PROJECT REPORT

1.INTRODUCTION

1.1 Project Overview

Swimming is one of the best exercises that helps people to reduce stress in this urban lifestyle. Swimming pools are found larger in number in hotels, and weekend tourist spots and barely people have them in their house backyard. Beginners, especially, often feel it difficult to breathe underwater which causes breathing trouble which in turn causes a drowning accident. Worldwide, drowning produces a higher rate of mortality without causing injury to children. Children under six of their age are found to be suffering the highest drowning mortality rates worldwide. Such kinds of deaths account for the third cause of unplanned death globally, with about 1.2 million cases yearly. To overcome this conflict, a meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analyzing the video feeds to detect any anomalies. but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher then an alert will be generated to attract lifeguards' attention.

1.2 Purpose

A meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analyzing the video feeds to detect any anomalies. but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher then an alert will be generated to attract lifeguards' attention.

2.LITERATURE SURVEY

2.1 Existing Problem

Swimming pools are found larger in number in hotels, and weekend tourist spots and barely people have them in their house backyard. Beginners, especially, often feel it difficult to breathe underwater which causes breathing trouble which in turn causes a drowning accident. Worldwide, drowning produces a higher rate of mortality without causing injury to children. Children under six of their age are found to be suffering the highest drowning mortality rates worldwide. Such kinds of deaths account for the third cause of unplanned death globally, with about 1.2 million cases yearly. To overcome this conflict, a meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and

ceiling and analyzing the video feeds to detect any anomalies. but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher then an alert will be generated to attract lifeguards' attention.

2.2 References

Abdel Ilah N. Alshbatat, Shamma Alhameli, Shamsa Almazrouei, Salama Alhameli, Wadhha Almarar

U. Handalage, N. Nikapotha, C. Subasinghe, T. Prasanga, T. Thilakarthna and D. Kasthurirathna

Anjana Unnikrishnan, Roshni A T, Anusha P R, Anju M Vinny, Anuraj C K

Jia-Xian Jian, Chuin-Mu Wang

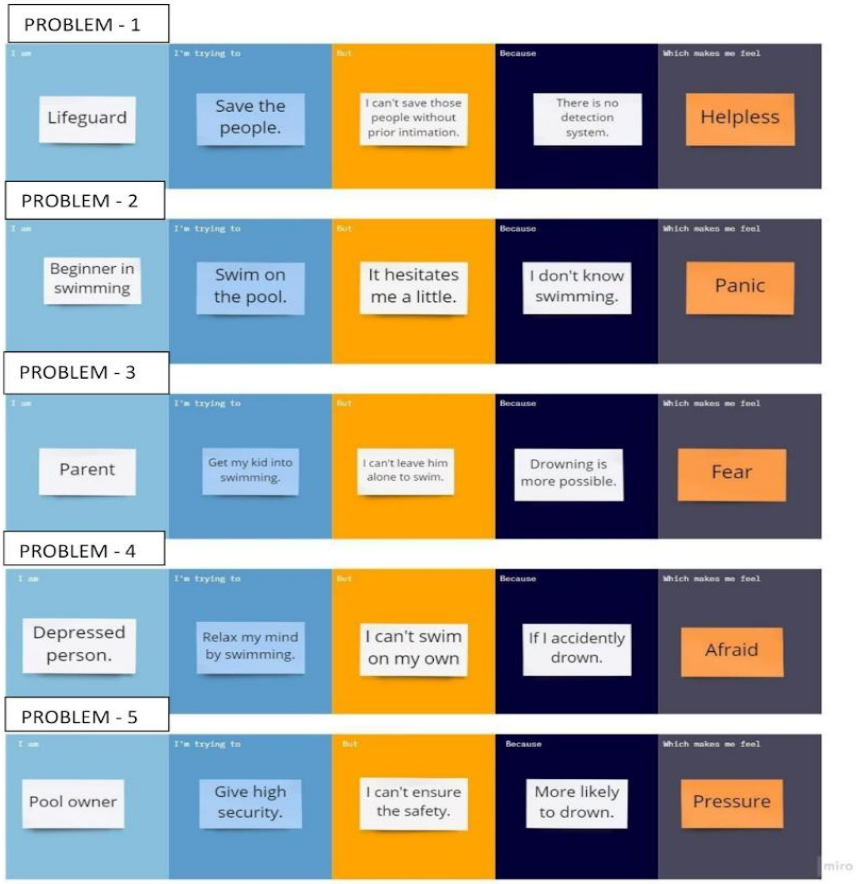
Pavithra P, Nandini S,Nanthana A,Noor Tabreen Aslam, Praveen Kumar P

2.3 Problem Statement Definition

Ideation Phase
Define the Problem Statements

Date	26 September 2022
Team ID	PNT2022TMID28949
Project Name	VirtualEye - Life Guard for Swimming Pools to Detect Active Drowning
Maximum Marks	2 Marks

Problem statements:



3.IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas

An empathy map is a collaborative tool teams can use to gain a deeper insight into their customers. Much like a user persona, an empathy map can represent a group of users, such as a customer segment. The empathy map was originally created by Dave Gray and has gained much popularity within the agile community. An Empathy Map consists of four quadrants. The four quadrants reflect four key traits, which the user demonstrated/possessed during the observation/research stage. The four quadrants refer to what the user: Said, Did, Thought, and Felt.

