

Project Documentation

Date	10 November 2022
Team ID	PNT2022TMID32676
Project Name	Virtual eye - Lifeguard for swimming pools for active drowning

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

1. INTRODUCTION

1.1 Project Overview

Virtual-eye is a cohesive all-in-one system used for poolside safety and drowning detection. At its core, it is a computational AI model named YOLO. An algorithm that uses neural networks to provide real-time object detection. We source the benefits of YOLO to enable real-time frame-by-frame detection, which in our case is identifying "potentially drowning swimmers". Virtual-eye is a full-fledged web application that takes in a video stream or footage to compute the probability of a potential case of drowning. On detecting any signs of drowning, the application sounds an alarm to notify that there is someone in need of help. The application additionally supports register & login features to authenticate users on a subscribe-to-use basis. Virtual-eye aims to be a minimal, hassle-free monitoring software to make pool days safe and fun.

1.2 Purpose

Virtual-eye aspires to be an all-encompassing solution for poolside safety and drowning detection. Its aim is to be the single standalone monitoring system in every pool. It is designed with a unit-purpose in mind which is to analyze the input feed and predict the probability of drowning. The purpose of the application is further enhanced by adding an alarm system that sounds in a situation of distress. This loud alarm notifies when drowning is detected so that steps for rescue can be taken appropriately and on time. Having the alarm contained in the same system makes it portable. An extended version would be when the footage is replaced by a live camera stream from the pool surveillance system. A complete application requires authentication, and it is achieved by implementing login/registration features. This feature is further tuned by enabling signing with OAuth.

2. LITERATURE SURVEY

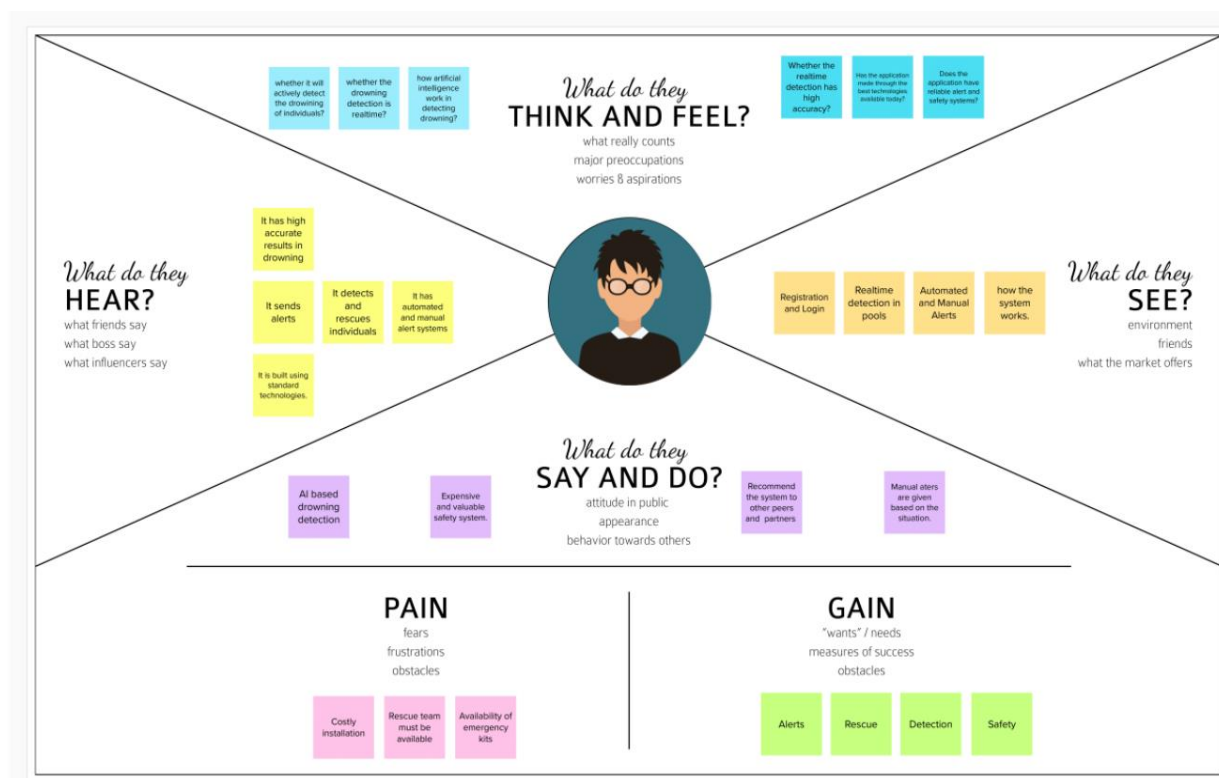
S.no	Paper Title	Year of publication	Journal or Conference name	Authors	Theme of the paper	Inference
1.	An automatic drowning detection surveillance system for challenging outdoor pool environments	2003	Computer Vision, 2003. Proceedings. Ninth IEEE International Conference.	A.H. Kam J.Wang	Automatic drowning detection surveillance system	Understanding Automatic drowning detection.
2.	Drowning Detection System using LRCN Approach	2022	Convergence in Technology Mumbai, India	Shardul Sanjay Chavan, Sanket Tukaram Dhake, Shubham Virendra Jadhav, Prof. Johnson Mathew	Drowning detection using LRCN	Understanding Approach of drowning using LRCN.
3.	A novel drowning detection method for safety of swimmers	2018	Proceedings of the National Power Systems Conference (NPSC) - 2018, December 14-16, NIT Tiruchirappalli, India	Ajil Roy, Dr. K Srinivasan National Institute of Technology Tiruchirappalli, India	Drowning detection for safety of swimmers	Understanding the safety measures provided by drowning detection

4.	Automated drowning detection and security in swimming pool	2017	International Research Journal of Engineering and Technology (IRJET)	A KANCHANA, KAVYA G.R, KAVITHA C, SOUMYASHREE V, SALILA HEGDE	Security in Drowning detection	Understanding the security measures provided by drowning detection
----	--	------	--	---	--------------------------------	--

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas


Empathy Map Canvas: An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviours and attitudes. It is a useful tool to help teams better understand their users. Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



3.2 Ideation & Brainstorming

Brainstorming provides a free and open environment that encourages everyone within a team to participate in the creative thinking process that leads to problem solving. Prioritizing volume over value, out-of-the-box ideas are welcome and built upon, and all participants are encouraged to collaborate, helping each other develop a rich number of creative solutions. Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

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



VIRTUAL EYE





Brainstorm & idea prioritization

In this session we aim to achieve a good base for beginning our project. With clear understanding of the task in hand, the next step would be to collectively put in our thoughts/ imagination and end with a proper feasibility study.

Ground Rules

- Be Creative
- Rule out every possible ideas and improvements
- Make your points clear and purposeful
- Don't hesitate. (Every point is noteworthy)
- Arguments are good ALA it lands beneficial
- Have various perspectives towards the problem

Team

-  Sasi
-  Satish
-  Soma (L)
-  Viswa

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 1

How might we detect and differentiate active drowning with the least possible error rate?

QUESTION 2

How might we automate the alert systems so as to provide crucial stats and info to the rescue team ?

QUESTION 3


How might we optimize the detection algorithm to yield results in the least time?


QUESTION 4


How might we bring more privacy, yet use camera for detection?


QUESTION 5


How might we optimally use minimal hardware to get the most accurate information in an around the environment?
















→











→

Step-2: Brainstorm, Idea Listing and Grouping

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

Soma

High level testing must be carried out before real world deployment.	Proper hyperparameters must be found for the model.	Systematic and Efficient algorithms to be followed
Requires HD cameras for good quality frames to be processed	Underwater cameras is a possible solution to detect humans under deep water	24/7 Power supply is must for the system to run & report
Provide critical and proper message to the rescue team	Make sure the stakeholders know, how the system works.	Make sure the stakeholders understand that there is a possibility for a false alarm as we

Visva

optimized food the refer to achieve live reality will less RM to get the desirable video of underwater footage	able to process (avoid drowning) and also meeting the rescue team of passive possibilities as a probable instance	setup an ACS and suggestive ways to ensure the information reaches in one or more ways as this deals with critical the saving situation
new strategy where there is 100% guarantee of finding & detecting bodies and passing info to rescue team as quickly as possible	ensuring the video food is not being recorded or saved instead being used only for detection which is later discarded	using alternative source of energy such as solar to make a green system but making sure to always have backup supply
having an integration with those who consumer to get all types of cameras to have better information and make decisions as a drowning incident	having info selective information sent to children and teenagers and teaching them ways to use the drowning detector easily	having considered the needs and variance of different age groups and also current swimming environment both controlled and leisure

Sasi

The AI should be trained with more samples for better results	There should be manual alert system in case of detection failure	More cameras should be used to improve accuracy.
How will be the accuracy level in the system?	Will the system detect properly if the pool is clumsy?	System should detect multiple drowning and should report the same
For privacy purpose the video stream should not be stored.	The system shouldnt annoy others	cameras can be mounted on the bottom of floating boards for large swimming pools.

