PROJECT REPORT

VIRTUAL EYE – LIFE GUARD FOR SWIMMING POOLS TO DETECT ACTIVE DROWNING

Team ID : PNT2022TMID09909

Team Members:

1. DANISH L (720719104041) [TEAM LEADER]
2. AFFRAN S M (720719104005)
3. ASHISH SURESH (720719104028)
4. DHANISH N ERSHAD (72071904045)

TABLE OF CONTENTS :

INTRODUCTION

Drowning incidents are potentially severe but thankfully rare for most lifeguards. Due to the infrequency of drowning incidents, the visual search for such occurrences is challenging (Lanagan-Leitzel, Skow & Moore, 2015). The difficulties involved in detecting infrequent drowning targets are reflected in other areas of real-world visual search with uncommon target items, such as airport security screenings (Wolfe, Horowitz & Kenner, 2005; Biggs & Mitroff, 2015). For example, Wolfe et al., (2005) found low-prevalence targets (occurring on 1% of trials) were missed more frequently than high-prevalence targets (occurring on 50% of trials), with error rates of 30% and 7%, respectively.

In regards to lifeguarding, visual search has been defined as observing part of an aquatic environment (beaches, pools, open water), and processing and assessing the events happening within that location (Fenner et al., 1999). While this definition suggests that the surveillance of the water is a fundamental and critical role of the lifeguard, there is relatively little focus on training in these areas (Lanagan-Leitzel & Moore, 2010). This is reflected in the UK National Pool Lifeguard Qualification (NPLQ) training manual (Blackwell, 2016), where only 6 out of 214 pages are dedicated to the education of scanning and observation behaviours (Blackwell et al., 2012). With this limited focus on visual training, lifeguards may be underprepared for detecting struggling swimmers in a timely manner.

PROJECT OVERVIEW:-

Lifeguard surveillance is a complex task that is crucial for swimmer safety, though few studies of applied visual search have investigated this domain. This current study compared lifeguard and non-lifeguard search skills using dynamic, naturalistic stimuli (video clips of confederate swimmers) that varied in set size and type of drowning. Lifeguards were more accurate and responded faster to drowning targets. Differences between drowning targets were also found: passive drownings were responded to less often, but more quickly than active drownings, highlighting that passive drownings may be less salient but are highly informative once detected. Set size effects revealed a dip in reaction speeds at an intermediate set-size level, suggesting a possible change in visual search strategies as the array increases in size. Nonetheless, the ability of the test to discriminate between lifeguards and non-lifeguards offers future possibilities for training and assessing lifeguard surveillance skills.

PURPOSE:-

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.

LITERARTURE SURVEY

➢ EXISTING PROBLEM:-

Paper 1 :

NAME : Aquatic competencies and drowning prevention in children 2-4 years

AUTHOR : Danielle H.Taylor ,Richard C.Franklin ,Amy E.Peden

MERITS : Findings suggest swimming ability may reduce risk of an incident (both fatal and non-fatal); however, the study was underpowered to validate this result. Risk reduction shows a positive correlation with increasing age.

CONCLUSION : Young children are not adults in miniature and supervision of children is required; however, developing aquatic competencies was not found to increase the risk of drowning.

Paper 2 :

NAME : A Novel drowning detection method for safety of swimmers

AUTHOR: A Novel drowning detection method for safety of swimmers

MERIT : limited time is given for the child or adult after the threshold time , even though it is same triggers the alarm.

CONCLUSION : reliable solution where the life guards have difficulty in monitoring the swimmers like a highly crowded sea.

PAPER 3 :

NAME : Drowning detection system using CNN

AUTHOR : valvi Priyanka , Prof. mr.Amar Palwankar

MERITS : if the swimmer faces any problem it will detect it by its motion and gives an alert.

CONCLUSION : proves it can be a reliable multimedia video-based surveillance system.

PAPER 4 :

NAME : DEWS: A Live Visual Surveillance System for Early Drowning Detection at Pool

AUTHOR : How - Lung ,Kar Ann toh , Junxian wang

MERITS : Examples of interesting behaviors, i.e., distress, drowning, treading and numerous swimming styles, are simulated and collected.

CONCLUSION : Experimental results show that we have established a prototype system which is robust and beyond the stage of proof-of-concept.

PAPER 5 :

NAME : Drowning Detection Based on Background Subtraction

AUTHOR : Lei Fei , Xang Xueli , Chen Donsheng

MERITS : method is effective to detect the drowners and eliminate the shadows.

CONCLUSION : gives a clear view of the drowners not by checking over their shadows.

PAPER 6 :

NAME : Off-time swimming pool surveillance using thermal imaging system

AUTHOR : Wai Kit Wong , Joe How Hui , Way Soong Lim

MERITS : the intruder detection algorithm achieved high accuracy of 95.58% for region outside swimming pool and 92.44% for region inside swimming pool.

CONCLUSION : enables night vision to detect people from drowning i and outside the pools.

PROBLEM STATEMENT DEFINITION:-

This project describes the drowning detection system for the prevention of drowning incidents in swimming pools. The problem boundary clearly distinguishes between the positive samples which are inside the boundary to those that are less relevant and outside the boundary. It works like an “extra lifeguard” for under the water of swimming pools. For instance, if it happens to someone to drown inside the swimming pool, it makes them take an excess amount of water content which affects the internal organs and sometimes it may be the cause of death. This detection system tracks the movements of everything inside the water bodies and will help to guard the lives by finding them easily.

Classification of drowning stages:

Stage 1*:* User needs a way to detect a person from drowning, because now a days lots of kids lose their lives by drowning.

Stage 2: User is a fireman who needs a way to detect a person from drowning because he needs to save people.

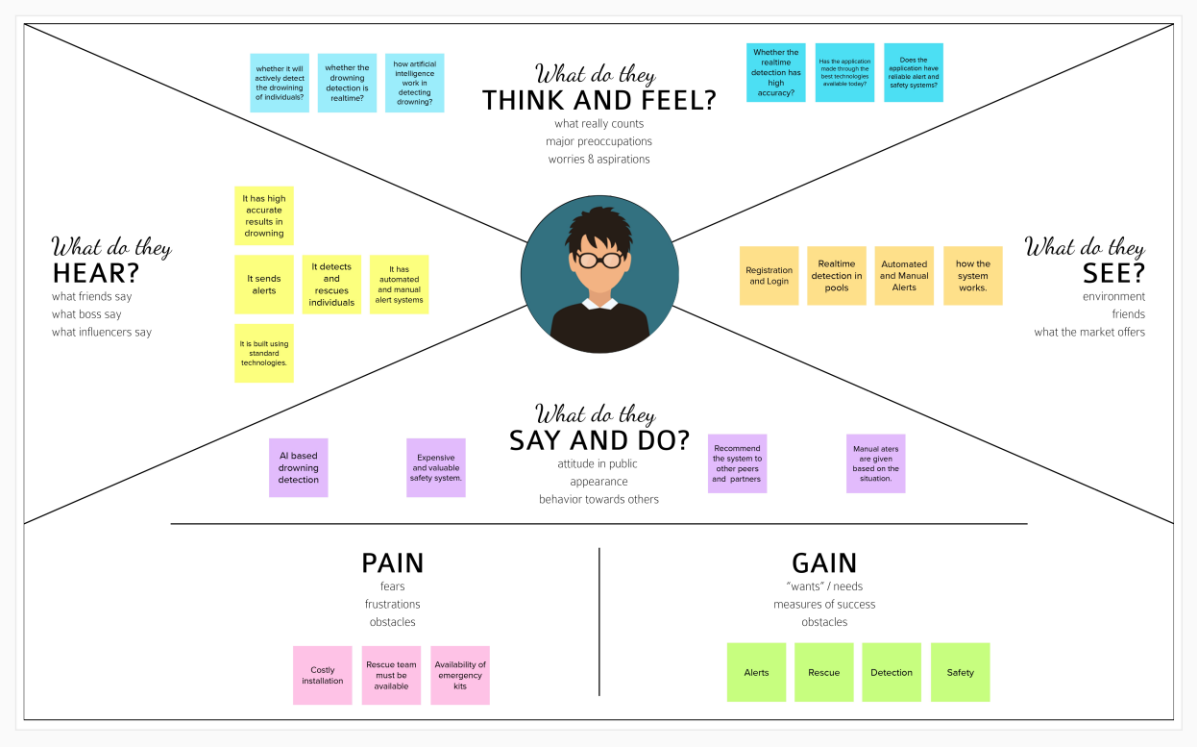
Stage 3: User need a way to detect the real time live environment and safeguard system.