3.2 Ideation and Brainstorming

Ideation is the process where you generate ideas and solutions through sessions such as Sketching, Prototyping, Brainstorming, Brainwriting, Worst Possible Idea, and a wealth of other ideation techniques. Ideation is also the third stage in the Design Thinking process

Brainstorming is a method of generating ideas and sharing knowledge to solve a particular commercial or technical problem, in which participants are encouraged to think without interruption. Brainstorming is a group activity where each participant shares their ideas as soon as they come to mind.

3.3 Proposed Solution

Your proposed solution should relate the current situation to a desired result and describe the benefits that will accrue when the desired result is achieved. So, begin your proposed solution by briefly describing this desired result.

Project Design Phase-I
Proposed Solution Template

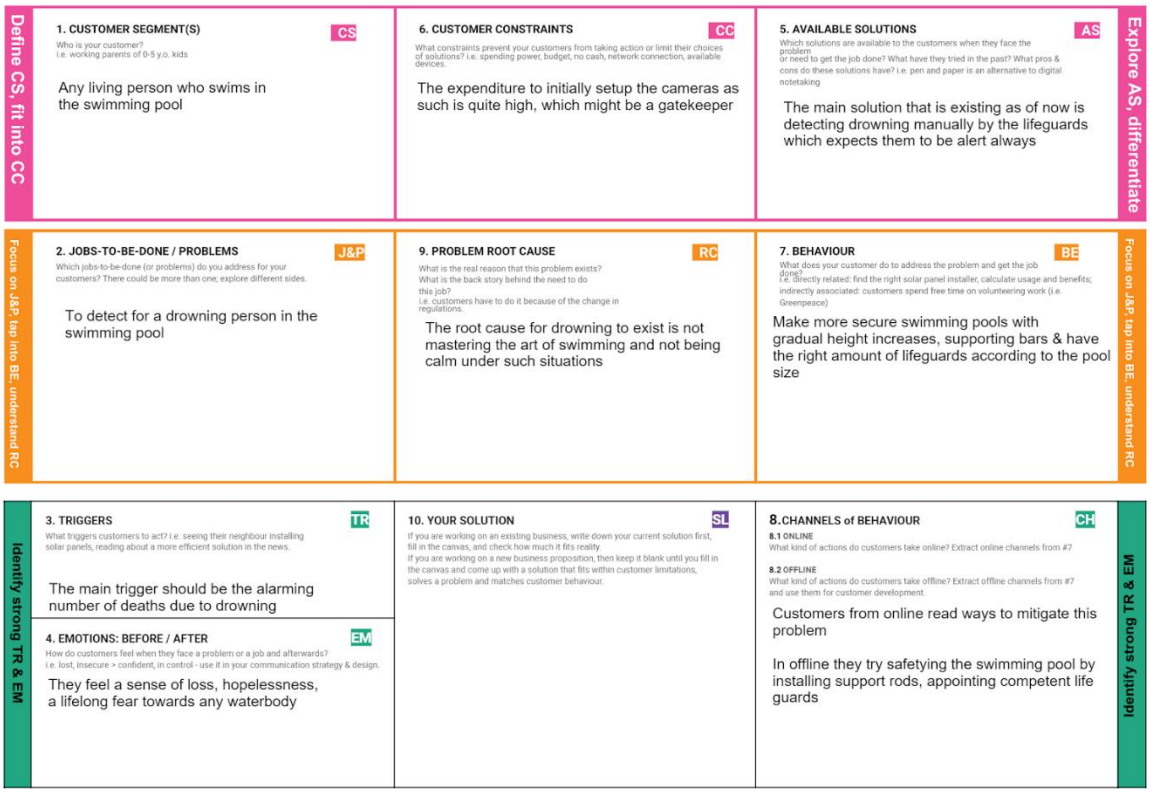
Date	10 OCTOBER 2022
Team ID	PNT2022TMID28949
Project Name	VirtualEye - Life Guard For Swimming Pools To Detect Active Drowning
Maximum Marks	2 Marks

Proposed Solution:

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Swimming pools are generally places of fun and healthy exercise, but they can be deadly as well. Even with a lifeguard observer on duty, swimmers may still have trouble in underwater or in parts of the pool beyond the lifeguard's field of view.
2.	Idea / Solution description	In this project, we use Artificial Intelligence. We install the cameras in underwater to detect the drowning people. Using deep learning, image can be recognized. If the image is detected, it triggers the alarm to alert the Life Guard who rescue the drowning peoples.
3.	Novelty / Uniqueness	The uniqueness of our system software to track the position and the location of a drowning person. We use YOLO Algorithm. Because of its high accuracy and fast detection speed. So it helps lifeguard to save people within seconds.
4.	Social Impact / Customer Satisfaction	Drowning globally has a higher death rate and is also the third leading cause of unexpected deaths worldwide, especially among children under the age of six. To overcome this conflict our drowning detection system will have an impact on society.
5.	Business Model (Revenue Model)	We can introduce the software-based approach for making a good income. It is extremely useful to lifeguards, swimmers and business operators. The number of features makes it attractive for end users to use our software system.
6.	Scalability of the Solution	Our software system can be used by the company driver who manages the pools. We use the IBM cloud server to collect and maintain the data. We will ensure the safety of the swimmers.

3.4 Problem Solution Fit

Problem-solution fit is a term used to describe the point validating that the base problem resulting in a business idea really exists and the proposed solution actually solves that problem.



4.REQUIREMENT ANALYSIS

4.1 Functional Requirement

Functional requirements are product features or functions that developers must implement to enable users to accomplish their tasks. So, it's important to make them clear both for the development team and the stakeholders. Generally, functional requirements describe system behavior under specific conditions.

4.2 Non-Functional Requirements

Non functional requirements are requirements that define 'how' the app must perform a certain function. In essence, they are the quality attributes of an app that define the user experience of the app. They are also known as non-behavioral requirements and are to be implemented according to their priority to the app function. This makes them flexible to an extent, making it possible to skip a few in case of time, budget or technology constraints.

5.PROJECT DESIGN

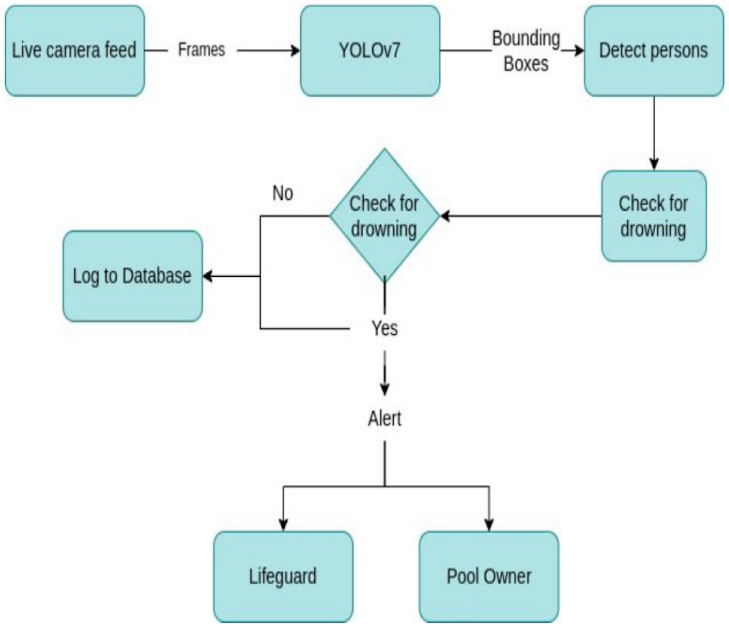
5.1 Data Flow Diagram

Project Design Phase-II
Data Flow Diagram & User Stories

Date	15 October 2022
Team ID	PNT2022TMID06492
Project Name	VirtualEye - LifeGuard for Swimming Pools to Detect Active Drowning
Maximum Marks	4 Marks

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.



5.2 Solution and Technical Architecture

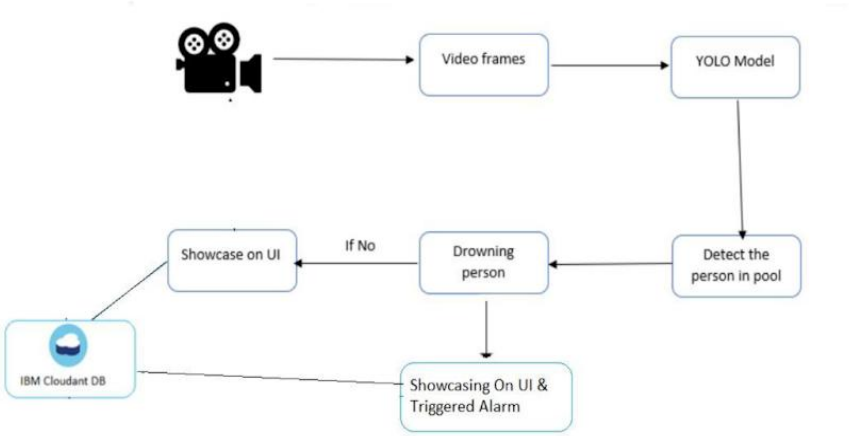
Project Design Phase-I
Solution Architecture

Date	16 October 2022
Team ID	PNT2022TMID28949
Project Name	Virtual Eye - Life Guard for Swimming Pools to Detect Active Drowning
Maximum Marks	4 Marks

Solution Architecture:

- By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning.
- Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analyzing the video feeds to detect any anomalies.
- but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher then an alert will be generated to attract lifeguards' attention.

