Software Design Document for IFish Farm: Monitoring and analysis of fish anomaly behavior in ubiquitous environment

Ali Ahmed, Hussam ElDin, Omar Anas, Youssef Mohamed
Supervised by: Dr. Ayman Ezzat, Dr. Ayman Nabil and Eng. Noha ElMasry

March 8, 2020

1 Introduction

1.1 Purpose

The purpose of this document is to provide a full description of how the iFish Farm monitoring system works. The monitoring system is an online web-based application system to monitor the fish ponds of a farm to ease the maintenance process for farmers to reduce the labor. This software design document (SDD) will describe the aim of the system and its functionalities. In addition, the document will show all constraints on the system, all interfaces' designs and all diagrams that were needed to build the system.

1.2 Scope

The purpose of the IFish Farm system is to ease the monitoring process and to create a convenient and easy-to-use web application for farmers to monitor their fish farms. The system is based on an unsupervised learning methodology to cluster the different fish behaviors occurring in the farm while providing necessary alerts to the farmers based on the events detected and suggesting a solution to the problem of the event. We will have a cloud server that will process the captured footage to be done there to speed up the monitoring process.

1.3 Overview

This document describes most of the system diagrams and architectures. It also previews how the system main functionalities work and how the user views and interacts with the software. The sections in this document gives a detailed description for the diagrams that help the developer developing the system. It includes the class diagrams, sequence diagrams and the 4 architectures diagrams.

1.4 Definitions and Acronyms

Term	Definition
YOLO	You Only Look Once
MSR	Multi-Scale Retinex algorithm.
ROI	Region of Interest.
MVC	Model-View-Controller

2 System Overview

The IFish Farm system aims to ease the monitoring of fish farms to the fish farmers. The system is divided into several phases. Firstly, the data is collected from a camera above the pond to get the video footage of the fish to detect abnormal behavior and another camera underwater that takes a picture periodically of the ammonia paper to detect toxic ammonia levels. Secondly, the preprocessing phase comes where the video footage and images are enhanced and compressed to get a smaller size and better quality of the video/picture on a cloud server to speed up the process. Thirdly, the classification part where we can detect fish using YOLO algorithm and detect the color of ammonia paper using thershold classfier. After that, features are extracted from the detected fish like there speed, direction, kurtosis, coordinates and many other features. Then, the clustering is applied using K-medoids algorithm to cluster the fish behavior and detect any abnormal behavior. Lastly, the farmer can monitor the farm through the web application and generate reports.

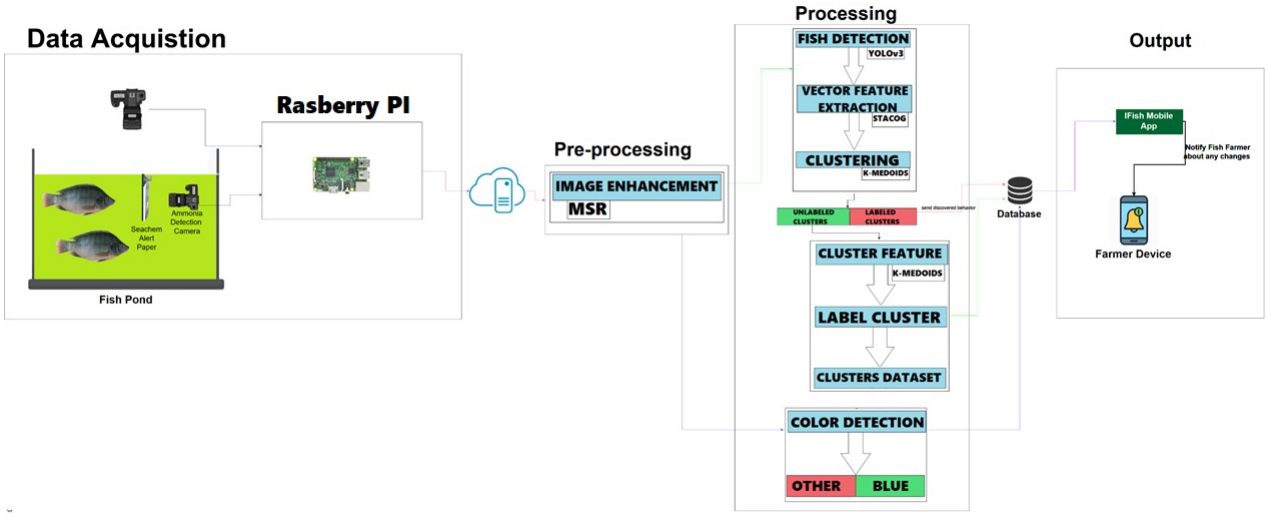

Figure 1: System Overview

3 System Architecture

3.1 Architectural Design

3.1.1 Software Diagram

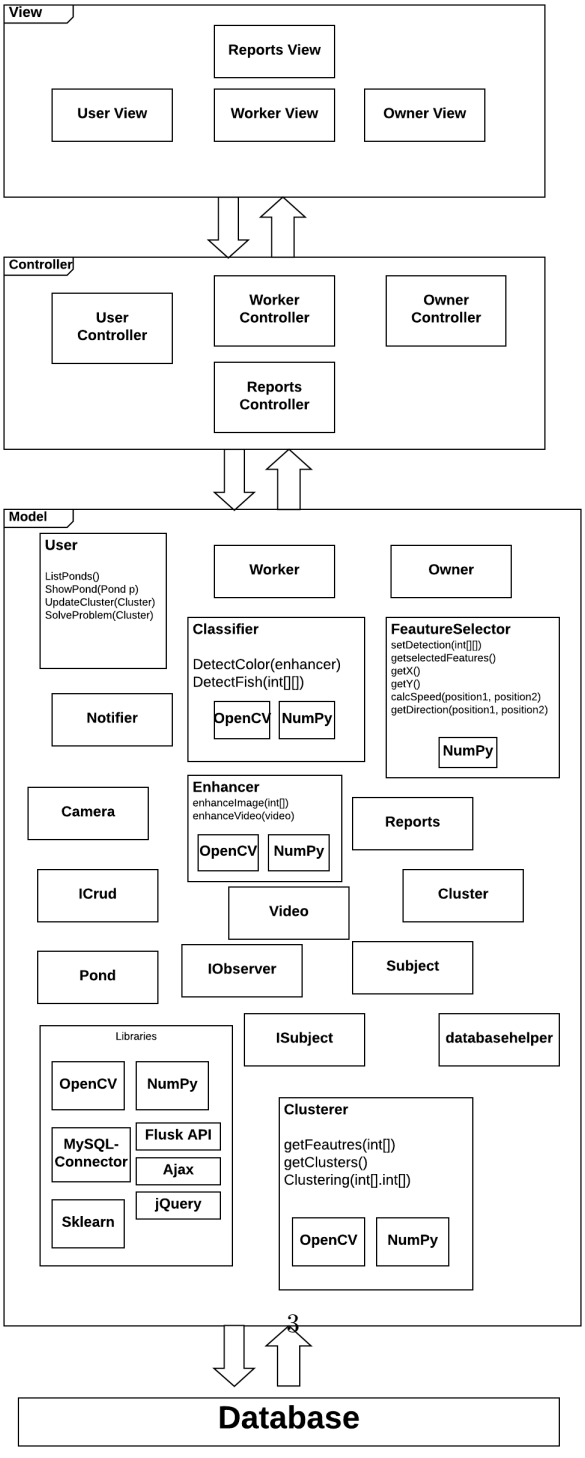

Figure 2: Software Diagram

View:
The view is responsible for presenting the data for in the graphical user interface. There are different views in the system such as the owner view , worker view and the reports view. Each one has the responsibility to view different views to the user.
Controller:
The main responsibility for the Controller is playing a role as an intermediate between the View and the model. It takes the data from the model and send this data for the view to display it for the users. For example, User Controller is responsible for handling common data between the owner and the worker. Reports Controller is responsible for the data between reports model to view it in the reports view.
Model:
The main responsibility for the model is dealing with the database to get the data needed for each class. It takes the data from the database and after that it sends the data to the controller. For example, the reports class takes the information required from the database to build the report.
Libraries used:
OpenCV: used in many operations in the system such as image and videos operations.
NumPy: used in many mathematical operations and handling arrays.
Flask API: used to connect the python code with the web application
SkLearn: provides us with many algorithms that is used in our system.
MySQL-Connector: provides the ability to write queries in the python code.
Ajax and JQuery: Used in the web applications as it provides many options regarding the web code.

