order": 2,
        "width": "8",
        "height": "6",
```

```
            "gtype": "donut",
            "title": "People Coming in",
            "label": "units",
            "format": "{{value}}",
            "min": 0,
            "max": "20",
            "colors": [
                "#00b500",
                "#e6e600",
                "#ca3838"
            ],
            "seg1": "",
            "seg2": "",
            "diff": false,
            "className": "",
            "x": 710,
            "y": 320,
            "wires": []
        },
        {
            "id": "c4f36429d4857ac1",
            "type": "function",
            "z": "c5934f78f4190c79",
            "name": "DOWN",
            "func": "msg.payload=msg.payload.down\n\n\nreturn msg;",
            "outputs": 1,
            "noerr": 0,
            "initialize": "",
            "finalize": "",
            "libs": [],
            "x": 420,
            "y": 380,
            "wires": [
                [
                    "6b754b60b6277219"
                ]
            ]
        },
        {
            "id": "6b754b60b6277219",
            "type": "ui_gauge",
            "z": "c5934f78f4190c79",
            "name": "",
            "group": "b7771ab55bf3a9e9",
            "order": 1,
            "width": "8",
            "height": "6",
            "gtype": "donut",
```

```
        "title": "People Coming out",
        "label": "units",
        "format": "{{value}}",
        "min": 0,
        "max": "20",
        "colors": [
            "#ca3838",
            "#e6e600",
            "#00b500"
        ],
        "seg1": "",
        "seg2": "",
        "diff": false,
        "className": "",
        "x": 670,
        "y": 480,
        "wires": []
    },
    {
        "id": "6d8be82fde335a13",
        "type": "ibmiot",
        "name": "",
        "keepalive": "60",
        "serverName": "",
        "cleansession": true,
        "appId": "",
        "shared": false
    },
    {
        "id": "b7771ab55bf3a9e9",
        "type": "ui_group",
        "name": "People Counter",
        "tab": "fac361559fed2381",
        "order": 1,
        "disp": true,
        "width": "22",
        "collapse": false,
        "className": ""
    },
    {
        "id": "fac361559fed2381",
        "type": "ui_tab",
        "name": "Smart IoT Based People Counter",
        "icon": "dashboard",
        "disabled": false,
        "hidden": false
    }
]
```

2.People Counter:

```
import numpy as np
import cv2
import Person
import time
import pyttsx3
import requests
import time
import sys
import ibmiotf.application
import ibmiotf.device
import random

organization = "iftbe4"
deviceType = "PeopleCounter"
deviceId = "123123"
authMethod = "token"
authToken = "12341234"
engine = pyttsx3.init()
engine.say('Hello')
engine.runAndWait()
#Contadores de entrada y salida
cnt_up   = 0
cnt_down = 0

#Fuente de video
#cap = cv2.VideoCapture(0)
#cap = cv2.VideoCapture('people.mp4')


#Propiedades del video
##cap.set(3,160) #Width
##cap.set(4,120) #Height

#Imprime las propiedades de captura a consola
cap = cv2.VideoCapture('people.mp4')
#cap = cv2.VideoCapture(0)
for i in range(19):
    print (i, cap.get(i))

w = cap.get(3)
h = cap.get(4)
frameArea = h*w
areaTH = frameArea/250
print ('Area Threshold', areaTH)
```

```
#Lineas de entrada/salida
line_up = int(2*(h/5))
line_down   = int(3*(h/5))

up_limit =   int(1*(h/5))
down_limit = int(4*(h/5))

print ("Red line y:",str(line_down))
print ("Blue line y:", str(line_up))
line_down_color = (255,0,0)
line_up_color = (0,0,255)
pt1 =  [0, line_down];
pt2 =  [w, line_down];
pts_L1 = np.array([pt1,pt2], np.int32)
pts_L1 = pts_L1.reshape((-1,1,2))
pt3 =  [0, line_up];
pt4 =  [w, line_up];
pts_L2 = np.array([pt3,pt4], np.int32)
pts_L2 = pts_L2.reshape((-1,1,2))

pt5 =  [0, up_limit];
pt6 =  [w, up_limit];
pts_L3 = np.array([pt5,pt6], np.int32)
pts_L3 = pts_L3.reshape((-1,1,2))
pt7 =  [0, down_limit];
pt8 =  [w, down_limit];
pts_L4 = np.array([pt7,pt8], np.int32)
pts_L4 = pts_L4.reshape((-1,1,2))