Solution Architecture Diagram:



5.3 User Stories

User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Pool owner)	Installation	USN-1	As a pool owner, I can install the cameras and set up the drowning detection system	I can connect the cameras to the cloud-hosted software	High	Sprint-1
	Detecting the drowning persons	USN-2	As a user, I can find the drowning persons by using the drowning detection system	I would receive an alert if a person is drowning	High	Sprint-1
	Notify the lifeguard	USN-3	As a user, I can notify the lifeguard when the system detects a drowning person	I can set up an alarm that would notify the lifeguard	High	Sprint-2
Customer (Lifeguard)	Rescue people	USN-4	As a user, I can rescue the drowning persons from the pool	I can save the drowning person	High	Sprint-2
Customer (Swimmers)	Safety	USN-5	As a user, I can swim without the fear of drowning	I can swim safely with the help of the system and the lifeguard	Medium	Sprint-2
Customer Care Executive	Contact	USN-6	resolve technical issues	I can contact the customer care executive to resolve any issues	Medium	Sprint-3
Adminitsrator	Dashboard	USN-7	Management of the drowning detection system and database management.	I can access the system's logs and any other data instantly	High	Sprint-4

6.PROJECT PLANNING AND SCHEDULING

6.1Sprint Planning and Estimation

Login.html

```
<!DOCTYPE html>
<html >

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <title>Virtual Eye</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo'
rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300'
rel='stylesheet' type='text/css'>
<link
href='https://fonts.googleapis.com/css?family=Open+Sans+Conde
nsed:300' rel='stylesheet' type='text/css'>
<!link rel="stylesheet" href="{{ url_for('static',
filename='css/style.css') }}">

<link
href='https://fonts.googleapis.com/css?family=Merriweather'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Josefin Sans'
rel='stylesheet'>
<link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>


<style>
.header {
                    top:0;
                    margin:0px;
                    left: 0px;
                    right: 0px;
                    position: fixed;
                    background-color: #28272c;
                    color: white;
                    box-shadow: 0px 8px 4px grey;
                    overflow: hidden;

                            padding-left:20px;
                            font-family: 'Josefin Sans';
                            font-size: 2vw;
                            width: 100%;
                            height:8%;
                            text-align: center;
                     }
    .topnav {
      overflow: hidden;
      background-color: #333;
    }

    .topnav-right a {
```

```
  float: left;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;
  font-size: 18px;
}

.topnav-right a:hover {
  background-color: #ddd;
  color: black;
}

.topnav-right a.active {
  background-color: #565961;
  color: white;
}

.topnav-right {
  float: right;
  padding-right:100px;
```

```
.login{
margin-top:-70px;
}
body {

 background-color:#ffffff;
 background-repeat: no-repeat;
 background-size:cover;
 background-position: 0px 0px;
 }
```

```
.login{
        margin-top:100px;
}
form {border: 3px solid #f1f1f1; margin-left:400px;margin-
right:400px;}

input[type=text],
input[type=email],input[type=number],input[type=password] {
width: 100%;
  padding: 12px 20px;
  display: inline-block;
  margin-bottom:18px;
  border: 1px solid #ccc;
  box-sizing: border-box;
}

button {
  background-color: #28272c;
  color: white;
  padding: 14px 20px;
  margin-bottom:8px;
  border: none;
```

```
 cursor: pointer;
 width: 100%;
 font-weight:bold;
}

button:hover {
 opacity: 0.8;
}

.cancelbtn {
 width: auto;

  padding: 10px 18px;
  background-color: #f44336;
}

.imgcontainer {
 text-align: center;
 margin: 24px 0 12px 0;
}

img.avatar {
 width: 30%;
 border-radius: 50%;
}

.container {
 padding: 16px;
}

span.psw {
 float: right;


   padding-top: 16px;

 }

 /* Change styles for span and cancel button on extra small screens
 */
 @media screen and (max-width: 300px) {
  span.psw {
     display: block;
     float: none;
  }
  .cancelbtn {
     width: 100%;
```

```
</style>
</head>

<body style="font-family:Montserrat;">

<div class="header">
 <div style="width:50%;float:left;font-size:2vw;text-
align:left;color:white; padding-top:1%">Virtual Eye</div>
  <div class="topnav-right" style="padding-top:0.5%;">

   <a  href="{{ url_for('index')}}">Home</a>
   <a class="active" href="{{ url_for('login')}}">Login</a>
   <a  href="{{ url_for('register')}}">Register</a>

  </div>
</div>
<div id="login" class="login">


       <form action="{{url_for('afterlogin')}}" method="post">
              <div class="imgcontainer">
                    <img style=""

 src="https://cdn.digitalhealth.net/wp-
 content/uploads/2017/03/eye_image_generic_555.jpg"
 alt="Avatar" class="avatar">
              </div>

              <div class="container">
                    <input type="email" placeholder="Enter
 registered email ID" name="_id" required><br>


                        <input type="password" placeholder="Enter
   Password" name="psw" required>

                        <button type="submit">Login</button><br>
      {{pred}}
                  </div>
          </form>

   </div>


   </body>
   </html>
```

