Project Report Format

1. **INTRODUCTION**
    1.1 Project Overview
    1.2 Purpose
2. **IDEATION & PROPOSED SOLUTION**
    2.1 Problem Statement Definition
    2.2 Empathy Map Canvas
    2.3 Ideation & Brainstorming
    2.4 Proposed Solution
3. **REQUIREMENT ANALYSIS**
    3.1 Functional requirement
    3.2 Non-Functional requirements
4. **PROJECT DESIGN**
    4.1 Data Flow Diagrams
    4.2 Solution & Technical Architecture
    4.3 User Stories
5. **CODING & SOLUTIONING (Explain the features added in the project along with code)**
    5.1 Feature 1
    5.2 Feature 2
    5.3 Database Schema (if Applicable)
6. **RESULTS**
    6.1 Performance Metrics
7. **ADVANTAGES & DISADVANTAGES**
8. **CONCLUSION**
9. **FUTURE SCOPE**
10. **APPENDIX**
    Source Code ,GitHub & Project Video Demo Link

INTRODUCTION

1.1 Project Overview

The project aims to develop a smart cafeteria management system that utilizes video processing and object detection algorithms to count the number of people entering and exiting the cafeteria. This system will provide real-time data and insights for efficient cafeteria management and resource allocation.

1.2 Purpose

The purpose of this project report is to document the development and implementation of the smart cafeteria management system. It serves as a comprehensive overview of the project, providing details on the technology stack, technical architecture, user stories, application characteristics, and data flow. The report is intended to provide a clear understanding of the project's objectives, methodologies, and outcomes to the stakeholders and project team members involved.

2. IDEATION & PROPOSED SOLUTION

2.1 Problem Statement Definition

Ideation Phase
Define the Problem Statements

Date	06 May 2023
Team ID	NM2023TMID17628
Project Name	Go No Queue-Rush Estimator for corporate cafeteria
Maximum Marks	2 Marks

Problem Statement :

A corporate cafeteria serves hundreds of employee's during peak hours often delaying the order which results in longer queues.

The goal of the system is to estimate the rush at the cafeteria and provide employees with real-time information about the expected waiting time, helping them make informed decisions about when to visit the cafeteria.

Problem Statement (PS)	I am (Customer)	I'm trying to	But	Because	Which makes me feel
A corporate cafeteria serves hundreds of employee's during peak hours often delaying the order which results in longer queues	An employee	Order food on the cafeteria	There is always a large queue	There is no way to predict the rush in the cafeteria accurately	Tired and frustrated

2.2 Empathy Map Canvas

Ideation Phase : Empathize & Discover

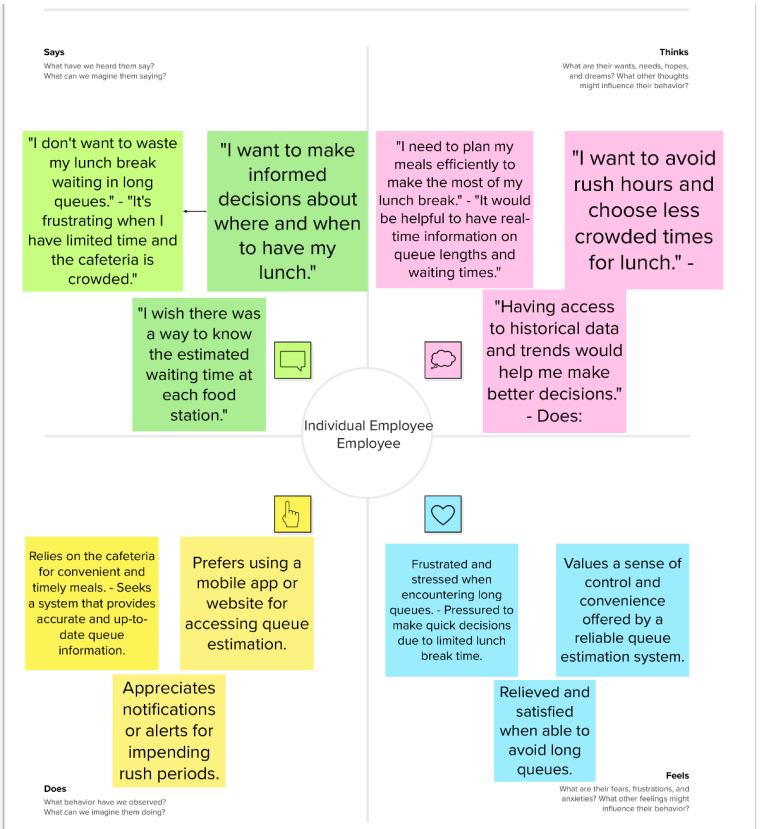
Date	18 May 2023
Team ID	NM2023TMID17628
Project Name	Go No Queue-Rush Estimator for corporate cafeteria
Maximum Marks	4 Marks