3.1.2 Hardware Diagram

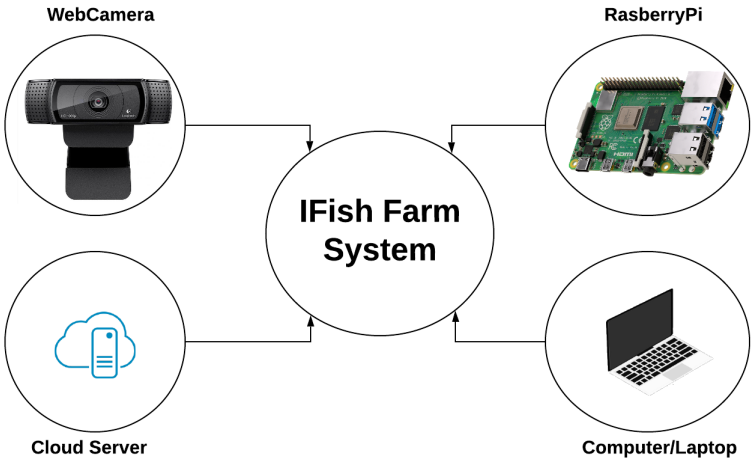

Figure 3: Hardware Diagram

This Diagram Shows all hardware components used in our system. They are four components and includes:
Webcamera: Settled above the pond to get the video footage responsible for monitoring the abnormal fish behaviors.
RasberryPi: Takes the video footage and uploads it to cloud server to be processed.
Cloud Server: It is where the data is processed. It is needed to provide a real-time feedback to the farmer.
Laptop/Computer: The owner of the fish farm might need a computer device to monitor the system.

3.2 Block Diagram

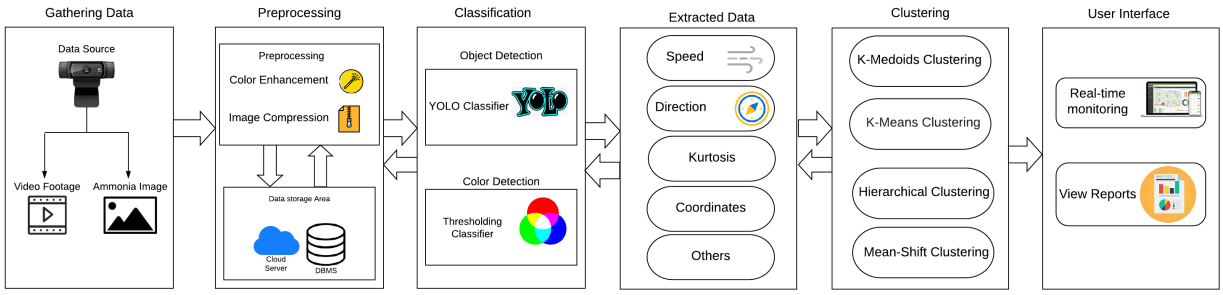

Figure 4: Block Diagram

The Diagram shows the system blocks. The system is divided into six different blocks as follows:

Gathering Data: This block is responsible for data acquisition where a camera above the pond gets the video footage for fish behavior and another camera takes a picture of the ammonia paper periodically to detect toxic ammonia.

Preprocessing: This block is responsible for editing the data to be better classified into the next phase. Firstly, the images and videos are enhanced to get better quality of unclear water. Secondly, the files are compressed to get a smaller size of the images and videos to speed up the process.

Classification: This block is responsible for taking the preprocessed data to classify it. It takes the video footage and detect the fish using YOLO algorithm. Also, It takes the ammonia image and use a color detection algorithm to detect ammonia levels.

Extracted Data: This block is mainly made for extracting the data from videos after fish detection. It takes the video and extract data like speed, direction, kurtosis, coordinated,.. etc. which then helps in detecting abnormal fish behavior.

Clustering: This block is responsible for clustering the fish behaviors into different clusters. It is mainly responsible for detecting any abnormal behavior in fish ponds.

User Interface: This block is responsible for viewing the fish pond in live stream and makes the fish farm owner be able to generate reports.

3.3 Process Diagram

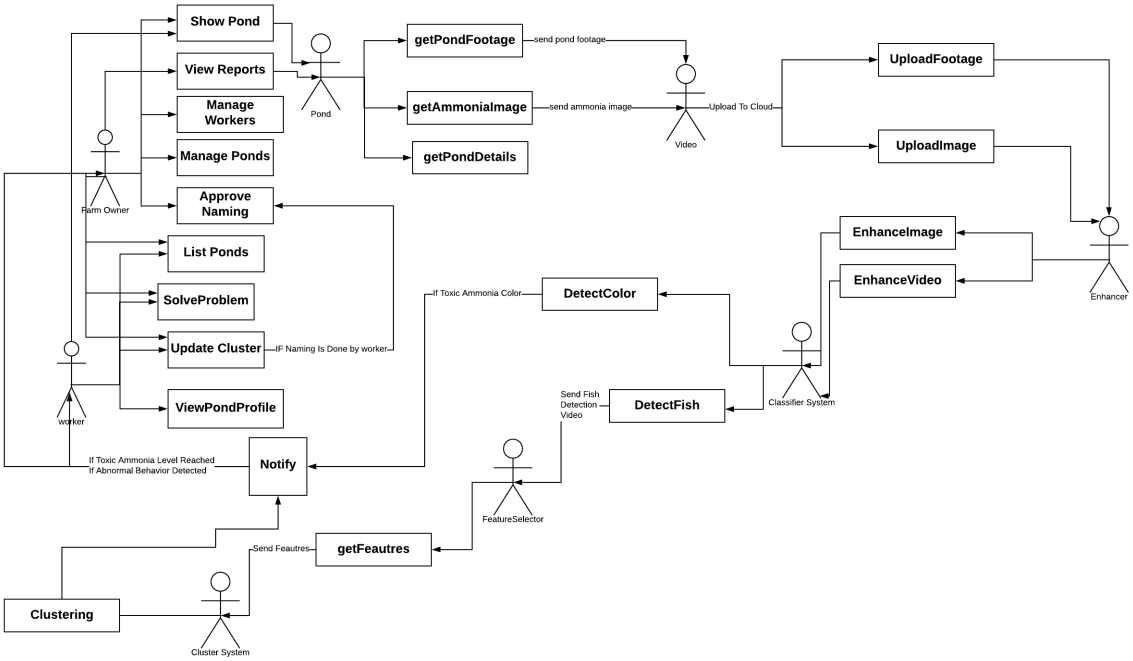

Figure 5: Process Diagram

This diagram shows the process flow of the system. The user shows the pond where it goes to the pond class to get the video footage and ammonia image. After that, the image and video are uploaded to the cloud server. In the cloud server the image/video are enhanced then the classifier detects the color in the ammonia image and detects the fish in the video footage. The features are then extracted by the featureSelector class after that it goes to the clusterer class to detect abnormal behavior. The system then notifies about any anomalies in the pond like the fish abnormal behavior and the toxic ammonia. The user has some other actions like listing all ponds, problem solutions, updating cluster names and other actions as shown in the diagram.

3.4 Context Diagram

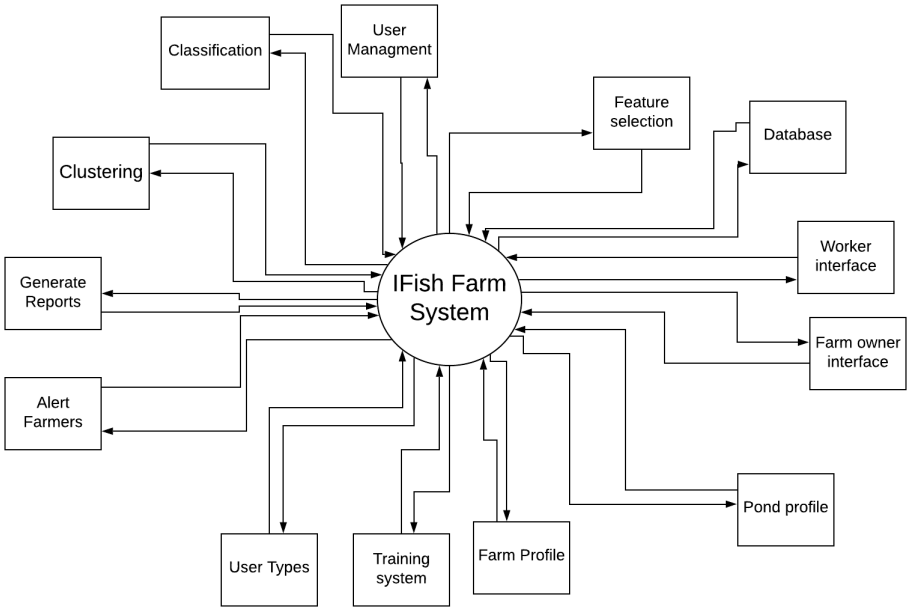

Figure 6: Context Diagram

This Diagram shows the relation between the systems' entities and the actual IFish Farm System. The systems' entites in our system is about 13 entity they include training system, classification, feature selection, database and many others.