Stage 4: Distance: Z-axis coordinates from the depth information acquired by the camera, corrected by adding from the deviation in the y-axis direction (straight line distance from the camera to the nose).

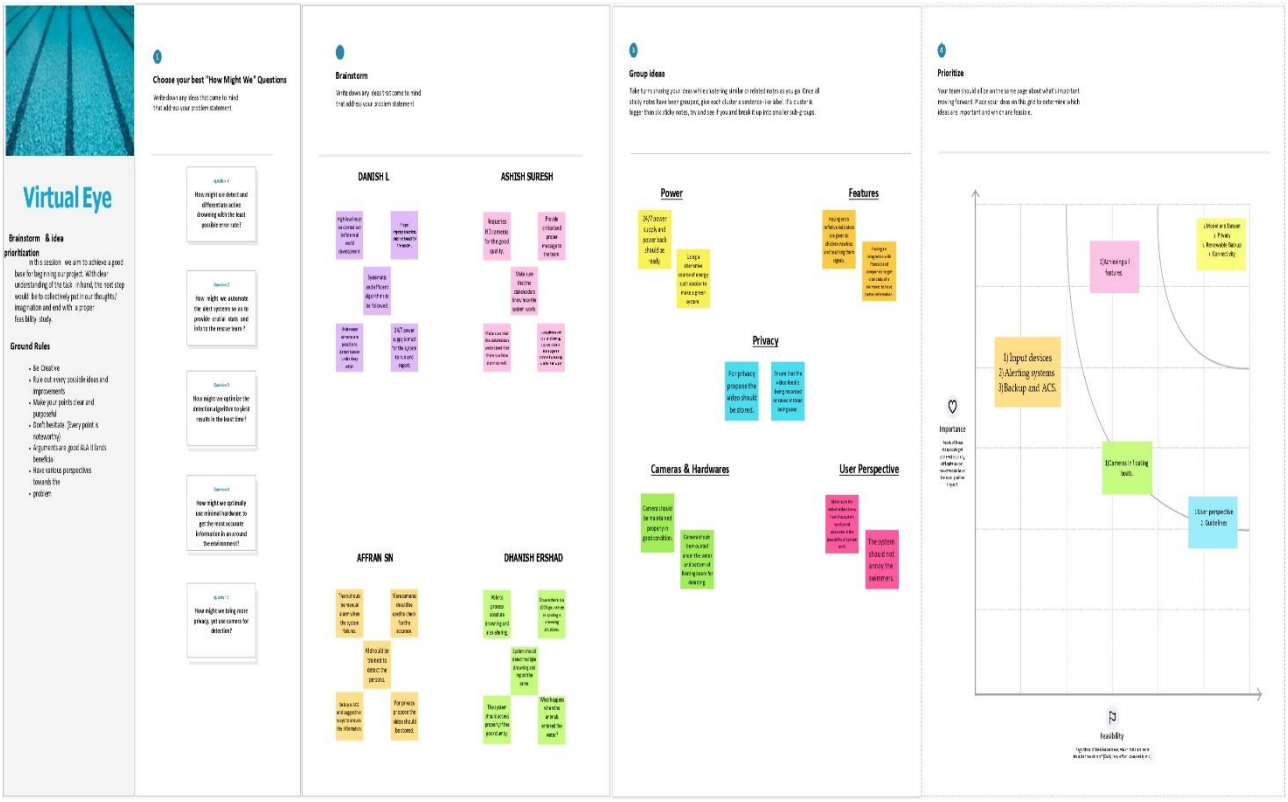
Stage 5: Depth: Z-axis coordinates obtained from the depth information acquired by the camera.

IDEATION AND PROPOSED SOLUTION

EMPATHY MAP CANVAS:-



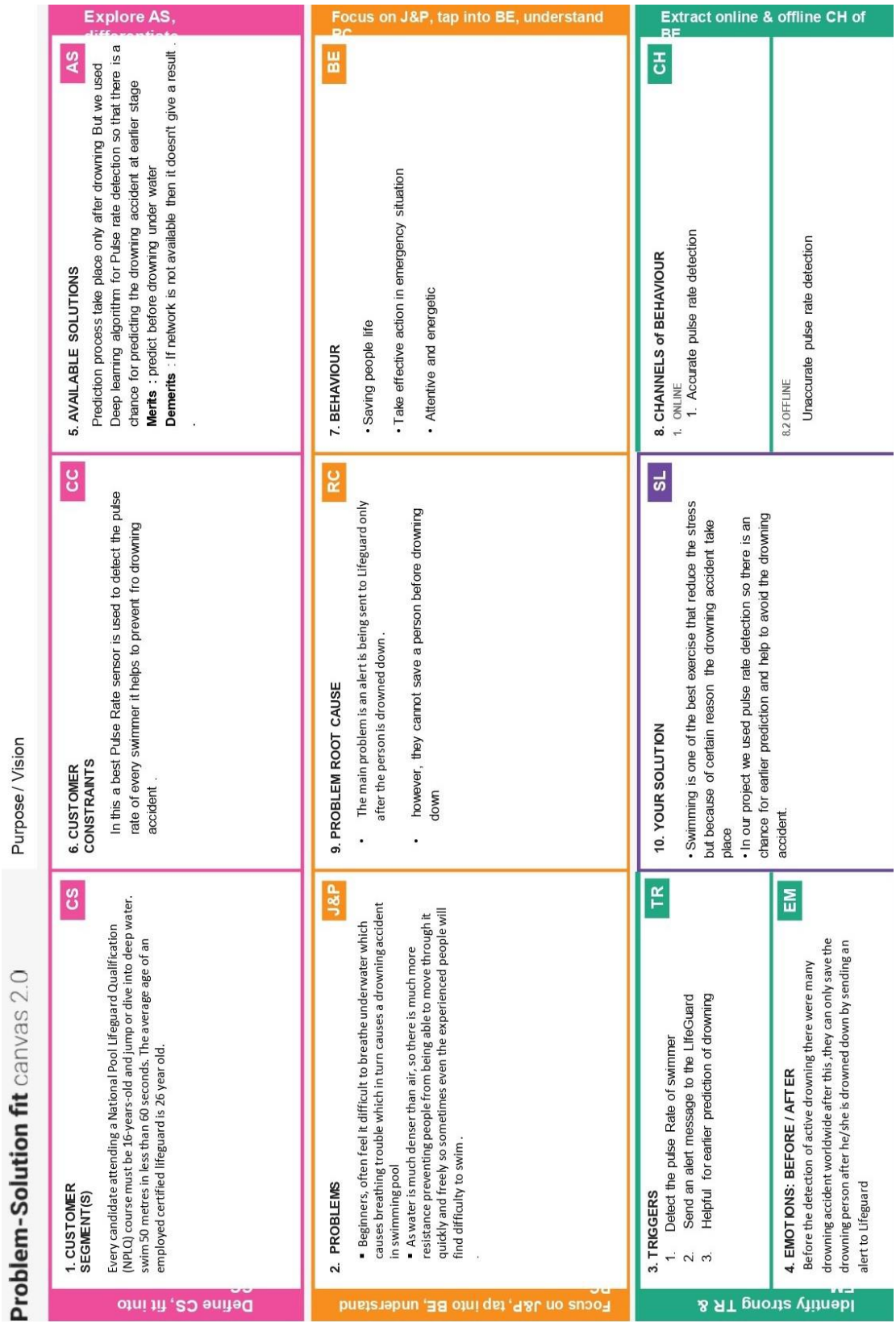
IDEATION AND BRAINSTORMING:-



PROPOSED SOLUTION:-

S.No.	Parameters	Description
1.	Problem Statement (Problem to be solved)	VirtualEye - LifeGuard for Swimming Pools To Detect Active Drowning.
2.	Idea / Solution description	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 so In This is project a Accurate Pulse Rate of every individual swimmer is also detected and sended as signal to the LifeGuard through alert message so it help LifeGuard to do earlier prediction of a swimmer pulse rate is reduced or increased By doing this they can get alert in advance and can save more then one person from Drowning.
3.	Novelty / Uniqueness	Accurate pulse rate detection using Deep learning.
4.	Social Impact / Customer Satisfaction	In case of an incident it is possible to extract and store not only the videos but also Pulse rate of a victim so it will be usefull to indentify the reason behind his/her drowness.
5.	Business Model (Revenue Model)	Can generate revenue from direct customers,like Lifeguard and collaborate with maritime sector and other swimming pool authorities.
6.	Scalability of the Solution	Deep learning Algorithm for the Pulse rate detection : It helps the LifeGuard for earlier prediction of drowning along with the Reason behind his/her drowning.

