Virtual eye – Life guard for swimming pools to detect active drowning

Submitted by:

Fredrick Dani (312319205039)
Ghokulesh (312319205040)
Arun R (312319205019)
Kailash V (312319205064)

TABLE OF CONTENTS

1.INTRODUCTION

1.1 PROJECT OVERVIEW

Designed for whom has to guarantee every day the safety in public and intensive-use pools,VirtualEye LifeGuard detects potential drowning and promptly notifies you.It features the latest artificial intelligence technology and adapts to the need of the users.It's the ultimate drowning detection system for those who demand the ultimate in safety.Virtual Eye works like an “extra lifeguard” under the water of your pool. Our object recognition software tracks the movements of all swimmers in a pool. And in the event of a serious drowning incident, Virtual Eye will provide an alarm to pool lifeguards. Thiswill help lifeguards improve their reaction-time, as they initiate a rescue.The live video stream from our underwater cameras is automatically monitored by our “State of the art” object recognition software. The Virtual Eye detects a swimmer in distress on the bottom of the pool, it will raise a radio alarm to pool lifeguards and a visual alarm to our Monitoring and Control Station. Lifeguards can visually assess the developing situation within seconds of the event first occurring, and initiate their rescue procedure when necessary.Virtual Eye is a leader in the pool safety market and offers systems for drowning and prevention.This Virtual Eye provides safety to the users.The system is not designed to replace a lifeguard or other human monitor, but to act as an additional tool. It helps the lifeguard to detect the underwater situation where they can’t easily observe.

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 systems we can devise an underwater pool safety system that reduces the risk of drowning. Usually, such systems can be developed by installing 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 analyzes the position of swimmers to assess the probability of drowning, if it is higher than an alert will be generated to attract lifeguards' attention.

2. LITERATURE SURVEY

2.1 <u>EXISTING PROBLEM</u>

Existing drowning detection systems can’t detect during night time and when the pool is crowded.Current systems make use of only the single input string for solving the problem that is to detect any drowning.It only detects humans,but does not any animals.Current system needs high resolution input string for accurate detection.An another drawback is the existing system can’t predict any accidents.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.

2.2 <u>REFERENCES</u>

Paper 1: The Swimmers Motion Detection Using Improved VIBE Algorithm

Authors: Muhammad Aftab Hayat, Goutian Yang, Atif Iqbal, Adeel Saleem, Adil hussain, Muhammad Mateen.

This paper proposed a novel method for drowning person detection in the swimming pool using video images. For background extraction and to update the exact motion area from the whole video using frame by frame difference, the vibe algorithm is used. Static and dynamic features are detected to recognize the normal swimmer and drowning person. The present invention discloses video based swimming pools drowning event detection method. In the detection process Time of map(Tom), the method is used to improve the traditional VIBE result. The sequence of video images of the swimming pool is collected in real-time by using a camera installed above the water surface, which mainly includes three steps of swimmers detection, swimmers tracking and drowning person behavior analysis. In the aspect of swimmer detection, an improved VIBE swimmer detection algorithm is proposed, and the algorithm is used to determine the swimmer's position. The swimmer tracking and particle filter based on the color distribution model which is combined with the nearest neighbor data association algorithm to achieve tracking of multiple swimmers. In the analysis of drowning behavior, three characteristics of drowning behavior are proposed to determine whether the swimmer is drowning. The invention can monitor the swimming pool in real-time through the camera installed above the water surface in a real public swimming place, and automatically detect the drowning person, which has great engineering application value.

Paper 2: An Automatic Video-Based Drowning Detection System for Swimming Pools using Active Contours

Authors: Nasrin Salehi and Maryam Keyvanara,Seyed Amirhassan Monadjemmi.

Safety in swimming pools is a crucial issue. In this paper, a real time drowning detection method based on HSV color space analysis is presented which uses prior knowledge of the video sequences to set the best values for the color channels. Our method uses a HSV thresholding mechanism along with Contour detection to detect the region of interest in each frame of video sequences. The presented software can detect drowning persons in indoor swimming pools and sends an alarm to the lifeguard rescues if the previously detected person is missing for a specific amount of time. The presented algorithm for this system is tested on several video sequences recorded in swimming pools in real conditions and the results are of high accuracy with a high capability of tracking individuals in real time. According to the evaluation results, the number of false alarms generated by the system is minimal and the maximum alarm delay reported by the system is 2.6 sec which can be relatively reliable compared to the acceptable time for rescue and resuscitation.

Paper 3: Design of a drowning rescue alert system

Authors: Samuel Ndueso John, Ukpabio Imelda Godswill, Omoruyi Osemwegie, Godfrey Onyiagha, Etinosa Noma-Osaghae, and Kennedy Okokpujie.

Dating back in time, drowning has been a significant ground for death worldwide; it accounts for the third cause of unplanned death globally, with about 1.2 million cases yearly. Characteristically it affects swimmers, accident victims, children and recreational seeking individuals. Although there have been various provisions put in place from drowning in some countries, it still accounts for the primary cause of unplanned death. Eradication rather than cure has been able to minimize the number of individuals who drown generally, except in developing nations, who lack adequate educational facilities and enforcement of safety measures on the dangers of drowning, thereby making the burden of drowning to escalate. The proposed drowning rescue system aims to curb deaths from drowning by observing the rise and fall of the heart rate and blood pressure of a swimmer or non-swimmer in water and if endangered, sends signals from the wearable device attached to the wrist of the victim who maybe undergoing a near drowning experience to the receiver or rescuer who could be a lifeguard, parent or neighbor, in order to enable the rescuer render immediate help.

Paper 4: Automated and Intelligent System for Monitoring Swimming Pool Safety Based on the IoT and Transfer Learning

Authors: Aziz Alotaibi.

Recently, integrating the Internet of Things (IoT) and computer vision has been utilized in swimming pool automated surveillance systems. Several studies have been proposed to overcome off-time surveillance drowning incidents based on using a sequence of videos to track human motion and position. This paper proposes an efficient and reliable detection system that utilizes a single image to detect and classify drowning objects, to prevent drowning incidents. The proposed system utilizes the IoT and transfer learning to provide an intelligent and automated solution for off-time monitoring swimming pool safety. In addition, a specialized transfer-learning-based model utilizing a model pre trained on “ImageNet”, which can extract the most useful and complex features of the captured image to differentiate between humans, animals, and other objects, has been proposed. The proposed system aims to reduce human intervention by processing and sending the classification results to the owner’s mobile device. The performance of the specialized model is evaluated by using a prototype experiment that achieves higher accuracy, sensitivity, and precision, as compared to other deep learning algorithms.

Paper 5: Computer Vision Enabled Drowning Detection System

Authors: Upulie Handalage, Nisansali Nikapotha, Chanaka Subasinghe, Tereen Prasanga, Thusithanjana Thilakarthna, Dharshana Kasthurirathna

Safety is paramount in all swimming pools. The current systems expected to address the problem of ensuring safety at swimming pools have significant problems due to their technical aspects, such as underwater cameras and methodological aspects such as the need for human intervention in the rescue mission. The use of an automated visual-based monitoring system can help to reduce drownings and assure pool safety effectively. This study introduces a revolutionary technology that identifies drowning victims in a minimum amount of time and dispatches an automated drone to save them. Using convolutional neural network (CNN) models, it can detect a drowning person in three stages. Whenever such a situation like this is detected, the inflatable tube-mounted self-driven drone will go on a rescue mission, sounding an alarm to inform the nearby lifeguards. The system also keeps an eye out for potentially dangerous actions that could result in drowning. This system’s ability to save a drowning victim in under a minute has been demonstrated in prototype experiments' performance evaluations.

Paper 6: Video Based Drowning Detection System

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

At present, there are swimming pools in every part of the world. Most of the swimming pool accidents or incidents occur due to improper security. Therefore, Accidental deaths in swimming pools are actually increasing. So, Video based drowning detection system is designed in this article. The proposed system structure comprises raspberry pi (Single Board Computer) equipped with a USB camera for taking the live feed from the pool area. The system also covers the alerting phenomena using a buzzer so that necessary actions are taken intermittently without any delay. The working structure starts from the raspberry pi with image processing for video feed intake, deep learning for activity recognition and finally GPIO system for alerting and short message service.