3.5 Decomposition Description

3.5.1 Class Diagram

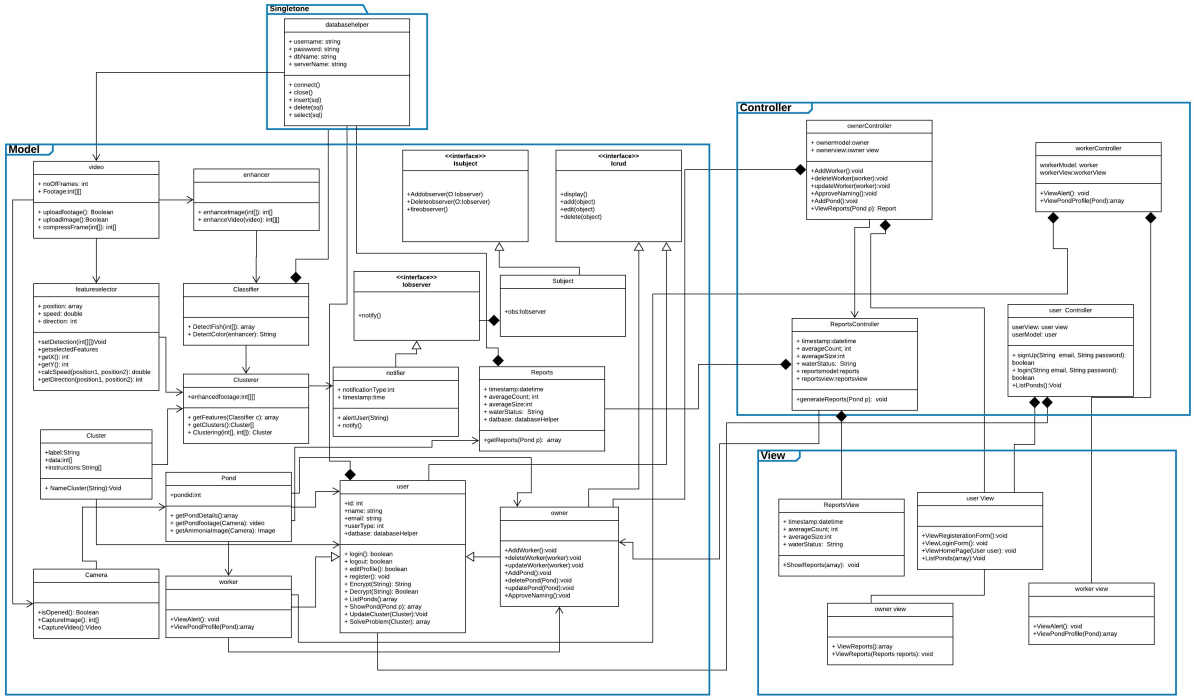

Figure 7: Class Diagram

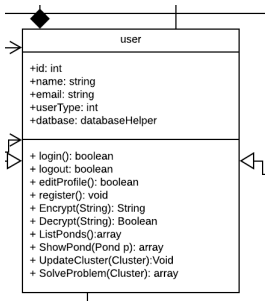

Figure 8: User Class

Class name: user
List of super classes: None.
List of sub classes: Owner, Worker.
Purpose: Class to encapsulate different user-types with their common attributes.
Collaboration:
- Aggregates class databaseHelper and UserController.
- Extended by Owner and Worker.
Attributes: Id, name, email, userType, object from databasehelper
Operations: login(): boolean
logout: boolean
editProfile(): boolean
register(): void
Encrypt(String): String
Decrypt(String): Boolean
ListPonds():array
ShowPond(Pond p): array
UpdateCluster(Cluster):Void
SolveProblem(Cluster): array

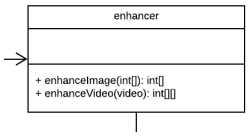

Figure 9: Enhancer Class

Class name: enhancer
List of super classes: None.
List of sub classes: None.
Purpose: Class to get images and videos to enhance their colors for better classification.
Collaboration:

- Assisted by video class.
- Assists Classifier Class
. Attributes: None.
Operations: enhanceImage(int[]): int[]
enhanceVideo(video): int[][]

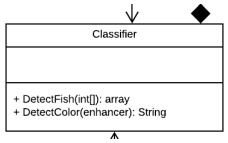

Figure 10: Classifier Class

Class name: classifier
List of super classes: None.
List of sub classes: None.
Purpose: Needed to classify objects (fish) and detect ammonia colors for toxic ammonia levels.
Collaboration:
- Aggeregates class databaseHelper.
- Assisted by enhancer class.
- Assists clusterer class.
Attributes: None.
Operations: DetectFish(int[]): array
DetectColor(enhancer): String

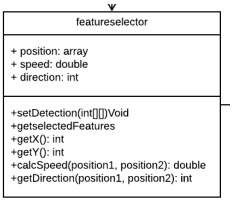

Figure 11: Feauture Selector Class

Class name: feautureselector
List of super classes: None.
List of sub classes: None.
Purpose: Class to extract features from the videos to get the needed features.
Collaboration:
- Assisted by video class.
- Assists clusterer class.
Attributes: position, speed, direction
Operations: setDetection(int[][]): void
getselectedFeatures(): void

getX(): int
getY(): int
calcSpeed(position1, position2): double
getDirection(position1, position2): int

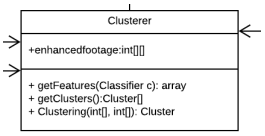

Figure 12: Clusterer Class

Class name: clusterer
List of super classes: None.
List of sub classes: None.
Purpose: Class to detect different behaviors of fish in the system by clustering them into different clusters.
- Assisted by feautureselector , cluster and classifier class.
- Assists notifier class.
Attributes: enhancedfootage:int[][]
Operations: getFeatures(Classifier c): array
getClusters():Cluster[]
Clustering(int[], int[]): Cluster