6.2Sprint Delivery Schedule

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	6	6 Days	24 Oct 2022	29 Oct 2022	6	29 Oct 2022
Sprint-2	3	6 Days	31 Oct 2022	05 Nov 2022	3	05 Nov 2022
Sprint-3	4	6 Days	07 Nov 2022	12 Nov 2022	4	12 Nov 2022
Sprint-4	3	6 Days	14 Nov 2022	19 Nov 2022	3	19 Nov 2022

Velocity:

The Team's average velocity (AV) per iteration unit (story points per day)

$$AV=Velocity/Sprint\ Duration = 4/6$$

Burndown Chart:

A burn down chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

6.3 Reports From JIRA

Jira helps teams plan, assign, track, report, and manage work and brings teams together for everything from agile software development and customer support to start-ups and enterprises.

7.CODING AND SOLUTIONING(Explain the features added in the project along with code)

A meticulous system is to be implemented along the swimming pools to save human life. By studying body movement patterns and connecting cameras to artificial intelligence (AI) systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing more than 16 cameras underwater and ceiling and analysing the video feeds to detect any anomalies. but AS a POC we make use of one camera that streams the video underwater and analyses the position of swimmers to assess the probability of drowning, if it is higher, then an alert will be generated to attract lifeguards' attention.

```
#import necessary packages
import cv2
import os
import numpy as np
from .utils import download_file

initialize = True
net = None
dest_dir = os.path.expanduser('~') + os.path.sep + '.cvlib' + os.path.sep + 'object_detection' +
os.path.sep + 'yolo' + os.path.sep + 'yolov3'
classes = None
#colors are BGR instead of RGB in python
COLORS = [0,0,255], [255,0,0]

def populate_class_labels():

    #we are using a pre existent classifier which is more reliable and more efficient than one
    #we could make using only a laptop
    #The classifier should be downloaded automatically when you run this script
    class_file_name = 'yolov3_classes.txt'
    class_file_abs_path = dest_dir + os.path.sep + class_file_name
    url = 'https://github.com/Nico31415/Drowning-Detector/raw/master/yolov3.txt'
    if not os.path.exists(class_file_abs_path):
        download_file(url=url, file_name=class_file_name, dest_dir=dest_dir)
    f = open(class_file_abs_path, 'r')
    classes = [line.strip() for line in f.readlines()]
```

```
    #the number of output layers in a neural network is the number of possible
    #things the network can detect, such as a person, a dog, a tie, a phone...
    layer_names = net.getLayerNames()

    output_layers = [layer_names[i[0] - 1] for i in net.getUnconnectedOutLayers()]

    return output_layers


def draw_bbox(img, bbox, labels, confidence, Drowning, write_conf=False):

    global COLORS
    global classes

    if classes is None:
        classes = populate_class_labels()

    for i, label in enumerate(labels):

        #if the person is drowning, the box will be drawn red instead of blue
        if label == 'person' and Drowning:
            color = COLORS[0]
            label = 'DROWNING'
        else:
            color = COLORS[1]

        if write_conf:
            label += ' ' + str(format(confidence[i] * 100, '.2f')) + '%'
```

```
        cv2.rectangle(img, (bbox[i][0],bbox[i][1]), (bbox[i][2],bbox[i][3]), color, 2)

        cv2.putText(img, label, (bbox[i][0],bbox[i][1]-10), cv2.FONT_HERSHEY_SIMPLEX, 0.5, color, 2)

    return img

def detect_common_objects(image, confidence=0.5, nms_thresh=0.3):

    Height, Width = image.shape[:2]
    scale = 0.00392

    global classes
    global dest_dir

    #all the weights and the neural network algorithm are already preconfigured
    #as we are using YOLO

    #this part of the script just downloads the YOLO files
    config_file_name = 'yolov3.cfg'
    config_file_abs_path = dest_dir + os.path.sep + config_file_name

    weights_file_name = 'yolov3.weights'
    weights_file_abs_path = dest_dir + os.path.sep + weights_file_name

    url = 'https://github.com/Nico31415/Drowning-Detector/raw/master/yolov3.cfg'

    if not os.path.exists(config_file_abs_path):
        download_file(url=url, file_name=config_file_name, dest_dir=dest_dir)
```

```
url = 'https://pjreddie.com/media/files/yolov3.weights'

if not os.path.exists(weights_file_abs_path):
    download_file(url=url, file_name=weights_file_name, dest_dir=dest_dir)



global initialize
global net

if initialize:
    classes = populate_class_labels()
    net = cv2.dnn.readNet(weights_file_abs_path, config_file_abs_path)
    initialize = False

blob = cv2.dnn.blobFromImage(image, scale, (416,416), (0,0,0), True, crop=False)

net.setInput(blob)

outs = net.forward(get_output_layers(net))

class_ids = []
confidences = []
boxes = []

for out in outs:
    for detection in out:
        scores = detection[5:]
```

```
            class_id = np.argmax(scores)
            max_conf = scores[class_id]
            if max_conf > confidence:
```

```
                center_x = int(detection[0] * Width)
                center_y = int(detection[1] * Height)
                w = int(detection[2] * Width)
                h = int(detection[3] * Height)
                x = center_x - w / 2
                y = center_y - h / 2
                class_ids.append(class_id)
                confidences.append(float(max_conf))
                boxes.append([x, y, w, h])


        indices = cv2.dnn.NMSBoxes(boxes, confidences, confidence, nms_thresh)

    bbox = []
    label = []
    conf = []

    for i in indices:
        i = i[0]
        box = boxes[i]
        x = box[0]
        y = box[1]
        w = box[2]
        h = box[3]
        bbox.append([round(x), round(y), round(x+w), round(y+h)])
        label.append(str(classes[class_ids[i]]))
        conf.append(confidences[i])

    return bbox, label, conf
```