Sathish

power backup should be there in case of powercut.	The network connectivity should be good for faster alert transmission.	cameras should be maintained properly for good results
What happens if animals were encountered in the pool?	When more people are drowning there will be a problem to detect all so multiple cameras are needed to eliminate such problems.	Use powerful algorithm to get trained from various datasets.
AI should be trained in such a way that it should detect multiple drowning		

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

TIP

You can use the Voting section next above to focus on the strongest ideas.

Privacy

ensuring the video feed is not being recorded or saved instead being used only for detection which is later discarded

For privacy purpose the video stream should not be stored.

User Perspective

Make sure the stakeholders know how the system works and understand the possibility for system work.

The system should not annoy the swimmers

Make sure the stakeholders understand that there is a possibility for a false alarm as well

Features

having an integration with third party companies to generate system or camera to have better information and prevent possibility of a drowning incident.

When more people are drowning there will be a problem to detect all so multiple cameras are needed to eliminate such problems.

Will the system detect properly if the pool is clumsy?

Cameras & Hardware

cameras should be maintained properly for good results

Cameras should be mounted on Underwater and bottom of floating boards for detecting drowning effectively especially on large swimming pools.

System should detect multiple drowning and should report the same

Network and Connectivity

The network connectivity should be good for faster alert transmission.

optimized food transfer to achieve live reality will less RM to get the desirable video of underwater footage

Power

24/7 Power supply and power backup must for this system to run & report proper alerts to rescue team.

power backup should be there in case of powercut.

using alternative source of energy such as solar to make a green system but making sure to always have backup supply

AI and ML

Proper hyperparameters must be found for the model.

The AI should be trained with more samples for better results

able to process (avoid drowning) and also meeting the rescue team of passive possibilities as a probable instance

AI should be trained in such a way that it should detect multiple drowning

High level testing must be carried out before real world deployment.

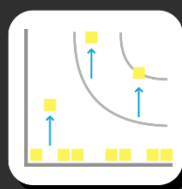
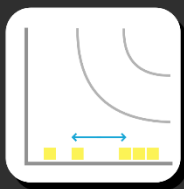
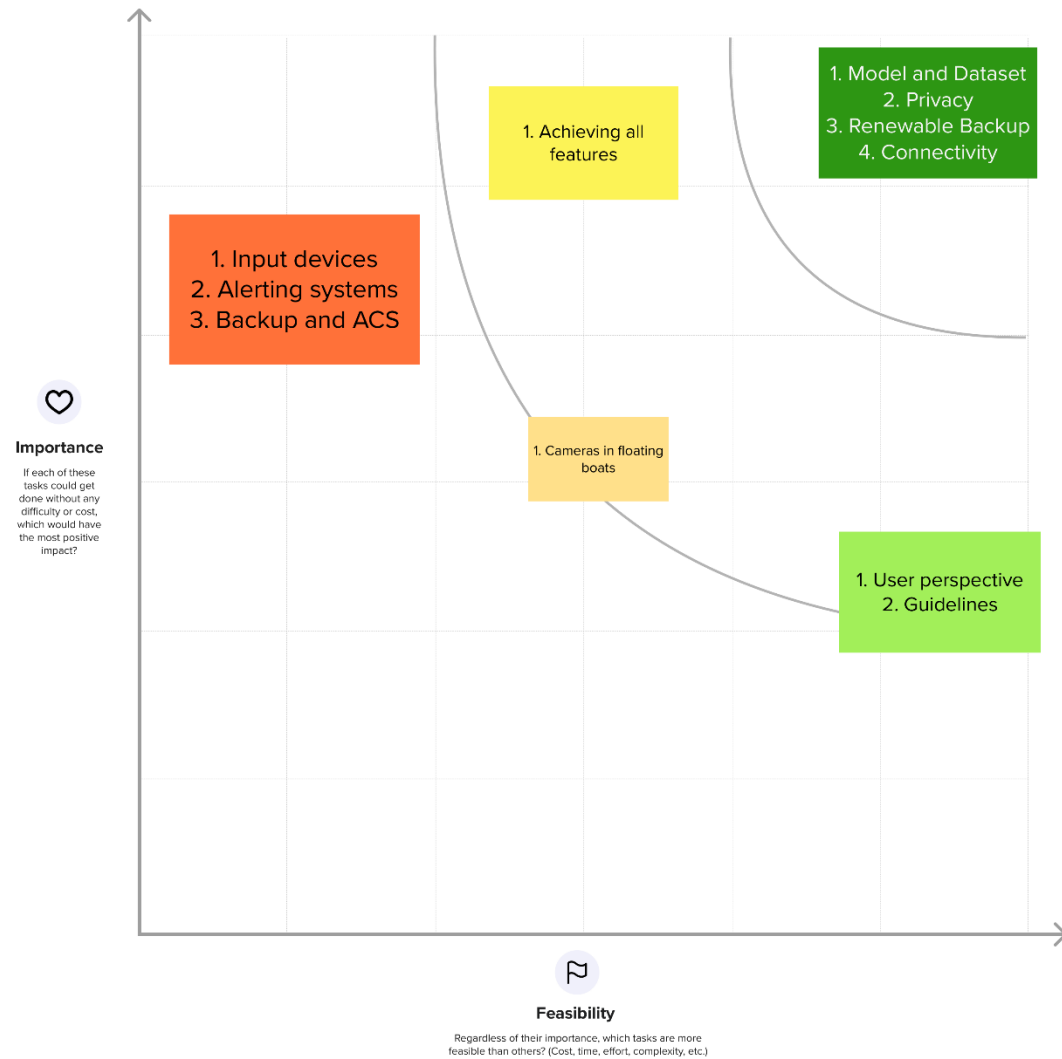
Step-3: Idea Prioritization

4

Prioritize

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

🕒 20 minutes



3.3 Proposed solution

S.No	Parameter	Description
1.	Problem Statement (Problem to be solved)	detection of abnormal activity or drowning and to alert staff.
2.	Idea / Solution description	Using our model and the CNN and YOLO algorithms, we can forecast drowning incidents in swimming pools. To obtain very accurate results, we specifically use version 7 of the YOLO algorithm.
3.	Novelty / Uniqueness	We can predict drowning incidences in swimming pools using our model, the CNN algorithm, and the YOLO algorithm. Version 7 of the YOLO algorithm is specifically used to get extremely accurate results.
4.	Social Impact / Customer Satisfaction	Annually 1.2 million individuals confront spontaneous passing due to suffocating globally. This passing rate will be decreased by actualizing this solution.
5.	Business Model (Revenue Model)	Thus it may be a lifesaving show, it can be utilized by fledglings and unpredictable swimmers. It cautions close by swimmers who protect the suffocating one.
6.	Scalability of the Solution	YOLO V7 includes a extraordinary adaptability engineering compare to others. Increase in depth scaling and width scaling and determination scaling we are able increment the adaptability of this model. Real-time Question location of yolo calculation has tall precision and fast