3.5.2 Activity Diagram

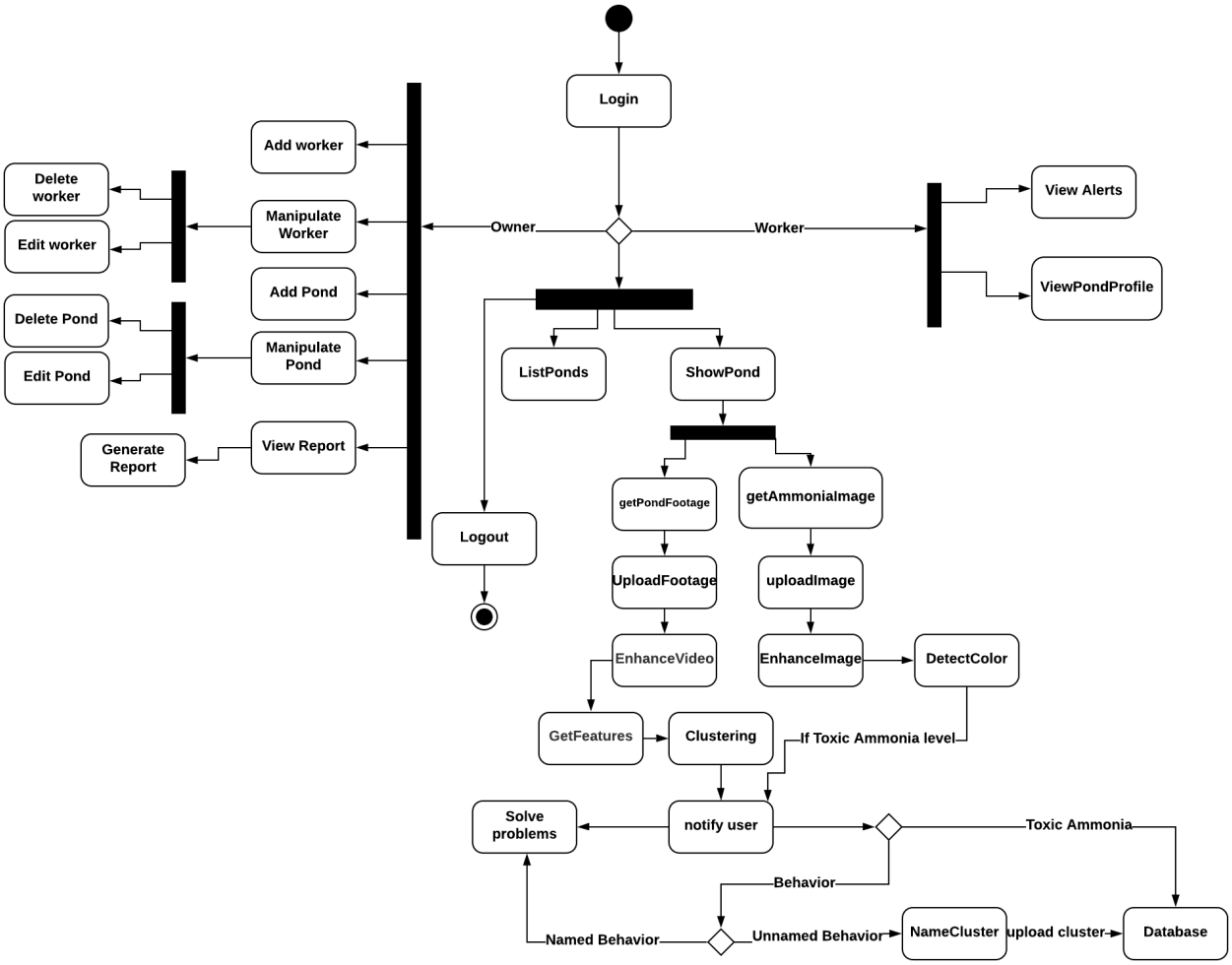

Figure 13: Activity Diagram

3.6 State Diagram

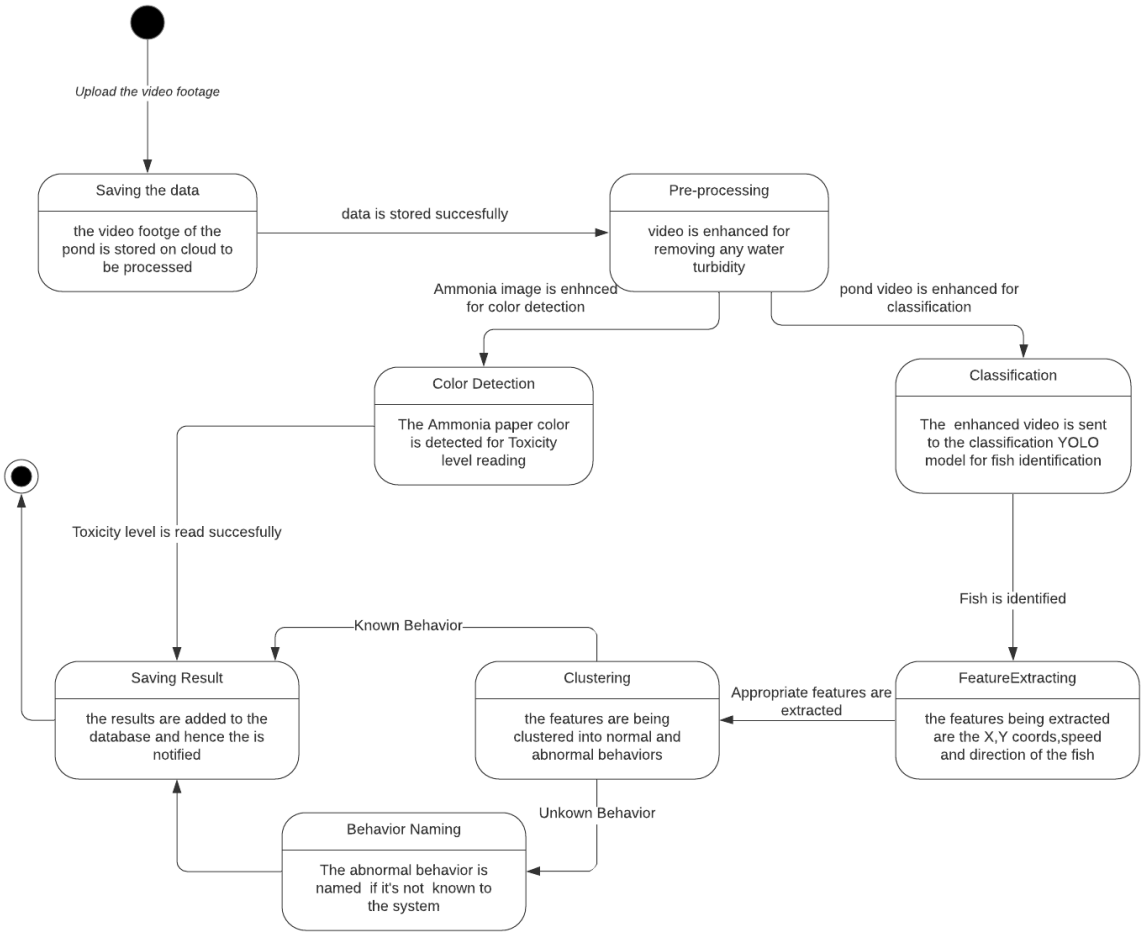

Figure 14: State Diagram

3.6.1 Sequence Diagrams

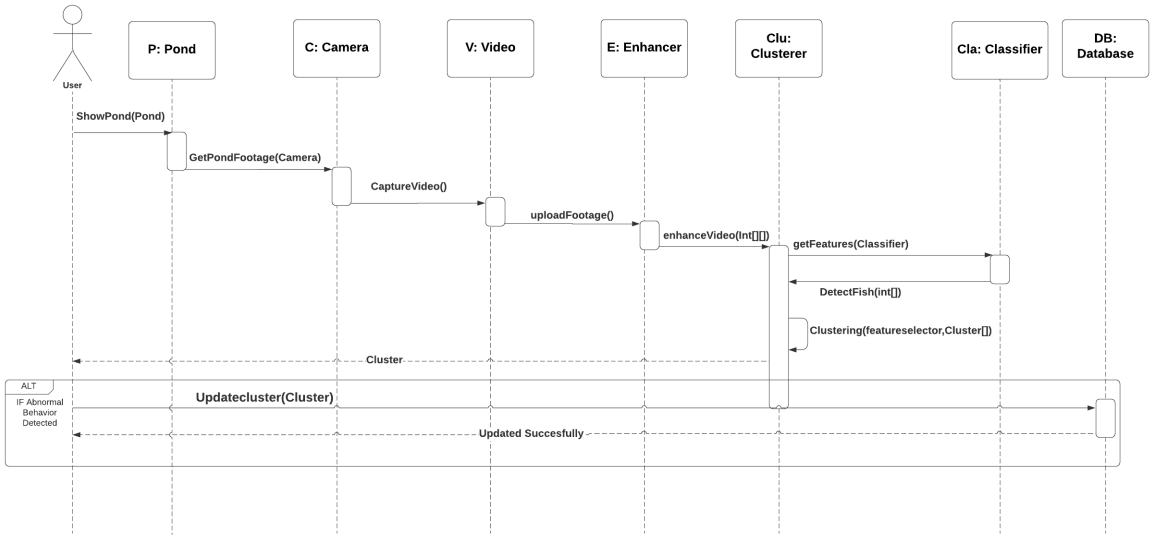

Figure 15: Clustering Sequence Diagram

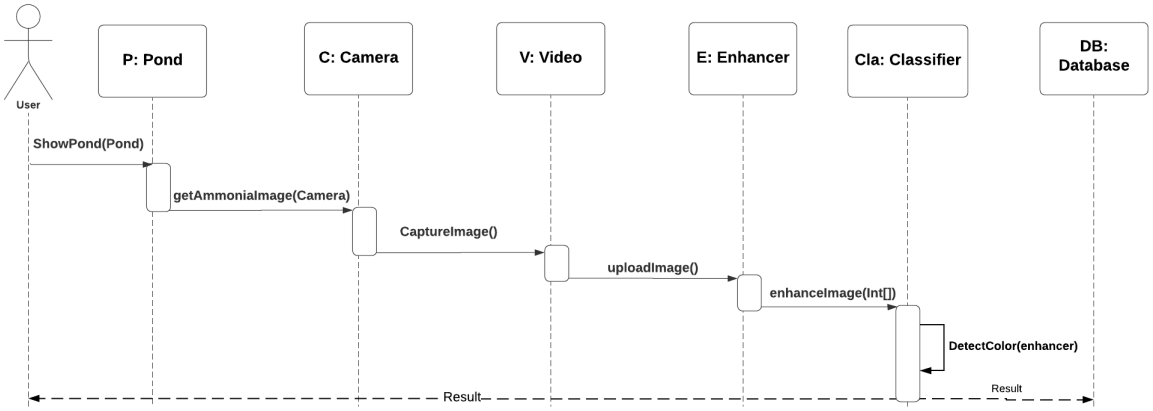

Figure 16: Ammonia Detection Sequence Diagram

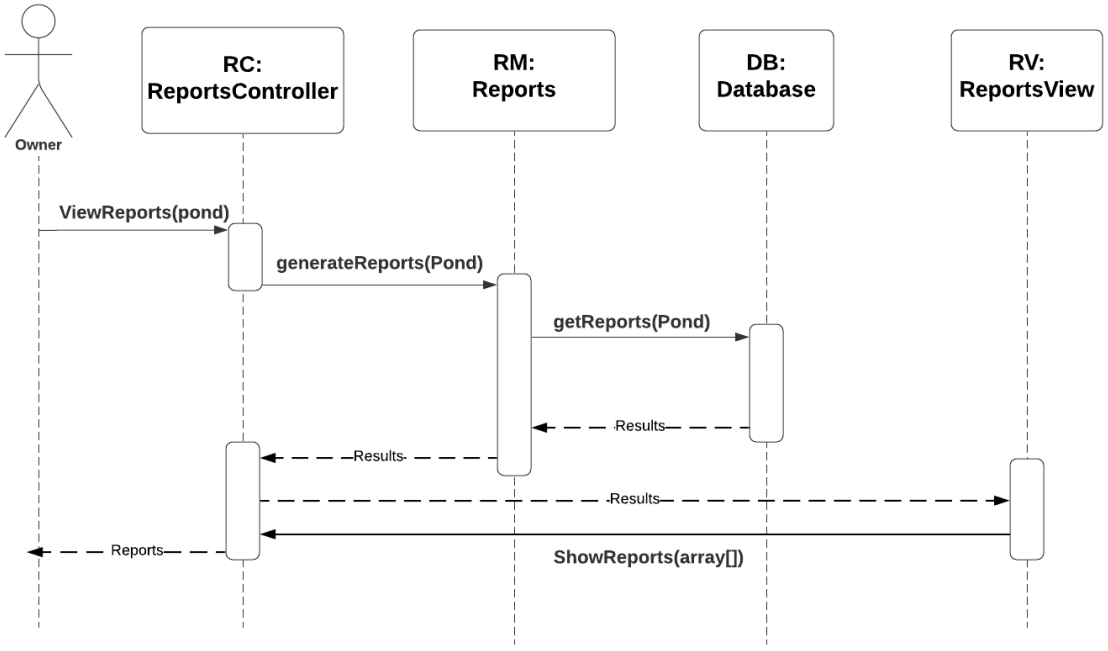

Figure 17: Reports Sequence Diagram

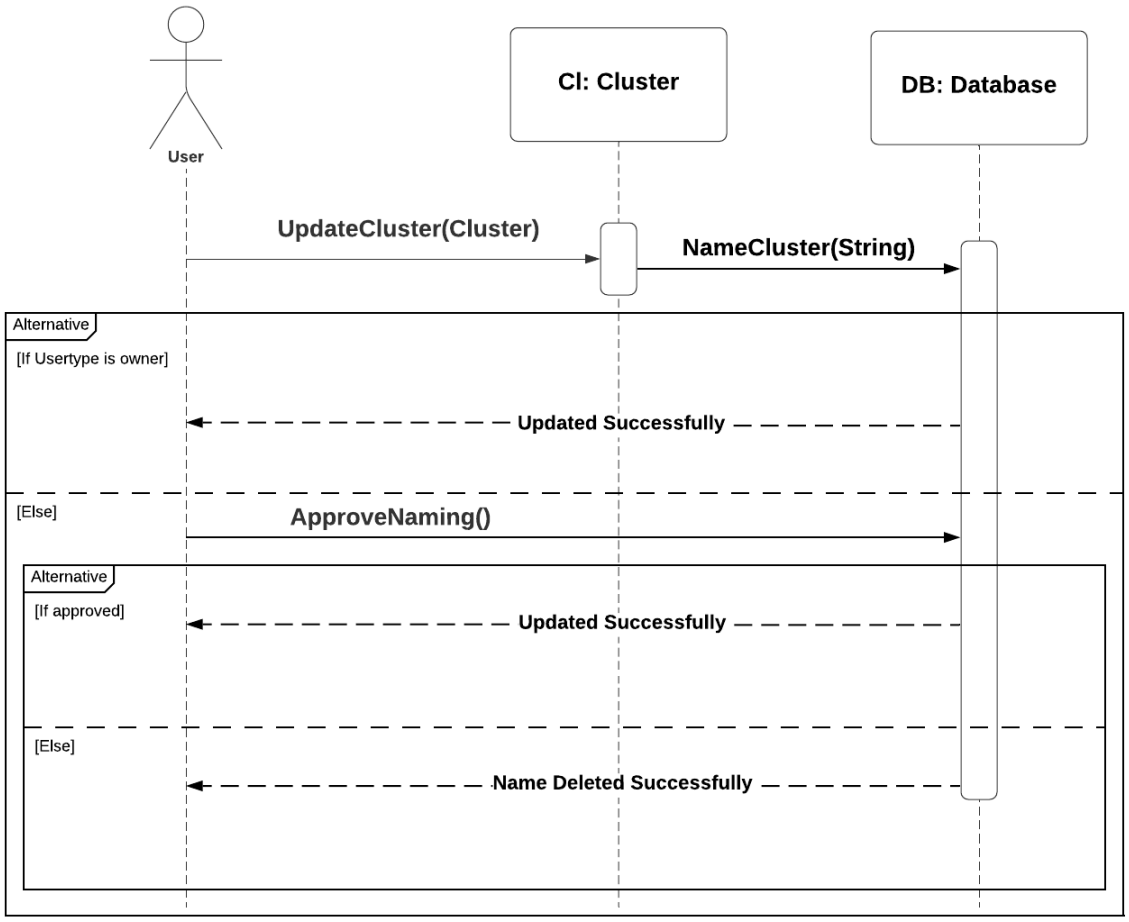

Figure 18: Behavior Renaming Diagram

3.7 Design Rationale

We use the Model-View-Controller (MVC) architecture as mentioned before. The main reason we used this architecture is that it provides us with flexibility to change in the code with less complexity. Also, the system is developed for fis farms that need accurate and reliable results. We had to choose between different algorithms that are applied in different phases in the system (image enhancement, classification, clustering).

Image Enhancement Algorithms:

- ACE: This algorithm is based on an approach that merges the gray World and white patch mechanisms, while taking into account the spatial distribution of color information.
- MSRCR: The algorithm is based on retinex theory which is greying out the image, either the whole image or in specific regions while applying color restoration mechanism to avoid desaturation of the image.
- MSRCP: This algorithm is also based on retinex but with slight difference in the color preservation techniques which fixes some parameters that needed to be put randomly.

- We chose the MSRCR algorithm to enhance our images after testing with some images with the other algorithms. The MSRCR algorithm was the best enhancement algorithm among the others that gives the better results regarding the enhancement quality of the picture.

Classification Algorithms:

- R-CNN: The main idea of this algorithm is using selective search, it identifies a number of bounding-box objects (region of interest) and then it extracts CNN features from each region independently for classification.
- Fast R-CNN: It's a better version of R-CNN algorithm. The difference is instead of extracting CNN feature independently it combines them into one CNN over the entire image which make it faster than R-CNN.
- YOLO: This algorithm works by applying a single neural network to the image as a whole. Then, the network divides the image into regions and predicts the probabilities for each region.

- The YOLO algorithm was chosen to detect objects (fish) in our system as it is better and faster than competitors like R-CNN or Fast R-CNN as it uses only uses one neural network for its predictions, unlike R-CNN which needs thousands of neural networks for a single image.