.Go No Queue-Rush Estimator for corporate cafeteria

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

Brainstorm & Idea Prioritization Template

Go No Queue -Rush Estimator for Corporate Cafeteria

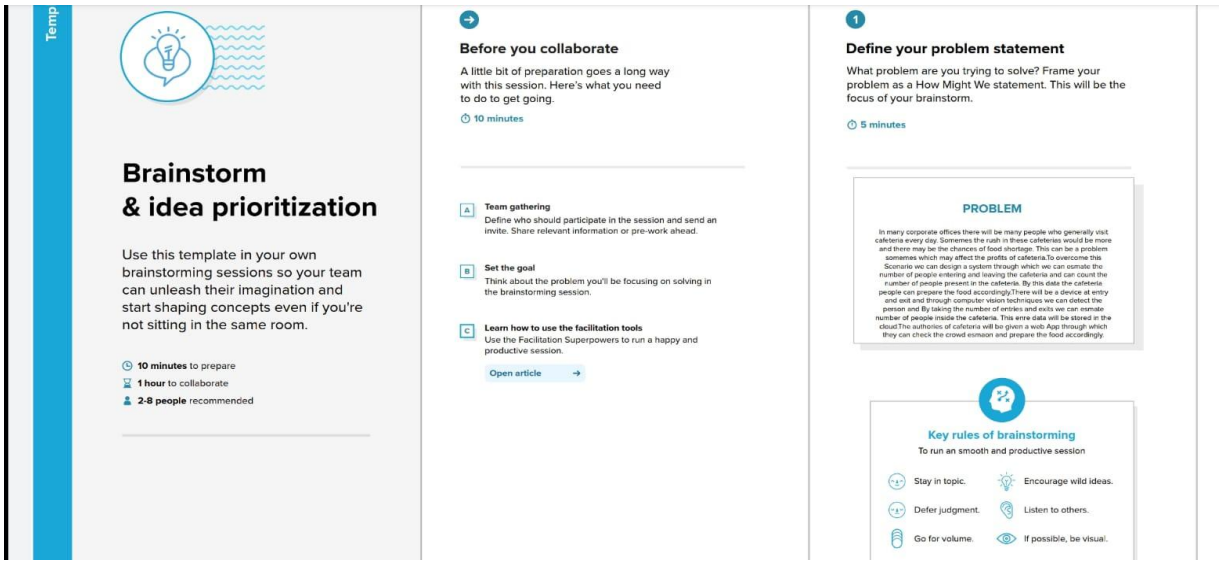
Date	20 May 2023
Team ID	NM2023TMID17628
Project Name	Go No Queue -Rush Estimator for Corporate Cafeteria
Maximum Marks	4 Marks

Brainstorm & Idea Prioritization Template:

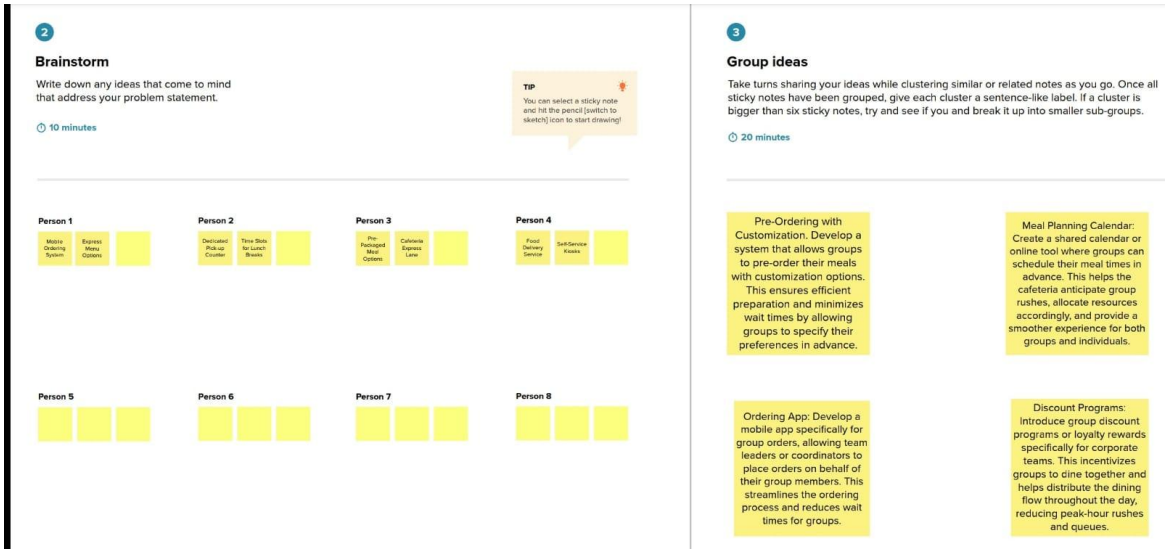
Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich amount of creative solutions.

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

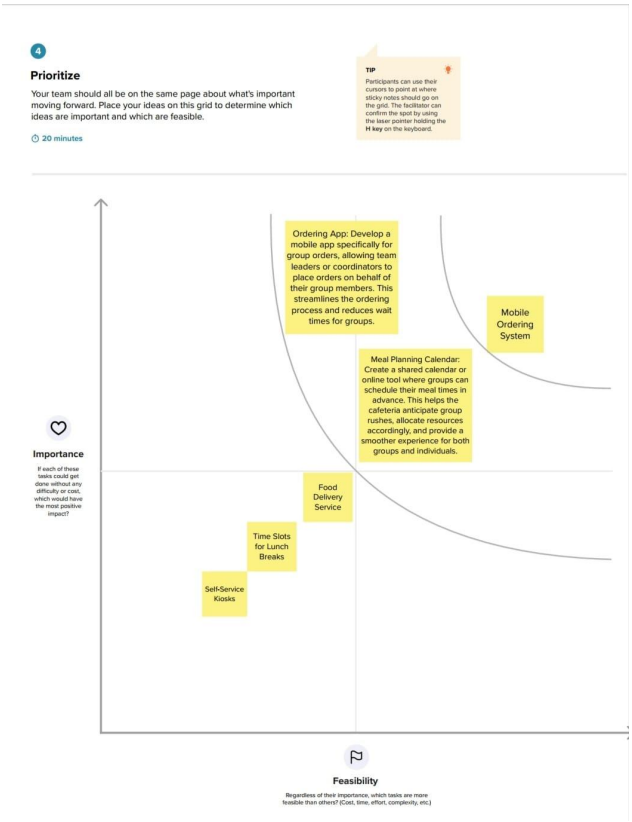
Step-1: Team Gathering, Collaboration and Select the Problem Statement



Step-2: Brainstorm, Idea Listing and Grouping



Step-3: Idea Prioritization



2.4 Proposed Solution :

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	A corporate cafeteria serves hundreds of employee's during peak hours often delaying the order which results in longer queues
2.	Idea / Solution description	We implement the solution using an IoT device integrated with a camera module which is connected with IBM IoT platform which communicates with node red. By the Web UI through which the end user gets the people count.
3.	Novelty / Uniqueness	1. Integration with IBM IoT platform 2. Node-RED is utilized as the backend to handle data from IoT device 3. Web based User Interface is implemented
4.	Social Impact / Customer Satisfaction	1. Improved customer Experience 2. Efficient resource allocation 3. Reduced queue time 4. Improved planning
5.	Business Model (Revenue Model)	1. Installation and Set-up up fee 2. Yearly basis subscription fee 3. Extra cost for UI interface specific design 4. Additional hardware: Multiple IoT device for various regions of the cafeteria
6.	Scalability of the Solution	1. Handling increased user base 2. Optimal performance and response time is to be maintained 3. By leveraging cloud based infrastructure