3.4 Problem Solution fit

Project Title: VirtualEye - Life Guard for Swimming Pools to Detect Active Drowning			Project Design Phase-I - Solution Fit		Team ID: PNT2022TMID32676
Define CS, fit into CC	1. CUSTOMER SEGMENT(S) CS Owners of stadium,schools and localities having swimming pools were the customers.	6. CUSTOMER CONSTRAINTS CC <ul style="list-style-type: none"> ❑ Installation cost is high. ❑ Cameras should be maintained properly for good results. ❑ Network connectivity should be good for faster alert transmission. ❑ 24/7 Power supply should be available. 	5. AVAILABLE SOLUTIONS AS Earlier days drowning of individuals were identified by manual monitoring by the swimming pool attendant but it has some difficulties like not able to monitor all the individuals in the swimming pool. SOLUTION: We use YOLO model in drowning detection, the accuracy of detecting active drowning is high As many cameras were installed everyone is being monitored at a time and the alerts are given instantly. MERITS: Alerts were given instantly. DEMERITS: Detection becomes difficult if the pool is clumsy.	Explore AS, differentiate	
	2. JOBS-TO-BE-DONE / PROBLEMS J&P <ul style="list-style-type: none"> ❑ There Is A Safety Flaw To Swimmers As They Might Drown In Certain Conditions as There Is No Assurance That The Swimmer Is Always Healthy And Conscious In The Pool. ❑ It is a difficult task to keep an eye on all the swimmers/individuals at the same time. 	9. PROBLEM ROOT CAUSE RC The root cause for the problem to occur is that many don't have health conscious and leads to drowning. Another reason is not having proper training. And for a attendant during clumsy situation it is difficult to monitor all the individuals.	7. BEHAVIOUR BE DIRECTLY RELATED: <ul style="list-style-type: none"> ❑ Finding the best drowning system by analysing the performance and rating of the system,checking cost efficiency,feasibility and the total capital cost needed for installation . INDIRECTLY RELATED: <ul style="list-style-type: none"> ❑ Customers hire for pool attendants to monitor the swimmers individually. 		
Identify strong TR & EM	3. TRIGGERS TR <ul style="list-style-type: none"> ➤ It has high accurate results in drowning detection. ➤ It sends Alerts. ➤ It has an automated and manual alert system. ➤ It is built using Standard Technologies. 	10. YOUR SOLUTION SL <ul style="list-style-type: none"> ❖ Using our model and the CNN and YOLO algorithms, we can forecast drowning incidents in swimming pools.To obtain very accurate results, we specifically use version 7 of the YOLO algorithm. ❖ Annually 1.2 million individuals confront spontaneous passing due to suffocating globally. This passing rate will be decreased by actualizing this solution. ❖ Thus it may be a lifesaving show, it can be utilized by fledglings and unpredictable swimmers. It cautions close by swimmers who protect the suffocating one. 	8.CHANNELS of BEHAVIOR CH 8.1 ONLINE Dashboard access to live AI detection. 8.2 OFFLINE Customers can use Customer support.	Identify strong TR & EM	
	4. EMOTIONS: BEFORE / AFTER EM BEFORE: Before the usage of Active drowning detection,identifying and rescuing drowning individuals was difficult. AFTER: After the introduction of Active drowning detection,the drowning individuals were detected and the alerts were given instantly.				

4.REQUIREMENT ANALYSIS

4.1 Functional Requirements

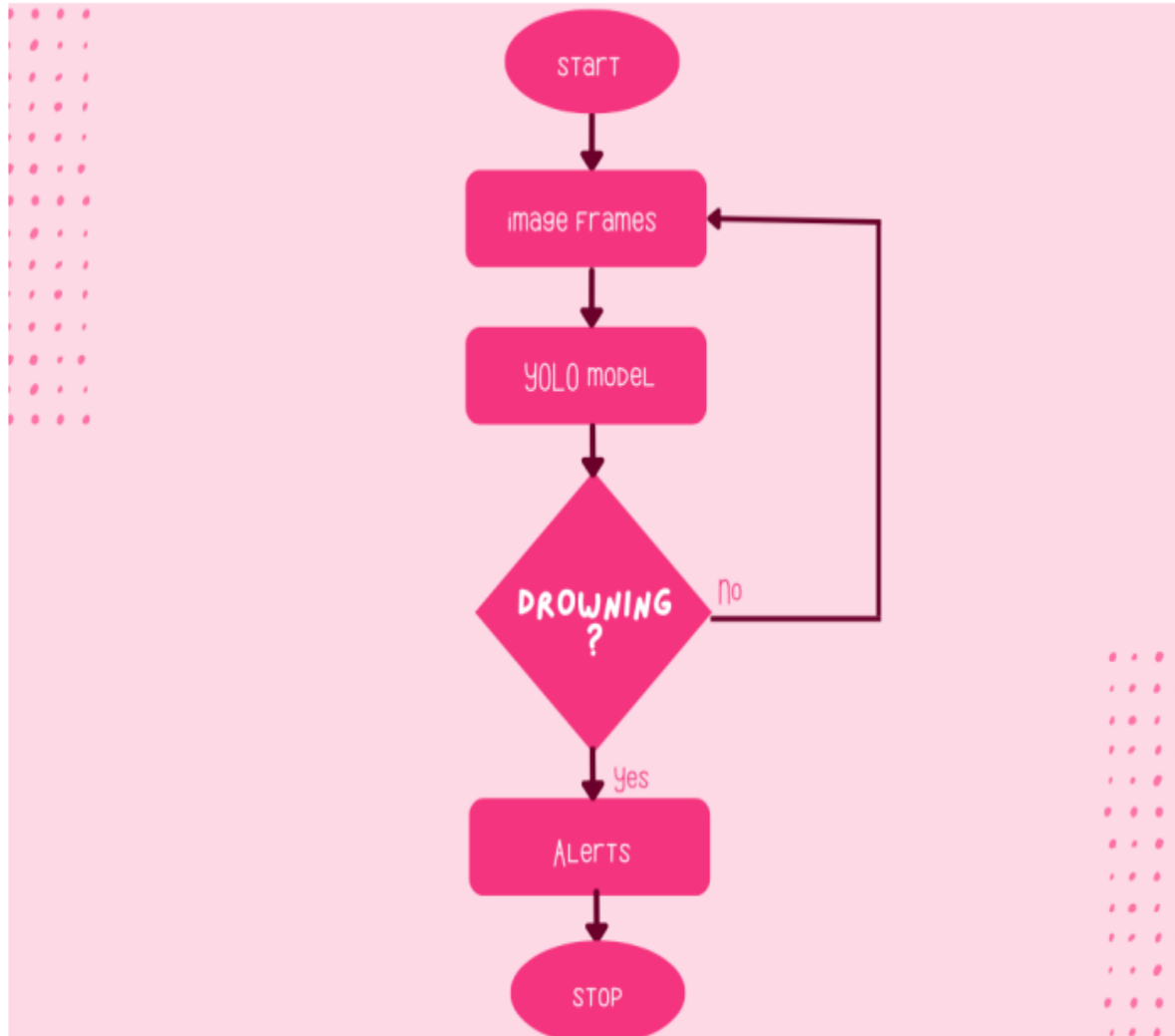
FR No.	Functional Requirements (EPIC)	Sub Requirement (Story/Sub-Task)
FR-1	Installation	Camera will be fixed under the water in the swimming pool.
FR-2	Login/Registration	User Register and login using their email.
FR-3	Audio	Will be calm if the person is unconscious or ask for help
FR-4	Support	Take the help from the medical team
FR-5	Prior Alert	Alert message sends to the rescue team
FR-6	Dashboard	AI detections are clearly displayed to the user with respect to the input.

4.2 Non-Functional Requirements

FR No.	Functional Requirements (EPIC)	Sub Requirement (Story/Sub-Task)
NFR-1	Usability	To provide safety for every person in the swimming pool from drowning.
NFR-2	Security	An alarm will be there to alert and alert message will send to rescue team to save the live of the person in the swimming pool
NFR-3	Reliability	Virtual eye lifeguard alerts instant alerts by alarms and the result will be accurate.
NFR-4	Performance	The alert were given immediately when drowning is detected.
NFR-5	Availability	1. All necessary equipment like water tubes, ropes, first aid kit, life hooks will be available
NFR-6	Scalability	2. Virtual eye lifeguard identifies drowning and it will alert immediately. It features the latest artificial intelligence technology and work with the need of the user.