Clustering Algorithms:

- Hierarchical clustering: The algorithm starts by treating each observation as a separate cluster. Then, it continuously executes the following two steps: identify the two clusters that are closest together, and merge the two most similar clusters. This continues until all the clusters are merged together.

- K-means: The algorithm is based on calculating the mean of the data to get the center of a cluster.

- K-medoids: The algorithm is based on sorting the data and taking the middle one as the center. It take each data and compute its distance with the other. the we select the diamond with the minimum distance.

- We chose the K-medoids algorithm as it is more robust algorithm than others. Also, it suits our situation in detecting different behaviors of fish as in unsupervised learning.

4 Data Design

4.1 Data Description

Our Data will be stored in a database using MySQL phpmyadmin.
Some of the database tables are explained as follows :
1.User : Contains all user data required to make the user access the system.
2.Alert : Contains all information about alerts to the farmers.
3.Behavior : Contains all fish behaviors that are detected by the system including their name.

Database Scheme:

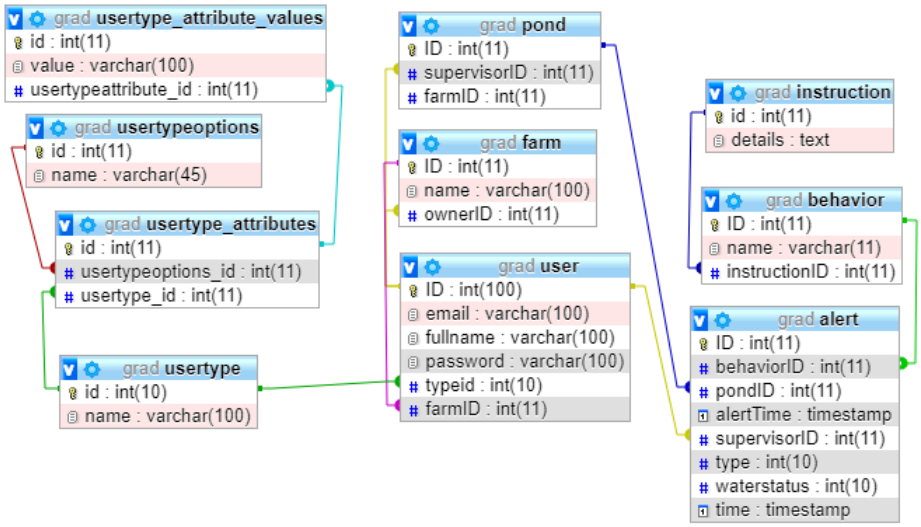

Figure 19: Database scheme

4.2 Data Dictionary

Table	Column	Type
User	id	int(100)
	email	varchar(100)
	fullname	varchar(100)
	password	varchar(100)
	typeid	int(10)
	farmID	int(11)
Alert	id	int(11)
	behaviorID	int(11)
	pondID	int(11)
	alertTime	timestamp
	supervisorID	int(11)
	type	int(10)
	waterstatus	int(10)
	time	timestamp
Behavior	id	int(11)
	name	varchar(11)
	instructionID	int(11)
Instruction	id	int(11)
	details	text
pond	id	int(11)
	supervisorID	int(11)
	farmID	int(11)
farm	id	int(11)
	name	varchar(100)
	ownerID	int(11)
usertype	id	int(10)
	name	varchar(100)
usertype_attributes	id	int(11)
	userTypeOptions_ID	int(11)
	userType_ID	int(11)
usertype_attribute_values	id	int(11)
	value	varchar(100)
	userTypeAttribute_ID	int(11)
usertypeoptions	id	int(11)
	name	varchar(45)

Table 1: Database tables

5 Component Design

In this section we describe the blocks of the system in detail. The blocks of the system are data acquisiton, preprocessing, classification and clustering.

5.1 Data Acquisition

This is the first phase were we collect our data to be further preprocessed, classified and clustered. The data is collected from a camera placed above the pond which is responsible for visual surveillance of the behaviors of the fish and another camera which is placed underwater and responsible for watching the ammonia paper for any changes in the toxicity levels.