3.REQUIREMENT ANALYSIS

Project Design Phase-II
Solution Requirements (Functional & Non-functional)

Date	20May2023
Team ID	NM2023TMID17628
Project Name	Go No Queue -Rush Estimator for Corporate Cafeteria

3.1 Functional Requirements:

Following are the functional requirements of the proposed solution.

FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration through Form Registration through Gmail Registration through LinkedIN
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP
FR-3	User login	Confirmation via Registered Gmail Confirmation via Password

3.2 Non-functional Requirements:

Following are the non-functional requirements of the proposed solution.

FR No.	Non-Functional Requirement	Description
NFR-1	**Usability**	The system should provide clear and understandable crowd estimation data to cafeteria authorities.
NFR-2	**Security**	Implement strong encryption protocols to protect data during transmission and storage.
NFR-3	**Reliability**	The computer vision techniques for person detection should have low error rates and adaptability to different environmental conditions.
NFR-4	**Performance**	The system should deliver quick response times, ensuring timely information for decision-making.
NFR-5	**Availability**	The web app should have high uptime, with minimal scheduled maintenance or updates during peak hours.
NFR-6	**Scalability**	The system should be designed to handle a growing number of visitors and accommodate multiple cafeteria locations.

4. PROJECT DESIGN

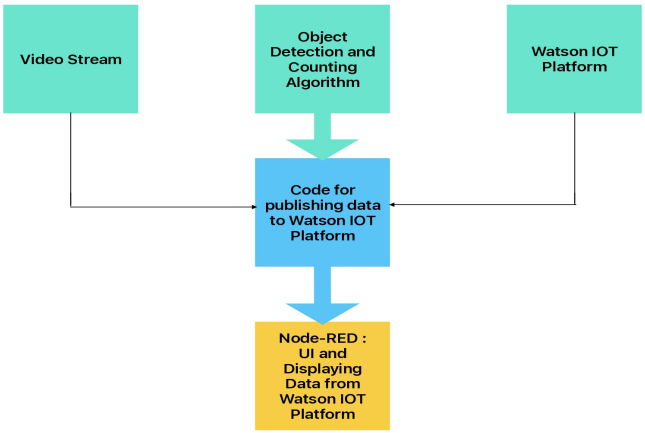
4.1 Data Flow Diagrams

Project Design Phase-II
Data Flow Diagram & User Stories

Date	18 May 2023
Team ID	NM2023TMID17628
Project Name	Go No Queue-Rush Estimator for corporate cafeteria

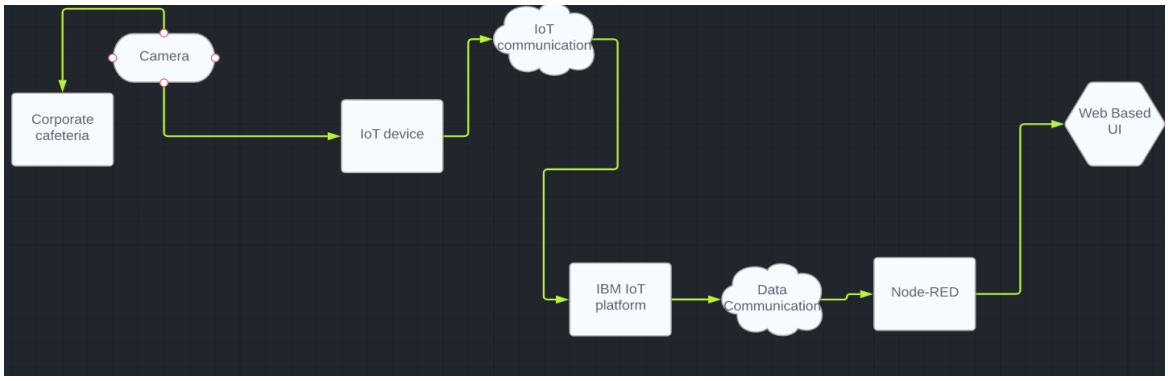
Data Flow Diagrams:

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It shows how data enters and leaves the system, what changes the information, and where data is stored.