5.PROJECT DESIGN

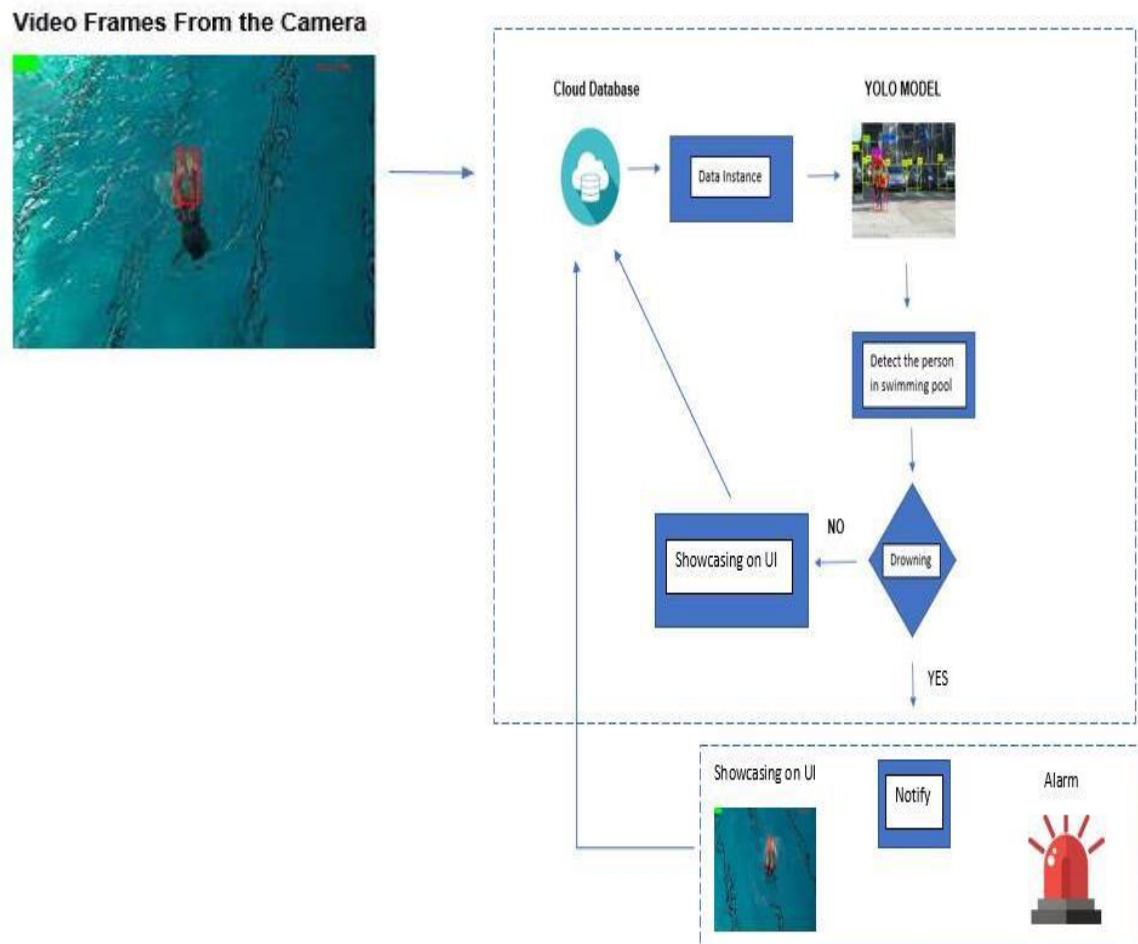
5.1 Data Flow Diagrams



5.2 Solution and Technical Architecture

Solution architecture is a complex process – with many sub-processes – that bridges the gap between business problems and technology solutions. Its goals are to:

- Find the best tech solution to solve existing business problems.
- Describe the structure, characteristics, behaviour, and other aspects of the software to project stakeholders.
- Define features, development phases, and solution requirements.
- Provide specifications according to which the solution is defined, managed, and delivered.



5.3 User Stories

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming the password.	I can access my account/dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I get a dashboard to view detections by the AI.		High	Sprint-1
Customer (Webuser)	Registration	USN-1	As a user, I can register for the application by entering my email, password, and confirming the password.	I can access my account/dashboard	High	Sprint-1
		USN-2	As a user, I will receive confirmation email once I have registered for the application	I can receive confirmation email & click confirm	High	Sprint-1
		USN-3	As a user, I can register for the application through Facebook	I can register & access the dashboard with Facebook Login	Low	Sprint-2
		USN-4	As a user, I can register for the application through Gmail		Medium	Sprint-1
	Login	USN-5	As a user, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-6	As a user, I get a dashboard to view detections by the AI.		High	Sprint-1
Customer Care Executive	Issue Threads & Feedback	USN-7	Threads to reports issues & feedbacks.		High	Sprint-1
Administrator	Login	USN-8	As an administrator, I can log into the application by entering email & password		High	Sprint-1
	Dashboard	USN-9	As an administrator, I get a dashboard to view detections by the AI. Also, options for manual alerts for rescue.		High	Sprint-1

6. PROJECT PLANNING & SCHEDULING

6.1 Sprint Planning & Estimation

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Registration	USN-1	As a user, I can use a simple login function using email, password and confirming the password	2	High	Soma, Viswa, Sasi, Sathish
Sprint-1		USN-2	As a user, I receive a confirmational mail when registered successfully	1	High	Soma, Viswa, Sasi, Sathish
Sprint-2		USN-3	As a user, I can sign in using OAuth or SAML	2	Low	Soma, Viswa, Sasi, Sathish
Sprint-1	Login	USN-4	As a user, I can log into the application by entering email & password	1	High	Soma, Viswa, Sasi, Sathish
Sprint-1		USN-5	As an Admin, I can login as a super-user and view/modify other user accounts	2	High	Soma, Viswa, Sasi, Sathish
Sprint-2		USN-6	As a user, I can Login using OAuth	1	Low	Soma, Viswa, Sasi, Sathish
Sprint-1	Dashboard	USN-7	As a user, I get a dash view of the probable prediction by the AI	2	High	Soma, Viswa, Sasi, Sathish
Sprint-2	Issue Thread and feedback	USN-8	As a user, I can post queries in issue thread and get feedback	1	Low	Soma, Viswa, Sasi, Sathish

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	8	10 Days	1 Nov 2022	10 Nov 2022	8	17 Nov 2022
Sprint-2	3	4 Days	10 Nov 2022	13 Nov 2022	2	17 Nov 2022
Sprint-3	-	-	-	-	-	-
Sprint-4	-	-	-	-	-	-

Velocity:

Imagine we have a 10-day sprint duration, and the velocity of the team is 20 (points per sprint). Let us calculate the team's average velocity (AV) per iteration unit (story points per day)