#Substractor de fondo
fgbg = cv2.createBackgroundSubtractorMOG2(detectShadows = True)

#Elementos estructurantes para filtros morfoogicos
kernelOp = np.ones((3,3),np.uint8)
kernelOp2 = np.ones((5,5),np.uint8)
kernelCl = np.ones((11,11),np.uint8)

#Variables
font = cv2.FONT_HERSHEY_SIMPLEX
persons = []
max_p_age = 5
pid = 1
def ibmwork(cnt_up,cnt_down,deviceCli):
    data = { 'UP' : cnt_up, 'down': cnt_down}
        #print data
    def myOnPublishCallback():
```

```
        print ("Published Up People Count = %s" % str(cnt_up), "Down People
Count = %s " % str(cnt_down), "to IBM Watson")

    success = deviceCli.publishEvent("PeopleCounter", "json", data, qos=0,
on_publish=myOnPublishCallback)
    if not success:
        print("Not connected to IoTF")



    deviceCli.disconnect()


def ibmstart(cnt_up,cnt_down):

    try:
            deviceOptions = {"org": organization, "type": deviceType, "id":
deviceId, "auth-method": authMethod, "auth-token": authToken}
            deviceCli = ibmiotf.device.Client(deviceOptions)
            print(type(deviceCli))
            #.....................................

    except Exception as e:
            print("Caught exception connecting device: %s" % str(e))
            sys.exit()
    deviceCli.connect()
    ibmwork(cnt_up,cnt_down,deviceCli)


while(cap.isOpened()):
##for image in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=True):
    #Lee una imagen de la fuente de video
    ret, frame = cap.read()
##    frame = image.array

    for i in persons:
        i.age_one() #age every person one frame
    ############################
    #   PRE-PROCESAMIENTO   #
    ############################

    #Aplica substraccion de fondo
    fgmask = fgbg.apply(frame)
    fgmask2 = fgbg.apply(frame)

    #Binariazcion para eliminar sombras (color gris)
    try:
```

```
        ret,imBin= cv2.threshold(fgmask,200,255,cv2.THRESH_BINARY)
        ret,imBin2 = cv2.threshold(fgmask2,200,255,cv2.THRESH_BINARY)
        #Opening (erode->dilate) para quitar ruido.
        mask = cv2.morphologyEx(imBin, cv2.MORPH_OPEN, kernelOp)
        mask2 = cv2.morphologyEx(imBin2, cv2.MORPH_OPEN, kernelOp)
        #Closing (dilate -> erode) para juntar regiones blancas.
        mask =  cv2.morphologyEx(mask , cv2.MORPH_CLOSE, kernelCl)
        mask2 = cv2.morphologyEx(mask2, cv2.MORPH_CLOSE, kernelCl)
    except:
        print('EOF')
        print ('UP:',cnt_up)
        print ('DOWN:',cnt_down)
        break
    ##################
    #   CONTORNOS    #
    ##################

    # RETR_EXTERNAL returns only extreme outer flags. All child contours are
left behind.
    contours0, hierarchy =
cv2.findContours(mask2,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_SIMPLE)
    for cnt in contours0:
        area = cv2.contourArea(cnt)
        if area > areaTH:
            ##################
            #   TRACKING    #
            ##################

            #Falta agregar condiciones para multipersonas, salidas y entradas
de pantalla.