8.TESTING

8.1 Test cases

```
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
#for PiCamera
#from picamera Import PiCamera
#camera = PiCamera
#camera.start_preview()
# open webcam
webcam = cv2.VideoCapture(0)

if not webcam.isOpened():
    print("Could not open webcam")
    exit()



t0 = time.time() #gives time in seconds after 1970

#variable dcount stands for how many seconds the person has been
standing still for
centre0 = np.zeros(2)
isDrowning = False

#this loop happens approximately every 1 second, so if a person
doesn't move,
#or moves very little for 10seconds, we can say they are drowning

#loop through frames
while webcam.isOpened():

    # read frame from webcam
    status, frame = webcam.read()

    if not status:
        print("Could not read frame")
        exit()
```

```
    # apply object detection
    bbox, label, conf = cv.detect_common_objects(frame)
    #simplifying for only 1 person

    #s = (len(bbox), 2)

    if(len(bbox)>0):
            bbox0 = bbox[0]
            #centre = np.zeros(s)
```

```
            centre = [0,0]


            #for i in range(0, len(bbox)):
                #centre[i] =[(bbox[i][0]+bbox[i][2])/2,(bbox[i]
[1]+bbox[i][3])/2 ]

            centre =[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2 ]

            #make vertical and horizontal movement variables
            hmov = abs(centre[0]-centre0[0])
            vmov = abs(centre[1]-centre0[1])

            #there is still need to tweek the threshold
            #this threshold is for checking how much the centre has
moved

            x=time.time()

            threshold = 10
            if(hmov>threshold or vmov>threshold):
```

```
            if(hmov>threshold or vmov>threshold):
                print(x-t0, 's')
                t0 = time.time()
                isDrowning = False

            else:

                print(x-t0, 's')
                if((time.time() - t0) > 10):
                    isDrowning = True



            #print('bounding box: ', bbox, 'label: '
label ,'confidence: ' conf[0], 'centre: ', centre)
            #print(bbox,label ,conf, centre)
            print('bbox: ', bbox, 'centre:', centre, 'centre0:',
centre0)
            print('Is he drowning: ', isDrowning)

            centre0 = centre
            # draw bounding box over detected objects

    out = draw_bbox(frame, bbox, label, conf,isDrowning)

    #print('Seconds since last epoch: ', time.time()-t0)
```

```
    # display output
    cv2.imshow("Real-time object detection", out)
```

8.2 User Acceptance Testing

```
[net]
# Testing
# batch=1
# subdivisions=1
# Training
batch=64
subdivisions=16
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=.1

learning_rate=0.01
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

# Downsample

[convolutional]
batch_normalize=1
filters=64
size=3
stride=2
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear
```

```
[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
```

```
stride=1
pad=1
activation=leaky
```

```
[shortcut]
from=-3
activation=linear

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[shortcut]
from=-3
activation=linear


[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky
```