$$AV = \text{Sprint Duration/velocity}$$

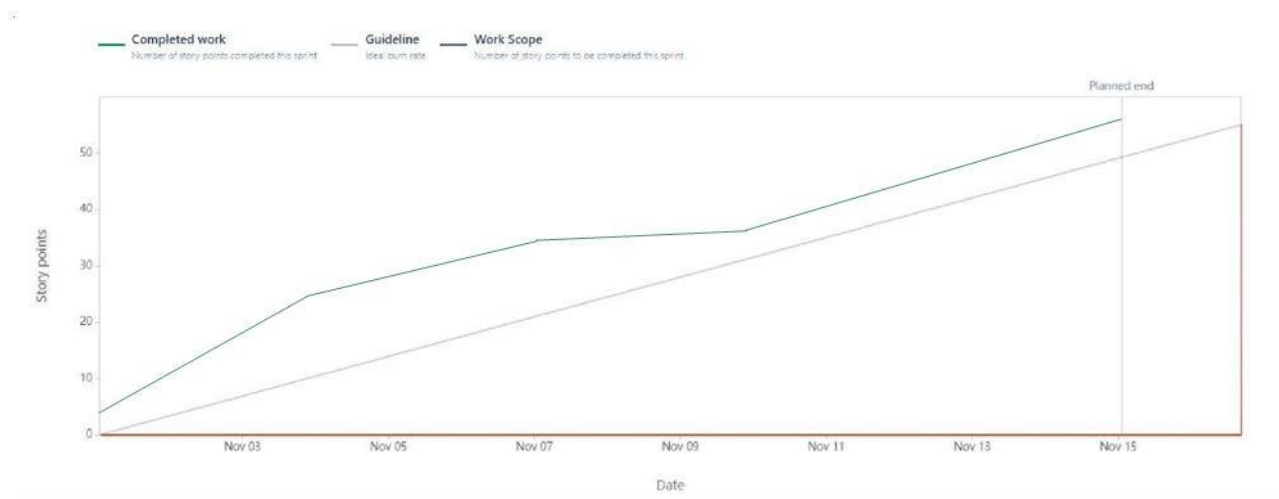
$$= 10/8$$

$$= 1.25$$

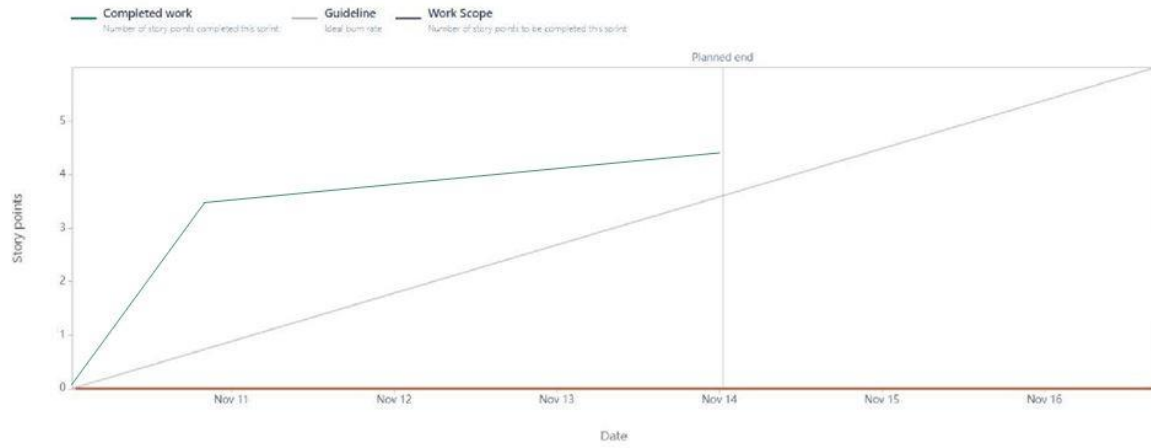
6.3 Reports From JIRA

Burndown Chart:

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

Sprint 1

Sprint 2




```

        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)

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

    if(isDrowning == True):

        webcam.release()
        cv2.destroyAllWindows()

        playsound("http://localhost:5000/static/sound3.mp3")

        return render_template("prediction.html",
message="Emergency!!! The Person is Drowning")

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

    webcam.release()
    cv2.destroyAllWindows()

    return render_template("prediction.html")
else:
    return render_template("login.html", message="You must be logged in
first!")

```

7.2 Feature 2

Admin Login, Dashboard and feedback – Admin who is the user can register through the site and login with the credentials. After the admin is directed to the dashboard page where the detection can be done & feedbacks will be shown given by other users.

dashboard.html

```

{% extends "base.html" %} {% block content %}

<section id="dashboard">
    {% if bad %}
    <div id="message">{{ message }}</div>

```



```
</div>
{% endif %} {% endfor %}
</section>
{% endblock %}
```

Server code for “/feedbacks”:

```
@app.route("/feedbacks", methods=["GET"])
def adminDashboard():
    if request.cookies.get("isLoggedIn") == "True" and
request.cookies.get("isAdmin") == "True":

        feedbacks = []

        for document in my_database:
            feedbacks.append(document)
            print(document)

        return render_template("feedbacks.html", feedbacks=feedbacks)
    else:
        return render_template("login.html", message="You must be logged in
first!")
```

7.3 Database Schema

User Account has defined database schema.

Schema Structure:

```
{
  "_id": {
    Type: String
    Required: True
  },
  "name": {
    Type: String
    Required: True
  }
,
  "psw": {
    Type: String
    Required: True
  }
}
```

```
,
  "isAdmin": {
    Type: String
  },
  "feedback": {
    Type: String
  }
}
```

Schema fields and their use:

_id – email of the user

name – name of the user

psw – password for the user

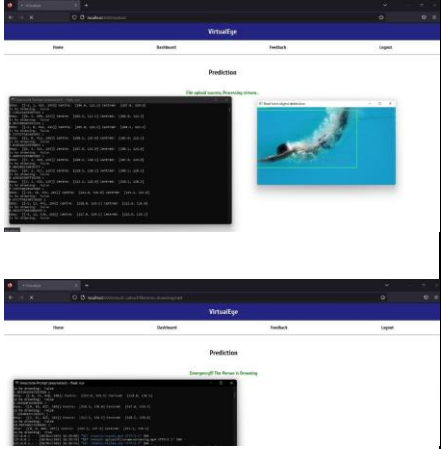
isAdmin – Specifies if the user has admin privileges

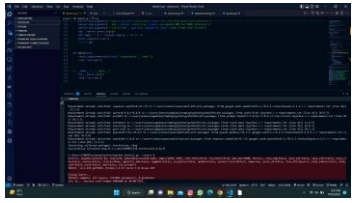
feedback – User feedback

8. TESTING

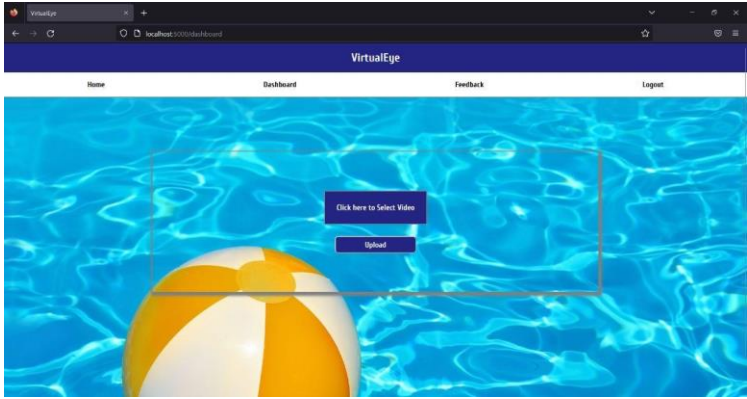
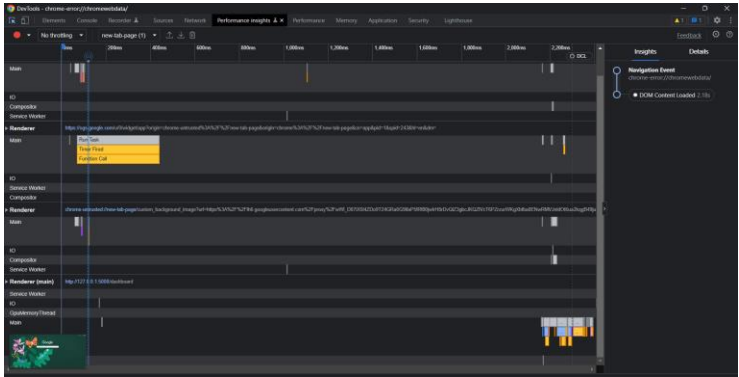
8.1 Test Cases