PROBLEM SOLUTION FIT



REQUIREMENT ANALYSIS

Functional requirement:-

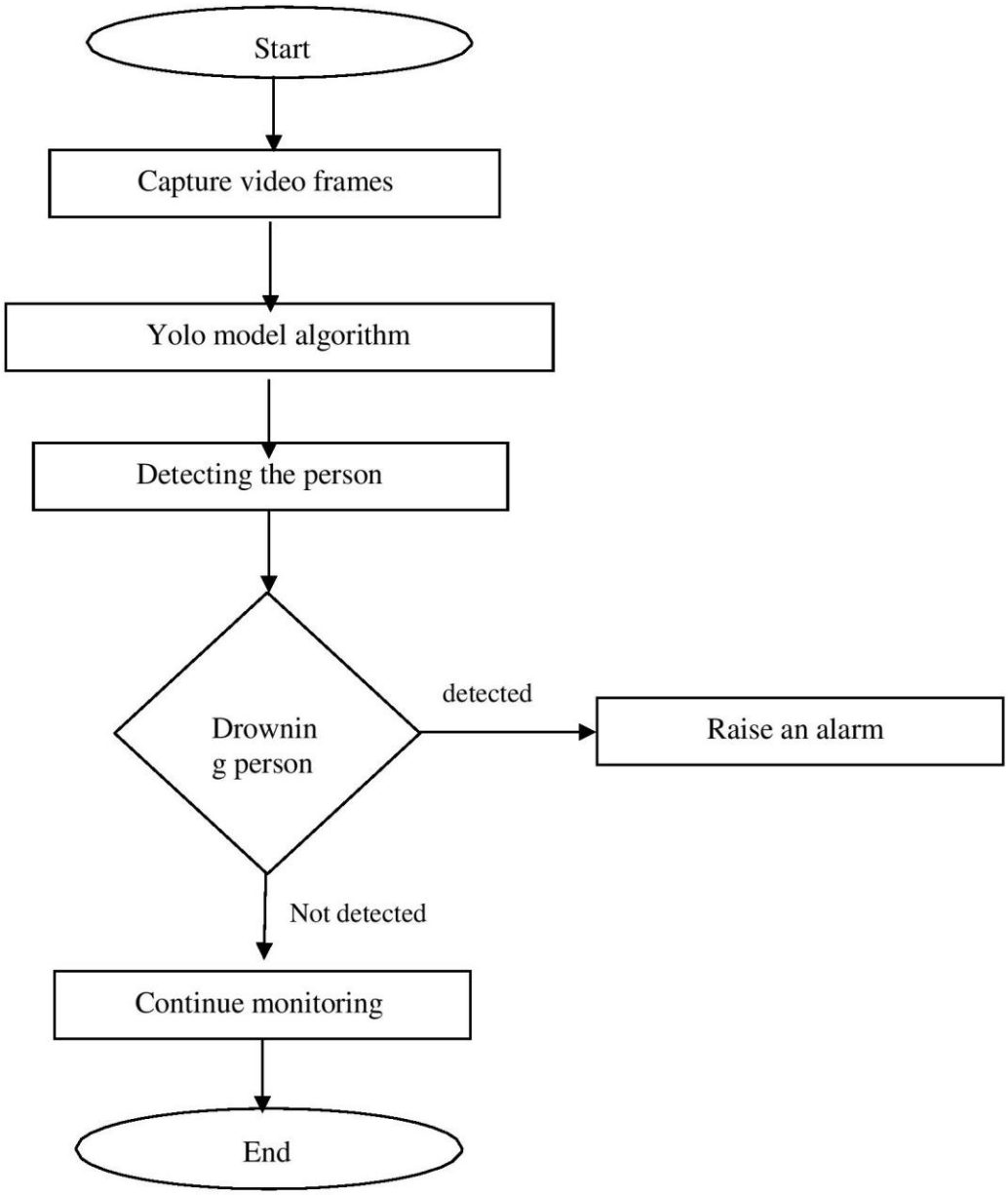
FR.No.	Functional Requirement	Sub Requirement
FR-1	Camera Installation	Cameras should be installed inside water and in the walls of the building.
FR-2	Sensor Installation	Installed under the water without disturbing the people.
FR-3	Deduction	Detected by pulse rate and movements.
FR-4	Alert	Sends an alert message to the lifeguard.
FR-5	Support	Lifeguard help or swim tubes.
FR-6	Alarm	Rings alarm with drowning detected.

Non-Functional Requirement

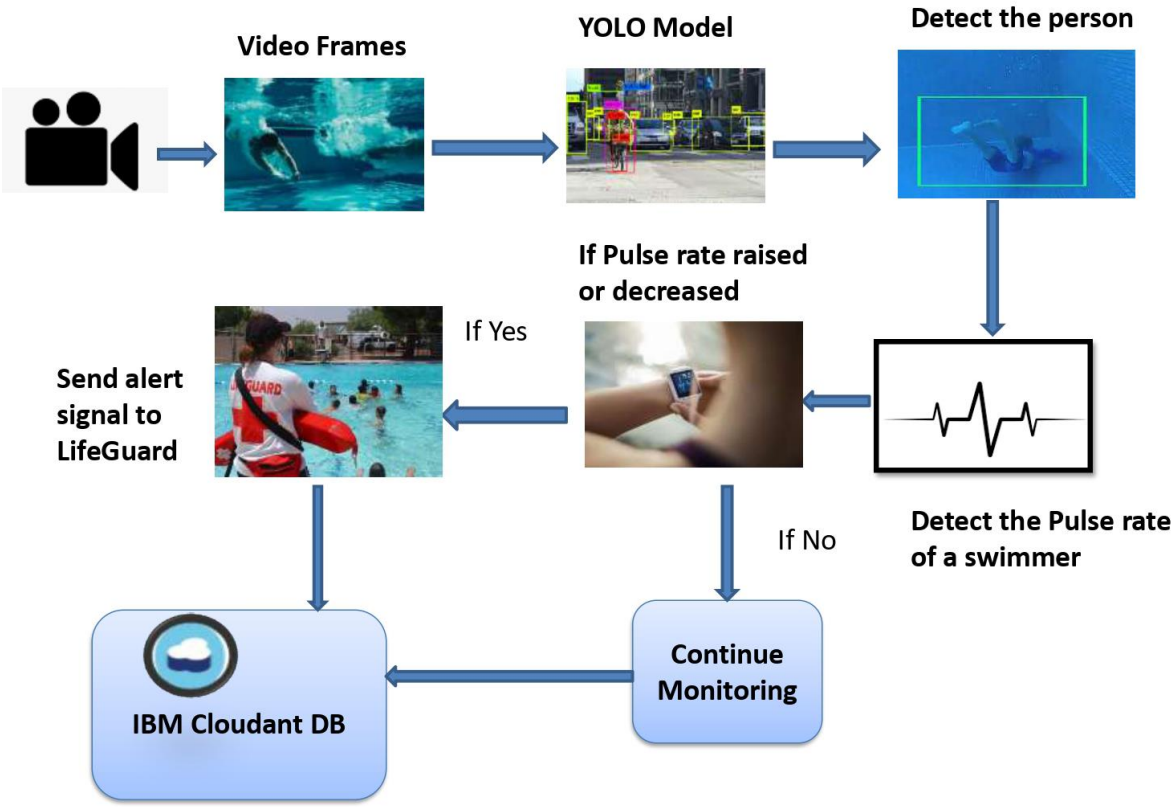
FR No.	Non-Functional Requirement	Description
NFR-1	Usability	When someone is drowning, the sensor detects the pulse rate and locate the swimmer who is drowning and alert the people.
NFR-2	Security	Lifeguards will be present in the pool and the cameras are secured by the management and are safe.
NFR-3	Reliability	The process will be a reliable multimedia videobased surveillance system.
NFR-4	Performance	When the pulse rate of the swimmer reduces then the alarm will be triggered.
NFR-5	Availability	Detection equipment includes safety wheel, pool hook, rescue tubes, first aid box etc.