9.RESULTS

9.1Performance Metrices

				09-Nov-22 PNT2022TMID28949 Virtuale eye-Lifeguard for swimming 4 marks			
Test case ID	Feature Type		Test Scenario	Steps TO Execute	Test	Expected Result	Actual Result
LoginPage_TC_001	Functional	Home Page	Verify user is able to see the Login/Signup popup when user clicked on My account button	1.Enter URL and click go 2.Click on My Account dropdown button 3.Verify login/Singup popup displayed or not	Login.html	Login/Signup popup should display	Working as
LoginPage_TC_002		Home Page	Verify the UI elements in Login/Signup popup	1.Enter URL and click go 2.Click on My Account dropdown 3.Verify login/Singup popup with below UI elements: a.email text box b.password text box c. LYoin button d.New customer? Create account link e. Last password? Recovery password link	Login.html	Application should show below elements: a.email text box b.password text box c.Login button with orange colour d. New customer? Create account link e.Last password? Recovery password link	Working as expected
LoginPage_TC_003	Functional	Home page	Verify user is able to log into application with Valid credentials	1.Enter URL and click go 2.Click on My Account dropdown 3.Enter Valid username/email in Email text 4.Enter valid password in password text box 5. Click On in button	Username:lax@gmail password : lax26	User should navigate to prediction homepage	working as
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with invalid credentials	1, Enter URL and click go 2.Click on My Account dropdown button 3.Enter Invalid username/email in Email text box 4.Enter valid password in password text box 5.Click on •n button	Username:lax password:lax26	Application should show "Incorrect email or password " validation message.	working as
LoginPage_TC_004	Functional	Login page	Verify user is able to log into application with invalid credentials	I-Enter URL and click go 2.Click On My Account dropdown 3.Enter Valid username/email in Email text box 4.Enter Invalid password in password text box 5.Click on in button	username:lax26@mail password:lax26	Application should show •Incorrect email or password " validation message.	working as
LoginPage_TC_005	Functional	Login page	Verify user is able to into application with inValid credentials	1.Enter URL and click go 2.Click on My Account dropdown 3.Enter Invalid username/email in Email text box 4. Enter Invalid password in password text box 5. Click on I in button	username:lax26@mail password:1803	Application should show "Incorrect email or password " validation message.	working as
Predictionpage_TC_006	Functional	Prediction Page	Page should display whether the person is drowning or not	1, Camera should take pictures of people swimming in pools 2. It should predict the probability of drowning 3. It should show a bounding box displaying the probability Of drowning	Image Of people drowning	generate a alert to lifeguard if people are drowning	Working as

10.ADVANTAGES AND DISADVANTAGES

ADVANTAGES

- Safety first. Active drowning detection.
- Position and image of the drowning. When it comes to swimmers in trouble, every second counts.
- Recording of events.
- An additional level of security.

DISADVANTAGES

Designed for whom has to guarantee every day the safety in public and intensive use pools, this life guard detects potential drownings and promptly notifies you. It features the latest artificial intelligence technology and adapts to the needs of the user. It's the ultimate drowning detection system for those who demand the ultimate in safety.

11.CONCLUSION

Consistently numerous people, including kids, are suffocated or near suffocating in the deeps of the swimming pools, and the lifeguards are not prepared all around to deal with these issues. In this manner raises the necessities for having a framework that will thus recognize the suffocating people and alert the lifeguards at such hazard. It can be installed in International standardized schools where classes are held for training kids.

12.FUTURE SCOPE

1) Test results show that the mean precision rate of drowning is 94.62%, the mean false rate is 1.43%, and the mean missing rate is 3.57%.

2) The mean precision rate of swimming is 97.86%, the mean false rate is 7.93%, the mean missing rate is 5.93%, and the average frame rate is 33f/s.

3) This study will attempt to identify superiority in trained lifeguards through the use of videoed pool swimming scenarios that vary in set size.