5.2 Preprocessing

In this phase preprocessing takes place where we prepare the images and videos to be taken to the next phase which is classification to better classify and detect fish. Specifically, in this phase we use our chosen image enhancement algorithm to enhance videos and images to better detect fish. Image enhancement is important for our system as it provides better visualization for the turbid water images. As shown in the figure below it shows the difference between before enhancement (left image) and after enhancement (right image).

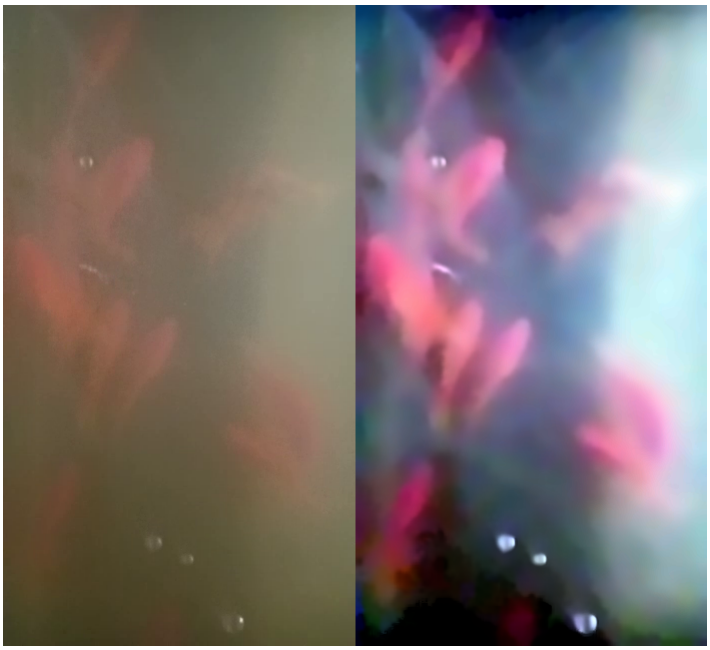

Figure 20: Left: Before Enhancement , Right: After Enhancement

5.3 Classification

This phase is essential for the system as it acts as an intermediate between preprocessing and clustering phases. The classification phase is divided into two parts. Firstly, the part were we use an object detection algorithm to detect fish which are then used to extract features. Secondly, the part were we use threshold color detection algorithm to detect different colors of ammonia paper to alert farmers if ammonia reaches toxic level.

5.3.1 Object Detection

In this section, we describe the object detection algorithm used to detect fish in fish farms. Object detection is important for the system as it detects fish where we can extract features from it and then do the clustering part. For this, we used YOLO to detect objects (fish) which has an acceptable real-time accuracy. The algorithm is one of the regression-based object detection algorithms where it estimates the classes and region of interests for the image in a single run for the algorithm. After detecting fish, we use the bounding box (region of interest) in each single frame of the video to extract from it some features like the coordinates, speed, direction,.. etc. As shown in the figure below an example of fish detection by YOLO.

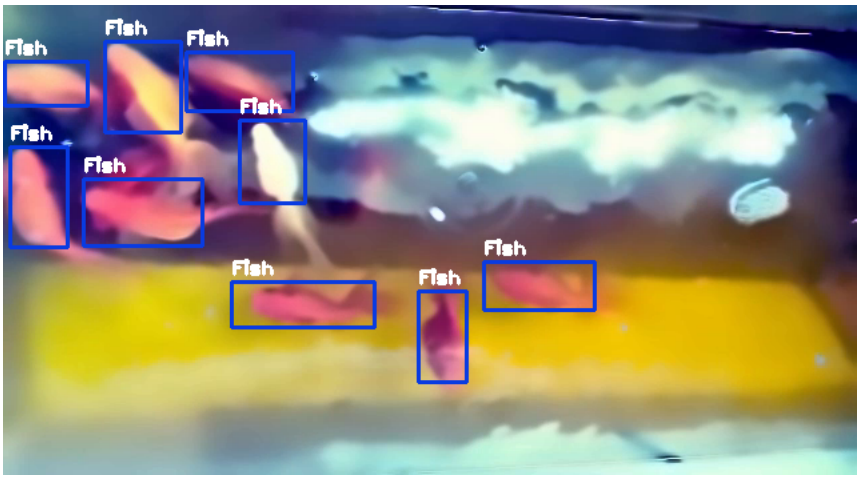

Figure 21: Fish Detection

5.3.2 Color Detection

In this section, we apply a threshold color detection algorithm to detect different stages of ammonia in the water. Ammonia detection is essential for fish farms as when it reaches toxic levels it leads to fish death. The algorithm works as follows. Firstly, it takes upper and lower BGR pixels to define which region of colors it should take and which it can discard. After that, giving the upper,

lower and the image to a function that proceeds to perform a binary mask on the image where the white pixels are the detected region and black pixels are the discarded ones. Finally, the binary image is passed to another function that takes the original input image and the binary image to retain the colors of the detected object. As shown in the figure below this is an example of detecting the color of toxicity in the ammonia paper.

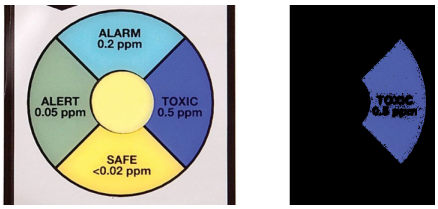

Figure 22: Left: Orginal Image , Right: Detected Region

5.4 Clustering

In this section, the clustering algorithm we used will be explained in detail. The clustering phase is the main phase in the system as it divides the fish behaviors into different clusters so the fish farmer can be alerted when any abnormal behavior occurs. K-Medoids was chosen to be the algorithm used in our system.

5.4.1 K-Medoids Algorithm

K-medoids is a clustering algorithm that partitions the dataset into groups and chooses data points as medoids (centers) for each cluster as its most centrally located point with the least dissimilarity to other objects in the cluster. It can be used in supervised and unsupervised learning techniques.

5.5 Advantages of K-Medoids

Compared to K-means, it is more flexible and robust when it comes to noise and outliers because it minimizes the absolute distance between the points and the selected centroid, as opposed to minimizing the square distance in k-means. A medoid has to belong to the set (cluster), while a centroid doesn't. Also, it is fast and executes in fixed number of steps.
This makes K-medoids a suitable clustering algorithm to accurately cluster the behaviours in our system.

5.6 How K-Medoids Works

The algorithm works in sequence of known steps as follows:

1. Select K random points out of N data points as the medoids.

2. Assign each data point to its closest medoid.

3. Check each cluster for any point that decreases the dissimilarity coefficient, if it does, select the point that decreases it the most as the new medoid for this cluster then repeat step 2.

The following figure shows a flowchart of how the algorithm works.

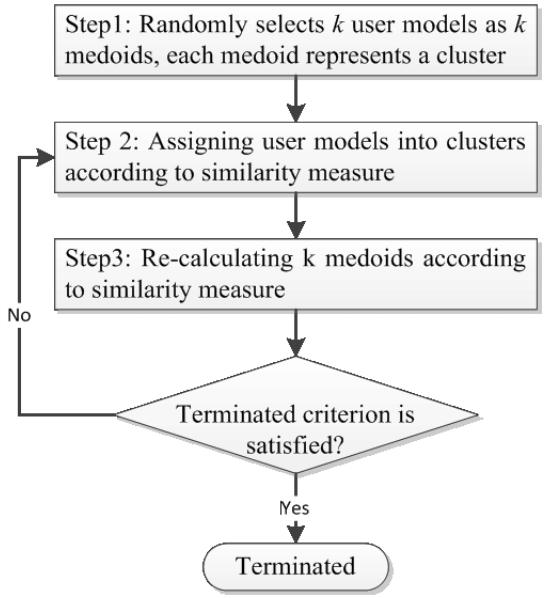

Figure 23: Flow Chart of K-medoids algorithm

6 Humnan Interface Design

6.1 Overview of User Interface

The IFish Farm user interface (UI) is made to be easy to use as the users will not be familiar with using technologies. The user will be able to login as worker or owner depending on his type. Each user type has different tasks to be done thus different screens are shown to each user. The upcoming sections show the screens in detail.