Paper 7: A Survey of Drowning Detection Techniques

Authors: Abdelaziz M. Shehata, Eslam M. Mohamed, Khaled L. Salem, Ahmed M. Mohamed, Mustafa Abdul Salam, Mennatullah M. Gamil.

Drowning is one cause of unintentional injury death worldwide, which made it a public health problem. It gained the interest of many engineers to create drowning detection systems by applying different technologies. This paper reviews different methods used for drowning detection in swimming pools, that applied the concepts of image processing, accelerometer, pulse and pressure sensing and LASER-LDR techniques. The reviews discussed the process, reliability and goals of each system. By surveying this we represented a comparison between the provided systems. A further discussion of the future challenges facing these systems is also mentioned with ideas to overcome them.

Paper 8: A Survey of Drowning Detection Techniques

Authors: Abdelaziz M. Shehata, Eslam M. Mohamed, Khaled L. Salem, Ahmed M. Mohamed, Mustafa Abdul Salam, Mennatullah M. Gamil.

Drowning is one cause of unintentional injury death worldwide, which made it a public health problem. It gained the interest of many engineers to create drowning detection systems by applying different technologies. This paper reviews different methods used for drowning detection in swimming pools, that applied the concepts of image processing, accelerometer, pulse and pressure sensing and LASER-LDR techniques. The reviews discussed the process, reliability and goals of each system. By surveying this we represented a comparison between the provided systems. A further discussion of the future challenges facing these systems is also mentioned with ideas to overcome them.

Paper 9: An Underwater Sonar-Based Drowning Detection System

Authors: Guoliang Xing, Zhenvu Yan, Haozheng Hou, Lixing He.

Drowning is a major cause of unintentional deaths in swimming pools. Most swimming pools hire lifeguards for continuous surveillance, which is labor-intensive and hence unfeasible for small private pools. The existing unmanned surveillance solutions like camera arrays require non-trivial installations, only work in certain conditions (e.g., with adequate ambient lighting), or raise privacy concerns. This demo presents SwimSonar, the first practical drowning detection system based on underwater sonar. SwimSonar employs an active ultrasonic sonar and features a novel sonar scanning strategy that balances the time and accuracy. Lastly, SwimSonar leverages a deep neural network for accurate drowning detection. Our experiments in real swimming pools show that the system achieves 88 % classification accuracy with a scan time of 1.5 seconds.

Paper 10: Drowning Detection Based on Background Subtraction

Authors: Chen Dongsheng, Wang Xueli, Lei Fei.

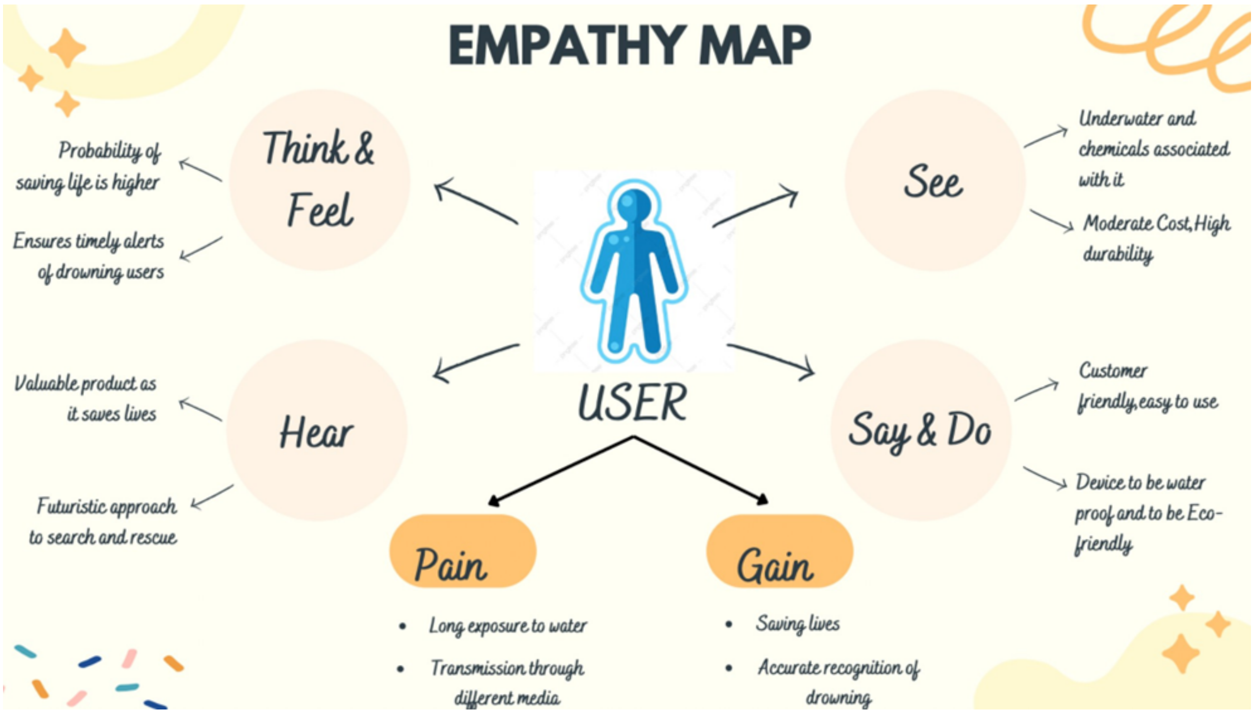
The main research subject in this paper is swimmer detection for visual surveillance of pools. A drowning detection method based on background subtraction is presented in this paper. The consecutive sequence of visual surveillance was obtained by the fixed camera installed in the pool wall. Each pixel is described by a Gaussian mixed model, set up by a self-adapted background model and updated timely. When the foreground objects are separated, for getting good results, the shadows and noises must be removed. The experiment's results show that this method is effective to detect the drowners and eliminate the shadows.

2.3 PROBLEM STATEMENT DEFINITION

An AI powered detector to be deployed at swimming pools.We are trying to minimize the chances of people drowning in the water by recognizing the possible actions people make at the time of distress or panic and sending an alert to the control room for taking suitable measures to save them. The barriers here are to differentiate between swimming and drowning.The major that leads to lots of drowning accidents are careless lifeguards and swimming pool authorities.Most time carelessness can be attributed to the ambiguity whether someone is drowning or not.