13.APPENDIX

Source code

```
import re import numpy
as np import os

from flask import Flask, app,request,render_template from
tensorflow.keras import models from tensorflow.keras.models import
load_model from tensorflow.keras.preprocessing import image from
tensorflow.python.ops.gen_array_ops import concat

from tensorflow.keras.applications.inception_v3 import preprocess_input import cvlib as cv

from cvlib.object_detection import draw_bbox import cv2
import time import numpy as np from playsound import
playsound import requests

from flask import Flask, request, render_template, redirect, url_for

#Loading the model

 from cloudant.client import Cloudant

 # Authenticate using an IAM API key

client = Cloudant.iam('apikey-v2-2qq2z28xgpj28weetcerewjn5cpm2ctxr5g9a6q9lai7',

'TOicEd3GsLC3SNnedbCK6zhbfApga3965ChxL0VH2hHc', connect =True)

 # Create a database using an initialized client

my_database = client.create_database('my_database')app=Flask(__name__)

 #default home page or route

@app.route('/') def index():

   return render_template('index.html')
```

```
@app.route('/index.html') def home():

   return render_template("index.html")

 #registration page @app.route('/register') def register():

   return render_template('register.html')

 @app.route('/afterreg', methods=['POST']) def afterreg():

   x = [x for x in request.form.values()]     print(x)     data = { '_id': x[1],

# Setting _id is optional

'name': x[0],

    'psw':x[2]     }     pri
nt(data)

        query = {'_id': {'$eq': data['_id']}}

    docs = my_database.get_query_result(query)     print(docs)

    print(len(docs.all()))

       if(len(docs.all())==0):

      url = my_database.create_document(data)        #response = requests.get(url)

          return  render_template('register.html',  pred="Registration  Successful,  please  login
using your details")     else:

        return render_template('register.html', pred="You are already a member, please login using your
details")

 #login page @app.route('/login') def login():

   return render_template('login.html')

 @app.route('/afterlogin',methods=['POST']) def afterlogin():

        user   =   request.form['_id']           passw   =
request.form['psw']     print(user,passw)

      query = {'_id': {'$eq': user}}

    docs = my_database.get_query_result(query)     print(docs)

    print(len(docs.all()))

          if(len(docs.all())==0):

      return render_template('login.html', pred="The username is not found.")     else:

      if((user==docs[0][0]['_id'] and passw==docs[0][0]['psw'])):

         return redirect(url_for('prediction'))        else:

         print('Invalid User')
```

```
@app.route('/logout') def logout():
  return render_template('logout.html')
@app.route('/prediction') def prediction():
  return render_template('prediction.html')
@app.route('/result',methods=["GET","POST"]) def res():
  webcam = cv2.VideoCapture('drowning.mp4')
   if not webcam.isOpened():
     print("Could not open webcam")       exit()
     t0 = time.time() #gives time in seconds after 1970
#variable dcount stands for how many seconds the person has been standing still for
centre0 = np.zeros(2)    isDrowning = False
   #this loop happens approximately every 1 second, so if a person doesn't move,
  #or moves very little for 10seconds, we can say they are drowning
  #loop through frames
while webcam.isOpened():
     # read frame from webcam
  status, frame = webcam.read()
      if not status:
        print("Could not read frame")          exit()
     # apply object detection
     bbox, label, conf = cv.detect_common_objects(frame)
     #simplifying for only 1 person
#s = (len(bbox), 2)       if(len(bbox)>0):          bbox0 = bbox[0]
        #centre = np.zeros(s)          centre = [0,0]
        #for i in range(0, len(bbox)):
          #centre[i]=[(bbox[i][0]+bbox[i][2])/2,(bbox[i][1]+bbox[i][3])/2 ]
 centre =[(bbox0[0]+bbox0[2])/2,(bbox0[1]+bbox0[3])/2 ]
   #make vertical and horizontal movement variables
 hmov = abs(centre[0]-centre0[0])          vmov = abs(centre[1]-centre0[1])
  #there is still need to tweek the threshold
```

```
        #this threshold is for checking how much the centre has moved

 x=time.time()

          threshold = 10          if(hmov>threshold or vmov>threshold):

                    print(x-t0, 's')                    t0 =
time.time()              isDrowning = True

                      else:

              print(x-t0, 's')              if((time.time() - t0) > 10):

                isDrowning = True

 #print('bounding box: ', bbox, 'label: ' label ,'confidence: ' conf[0], 'centre: ', centre)

          #print(bbox,label ,conf, centre)

               print('bbox: ', bbox, 'centre:', centre, 'centre0:', centre0)                print('Is he drowning: ',
isDrowning)

   centre0 = centre

         # draw bounding box over detected objects

  out = draw_bbox(frame, bbox, label, conf,isDrowning)

 #print('Seconds since last epoch: ', time.time()-t0)

  # display output

        cv2.imshow("Real-time object detection", out)          if cv2.waitKey(1) &
0xFF == ord('q'):

          break

                                    playsound(r'C:\Users\indum\Downloads\emergency-alarm-with-reverb-
29431.mp3')        webcam.release()

     cv2.destroyAllWindows()        return render_template('prediction.html',prediction="Emergency !!!
The Person is drowining")

          #return render_template('base.html')

   # press "Q" to stop

  # release resources

    #webcam.release()

    #cv2.destroyAllWindows()

    #return render_template('prediction.html',)

 """ Running our application """ if __name__ == "__main__":     app.run(debug=True)
```

-

Github and project demo link
https://github.com/IBM-EPBL/IBM-Project-12398-1659450167