PROJECT DESIGN

Data Flow Diagram:-



Solution and Technical Architecture:-



PROJECT PLANNING AND SCHEDULING

Sprint Planning and Estimation:-

Sprint	User Story / Task	Story Points	Priority	Team Members
Sprint-1	I can register for the application by entering my phone number.	1	High	Danish
	I will receive confirmation OTP once I have registered for the application	2	Low	Affran
	I can also register for the application through Gmail	2	Medium	Dhanish
	I can login into the application by entering email or phone number & password.	1	High	Ashish
	In prediction page, the data uploaded will help the user to detect the drowning movements	2	Medium	Affran
	The dataset collected will give high accuracy on the drowning details of the person.	2	High	Ashish

Sprint	User Story / Task	Story Points	Priority	Team Members
Sprint-2	The dataset is extracted and is used to train the model	4	High	Affran
	We will train the model	8	High	Ashish
	We will test the model	6	High	Danish
Sprint-3	The tested model will be loaded	3	High	Dhanish
	To identify the person by collecting realtime data.	5	Medium	Affran
	The data collected at present is checked with the pre-fed data	8	High	Ashish
Sprint-4	When the abnormal movement is detected the system will ring an alarm to notify the lifeguard to rescue the person	7	High	Danish
	We will be able to detect the drowning	3	Medium	Dhanish

Sprint Delivery Schedule:-

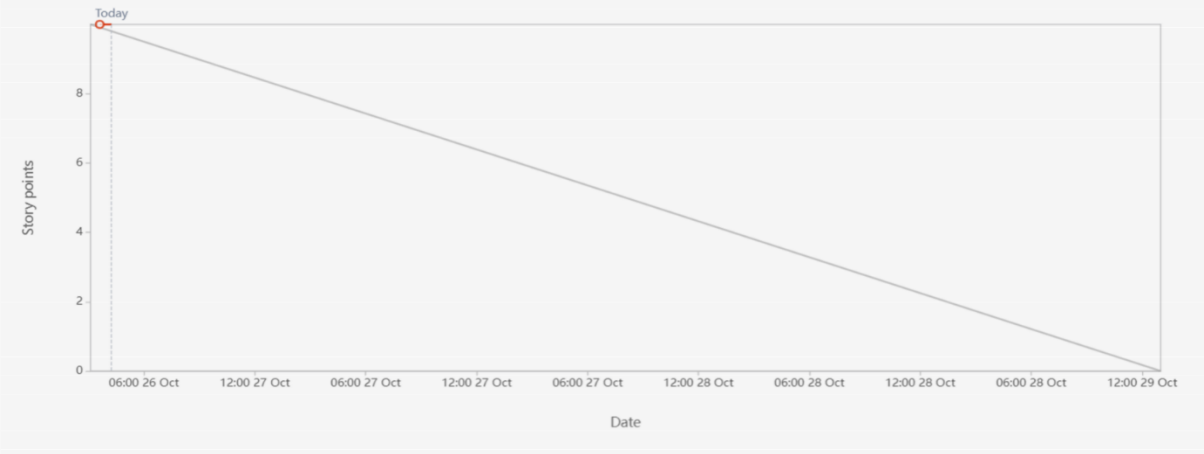
Sprint	Total Story Points	Duration	Sprint start date	Sprint end date (planned)	Story points completed (as on planned end date)	Srint release date (Actual)
Sprint-1	10	6 Days	24 Oct 2022	29 Oct 2022		
Sprint-2	18	6 Days	31 Oct 2022	05 Nov 2022		
Sprint-3	16	6 Days	07 Nov 2022	12 Nov 2022		
Sprint-4	12	6 days	14 Nov 2022	19 Nov 2022		

Velocity:

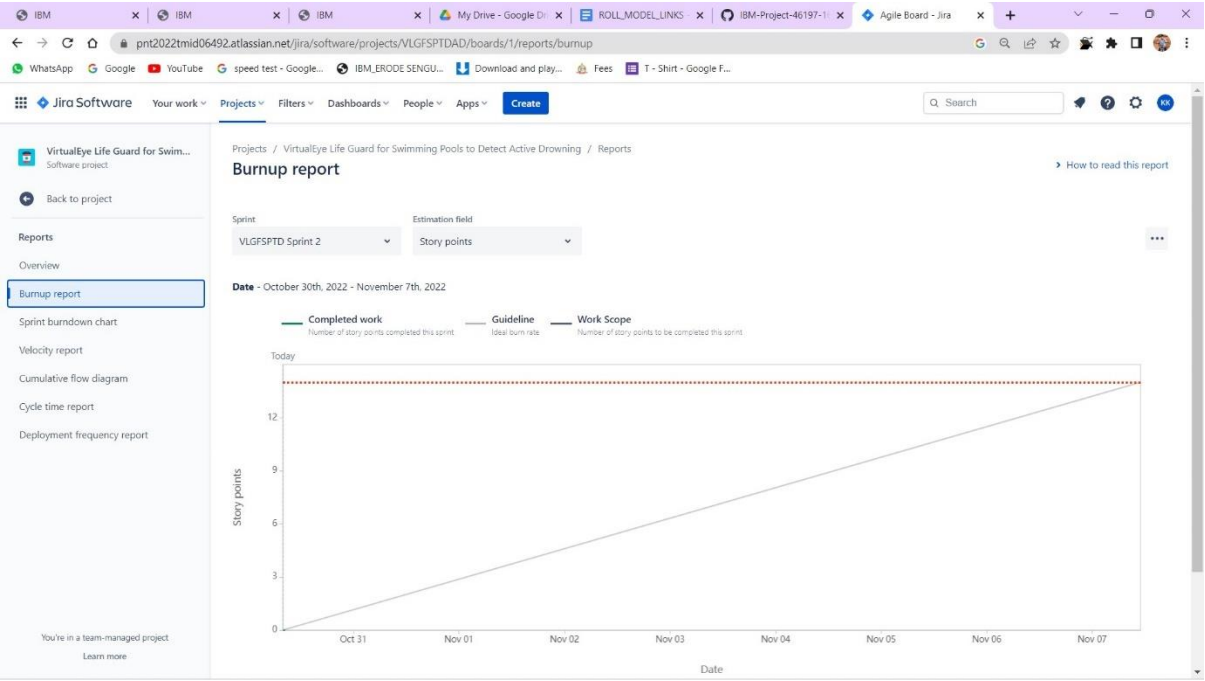
For Sprint-1 the Average Velocity (AV) is:
AV = Sprint Duration / velocity = 10 / 6 = 1.6
For Sprint-2 the Average Velocity (AV) is:
AV = Sprint Duration / velocity = 18 / 6 = 3.0
For Sprint-3 the Average Velocity (AV) is:
AV = Sprint Duration / velocity = 16 / 6 = 2.6
For Sprint-4 the Average Velocity (AV) is:
AV = Sprint Duration / velocity = 12/ 6 = 2.0

Reports from JIRA:-

BURN DOWN CHART:



BURN UP CHART :



CODING AND SOLUTIONING

NOTE : codes are available in the APPENDIX

FEATURE 1 :

FOR HTML CODE :

For this project create three HTML files namely

- **index.html**
- **base.html**
- **register.html**
- **login.html**
- **prediction.html**
- **logout.html**

and save them in the templates folder.

FEATURE 2 :

FOR PYTHON CODE ;

A. Import the libraries
B. Create a database using an initiated client.
C. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument.
D. Configure the registration page
E. Configure the login page

F. For logout from web application.

G. Create res() function for drowning detection

H. Creating bounding box

I. Main Function

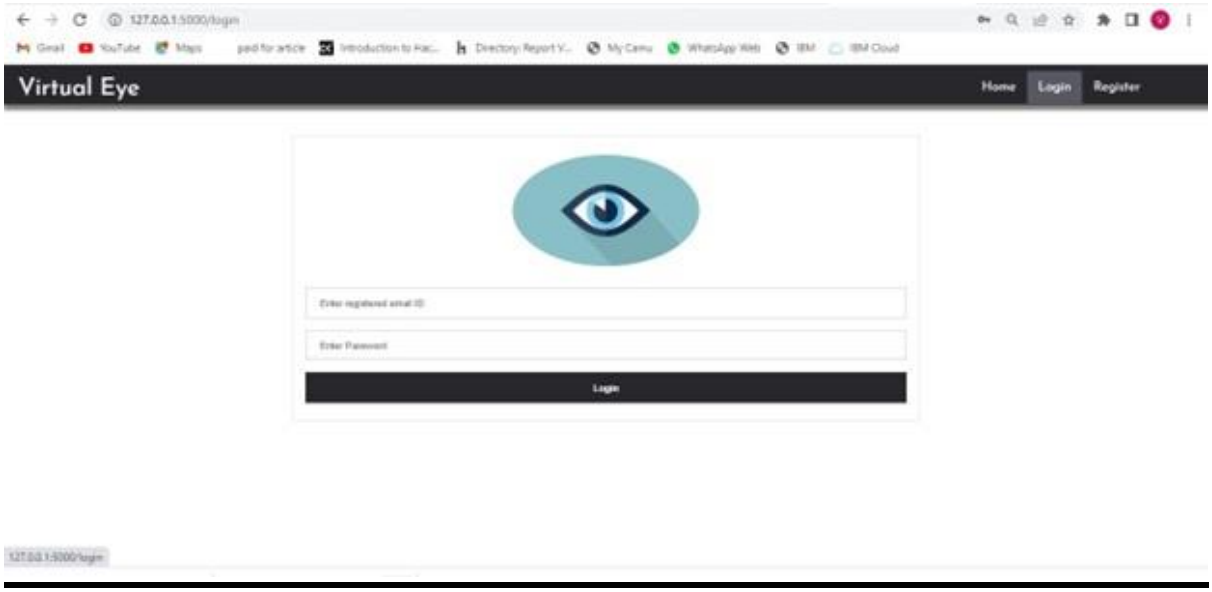
RUN THE APPLICATION :