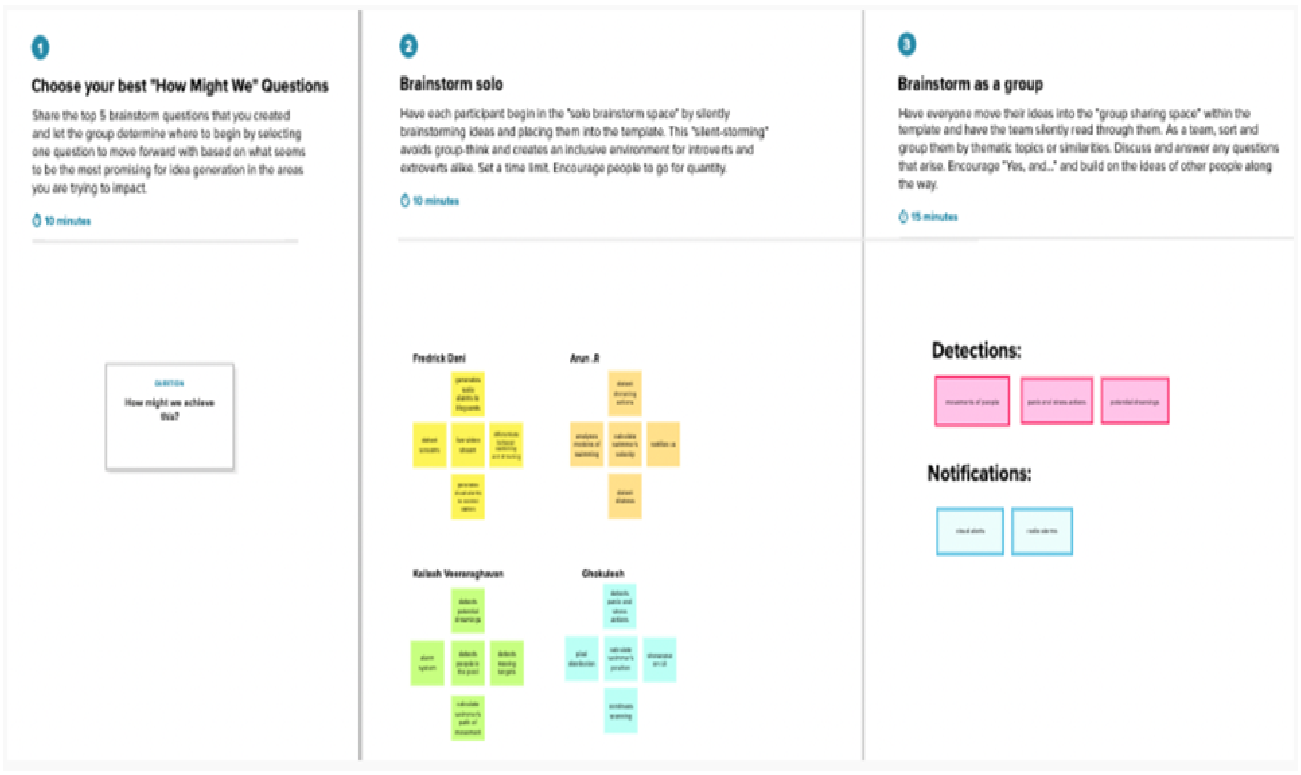
3. IDEATION & PROPOSED SOLUTION

3.1 <u>EMPATHY MAP CANVAS</u>



EMPATHY MAP
Think & Feel
Probability of saving life is higher
Ensures timely alerts of drowning users
See
Underwater and chemicals associated with it
Moderate Cost, High durability
USER
Hear
Valuable product as it saves lives
Futuristic approach to search and rescue
Say & Do
Customer friendly, easy to use
Device to be water proof and to be Eco-friendly
Pain
Long exposure to water
Transmission through different media
Gain
Saving lives
Accurate recognition of drowning

3.2 IDEATION AND BRAINSTORMING


1

Choose your best "How Might We" Questions

Share the top 5 brainstorm questions that you created and let the group determine where to begin by selecting one question to move forward with based on what seems to be the most promising for idea generation in the areas you are trying to impact.

10 minutes

QUESTION

How might we achieve this?

2

Brainstorm solo

Have each participant begin in the "solo brainstorm space" by silently brainstorming ideas and placing them into the template. This "silent-storming" avoids group-think and creates an inclusive environment for introverts and extroverts alike. Set a time limit. Encourage people to go for quantity.

10 minutes

3

Brainstorm as a group

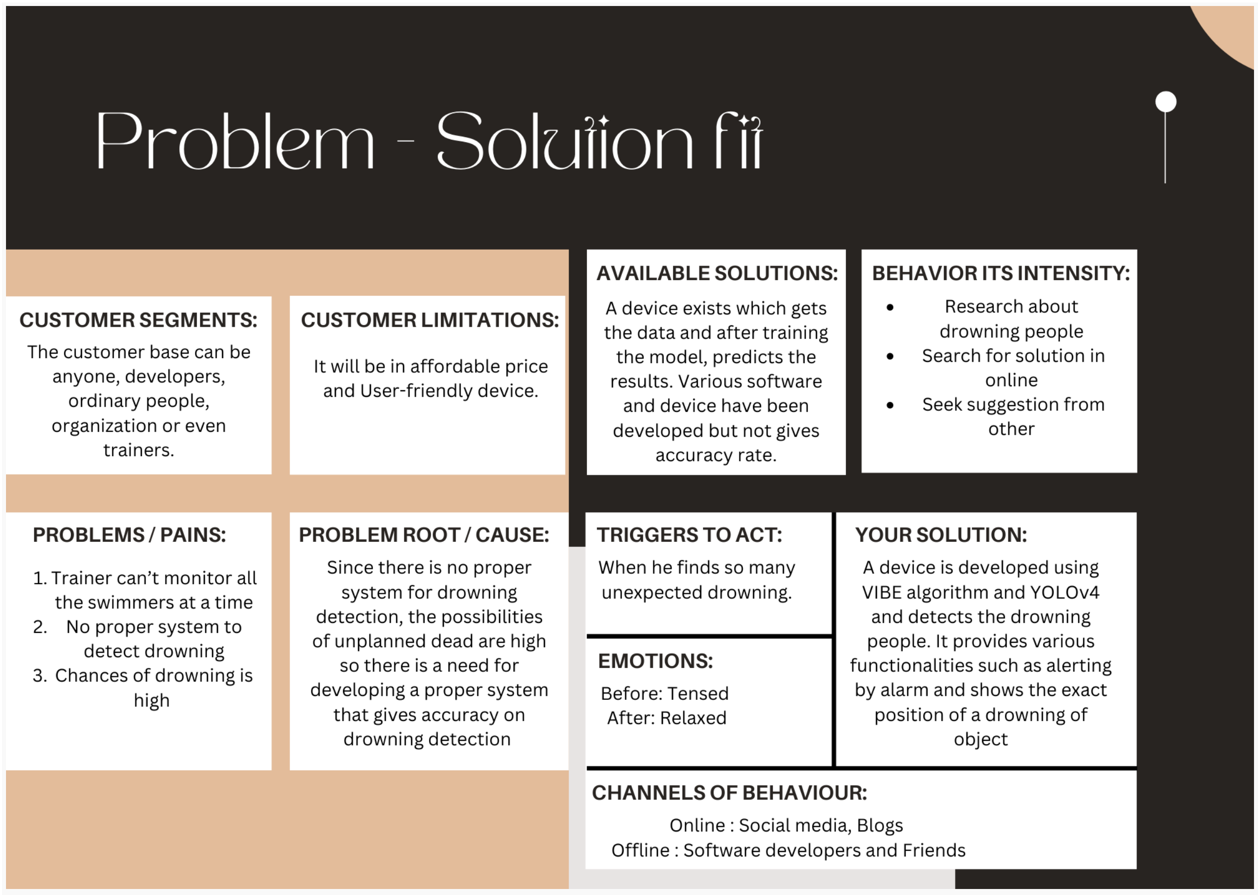
Have everyone move their ideas into the "group sharing space" within the template and have the team silently read through them. As a team, sort and group them by thematic topics or similarities. Discuss and answer any questions that arise. Encourage "Yes, and..." and build on the ideas of other people along the way.

15 minutes

3.3 PROPOSED SOLUTION

S.No	Parameter	Description
1.	Problem Statement (to be solved)	People are visiting swimming pools to Practice or to learn swimming .There is a possibility of someone drowning as they are new to these activities. So,to detect the active drowning of theperson, we have designed a "VIRTUAL EYE" program which is installed in the security camera available in the pool and it is connected with alarm and thus alertingthe rescue team about the drowning. Thus,a system is to be implemented along the swimming pools to save human life.

3.4 PROBLEM - FIT SOLUTION



Problem - Solution fit

CUSTOMER SEGMENTS:

The customer base can be anyone, developers, ordinary people, organization or even trainers.

CUSTOMER LIMITATIONS:

It will be in affordable price and User-friendly device.

AVAILABLE SOLUTIONS:

A device exists which gets the data and after training the model, predicts the results. Various software and device have been developed but not gives accuracy rate.