            M = cv2.moments(cnt)
            cx = int(M['m10']/M['m00'])
            cy = int(M['m01']/M['m00'])
            x,y,w,h = cv2.boundingRect(cnt)

            new = True
            if cy in range(up_limit,down_limit):
                for i in persons:
                    if abs(cx-i.getX()) <= w and abs(cy-i.getY()) <= h:
                        # el objeto esta cerca de uno que ya se detecto antes
                        new = False
                        i.updateCoords(cx,cy)   #actualiza coordenadas en el
objeto and resets age
                        if i.going_UP(line_down,line_up) == True:
                            cnt_up += 1;
                            print ("ID:",i.getId(),'crossed going up
at',time.strftime("%c"))
```

```
                            engine.say('A Person is Going UP ')
                            engine.runAndWait()
                        elif i.going_DOWN(line_down,line_up) == True:
                            cnt_down += 1;
                            print ("ID:",i.getId(),'crossed going down
at',time.strftime("%c"))
                            engine.say('A Person is Going Down')
                            engine.runAndWait()
                        break
                    if i.getState() == '1':
                        if i.getDir() == 'down' and i.getY() > down_limit:
                            i.setDone()
                        elif i.getDir() == 'up' and i.getY() < up_limit:
                            i.setDone()
                    if i.timedOut():
                        #sacar i de la lista persons
                        index = persons.index(i)
                        persons.pop(index)
                        del i     #liberar la memoria de i
                if new == True:
                    p = Person.MyPerson(pid,cx,cy, max_p_age)
                    persons.append(p)
                    pid += 1
            ##################
            #   DIBUJOS      #
            ##################
            cv2.circle(frame,(cx,cy), 5, (0,0,255), -1)
            img = cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,0),2)
            #cv2.drawContours(frame, cnt, -1, (0,255,0), 3)

    #END for cnt in contours0

    ############################
    # DIBUJAR TRAYECTORIAS  #
    ############################
    for i in persons:
##        if len(i.getTracks()) >= 2:
##            pts = np.array(i.getTracks(), np.int32)
##            pts = pts.reshape((-1,1,2))
##            frame = cv2.polylines(frame,[pts],False,i.getRGB())
##        if i.getId() == 9:
##            print str(i.getX()), ',', str(i.getY())
        cv2.putText(frame,
str(i.getId()),(i.getX(),i.getY()),font,0.3,i.getRGB(),1,cv2.LINE_AA)

    ##################
    #   IMAGANES    #
    ##################
```

```
    str_up = 'UP: '+ str(cnt_up)
    str_down = 'DOWN: '+ str(cnt_down)
    print('----------------------------------------')
    print ('UP:',cnt_up)
    print ('DOWN:',cnt_down)

    #r1 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&field
1='+str(cnt_up))
   # r2 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&field
2='+str(cnt_down))
   # print(r1.status_code)
   # print(r2.status_code)
    frame = cv2.polylines(frame,[pts_L1],False,line_down_color,thickness=2)
    frame = cv2.polylines(frame,[pts_L2],False,line_up_color,thickness=2)
    frame = cv2.polylines(frame,[pts_L3],False,(255,255,255),thickness=1)
    frame = cv2.polylines(frame,[pts_L4],False,(255,255,255),thickness=1)
    cv2.putText(frame, str_up ,(10,40),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame, str_up ,(10,40),font,0.5,(0,0,255),1,cv2.LINE_AA)
    cv2.putText(frame, str_down ,(10,90),font,0.5,(255,255,255),2,cv2.LINE_AA)
    cv2.putText(frame, str_down ,(10,90),font,0.5,(255,0,0),1,cv2.LINE_AA)

    cv2.imshow('Frame',frame)
    #cv2.imshow('Mask',mask)

    #preisonar ESC para salir

    ibmstart(cnt_up,cnt_down)




# Disconnect the device and application from the cloud

    k = cv2.waitKey(30) & 0xff
    if k == 27:
        break
#END while(cap.isOpened())

#################
#   LIMPIEZA    #
#################

cap.release()
cv2.destroyAllWindows()
```

GitHub & Project Video Demo Link

GitHub Link:

https://github.com/naanmudhalvan-SI/PBL-NT-GP-19150-1684125325/tree/main