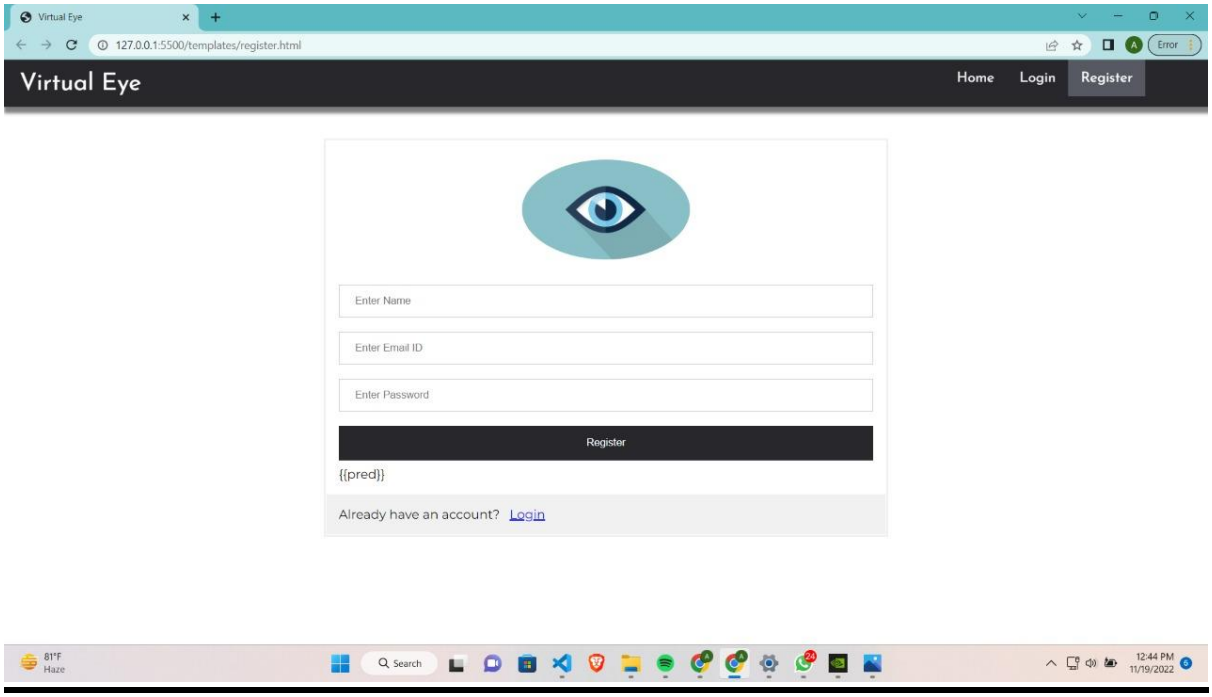
Run the application

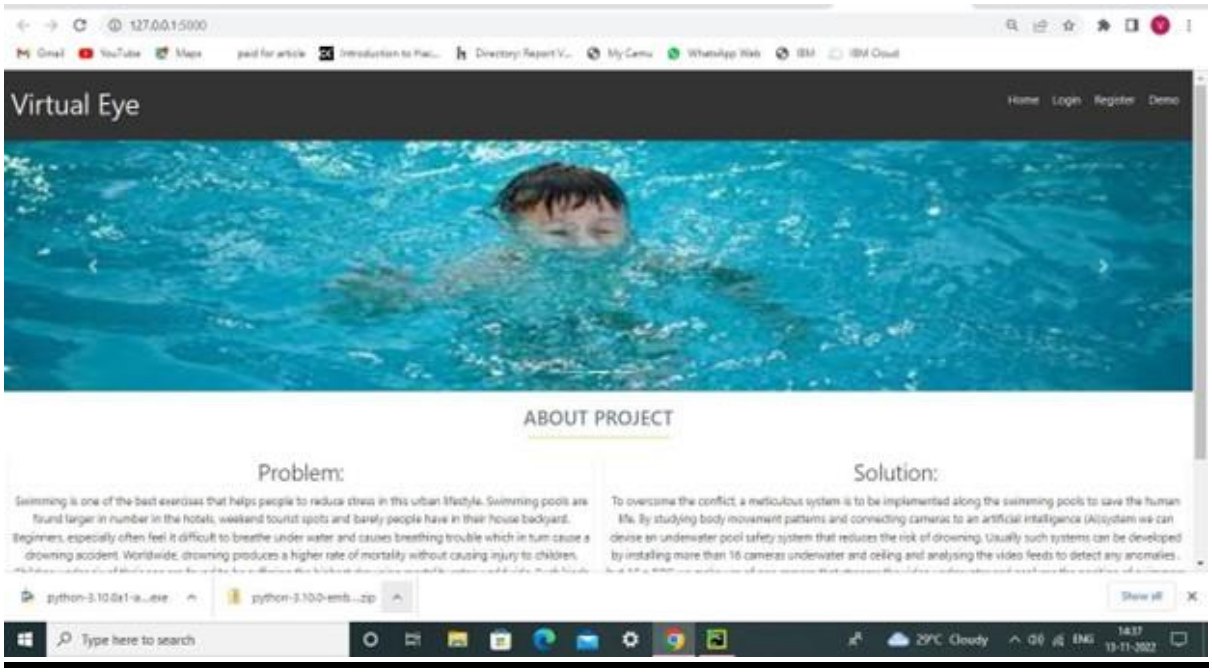
- Open the anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type the "python app.py" command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top right corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