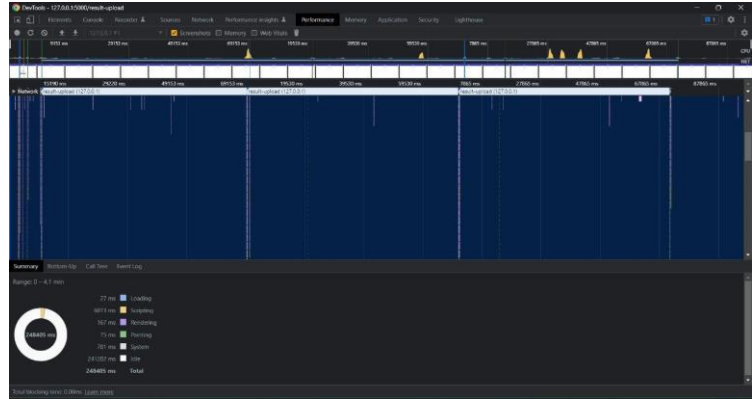
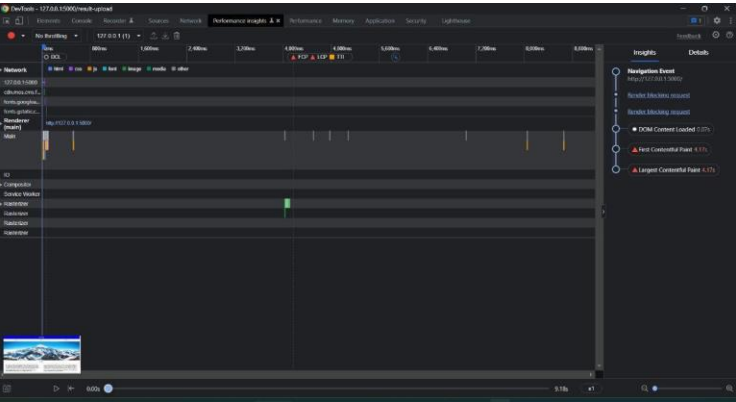
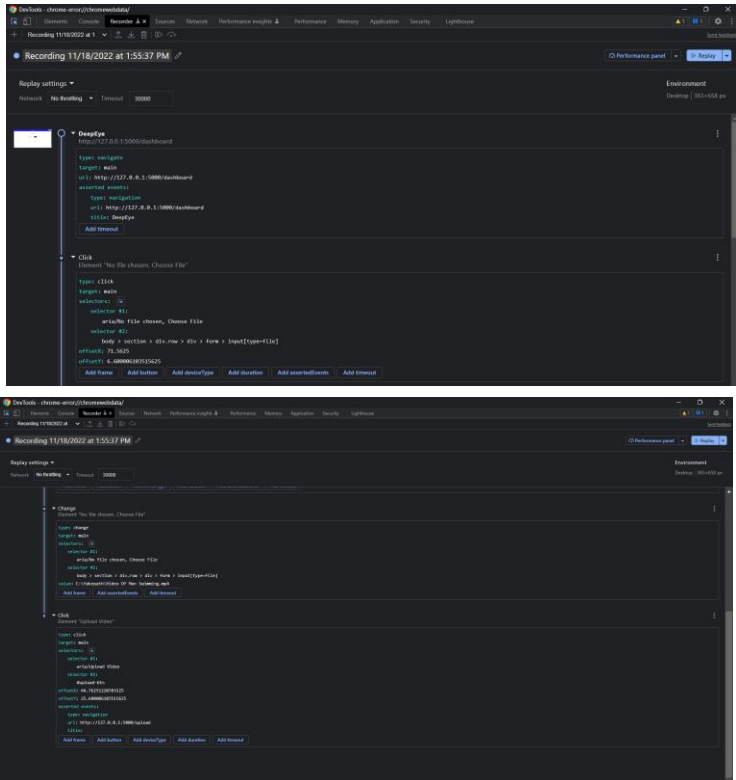
Model Performance Testing

S.No.	Parameter	Values	Screenshot
1.	Model Summary	<pre>detect1: weights=yolov5s.pt, source=0, data=data\coco128.yaml, imgsz=[640, 640], conf_thres=0.25, iou_thres=0.45, max_det=1000, device=, view_img=False, save_txt=False, save_conf=False, save_crop=False, nosave=False, classes=None, agnostic_nms=False, augment=False, visualize=False, update=False, project=runs\detect, name=exp, exist_ok=False, line_thickness=3, hide_labels=False, hide_conf=False, half=False, dnn=False, vid_stride=1 YOLOv5 v6.2-215-g575055c Python-3.9.13 torch-1.13.0+cpu CPUFusing layers... YOLOv5s summary: 213 layers, 7225885 parameters, 0 gradients 1/1: 0... Success (inf frames640x480 at 30.00 FPS)</pre>	

2.	Accuracy	Training Accuracy - 91.6% Validation Accuracy - 87.10%	(pre-trained model)
3.	Confidence Score (Only Yolo Projects)	Class Detected - 0 Confidence Score - Avg(82%)	

Data Analytics

S.No.	Parameter	Screenshot / Values
1.	Dashboard design	<p>No of Visualizations / Graphs - 2</p>  

2.	Data Responsiveness	
3.	Amount Data to Rendered (DB2Metrics)	
4.	Utilization of Data Filters	

5.	Effective User Story	No of Scene Added - 1
6.	Descriptive Reports	No of Visualizations / Graphs - 1

8.2 User Acceptance Testing

1.Purpose of Document

The purpose of this document is to briefly explain the test coverage and open issues of the Virtual-eye project at the time of the release to User Acceptance Testing (UAT).

2.Defect Analysis

This report shows the number of resolved or closed bugs at each severity level, and how they were resolved

Resolution	Severity 1	Severity 2	Severity 3	Severity 4	Subtotal
By Design	4	3	0	0	7
Duplicate	0	0	0	0	0
External	0	2	2	0	4
Fixed	6	4	3	1	14
Not Reproduced	0	0	1	0	1
Skipped	0	0	0	0	0
Won't Fix	0	1	0	0	1
Totals	10	10	6	1	27

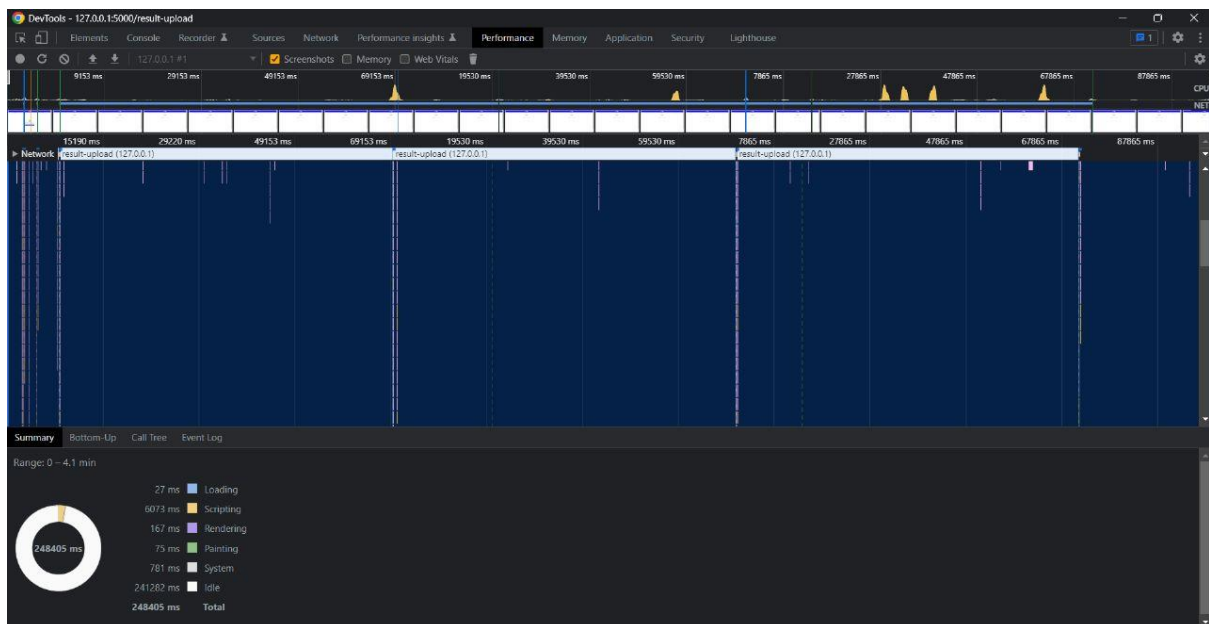
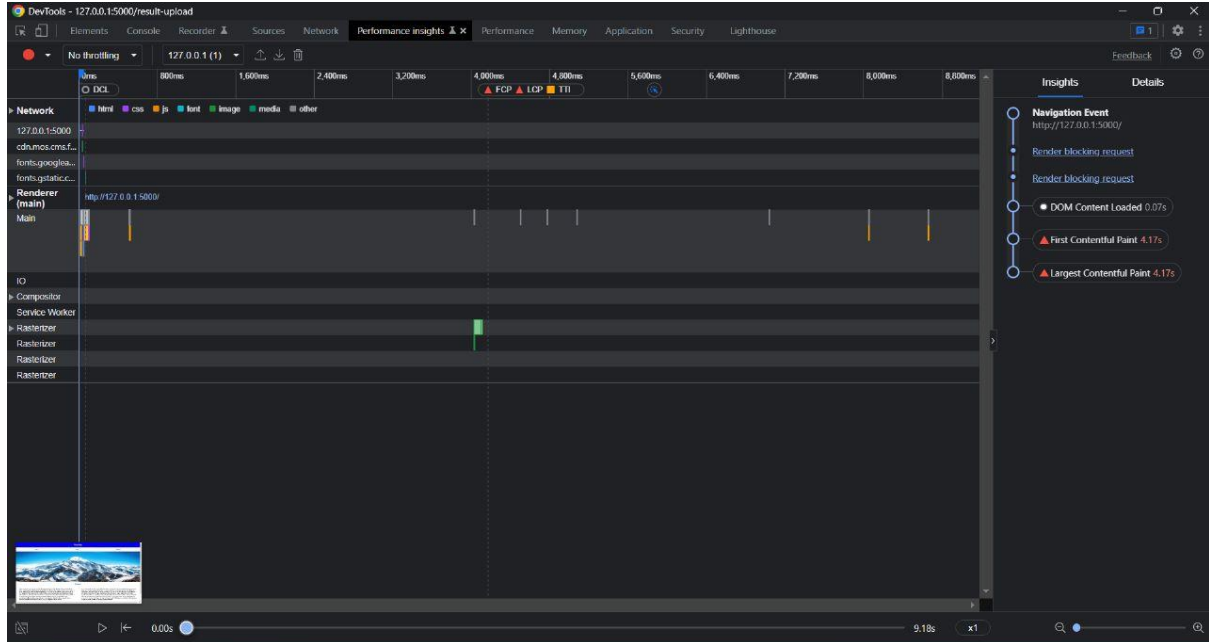
3.Test Case Analysis

This report shows the number of test cases that have passed, failed, and untested.