4.2 Solution & Technical Architecture

Technical Architecture:



4.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance Criteria	Priority	Team Member
End User	Object Detection	USN-1	As an end user, I want to detect objects in videos	1. The algorithm accurately detects objects in videos.	High	Vignesh
				2. The detection results are displayed on the UI.		
				3. The detection process is efficient and timely.		
Administrator	Real-time Monitoring	USN-2	As an administrator, I want real-time monitoring	1. The system displays the current occupancy of the cafe.	Medium	Uday Kiran
				2. The system provides historical occupancy data.		
Developer	Integration with Watson IoT Platform	USN-3	As a developer, I want to integrate with Watson IoT	1. The Python code successfully publishes data to Watson IoT Platform.	High	Macleish
				2. The data sent to Watson IoT Platform is accurate and reliable.		
				3. The integration is secure and follows best practices.		

5. CODING & SOLUTIONING (Explain the features added in the project along with code)

5.1 Feature 1:

Pre-Ordering and Queue Management:

- Enable employees to pre-order their meals through a mobile app or self-service
- Integration with Food Inventory:

5.2 Feature 2:

Real-Time Notifications:

- Implement a notification system that sends real-time alerts or updates to cafeteria users and staff. These notifications can be triggered based on specific conditions, such as when the cafeteria reaches a certain crowd density threshold or when wait times exceed a predefined limit.

CODE:

- **PersonCount.py:**

```
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 = "bx0msb"
deviceType = "Device3"
```

```
deviceId = "123456"
authMethod = "token"
authToken = "12345678"
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('peopl-------e.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   #
    ############################
    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
    # 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:
        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 Entering cafetaria')
                        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 Exiting cafetaria')
                        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
        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.putText(frame, str(i.getId()),(i.getX(),i.getY()),font,0.3,i.getRGB(),1,cv2.LINE_AA)
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&field1='+str(cnt_
```

```
up))
  # r2 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&field2='+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)
   ibmstart(cnt_up,cnt_down)
   k = cv2.waitKey(30) & 0xff
   if k == 27:
      break
cap.release()
cv2.destroyAllWindows()
```

● **Person.py:**

```
from random import randint
import time


class MyPerson:
    tracks = []
    def _init_(self, i, xi, yi, max_age):
        self.i = i
        self.x = xi
        self.y = yi
        self.tracks = []
        self.R = randint(0,255)
        self.G = randint(0,255)
        self.B = randint(0,255)
        self.done = False
        self.state = '0'
        self.age = 0
        self.max_age = max_age
        self.dir = None
    def getRGB(self):
        return (self.R,self.G,self.B)
    def getTracks(self):
        return self.tracks
    def getId(self):
        return self.i
```

```
def getState(self):
    return self.state
def getDir(self):
    return self.dir
def getX(self):
    return self.x
def getY(self):
    return self.y
def updateCoords(self, xn, yn):
    self.age = 0
    self.tracks.append([self.x,self.y])
    self.x = xn
    self.y = yn
def setDone(self):
    self.done = True
def timedOut(self):
    return self.done
def going_UP(self,mid_start,mid_end):
    if len(self.tracks) >= 2:
        if self.state == '0':
            if self.tracks[-1][1] < mid_end and self.tracks[-2][1] >= mid_end: #cruzo la linea
                state = '1'
                self.dir = 'up'
                return True
        else:
```

```
                return False
            else:
                return False
        def going_DOWN(self,mid_start,mid_end):
            if len(self.tracks) >= 2:
                if self.state == '0':
                    if self.tracks[-1][1] > mid_start and self.tracks[-2][1] <= mid_start: #cruzo la linea
                        state = '1'
                        self.dir = 'down'
                        return True
                else:
                    return False
            else:
                return False
        def age_one(self):
            self.age += 1
            if self.age > self.max_age:
                self.done = True
            return True
    class MultiPerson:
        def _init_(self, persons, xi, yi):
            self.persons = persons
            self.x = xi
            self.y = yi
            self.tracks = []
```

```
self.R = randint(0,255)
self.G = randint(0,255)
self.B = randint(0,255)
self.done = False
```