6.2 Screen Images

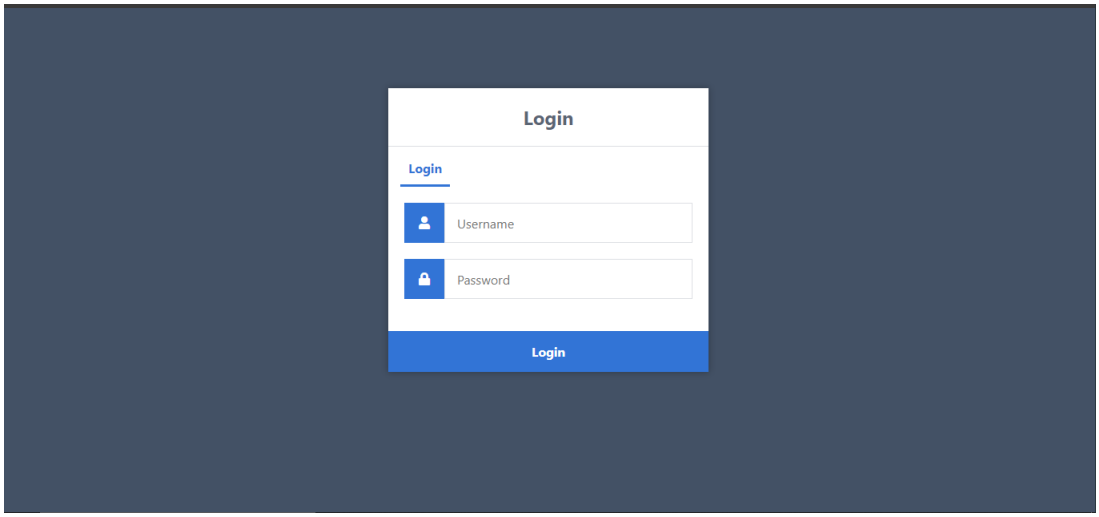

Figure 24: Login Screen

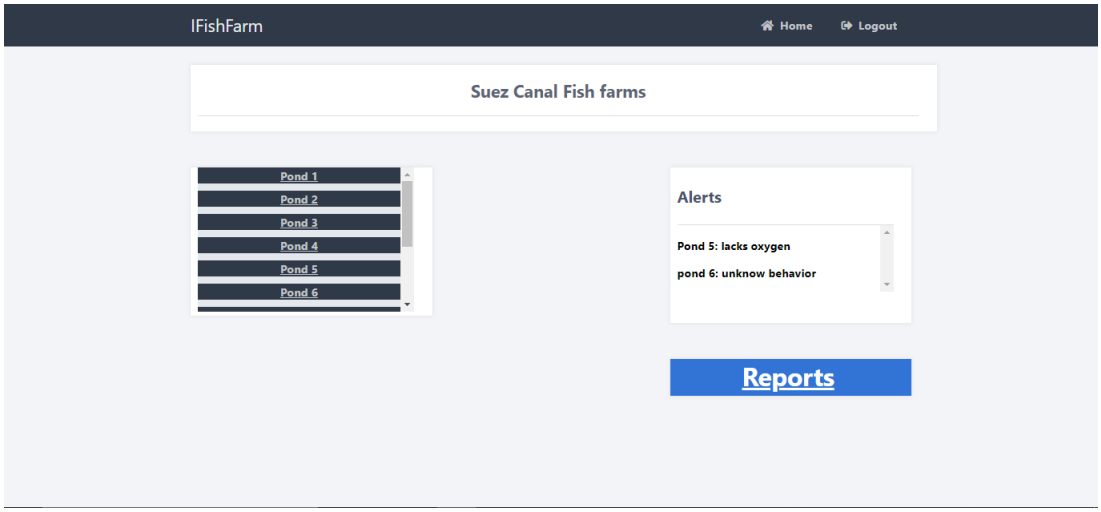

Figure 25: Main Screen

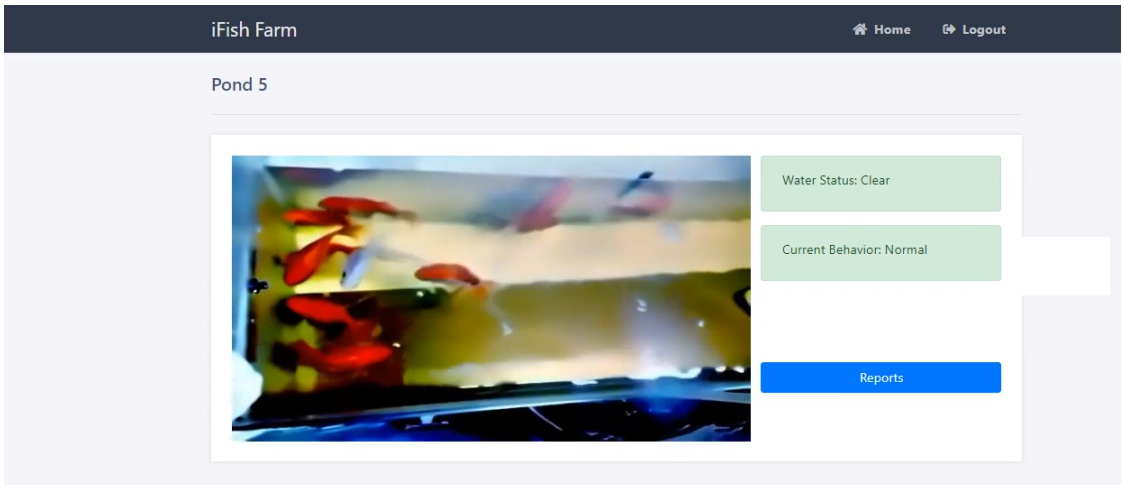

Figure 26: Pond Footage Screen Without any alerts happening

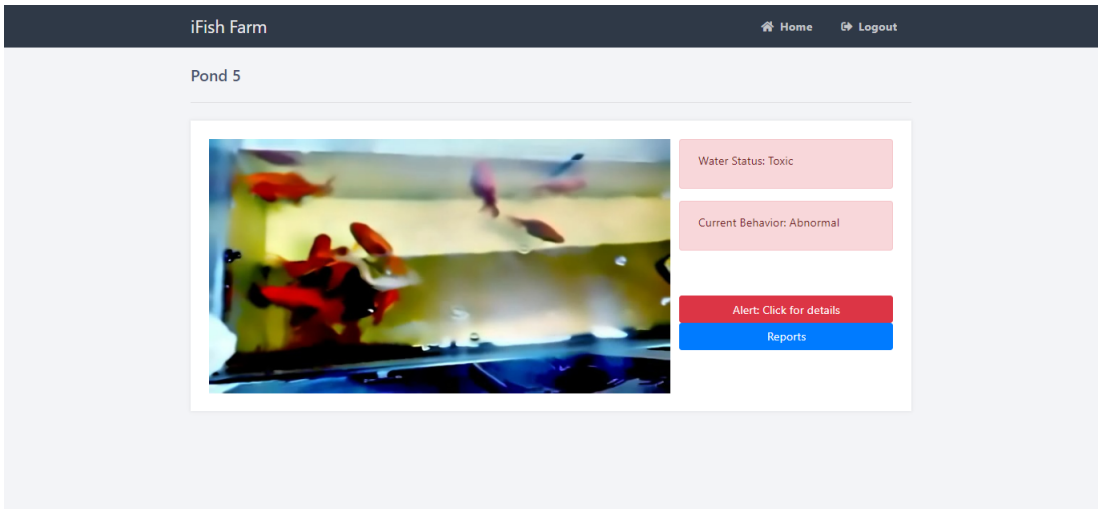

Figure 27: Pond Footage Screen with an alert

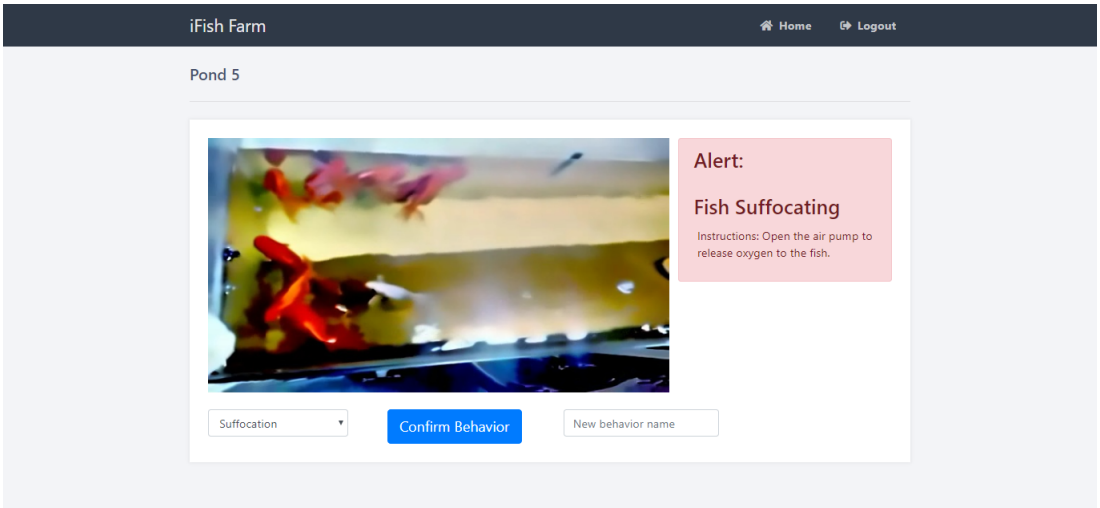