Section	Total Cases	Not Tested	Fail	Pass
Registration	2	0	0	2
Registration Confirmation mail	1	0	0	1
Login (correct credentials)	1	0	0	1
Login (incorrect credentials)	1	0	1	0
Dashboard video upload	2	0	0	2
Prediction	8	0	0	8
Predictions result accuracy	4	0	0	4
Result	2	0	0	2
Alarm	2	0	0	2
Feedback	1	0	0	1
Feedback Confirmation Mail	1	0	0	1
Version Control	2	0	0	2

9. RESULTS

9.1 Performance Metrics



10. ADVANTAGES & DISADVANTAGES

Advantages

- Actively detects drowning of individuals and instantly give alerts.
- Detection will be accurate as we use well trained Yolo model for drowning detection.
- Drowning detection will be showcased on the Dashboard of the application.

Disadvantages

- 24/7 Power supply and network connection should be available.
- Detection becomes difficult when the pool is clumsy.
- Cameras should be properly maintained.

11. CONCLUSION

Once we have the working drowning detection model, we can feed live video footage of the swimming pool to it so that it can keep detecting continuously for any drowning activities. If drowning is detected it will be highlighted on the system screen as well as alarms will be raised to alert security guards so that they can initiate rescue

12. FUTURE SCOPE

Virtual-eye 2.0 is the next iteration of our drowning detection system. With new and improved UI using, a solid AI model and novel functionalities version 2.0 will be a proper successor. We have planned to integrate our systems to get input from live surveillance feed which enables live monitoring. A list of new features on-way to version 2.0 are authentication using SAML, synthetic AI to train model from frames of surveillance feed, and improved UI with feature/data rich dashboard.

13. APPENDIX

Source Code :

base.html

```
<!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" />
    <link rel="stylesheet" href="/static/styles.css" />
    <title>VirtualEye</title>
  </head>
  <body>
    <nav>
      <h1>VirtualEye</h1>
      <ul>
        <li><a href="/">Home</a></li>
        {% if request.cookies.get("isLoggedIn") == "True" %}
        <li><a href="/dashboard">Dashboard</a></li>
        <li><a href="/feedback">Feedback</a></li>
        <li><a href="/logout">Logout</a></li>
        {% else %}
        <li><a href="/login">Login</a></li>
        <li><a href="/register">Register</a></li>
        {% endif %}
      </ul>
    </nav>
    <hr />

    {% block content %}{% endblock %}

    <footer>VirtualEye &copy; . All rights reserved.</footer>
  </body>
</html>
```

index.html

```
{% extends "base.html" %} {% block content %}
<section>
```



```

<div class="hero">
  
</div>
<div class="main">
  <div class="title">Virtual Eye</div>
  <div class="sub">
    <div class="left">
      <div class="para">
        Virtual-eye is a cohesive all-in-one system used for poolside safety
        and drowning detection. At its core, it is a computational AI model
        named YOLO. An algorithm that uses neural networks to provide
        real-time object detection. We source the benefits of YOLO to enable
        real-time frame-by-frame detection, which in our case is identifying
        "potentially drowning swimmers". Virtual-eye is a full-fledged web
        application that takes in a video stream or footage to compute the
        probability of a potential case of drowning. On detecting any signs
of
        drowning, the application sounds an alarm to notify that there is
        someone in need of help. The application additionally supports
        register & login features to authenticate users on a subscribe-to-
use
        basis. Virtual-eye aims to be a minimal, hassle-free monitoring
        software to make pool days safe and fun.
      </div>
    </div>
    <div class="right">
      <div class="para">
        <h3>STEPS</h3>
        <ol>
          <li>Login/Signup</li>
          <li>Upload Video File</li>
          <li>Be Patient</li>
          <li>Acknowledge the prediction</li>
          <li>Take According action</li>
          <li>Drop a feedback</li>
          <li>Logout</li>
        </ol>
      </div>
    </div>
  </div>
</div>
</section>
{% endblock %}

```

login.html

```
{% extends "base.html" %} {% block content %}

<section class="cont">
  <div class="message">{{ message }}</div>
  <form class="form" method="post" action="/afterlogin">
    <h1 class="head">Login</h1>
    <input type="email" name="_id" class="inp" placeholder="email" />
    <input type="password" name="psw" class="inp" placeholder="password" />
    <input type="submit" class="inp-btn" value="Login" />
  </form>
</section>

{% endblock %}
```

register.html

```
{% extends "base.html" %} {% block content %}

<section class="cont">
  {% if bad %}
  <div class="message">{{ message }}</div>
  {% else %}
  <div class="message" style="color: green">{{ message }}</div>
  {% endif %}
  <form class="form" method="post" action="/afterreg">
    <h1 class="head">Register</h1>
    <input type="text" name="name" class="inp" placeholder="username" />
    <input type="email" name="email" class="inp" placeholder="email" />
    <input type="password" name="password" class="inp" placeholder="password"
  />
    <input type="submit" class="inp-btn" value="Register" />
  </form>
</section>

{% endblock %}
```

dashboard.html

```
{% extends "base.html" %} {% block content %}

<section id="dashboard">
  {% if bad %}
  <div id="message">{{ message }}</div>
  {% else %}
```

```

<div id="message" style="color: green">{{message}}</div>
{% endif %}

<div style="display: flex; justify-content: center">
  <form
    class="form"
    action="/upload"
    method="post"
    enctype="multipart/form-data"
    style="margin-top: 7rem"
  >
    <label
      for="upload"
      style="
        cursor: pointer;
        padding: 1.5rem;
        border: 2px solid grey;
        margin-bottom: 1.5rem;
        color: white;
        background-color: #242582;
      "
    >
      Click here to Select Video
      <input
        class="inp-btn"
        style="display: none"
        type="file"
        name="video"
        id="upload"
        required
      />
    </label>

    <button id="upload-btn" class="inp-btn" type="submit">Upload</button>
  </form>
</div>
</section>

{% endblock %}

```

prediction.html

```

{% extends "base.html" %} {% block content %}

<section style="text-align: center">
  <h2 class="header">Prediction</h2>

```

```

{% if bad %}
<div id="message" style="color: red; margin-top: 3rem">{{ message }}</div>
{% else %}
<div id="message" style="color: green; margin-top: 3rem">{{message}}</div>
{% endif %}

<input type="hidden" value="{{filename}}" id="filename" />

<script>
  document.addEventListener("DOMContentLoaded", async (e) => {
    window.location.replace(
      "/result-upload?filename=" + document.getElementById("filename").value
    );
  });
</script>
</section>
{% endblock %}

```

feedback.html

```

{% extends "base.html" %} {% block content %}
<section id="feedback" class="cont">
  {% if admin %}
  <a href="/feedbacks" id="feedback-link">click here to see the feedbacks</a>
  {% endif %} {% if message %}
  <div id="message" style="color: green; margin-top: 3rem">{{message}}</div>
  {% else %}
  <form action="/feedback" method="post" class="form">
    <input
      type="email"
      class="inp"
      placeholder="email"
      name="email"
      value="{{email}}"
    />
    <textarea
      name="feedback"
      cols="60"
      rows="10"
      value="your feedback here..."
    ></textarea>
    <input
      type="submit"
      class="inp-btn"
      value="Submit"
      style="margin-top: 1rem"
    />
  </form>
  </section>
{% endblock %}