6. RESULTS

6.1 Performance Metrics:

When evaluating the success and effectiveness of the cafeteria crowd estimation system, several performance metrics can be considered. Here are some key performance metrics that can be used to assess the system's performance:

1. Accuracy of Crowd Estimation: Measure the accuracy of the crowd estimation algorithm by comparing the estimated crowd density with the actual number of people present in the cafeteria. This can be done by conducting periodic manual counts or using a separate validation system for verification.

2. Wait Time Reduction: Track the average wait times before and after implementing the system. Calculate the percentage reduction in wait times to assess the system's impact on improving the overall efficiency of the cafeteria.

3. Customer Satisfaction: Gather feedback from cafeteria users regarding their satisfaction with the system. Use surveys, ratings, or feedback forms to assess the level of satisfaction and identify areas for improvement.

4. Resource Utilization: Evaluate the system's effectiveness in optimizing resource allocation. Measure metrics such as the average number of staff members allocated during different time periods and the utilization rate of food counters. The goal is to ensure that resources are allocated efficiently based on crowd estimations.

5. Real-Time Updates: Monitor the system's ability to provide real-time updates and notifications accurately. Evaluate the frequency and relevance of the notifications sent to users and staff. Assess the responsiveness and effectiveness of the notification system.

6. System Reliability: Measure the system's uptime and reliability. Track any instances of system failures or downtime to assess the overall reliability of the crowd estimation system.

7. Scalability: If the system is implemented across multiple corporate offices or locations, assess its scalability by monitoring its performance in different environments and evaluating its ability to handle increasing numbers of users and data.

8. Cost Efficiency: Evaluate the cost-effectiveness of implementing the system by considering factors such as the initial setup cost, maintenance expenses, and the return on investment (ROI) achieved through improved operational efficiency and customer satisfaction.

7. ADVANTAGES & DISADVANTAGES

7.1 Advantages

- Real-time Occupancy Monitoring
- Data-Driven Decision Making
- Improved User Experience
- Enhanced Security
- Integration with IoT Platform

7.2 Disadvantages

- Initial Setup and Cost
- Privacy Concerns
- Technical Complexity
- Dependency on Internet Connectivity
- User Adoption and Training

8. CONCLUSION:

In conclusion, the cafeteria occupancy monitoring system utilizing video processing and IoT technology provides real-time insights into occupancy levels, enabling efficient resource management. Despite initial setup and cost requirements, the system enhances security measures and improves user experience. Implementing this system offers effective cafeteria management and a better overall experience for staff and users.

9.FUTURE SCOPE:

1. Integration with Machine Learning Algorithms: The system can be enhanced by incorporating machine learning algorithms to improve object detection and counting accuracy, enabling more advanced analytics and insights.

2. Mobile Application Development: Developing a dedicated mobile application can provide users with real-time occupancy updates, personalized notifications, and seamless access to cafeteria services, enhancing convenience and user experience.

3. Data-driven Decision Making: Leveraging the collected data, future enhancements can include predictive analytics and data-driven decision making, enabling proactive resource allocation and optimization based on historical trends and patterns.

4. Integration with Smart Building Systems: Integrating the cafeteria occupancy system with smart building systems, such as lighting and HVACsustainability benefits.

10. APPENDIX

Source Code :

Personcount.py

```
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 = "93ux2y"
deviceType = "Device1"
deviceId = "12345"
authMethod = "token"
authToken = "12345678"
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&field1=
'+str(cnt_up))
   # r2 =
requests.get('https://api.thingspeak.com/update?api_key=4BGMGGBRLQM3VRHO&field2=
'+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()
```

Node-Red jsn :


```
[
    {
        "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
    }
]
```


GitHub & Project Video Demo Link

Github Link:https://github.com/naanmudhalvan-SI/PBL-NT-GP--2691-1680616670/tree/main

Project Video Link :https://youtu.be/1u4m_Xmh2kM