BEHAVIOR ITS INTENSITY:

- Research about drowning people
- Search for solution in online
- Seek suggestion from other

PROBLEMS / PAINS:

1. Trainer can't monitor all the swimmers at a time
2. No proper system to detect drowning
3. Chances of drowning is high

PROBLEM ROOT / CAUSE:

Since there is no proper system for drowning detection, the possibilities of unplanned dead are high so there is a need for developing a proper system that gives accuracy on drowning detection

TRIGGERS TO ACT:

When he finds so many unexpected drowning.

EMOTIONS:

Before: Tensed
After: Relaxed

YOUR SOLUTION:

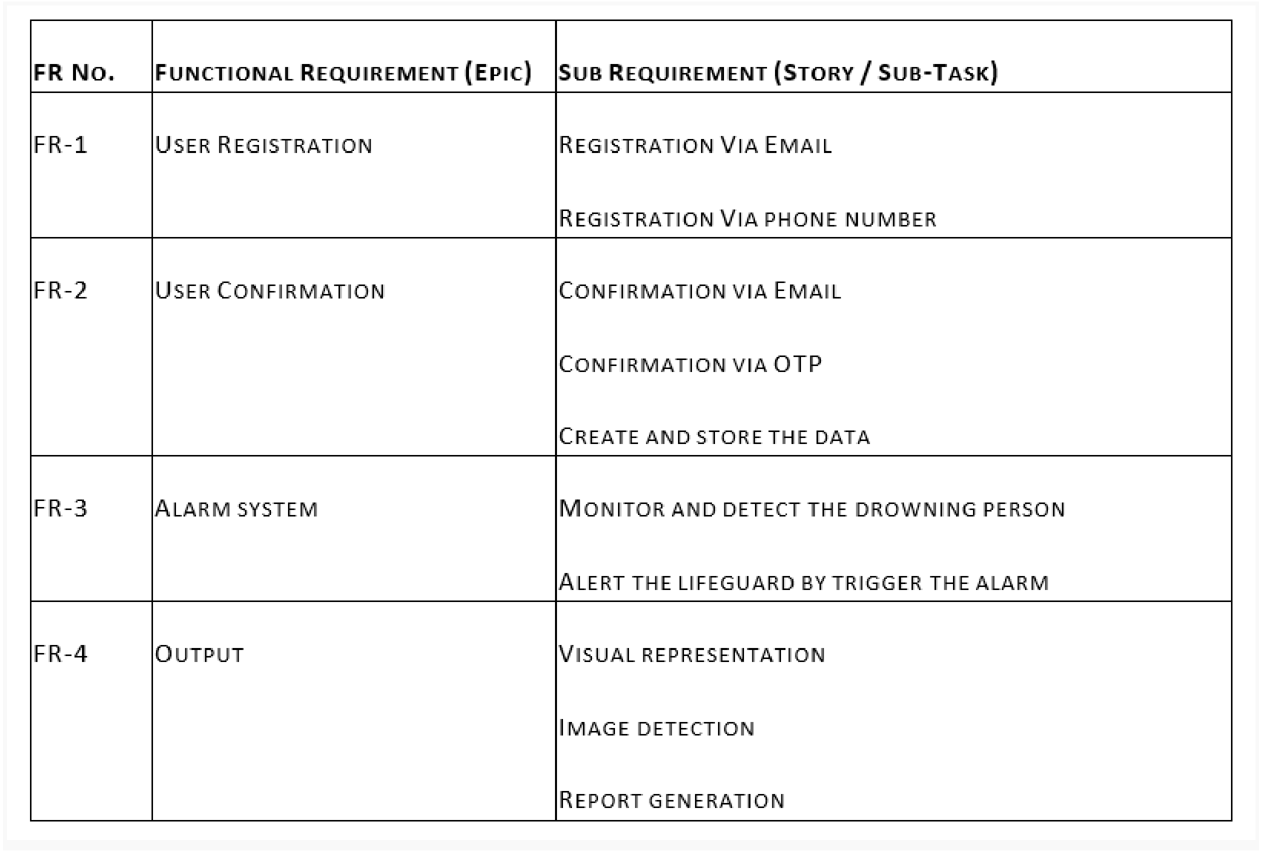
A device is developed using VIBE algorithm and YOLOv4 and detects the drowning people. It provides various functionalities such as alerting by alarm and shows the exact position of a drowning of object

CHANNELS OF BEHAVIOUR:

Online : Social media, Blogs
Offline : Software developers and Friends

4. REQUIREMENT ANALYSIS

4.1 FUNCTIONAL REQUIREMENTS



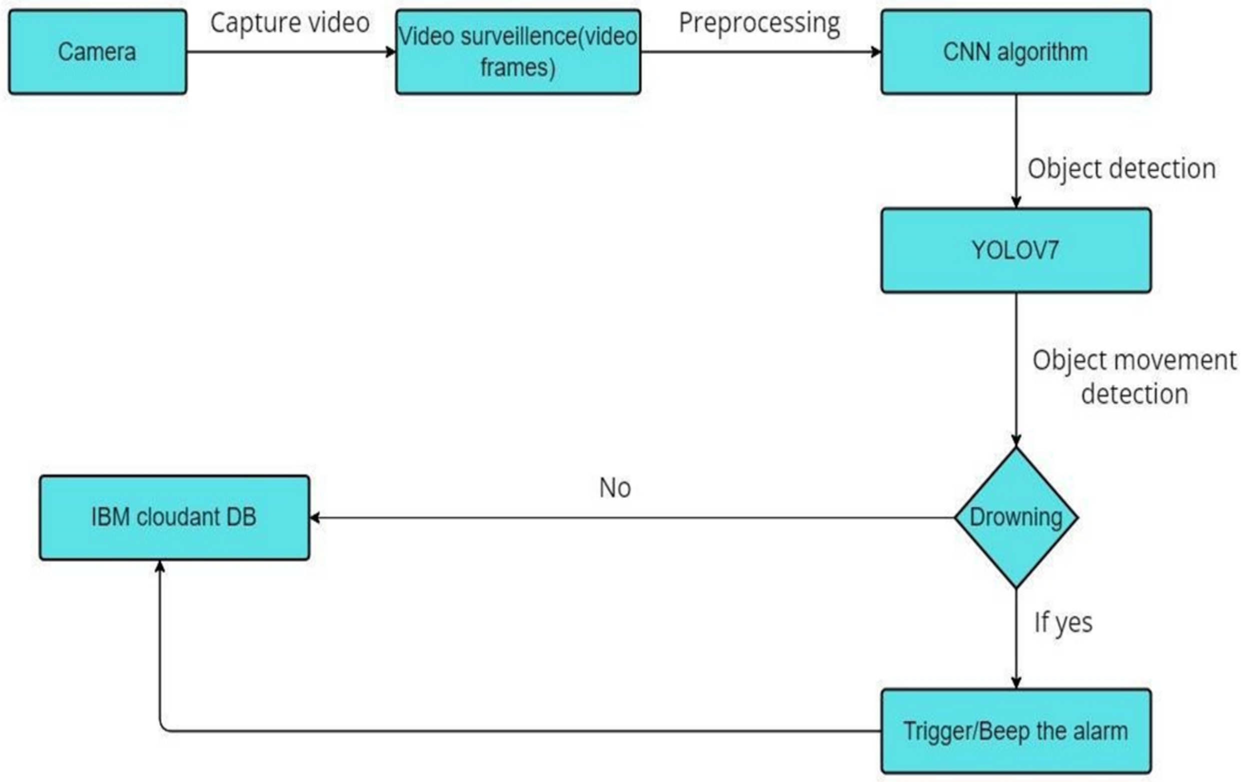
FR No.	Functional Requirement (Epic)	Sub Requirement (Story / Sub-Task)
FR-1	User Registration	Registration Via Email Registration Via phone number
FR-2	User Confirmation	Confirmation via Email Confirmation via OTP Create and store the data
FR-3	Alarm system	Monitor and detect the drowning person Alert the lifeguard by trigger the alarm
FR-4	Output	Visual representation Image detection Report generation