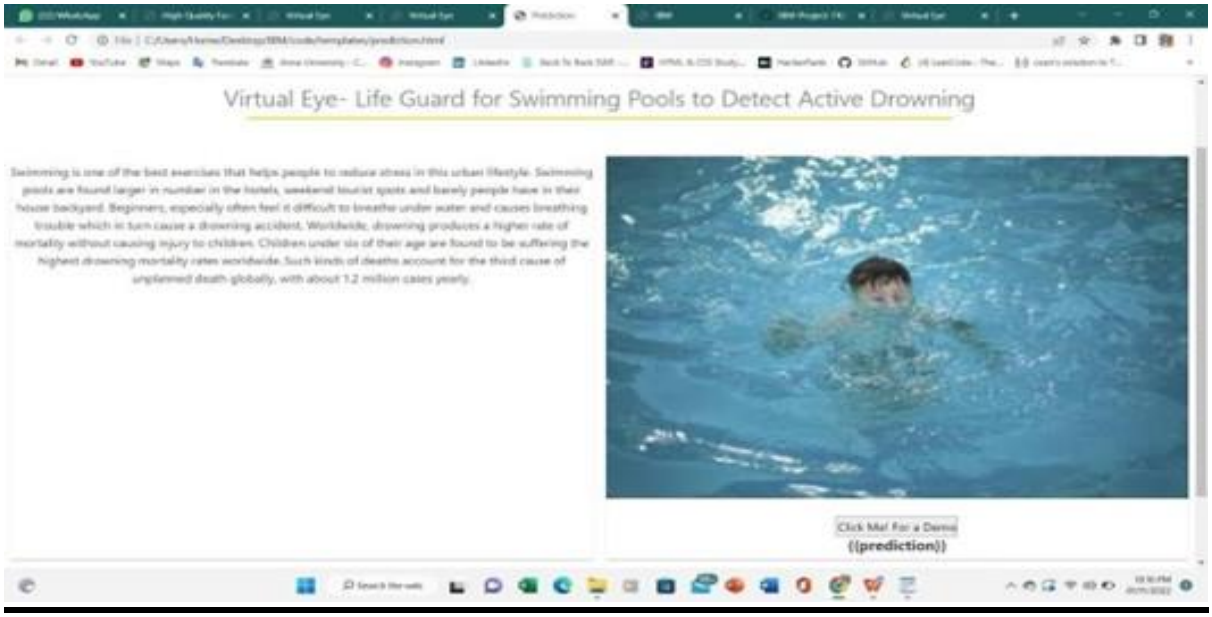
TESTING

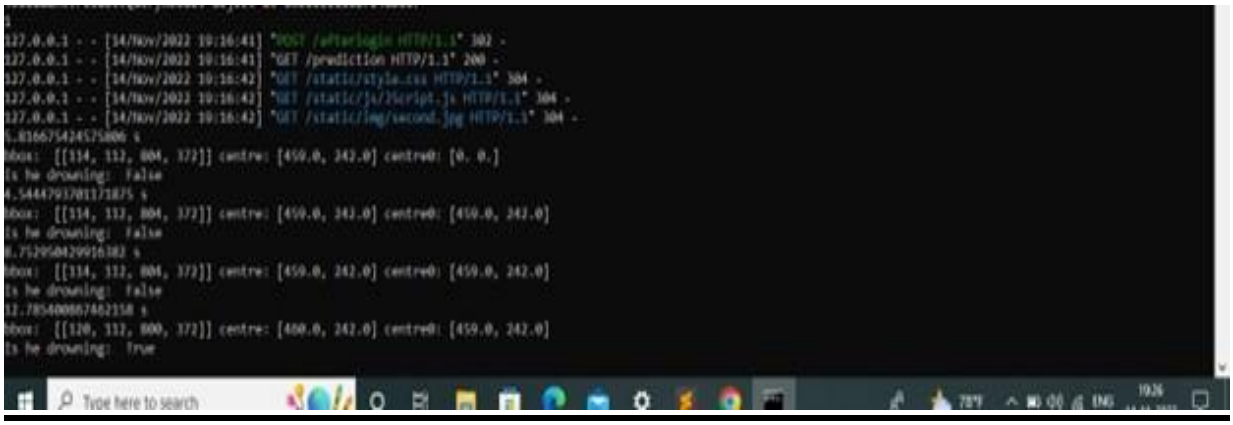
USER ACCEPTANCE TESTING :

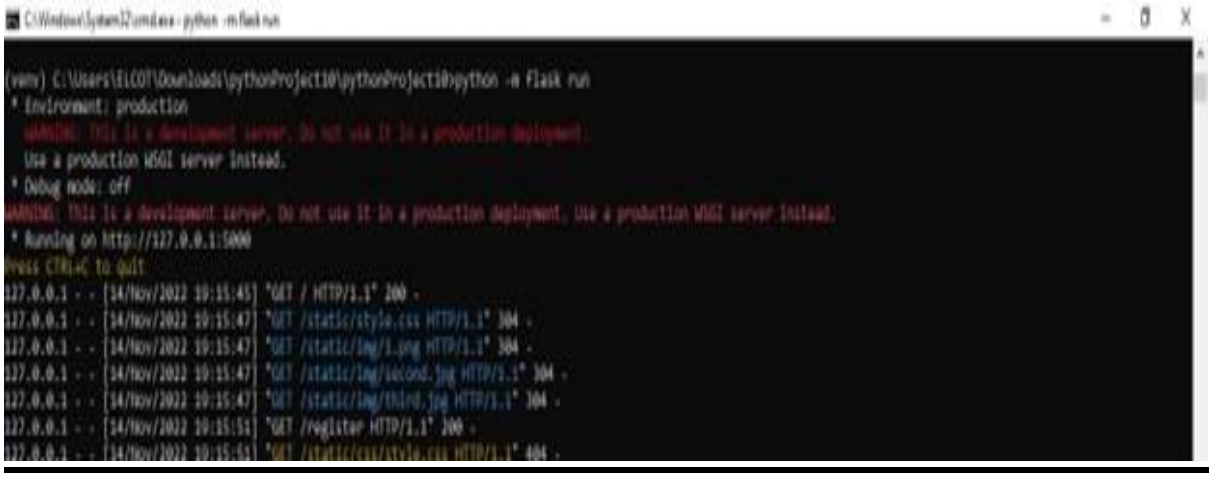




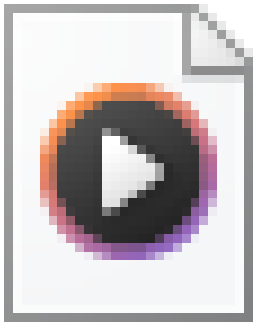

Virtual Eye
Home Login Register Demo
ABOUT PROJECT
Problem:
Solution:


Virtual Eye- Life Guard for Swimming Pools to Detect Active Drowning
Click Me! For a Demo
{{prediction}}


127.0.0.1 - - [14/Nov/2022 10:16:41] "POST /afterlogin HTTP/1.1" 302 -
127.0.0.1 - - [14/Nov/2022 10:16:41] "GET /prediction HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 10:16:42] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:16:42] "GET /static/js/JScript.js HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:16:42] "GET /static/img/second.jpg HTTP/1.1" 304 -
5.816675424575806 s
bbox: [[114, 112, 804, 372]] centre: [459.0, 242.0] centre0: [0. 0.]
Is he drowning: False
4.544479370117875 s
bbox: [[114, 112, 804, 372]] centre: [459.0, 242.0] centre0: [459.0, 242.0]
Is he drowning: False
8.752050429916382 s
bbox: [[114, 112, 804, 372]] centre: [459.0, 242.0] centre0: [459.0, 242.0]
Is he drowning: False
12.785400867462158 s
bbox: [[120, 112, 800, 372]] centre: [460.0, 242.0] centre0: [459.0, 242.0]
Is he drowning: True
Type here to search


C:\Windows\System32\cmd.exe - python -m flask run
(venv) C:\Users\ELCOT\Downloads\pythonProject10\pythonProject10>python -m flask run
* Environment: production
Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [14/Nov/2022 10:15:46] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [14/Nov/2022 10:15:47] "GET /static/style.css HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:15:47] "GET /static/img/1.png HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:15:47] "GET /static/img/second.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:15:47] "GET /static/img/third.jpg HTTP/1.1" 304 -
127.0.0.1 - - [14/Nov/2022 10:15:51] "GET /register HTTP/1.1" 200 -

RESULTS



DROWNING.mp4

ADVANTAGES AND DISADVANTAGES

ADVANTAGES :

A. It works like an “extra lifeguard” for under the water of swimming pools

B. This project describes the drowning detection system for the prevention of drowning incidents in swimming pools

C. This is project a Accurate Pulse Rate of every individual swimmer is also detected and send as signal to the Life Guard through alert message so it help Life Guard to do earlier prediction of a swimmer pulse rate is reduced or increased By doing this they can get alert in advance and can save more then one person from Drowning

DISADVANTAGES :

A. A limitation of this equipment is that if too many swimmers, the occlusion problem arises.

B. The reflection and refraction of light in air-water interference will affect the image quality, and drowning man

C. This method detected is not easy to distinguish swimmers and divers

D. This system needs constant observation which is the main disadvantage.

CONCLUSION

We provided a method to check human tracking and semantic event detection within the context of video surveillance system capable of automatically detecting drowning incidents in a swimming pool. In the current work, an effective background detection that incorporates prior knowledge using YOLO algorithm and contour detection enables swimmers to be reliably detected and tracked despite the significant presence of water ripples. The system has been tested on several instances of simulated water conditions such as water reflection, lightening condition and false alarms. Our algorithm was able to detect all the drowning conditions along with the exact position of the drowning person in the swimming pool and had an average detection delay of 1.53 seconds, which is relatively low compared to the needed rescue time for a lifeguard operation. Our results show that the proposed method can be used as a reliable multimedia video-based surveillance system.

FUTURE SCOPE :

Life safety in water has been a concern for many centuries. Latest technology advancements has enabled us to come up with effective drowning detection systems. However many of those solutions are costly and limited to few. Survey reports show us that highest numbers of deaths are reported in low and middle income countries. The survey report also mentions the children have the largest death ratio compared to adults. Also the deaths reported in these incidents are more from open water bodies than closed water bodies like swimming pools.