```

```
</form>
    {% endif %}
</section>
{% endblock %}
```

feedbacks.html

```
{% extends "base.html" %} {% block content %}
<section id="feedbacks" class="cont">
    {% for feed in feedbacks %} {% if feed["feedback"] %}
    <div class="card">
        <div class="feed">{{feed["feedback"]}}</div>
        <small>-{{feed["_id"]}}</small>
    </div>
    {% endif %} {% endfor %}
</section>
{% endblock %}
```

app.py

```
# from crypt import methods

from __future__ import print_function

from distutils.log import debug
from email import message
from gzip import BadGzipFile
from itertools import dropwhile
# from signal import alarm
from sqlite3 import connect
import cvlib as cv
from cvlib.object_detection import draw_bbox
import cv2
import time
import numpy as np
import requests

import time
import sib_api_v3_sdk
from sib_api_v3_sdk.rest import ApiException
from pprint import pprint

from werkzeug.utils import secure_filename
from playsound import playsound
```

```

import os
from dotenv import load_dotenv, find_dotenv

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

from cloudant.client import Cloudant

load_dotenv(find_dotenv())

client = Cloudant.iam(os.getenv("IBM_CLOUDANT_KEY"),
os.getenv("IBM_CLOUDANT_USER"), connect=True)

my_database = client.create_database("my_database")

app = Flask(__name__)

def sendMail(to_email, to_name, subject, content):

    configuration = sib_api_v3_sdk.Configuration()
    configuration.api_key['api-key'] = os.getenv("EMAIL_API_KEY")

    api_instance =
sib_api_v3_sdk.TransactionalEmailsApi(sib_api_v3_sdk.ApiClient(configuration))
    html_content = "<html><body><h1>"+ content + "</h1></body></html>"
    sender = {"name": "Admin@VirtualEye", "email": "fullstackdevme07@gmail.com"}
    to = [{"email": to_email, "name": to_name}]
    headers = {"Some-Custom-Name": "unique-id-1234"}
    params = {"parameter": "My param value", "subject": subject}
    send_smtp_email = sib_api_v3_sdk.SendSmtpEmail(to=to, headers=headers,
html_content=html_content, sender=sender, subject=subject)

    try:
        api_response = api_instance.send_transac_email(send_smtp_email)
        pprint(api_response)
    except ApiException as e:
        print("Exception when calling SMTPApi->send_transac_email: %s\n" % e)

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

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

@app.route("/prediction")

```

```

def prediction():
    if request.cookies.get("isLoggedIn") == "True":
        return render_template("prediction.html")
    else:
        return render_template("login.html", message="You must be logged in
first!")

@app.route("/dashboard")
def dashboard():
    if request.cookies.get("isLoggedIn") == "True":
        return render_template("dashboard.html")
    else:
        return render_template("login.html", message="You must be logged in
first!")

@app.route('/upload', methods = ['POST'])
def upload_file():
    if request.cookies.get("isLoggedIn") == "True":
        if request.method == 'POST':
            f = request.files['video']
            f.save(os.path.join(os.path.dirname(os.path.abspath(__file__)),
'static/uploads', secure_filename(f.filename)))
            return render_template("prediction.html", message="File upload
success, Processing stream...", bad=False, filename=f.filename)
        else:
            return render_template("login.html", message="You must be logged in
first!")

@app.route("/register")
def register():
    return render_template("register.html")

@app.route("/afterreg", methods=["POST"])
def afterreg():
    x = [x for x in request.form.values()]
    print(x)
    data = {
        "_id": x[1],
        "name": x[0],
        "psw": x[2],
        "feedback": ""
    }
    print(data)

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

    docs = my_database.get_query_result(query)

```

```

print(docs)

print(len(docs.all()))

if(len(docs.all()) == 0):
    url = my_database.create_document(data)
    content = "Hi, " + data["name"] + " You have successfully registered
with us!"
    sendMail(data["_id"], data["name"], "Registration Successfull",
content)
    return render_template("register.html", message="Registration
Successfull, Please login using your credentials", bad=False)
else:
    return render_template("register.html", message="You are already a
member, please login using your credentials", bad=True)

@app.route("/login")
def login():
    return render_template("login.html")

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

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

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

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

    print(len(docs.all()))

    if(len(docs.all()) == 0):
        resp = make_response(render_template("login.html", message="The email
is not found!"))
        return resp
    else:
        if((user == docs[0][0]["_id"]) and passw == docs[0][0]["psw"]):
            resp = make_response(redirect(url_for("dashboard")))
            resp.set_cookie('isLoggedIn', "True")
            print(docs[0][0]["_id"])
            if user == "admin@virtualeye.com":
                print("zsbdjsjbh")
                resp.set_cookie('isAdmin', "True")
            resp.set_cookie('email', user)

```



```

        return resp
    else:
        print("Invalid User")
        resp = make_response(render_template("login.html", message="The
email is not found!"))

@app.route("/logout")
def logout():
    if request.cookies.get("isLoggedIn") == "True":
        resp = make_response(render_template("login.html", message="You have
logged out successfully!"))
        resp.set_cookie('isLoggedIn', '', expires=0)
        resp.set_cookie('isAdmin', '', expires=0)
        resp.set_cookie('email', '', expires=0)
        return resp
    else:
        return render_template("login.html", message="You must be logged in
first!")

@app.route("/feedback", methods=["GET", "POST"])
def feedback():
    if request.cookies.get("isLoggedIn") == "True":
        if request.method == "GET":
            if request.cookies.get("isAdmin") == "True":
                return render_template("feedback.html",
email=request.cookies.get('email'), admin=True)
            return render_template("feedback.html",
email=request.cookies.get('email'))
        else:
            print(request.form)
            email = request.form["email"]
            print(email)
            feedback = request.form["feedback"]
            print(feedback)

            query = {"_id": {"$eq": email}}

            docs = my_database.get_query_result(query)

            print(docs)

            print(len(docs.all()))

            if(len(docs.all()) == 0):
                resp = make_response(render_template("feedback.html",
message="Something went wrong.. Plese try again later"))

```

```

        return resp
    else:
        if((email == docs[0][0]["_id"])):

            my_document = my_database[email]
            my_document['feedback'] = feedback
            my_document.save()

            print(my_document)

            content = "Thank you for your feedback!"
            sendMail(email, "Dear user", "Feedback Submitted
Successfully!", content)

            resp = make_response(render_template("feedback.html",
message="Thanks! Your feedback submitted successfully!!"))
            return resp
        else:
            print("Invalid User")
            resp = make_response(render_template("feedback.html",
message="Something went wrong.. Plese try again later"))
            return resp

    else:
        return render_template("login.html", message="You must be logged in
first!")

@app.route("/feedbacks", methods=["GET"])
def adminDashboard():
    if request.cookies.get("isLoggedIn") == "True" and
request.cookies.get("isAdmin") == "True":

        feedbacks = []

        for document in my_database:
            feedbacks.append(document)
            print(document)

        return render_template("feedbacks.html", feedbacks=feedbacks)
    else:
        return render_template("login.html", message="You must be logged in
first!")

@app.route("/result-upload", methods=["GET"])
def resUpload():

    if request.cookies.get("isLoggedIn") == "True":

```

```

filename = request.args.get("filename")

webcam = cv2.VideoCapture("static/uploads/" + filename)

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()
    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)

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

        if(isDrowning == True):

            webcam.release()
            cv2.destroyAllWindows()

            playsound("http://localhost:5000/static/sound3.mp3")

            return render_template("prediction.html",
message="Emergency!!! The Person is Drowning")

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

    webcam.release()
    cv2.destroyAllWindows()

    return render_template("prediction.html")
else:
    return render_template("login.html", message="You must be logged in
first!")

if __name__ == '__main__':
    app.run(debug=True, static_url_path="static", static_folder='static',
template_folder="templates")

```

GitHub & Project Demo Link:

Github: <https://github.com/IBM-EPBL/IBM-Project-17068-1659627537>

Demo:

https://drive.google.com/drive/folders/1BZDCURrGh4l9B5HEBEuwceTPu8_v5dvs?usp=sharing