4.2 NON FUNCTIONAL REQUIREMENTS

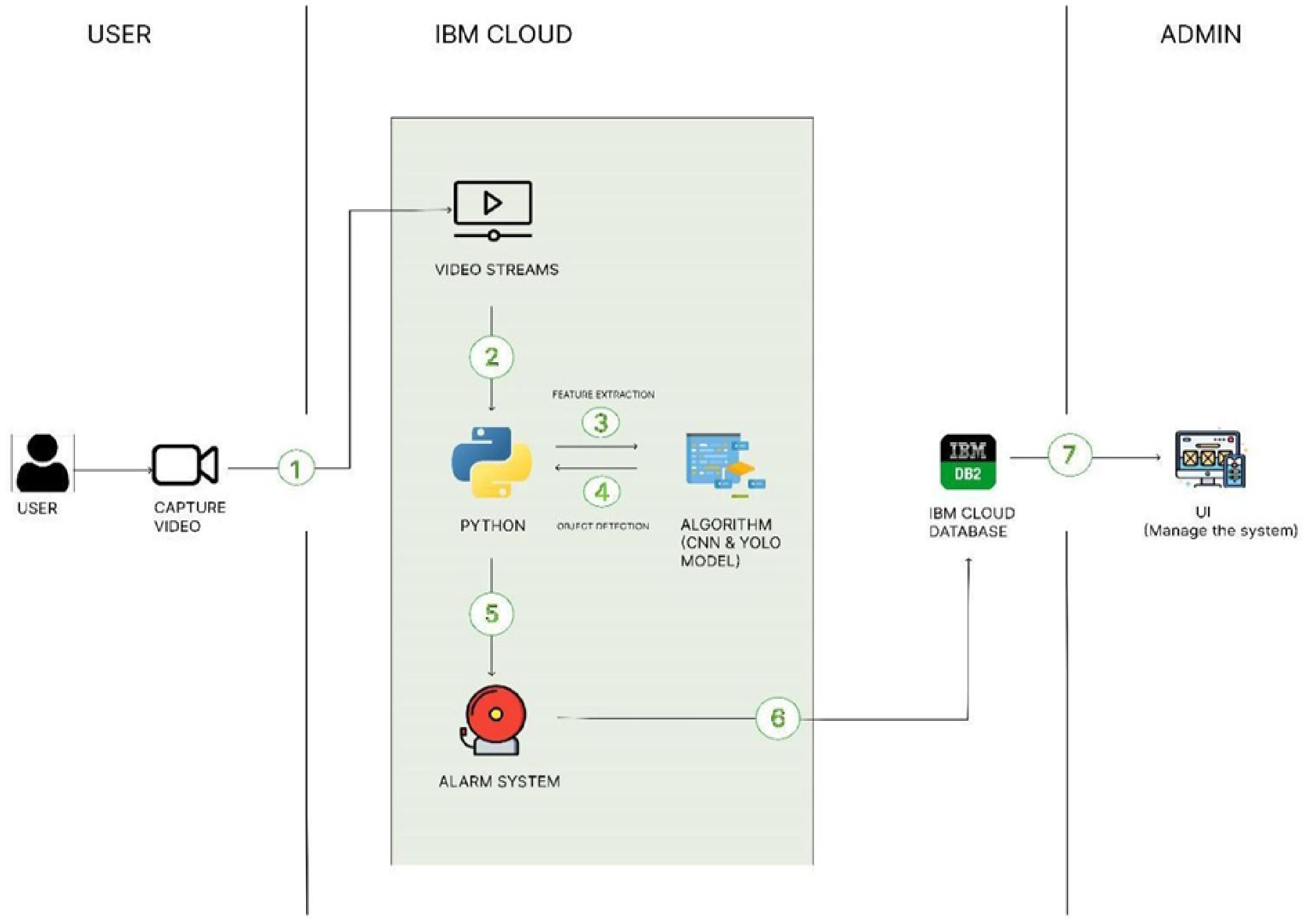
FR No.	**Non-Functional Requirement**	**Description**
NFR-1	**Usability**	Eco – Friendly.
NFR-2	**Security**	Observing each and every body movement of the swimmers.
NFR-3	**Reliability**	Suitable for all the swimming pools.

5. PROJECT DESIGN

5.1 DATAFLOW DIAGRAM



5.2 SOLUTION & TECHNICAL ARCHITECTURE



5.3 USER STORIES

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
	Registration	USN-1	I can be able to Register with My phone number but only one registration for single number.	1	High	Ghokulesh T
		USN-2	I will receive confirmation OTP on the registered phone number.	2	Medium	Arun R
		USN-3	I can also register Through Gmail	2	Low	Kailash V
Sprint-1	Login	USN-4	I can login into the application by entering phone number & password.	1	High	Arun R
		USN-5	In prediction page, the data uploaded will help the user to detect the drowning movements	2	Medium	Fredrik Dani

6. PROJECT PLANNING & SCHEDULING

6.1 SPRINT PLANNING & ESTIMATION

S.NO	MILESTONE	DESCRIPTION	DURATION
1	Prerequisites	They are the items that are required before execution of every phase	1WEEK
2	Create & Configure IBM cloud services	It provides solutions that enable higher levels security, management and having methods for rapid delivery for running critical workloads.	2WEEK
3	Develop the pythonscript	A Python script is a set of commands included in a file that is intended to be run similarly to a program.The concept is that the file will be run or performed from the command line or from within a Python interactive shell	1WEEK

4	Develop web application	A web application (or web app) is application softwarethat runs in a web browser	3WEEK
5	Ideation phase	Ideation is the process where you come up with ideas and solutions through Sketching, Prototyping, Brainstorming, Brain writing, Worst Possible Idea, and other ideation techniques.	1WEEK

6	Project design phases	Project design is an early phase of a project where the project's key features, structure, criteria for success, and major deliverables are planned out. The aim is to develop one or more designsthat can be used to achieve the desired project goals.	2WEEK

7	Project planningphase	Project planning is a discipline addressing how to complete a project in a certain timeframe, usually with defined stages and designated resources. One view of project planning divides the activity into these steps: Identifying deliverables,scheduling, planning tasks	2WEEK

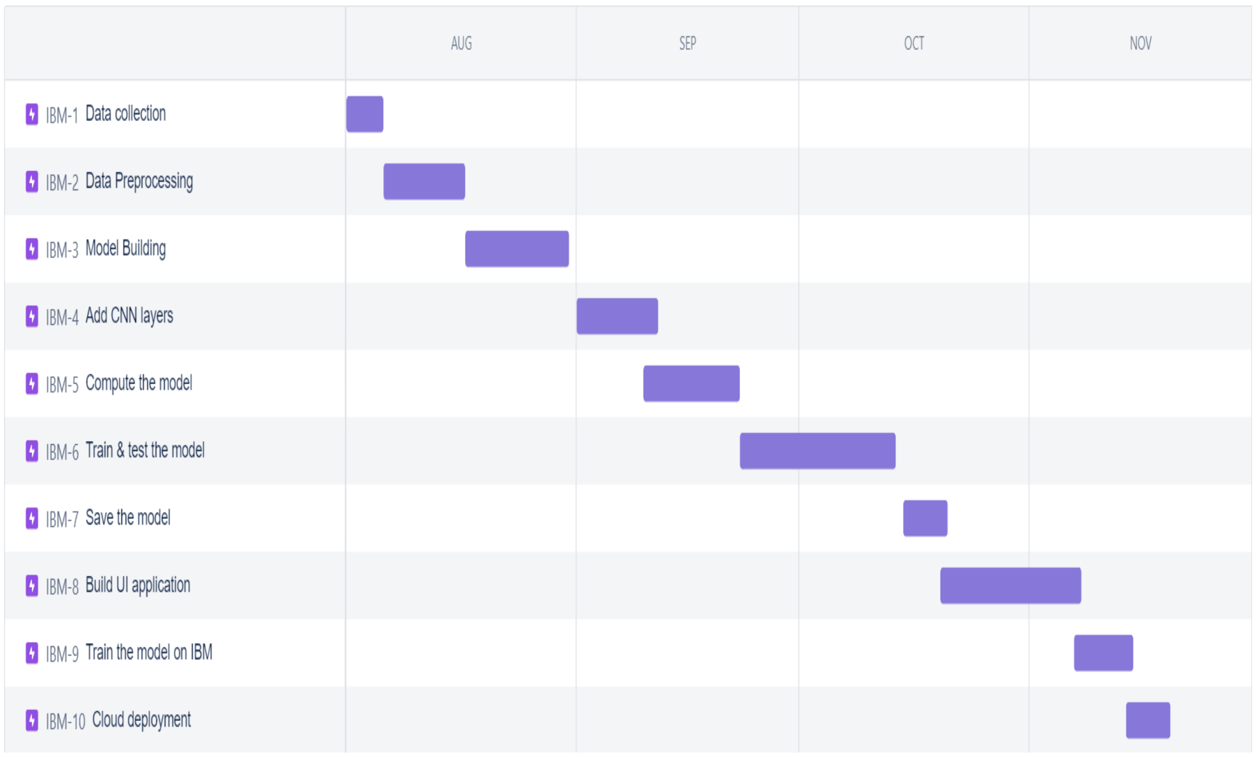
8	PROJECT DEVELOPMENT PHASE	PROJECT DEVELOPMENTIS THE PROCESS OF PLANNING AND ALLOCATING RESOURCES TO FULLY DEVELOP A PROJECT OR PRODUCT	4WEEK