The solution described above will be able to address these issues. The swimming goggles with drowning detection unit can be economically viable solution. The range of the alarms transmission can be improved by using underwater acoustics. Any age groups will be comfortable wearing the goggles, without hampering the recreational joy while swimming. The goggles can be useful even in sea. The alarm receivers can be placed at different locations in the water bodies which is having high chance of drowning. Another major advantage of this approach unlike other approach is the ease of use in all atmospheric conditions, like rain or wind to day or night. This solution is also a reliable solution where the life guards have difficulty to monitor the swimmers like a highly crowded se

APPENDIX

Source code :

Python :

```
from flask import Flask
from flask import flash, request, redirect, render_template, url_for
from cloudant.client import Cloudant
from cvlib.object_detection import draw_bbox

import time
import os
import cvlib as cv
import cv2
import time
import numpy as np

app = Flask(__name__)
client = Cloudant.iam(
    'c56fd99d-acbb-4081-8351-ed1e1e82ba02-bluemix',
    'RKYFESfjUudW-C8Pm-2WaAn3Q9N4Ud49q2PzIbVV4NdU',
    connect=True)
db = client['user_details']


@app.after_request
def add_header(r):
    """
    Add headers to both force latest IE rendering engine or Chrome Frame,
    and also to cache the rendered page for 10 minutes.
    """
    r.headers["Cache-Control"] = "no-cache, no-store, must-revalidate"
    r.headers["Pragma"] = "no-cache"
    r.headers["Expires"] = "0"
    r.headers['Cache-Control'] = 'public, max-age=0'
    return r


@app.route("/")
def home():
    return render_template('index.html')


@app.route("/login", methods=["GET", "POST"])
def login():
    if request.method == "POST":
        x = [x for x in request.form.values()]
        data = {
            '_id': x[0],
            'psw': x[1]
        }
        query = {'_id': {'$eq': data['_id']}}
```

```
        docs = db.get_query_result(query)
        if len(docs.all()) == 0:
            db.create_document(data)
            return render_template('login.html', title="VirtualEye -
Login", status='NR')
        else:
            if x[0] == docs[0][0]['_id'] and x[1] == docs[0][0]['psw']:
                return redirect(url_for('prediction'))
            else:
                return render_template('login.html', title="VirtualEye -
Login", status="Failed")
    return render_template('login.html', title="VirtualEye - Login")


@app.route("/register", methods=['GET', 'POST'])
def register():
    if request.method == "POST":
        x = [x for x in request.form.values()]
        data = {
            '_id': x[1],
            'name': x[0],
            'psw': x[2]
        }
        query = {'_id': {'$eq': data['_id']}}
        docs = db.get_query_result(query)
        if len(docs.all()) == 0:
            db.create_document(data)
            return render_template('register.html', title='VirtualEye -
Register', status='Success')
        else:
            return render_template('register.html', title='VirtualEye -
Register', status='Failed')
    return render_template('register.html', title='VirtualEye - Register')


@app.route("/demo", methods=['GET'])
def demo():
    return render_template('base.html', title="VirtualEye - Demo")


@app.route("/forgotpassword")
def forgotpass():
    return render_template('base.html', title="VirtualEye")


@app.route("/logout")
def logout():
    return render_template('logout.html', title="VirtualEye - Logged out")


@app.route('/result')
def prediction():
    webcam = cv2.VideoCapture('drowning.mp4')

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

    t0 = time.time()
    centre0 = np.zeros(2)
    isDrowning = False
```

```
    while webcam.isOpened():
        status, frame = webcam.read()

        bbox, label, conf = cv.detect_common_objects(frame)

        if len(bbox) > 0:
            centre = [0, 0]
            centre = [(bbox[0][0] + bbox[0][2]) / 2, (bbox[0][1] +
bbox[0][3]) / 2]
            hmov = abs(centre[0] - centre0[0])
            vmov = abs(centre[1] - centre0[1])

            x = time.time()
            threshold = 10

            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('bbox:', bbox, ' center:', centre, ' centre0:', centre0)
            print('Are they drowning: ', isDrowning)

            centre0 = centre

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

        cv2.imshow("Real-time object detection", out)

        if isDrowning:
            os.system("mpg123 -q alarm.mp3")
            webcam.release()
            cv2.destroyAllWindows()
            return render_template('prediction.html',
prediction="Emergency!!! The Person is drowning",
                                   title="VirtualEye - Prediction")

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

    webcam.release()
    cv2.destroyAllWindows()
    return render_template('prediction.html', title='VirtualEye -
Prediction', prediction='Waiting for footage')


if __name__ == '__main__':
    app.run(debug=True)
```

init.py:

```
from .object_detection import detect_common_objects
```

object detection.py :

```
#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()]

    return classes


def get_output_layers(net):

    #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')) + '%'


        #you only need to points (the opposite corners) to draw a rectangle.
These points
        #are stored in the variable bbox
        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
```

utils.py :

```
import requests
import progressbar as pb
import os

def download_file(url, file_name, dest_dir):

    if not os.path.exists(dest_dir):
        os.makedirs(dest_dir)

    full_path_to_file = dest_dir + os.path.sep + file_name

    if os.path.exists(dest_dir + os.path.sep + file_name):
        return full_path_to_file

    print("Downloading " + file_name + " from " + url)

    try:
```

```
        r = requests.get(url, allow_redirects=True, stream=True)
    except:
        print("Could not establish connection. Download failed")
        return None

    file_size = int(r.headers['Content-Length'])
    chunk_size = 1024
    num_bars = round(file_size / chunk_size)

    bar = pb.ProgressBar(maxval=num_bars).start()

    if r.status_code != requests.codes.ok:
        print("Error occurred while downloading file")
        return None

    count = 0

    with open(full_path_to_file, 'wb') as file:
        for chunk in  r.iter_content(chunk_size=chunk_size):
            file.write(chunk)
            bar.update(count)
            count +=1

    return full_path_to_file
```

drowndetect.py :

```
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):
                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)

    # press "Q" to stop
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# release resources
webcam.release()
cv2.destroyAllWindows()
```

HTML code :

INDEX PAGE :

```
<!-- NAVIGATION MENUS -->
   <!DOCTYPE html>
<html lang="en">
 <head>
   <meta charset="UTF-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <style>
     * {
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}
body {
 font-family: cursive;
}
a {
 text-decoration: none;
}
li {
 list-style: none;
}
.navbar {
 display: flex;
 align-items: center;
 justify-content: space-between;
 padding: 20px;
 background-color: teal;
 color: #fff;
}
.nav-links a {
 color: #fff;
}
/* LOGO */
.logo {
 font-size: 32px;
}
/* NAVBAR MENU */
.menu {
 display: flex;
 gap: 1em;
 font-size: 18px;
}
.menu li:hover {
 background-color: #4c9e9e;
 border-radius: 5px;
 transition: 0.3s ease;
}
.menu li {
 padding: 5px 14px;
}
.services {
 position: relative;
}
.dropdown {
```

```
 background-color: rgb(1, 139, 139);
 padding: 1em 0;
 position: absolute; /*WITH RESPECT TO PARENT*/
 display: none;
 border-radius: 8px;
 top: 35px;
}
.dropdown li + li {
 margin-top: 10px;
}
.dropdown li {
 padding: 0.5em 1em;
 width: 8em;
 text-align: center;
}
.dropdown li:hover {
 background-color: #4c9e9e;
}
.services:hover .dropdown {
 display: block;
}
#example1 {
  background: url(swim.jpg);

}
#swim
{
   height: 200px;
  width: 50%;
}
   </style>
   <title>Document</title>
 </head>
 <body>
   <nav class="navbar">
     <!-- LOGO -->
     <div class="logo">VIRTUAL EYE</div>
     <!-- NAVIGATION MENU -->
     <ul class="nav-links">
       <!-- USING CHECKBOX HACK -->

       <div class="menu">
         <li><a href="/">Home</a></li>
         <li><a href="/">About</a></li>
         <li class="services">
           <a href="/">Prediction</a>

         </li>
         <li><a href="/register">Register</a></li>
         <li><a href="/login">Login</a></li>
       </div>
     </ul>
   </nav>
   <div class="swim">
     <img style="height:670px; width:1520px"src="/static/swim.jpg">
     <a href="/result"><button style = "position:absolute; right:60px;
bottom:45px; height:40px; width:500px; color:cyan; background:black;">TRY
THIS PROJECT IN DEMO VERSION (CLICK HERE)</button></a>
   </div>
```

```
 </body>
</html>
```

LOGIN PAGE :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>login</title>
  <style>
    * {box-sizing: border-box}

/* Add padding to containers */
.container {
  padding: 16px;
}

/* Full-width input fields */
input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: 1px solid;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

/* Overwrite default styles of hr */
hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

/* Set a style for the submit/register button */
.registerbtn {
  background-color: #04AA6D;
  color: baby blue;
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity:1;
}

/* Add a blue text color to links */
a {
  color: aqua
```

```
;
}

/* Set a grey background color and center the text of the "sign in" section
*/
.signin {
  background-color: #89cfef;
  text-align: center;
}
p {
  background-image: url('img_girl.jpg');
}
  </style>
</head>
<body>
<form>
  <div class="container">
    <h1>LOGIN</h1>
    <p>Login with your credentials.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email" id="email"
required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" id="psw"
required>


    <p><a href="/index" class="registerbtn">LOGIN</a>.</p>
     < p style="background-image: url('img_girl.jpg');">
  </div>


</form>
</body>
</html>
```

LOGOUT PAGE :

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Virtual Eye - Home</title>
    <script
src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-
ratio,line-clamp"></script>
</head>

<body>
    <header>
        <nav class="flex items-center justify-between flex-wrap bg-gray-700
p-6">
```

```
            <div class="flex items-center flex-shrink-0 text-white mr-6
basis-4/5">
                <span class="font-semibold text-xl tracking-tight">Virtual
Eye</span>
            </div>
            <div class="w-full block flex-grow lg:flex lg:items-center
lg:w-auto">
                <div class="text-sm lg:flex-grow font-semibold">
                    <a href="index.html" class="block mt-4 lg:inline-block
lg:mt-0 text-gray-400 hover:text-white mr-4">
                        Home
                    </a>
                    <a href="register.html"
                        class="block mt-4 lg:inline-block lg:mt-0 text-
gray-400 hover:text-white mr-4">
                        Register
                    </a>
                    <a href="login.html" class="block mt-4 lg:inline-block
lg:mt-0 text-gray-400 hover:text-white mr-4">
                        Login
                    </a>
                    <a href="demo.html" class="block mt-4 lg:inline-block
lg:mt-0 text-gray-400 hover:text-white mr-4">
                        Demo
                    </a>
                </div>
            </div>
        </nav>
    </header>

    <div class="flex-col items-center">
        <p class="font-semibold text-2xl m-8" style="text-align:
center">Successfully Logged Out!</p>
        <p class="font-semibold text-l mb-4 text-green-500" style="text-
align: center">Login for more information</p>
        <button
            class="block m-auto bg-blue-500 hover:bg-blue-700 text-white
font-bold py-2 px-4 rounded-full">Login</button>
    </div>

</body>

</html>
```

PREDICTION PAGE :

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Virtual Eye - Home</title>
    <script
src="https://cdn.tailwindcss.com?plugins=forms,typography,aspect-
ratio,line-clamp"></script>
</head>
```

```
<body>
    <header>
        <nav class="flex items-center justify-between flex-wrap bg-gray-700
p-6">
            <div class="flex items-center flex-shrink-0 text-white mr-6
basis-4/5">
                <span class="font-semibold text-xl tracking-tight">Virtual
Eye</span>
            </div>
            <div class="w-full block flex-grow lg:flex lg:items-center
lg:w-auto">
                <div class="text-sm lg:flex-grow font-semibold">
                    <a href="index.html" class="block mt-4 lg:inline-block
lg:mt-0 text-gray-400 hover:text-white mr-4">
                        Home
                    </a>
                    <a href="logout.html"
                        class="block mt-4 lg:inline-block lg:mt-0 text-
gray-400 hover:text-white mr-4">
                        Logout
                    </a>
                </div>
            </div>
        </nav>
    </header>

    <p class="font-bold text-3xl px-8 pt-8 underline" style="text-align:
center">VirtualEye - Life Guard for Swimming
        Pools to Detect Active Drowning</p>
    <div class="grid grid-cols-2">
        <div class="flex-col bg-white shadow-md rounded px-8 pt-10 pb-8 m-8
mr-4">
            <p class="m-auto text-xl font-semibold mb-4">Problem</p>
            <p>
                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.
            </p>

        </div>
        <div class="flex-col bg-white shadow-md rounded px-8 pt-10 pb-8 m-8
ml-4">
            <img class="h-56 m-auto" src="static_images/background.jpg">
            <button
                class="bg-blue-500 hover:bg-blue-700 text-white font-bold
py-2 px-4 rounded-full block m-auto mt-4 mb-4">Click
                me! For a demo</button>
```

```
            <p style="text-align: center">{{ Prediction }}</p>
        </div>
    </div>

</body>

</html>
```

REGISTER PAGE :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title> Register </title>
  <style>
    * {box-sizing: border-box}


.container {
  padding: 16px;
}


input[type=text], input[type=password] {
  width: 100%;
  padding: 15px;
  margin: 5px 0 22px 0;
  display: inline-block;
  border: none;
  background: #f1f1f1;
}

input[type=text]:focus, input[type=password]:focus {
  background-color: #ddd;
  outline: none;
}

hr {
  border: 1px solid #f1f1f1;
  margin-bottom: 25px;
}

.registerbtn {
  background-color: #04AA6D;
  color: white;
  padding: 16px 20px;
  margin: 8px 0;
  border: none;
  cursor: pointer;
  width: 100%;
  opacity: 0.9;
}

.registerbtn:hover {
  opacity:1;
}
```

```
a {
  color: red;
}

.signin {
  background-color: #f1f1f1;
  text-align: center;
}

  </style>
</head>
<body>
<form>
  <div class="container">
    <h1>Register</h1>
    <p>Please fill in this form to create an account.</p>
    <hr>

    <label for="email"><b>Email</b></label>
    <input type="text" placeholder="Enter Email" name="email" id="email"
required>

    <label for="psw"><b>Password</b></label>
    <input type="password" placeholder="Enter Password" name="psw" id="psw"
required>

    <label for="psw-repeat"><b>Repeat Password</b></label>
    <input type="password" placeholder="Reenter Password" name="psw-repeat"
id="psw-repeat" required>
    <hr>


    <p><a href="/index" class="registerbtn">Register</a>.</p>
  </div>

  <div class="container signin">
    <p>Already a member? <a href="/login">Sign in</a>.</p>
  </div>
</form>
</body>
</html>
```

RESULT PAGE :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
<h1>result</h1>
</body>
</html>
```

GITHUB LINK :

https://github.com/IBM-EPBL/IBM-Project-34273-1660233691

REFERENCES:-

[1]Foresti, Gian Luca, Petri Mähönen, and Carlo S. Regazzoni, eds. Multimedia video-based surveillance systems: Requirements, Issues and Solutions. Vol. 573. Springer Science & Business Media, 2012.

[2]Jones, Graeme A., Nikos Paragios, and Carlo S. Regazzoni, eds. Video-based surveillance systems: computer vision and distributed processing. Springer Science & Business Media, 2012.

[3]Conde, Cristina, et al. "HoGG: Gabor and HoG-based human detection for surveillance in non-controlled environments." Neurocomputing 100 (2013): 19-30.

[4]Wang, Xiaogang. "Intelligent multi-camera video surveillance: A review." Pattern recognition letters 34.1 (2013): 3-19.

[5]Gudyś, Adam, et al. "Tracking people in video sequences by clustering feature motion paths." Computer Vision and Graphics. Springer International Publishing, 2014. 236-245.

[6]Vezzani, Roberto, Davide Baltieri, and Rita Cucchiara. "People reidentification in surveillance and forensics: A survey." ACM Computing Surveys (CSUR) 46.2 (2013): 29.

[7]Bierens, Joost, and Andrea Scapigliati. "Drowning in swimming pools." Microchemical journal 113 (2014):

53-58.

\[8]Zhang, Chi, Xiaoguang Li, and Fei Lei. "A Novel Camera-Based Drowning Detection Algorithm." Advances in Image and Graphics Technologies. Springer Berlin Heidelberg, 2015. 224-233.

[9]Fei, Lei, Wang Xueli, and Chen Dongsheng. "Drowning Detection Based on Background Subtraction." Embedded Software and Systems, 2009. ICESS'09. International Conference on. IEEE, 2009.

[10]Kharrat, Mohamed, et al. "Near drowning pattern detection using neural network and pressure information measured at swimmer's head level." Proceedings of the Seventh ACM International Conference on Underwater Networks and Systems. ACM, 2012.

[11]Kam, Alvin H., Wenmiao Lu, and Wei-Yun Yau. "A video-based drowning detection system." Computer Vision—ECCV 2002. Springer Berlin Heidelberg, 2002. 297-311.

[12]Chan, Kwok Leung. "Detection of swimmer using dense optical flow motion map and intensity information." Machine vision and applications 24.1 (2013): 75-101.

[13]Pleština, Vladimir, and Vladan Papić. "Features analysis for tracking players in water polo." 16th International Conference on Automatic Control, Modelling & Simulation. 2014.

[14]Wang, Hua, and Sing Kiong Nguang. "Intelligent and Comprehensive Monitoring System for Swimming Pool." International Journal of Sensors Wireless Communications and Control 3.2 (2013): 85-94.