6.2 SPRINT DELIVERY SCHEDULE

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	20	6 Days	24 Oct 2022S	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

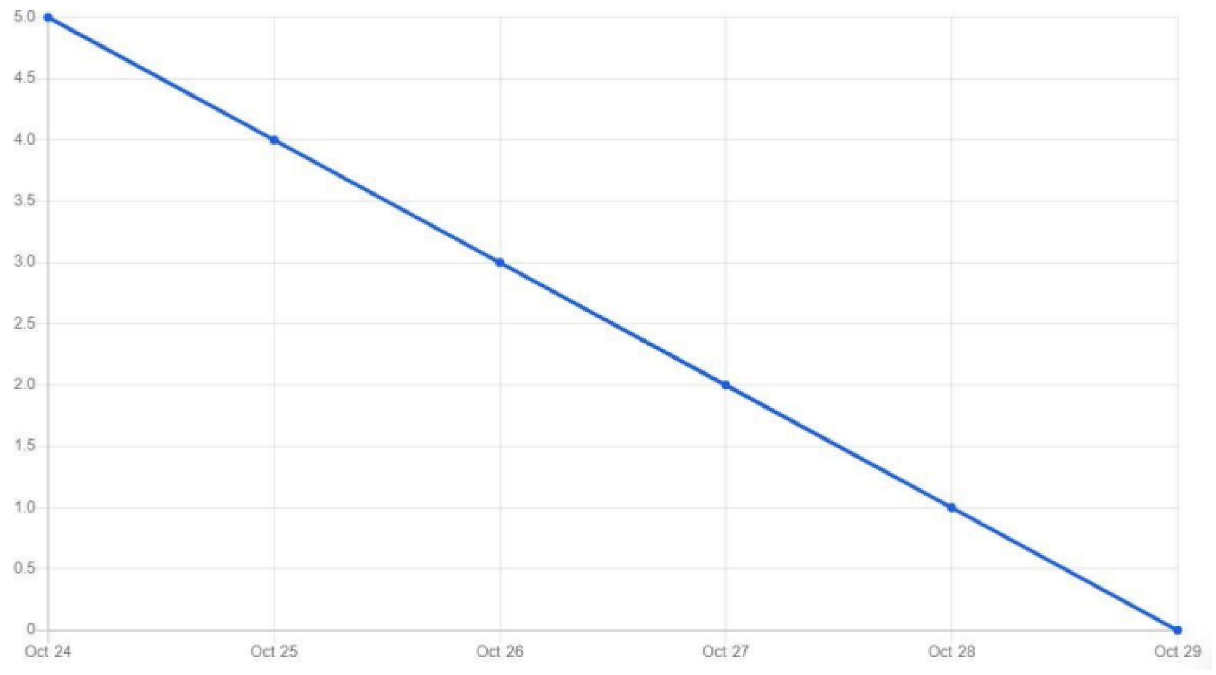
6.3 REPORTS FROM JIRA

Roadmap:

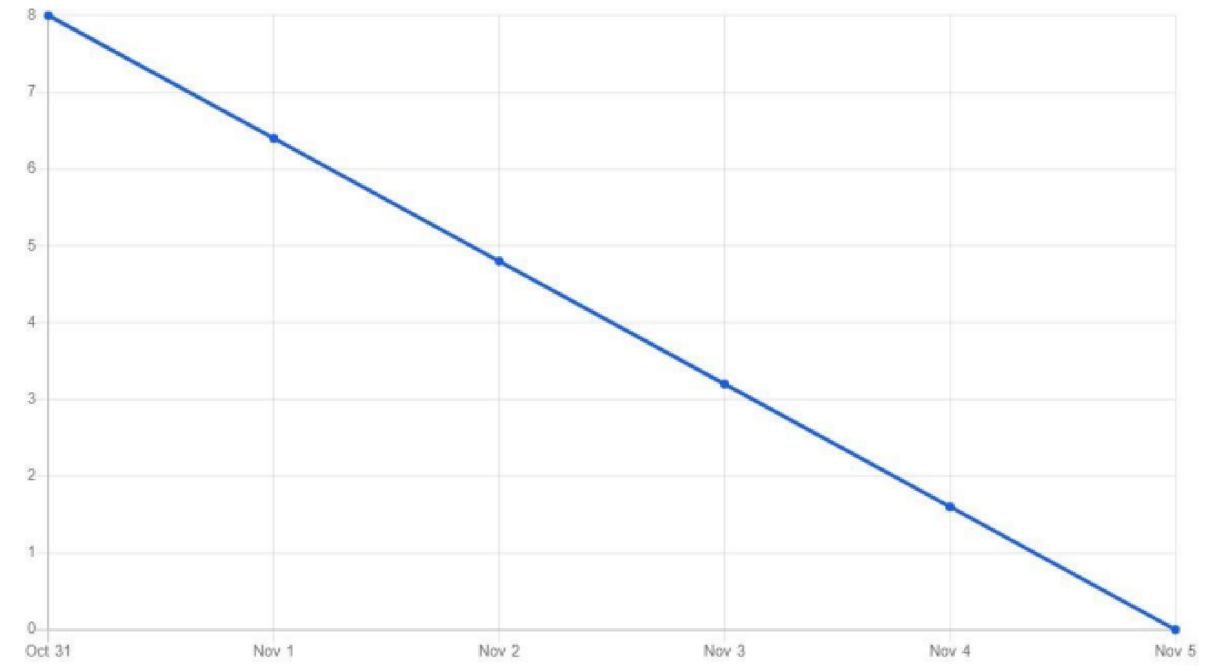


Burndown chart:

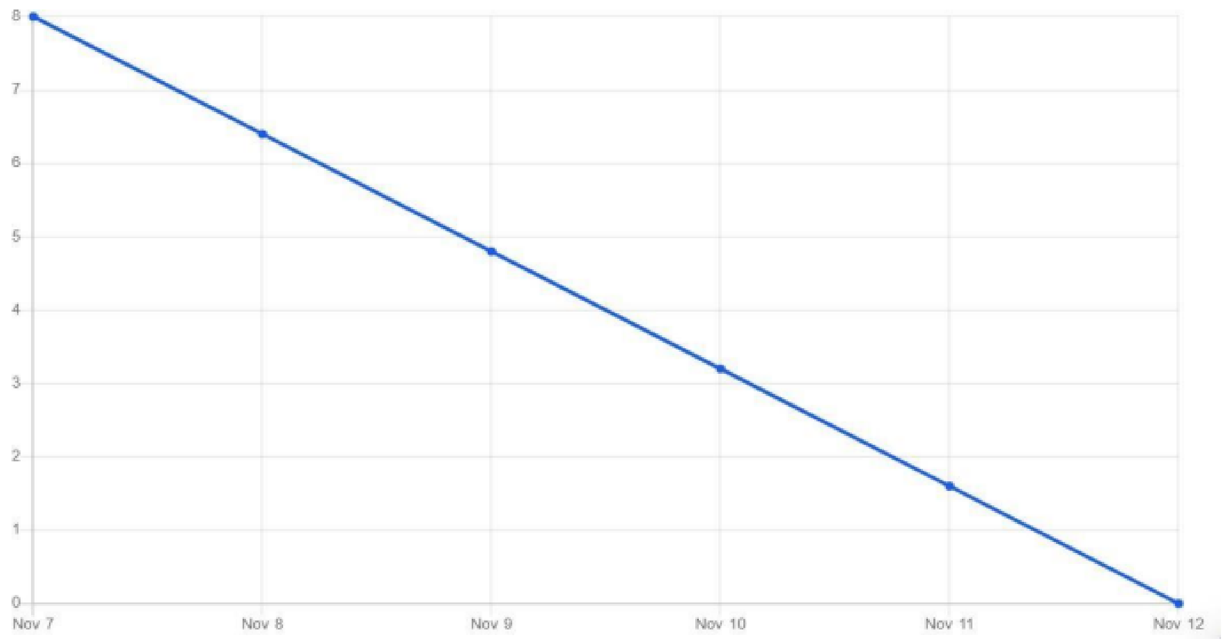
Sprint 1:



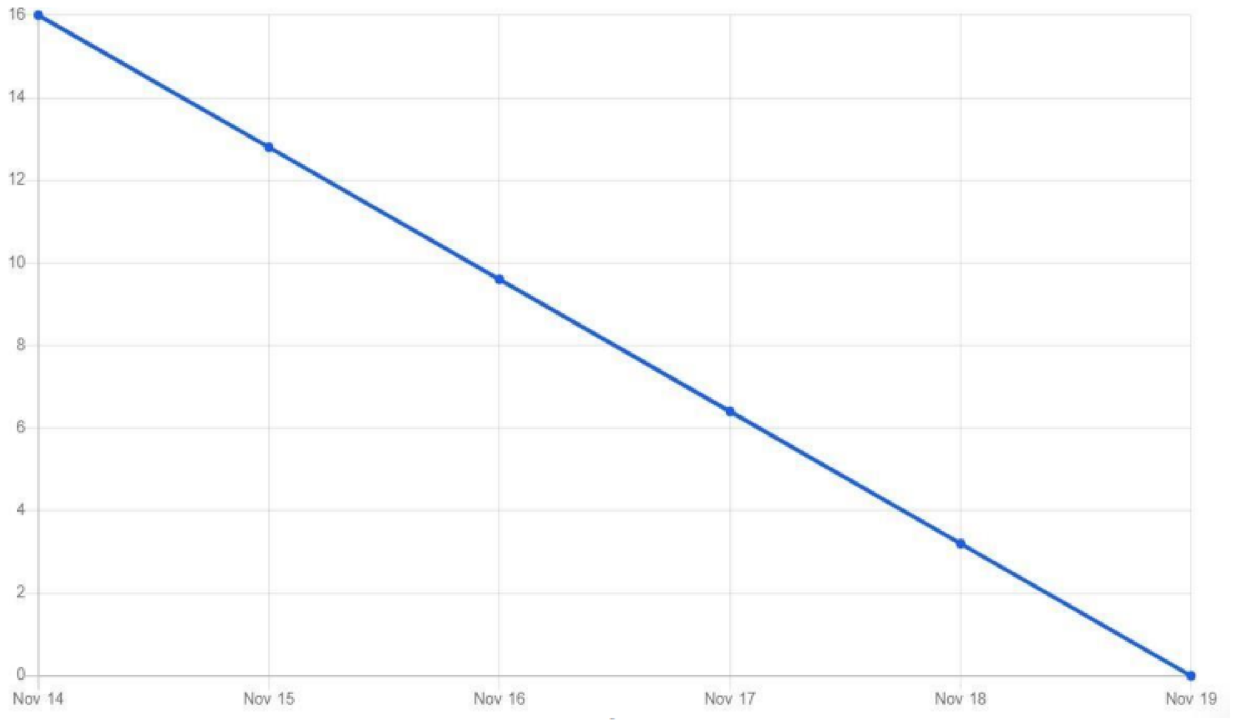
Sprint 2:



Sprint 3:



Sprint 4:



7.CODING AND SOLUTIONING

7.1 <u>FEATURE 1</u>

Objectdetection.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.tx
t'
```

```
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 - 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.cf
g'
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 = indices[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
```

Explanation:

YOLOv3("You Look Only Once", version 3),this is a neural network capable of detecting what is in an image and where stuff is, in one pass. It gives the bounding boxes around the detected objects, and it can detect multiple objects at a time. YOLOv3 is trained over Darknet which is a framework to train neural networks. It is open source and written in C/CUDA and serves as the basis for YOLO, meaning it sets the architecture of the network. The network structure basically looks like a normal CNN, with convolutional and max pooling layers, followed by 2 fully connected layers in the end. The input image is divided into an S x S grid of cells. For each object that is present on the image, one grid cell is said to be "responsible" for predicting it. That is the cell where the center of the object falls into. Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence). The (x, y) coordinates represent the center of the box, relative to the grid cell location (if the center of the box does not fall inside the grid cell, then this cell is not responsible for it). These coordinates are normalized to fall between 0 and 1. The (w, h) box dimensions are also normalized to [0, 1], relative to the image size. Finally, the confidence reflects the presence or absence of an object of any class. If no object exists in that cell, the confidence score is zero. Otherwise the confidence score equals the intersection over union (IOU) between the predicted box and the ground truth. The final output of the objects-detection module is thereby a S x S x (B * 5 +C) tensor. All of the detected objects are forwarded into a Drowning-Alerts phase for a further analysis.

7.2 FEATURE 2

Drowndetection.py

```
import cvlib as cv
from cvlib.Objectiondetection import detect_common_objects
import cv2
import time
import numpy as np
webcam = cv2.VideoCapture(0)
if not webcam.isOpened():
print("Could not open webcam")
exit()
t0 = time.time()
centre0 = np.zeros(2)
isDrowning = False
while webcam.isOpened():
status, frame = webcam.read()
if not status:
print("Could not read frame")
```

```
exit()
bbox, label, conf = cv.detect_common_objects(frame)
if (len(bbox) > 0):
bbox0 = bbox[0]
centre = [0, 0]
centre = [(bbox0[0] + bbox0[2]) / 2, (bbox0[1] + bbox0[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, 'centre:', centre, 'centre0:', centre0)
print('Is he drowning: ', isDrowning)
centre0 = centre
out = draw_bbox(frame, bbox, label, conf, isDrowning)
cv2.imshow("Real-time object detection", out)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
webcam.release()
cv2.destroyAllWindows()
```

Explanation:

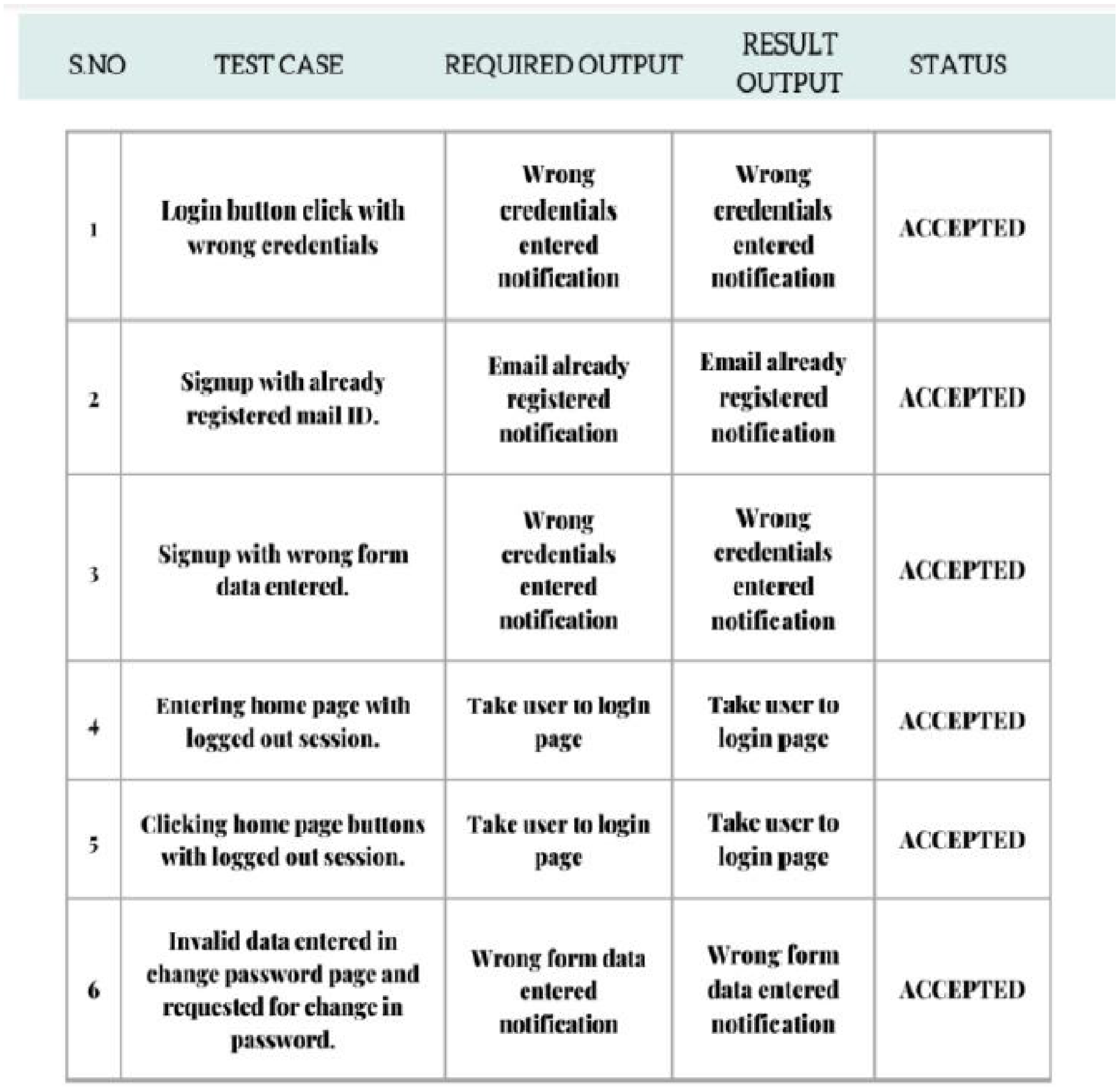
The Detection Filtering module goes over each the objects-detection results, and only keeps the most prominent ones. It takes into account user parameters for filtering according to a minimal confidence threshold (conf_thr) and to a maximal size threshold (size_thr). It also allows user filtering of all non-person objects (by setting detect_all to False). The results are converted into a frame-related coordinates, and pass through a Non-Maximum Suppression (NMS) procedure, to eliminate redundant overlapping boxes with lower confidence. The user may tweak this procedure by setting minimum thresholds for filtering boxes by score (conf_thr) and by overlapping amount (nms_thr). NMS suppresses overlapping bounding boxes and only retains the bounding box that has the maximum probability of object detection associated with it.The Detections Matching module iterates over the filtered detections, seeks for matching persons, and updates that person accordingly. Foreach bounding-box ('detection'), it goes over all of the already registered persons, and tries to find a person-match, i.e. the person with the highest IOU score, aka Jaccard Index. The user may tune this procedure by setting a minimal IOU score (iou_thr), where scores below that threshold will be filtered out. If there is no match, the 'winner' is then being registered as a new person. Otherwise, the corresponding person is getting updated with the detection findings. It shall be noted that each person carries a footprint tail of his recent tracking marks, tweaked by user parameter max_history. Seeking for a person match involves going through each of those history marks, which means a match may be obtained with a recent previous detection of a given person.

8.TESTING

8.1 TEST CASES

Test caseID	Feature Type	Component	Test Scenario	Expected Result	Actual Result	Status
TC_001	UI	Home Page	Verify UI elements in the Home Page	The Home page must be displayed properly	Working as expected	FAIL
TC_002	UI	Home Page	Check if the UI elements are displayed properly in different screen sizes	The Home page must be displayed properly in all sizes	The UI is not displayed properly in screen size 2560 x 1801 and 768 x 630	FAIL
TC_003	Functional	Home Page	Check if user can upload their file	The input image should be uploaded to the application successfully	Working as expected	PASS
TC_004	Functional	Home Page	Check if user cannot upload unsupported files	The application should not allow user to select a non image file	User is able to upload any file	FAIL
TC_005	Functional	Home Page	Check if the page redirects to the result page once the input is given	The page should redirect to the results page	Working as expected	PASS

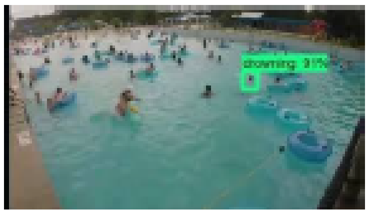
8.2 USER ACCEPTANCE TESTING



S.NO	TEST CASE	REQUIRED OUTPUT	RESULT OUTPUT	STATUS
1	Login button click with wrong credentials	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
2	Signup with already registered mail ID.	Email already registered notification	Email already registered notification	ACCEPTED
3	Signup with wrong form data entered.	Wrong credentials entered notification	Wrong credentials entered notification	ACCEPTED
4	Entering home page with logged out session.	Take user to login page	Take user to login page	ACCEPTED
5	Clicking home page buttons with logged out session.	Take user to login page	Take user to login page	ACCEPTED
6	Invalid data entered in change password page and requested for change in password.	Wrong form data entered notification	Wrong form data entered notification	ACCEPTED

9.RESULTS

9.1 PERFORMANCE METRICS

S.No.	Parameter	Values	Screenshot
1.	Model Summary	This project aims to create a system that will be able to automatically detect drowning incidents in the swimming pool using human action detection.	
2.	Accuracy	Training Accuracy - 90.3 Validation Accuracy -91.2	
3	Confidence Score (Only Yolo Projects)	Class Detected - coco object class Confidence Score - 0.86	

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.
- This system don’t have to wait until life guard comes to rescue because it has uplifting mesh.
- This is very fast process
- More effective and cost Efficient than previous other models

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.
- Internet connection is necessary to use GPS or sending alert messages. Sometimes to send messages SIM balance may be required.

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

13.1 SOURCE CODE

Objectdetection.py:

```
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 - 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.cf
g'

  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 = indices[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
```

Drowndetection.py:

```
import cvlib as cv
from cvlib.Objectiondetection import detect_common_objects
import cv2
import time
import numpy as np

webcam = cv2.VideoCapture(0)

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

t0 = time.time()

centre0 = np.zeros(2)
isDrowning = False


while webcam.isOpened():

    status, frame = webcam.read()

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

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

```
if (len(bbox) > 0):
    bbox0 = bbox[0]

    centre = [0, 0]

    centre = [(bbox0[0] + bbox0[2]) / 2, (bbox0[1] + bbox0[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, 'centre:', centre, 'centre0:', centre0)
    print('Is he drowning: ', isDrowning)

    centre0 = centre

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

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

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

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

13.2 GITHUB & DEMO LINK

https://github.com/IBM-EPBL/IBM-Project-10341-1659171127

https://drive.google.com/file/d/1kOvDrfMOrLvIFE5xrysiJCatRCTIGk9W/view?usp=share_link