Figure 28: Alert Confirmation Screen

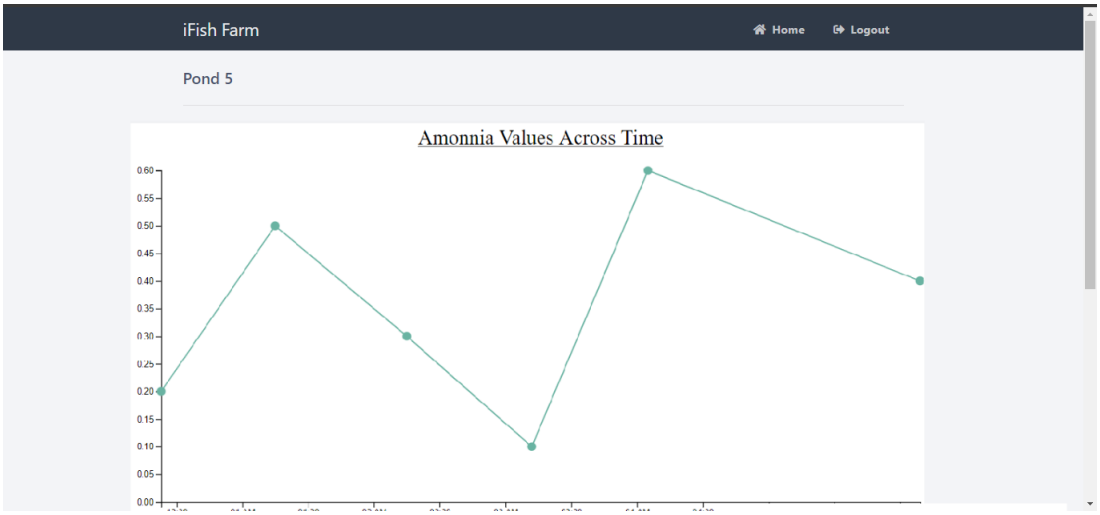

Figure 29: Reports Screen

Figure 30: Reports Screen

6.3 Screen Objects and Actions

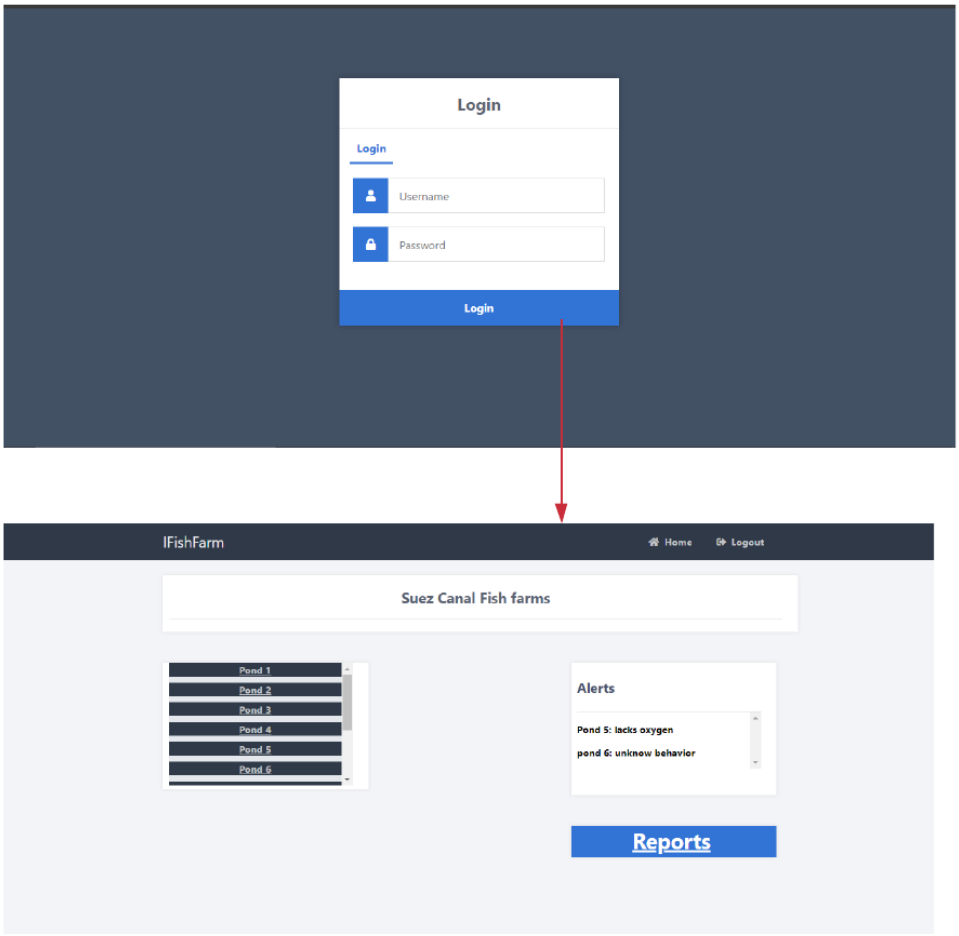

Figure 31: Login Action

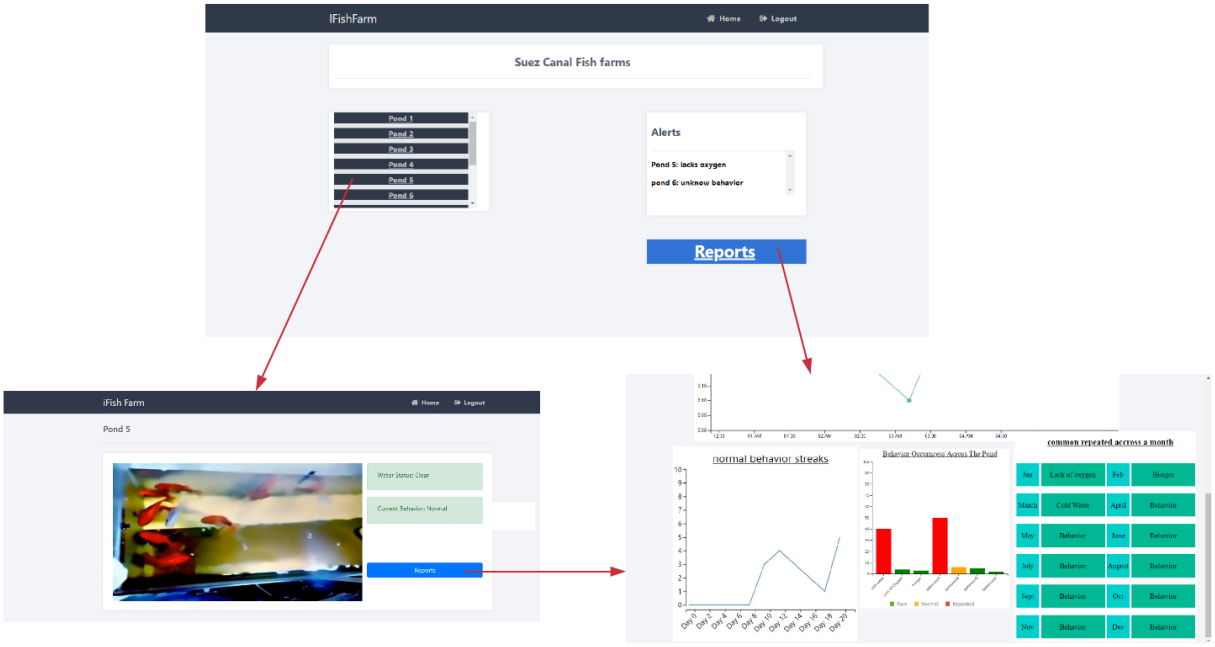

Figure 32: Main Page Actions

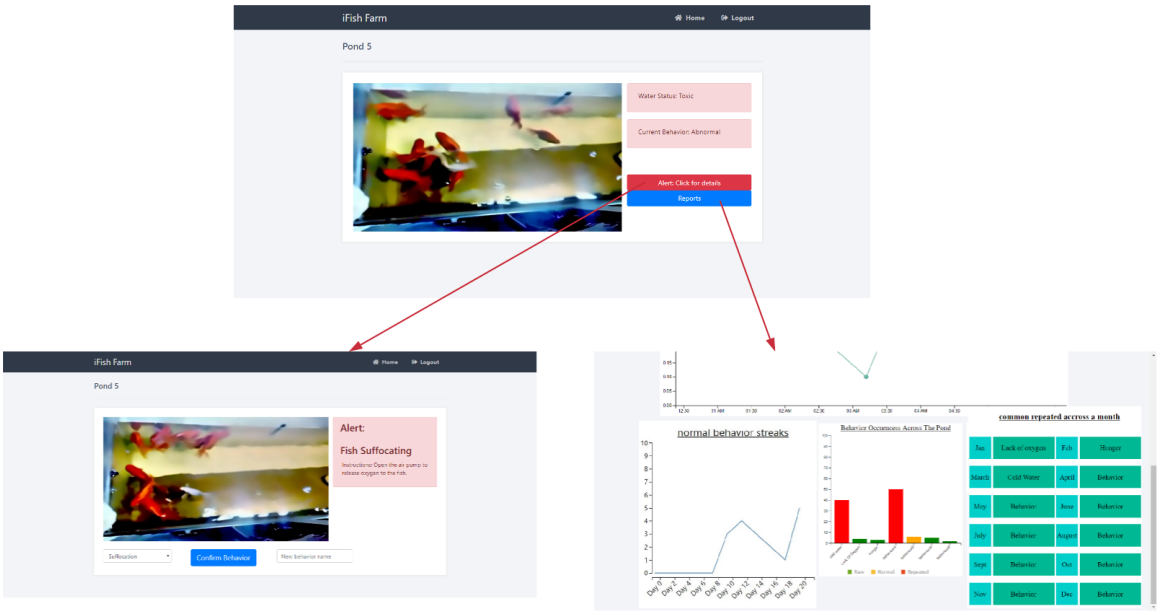

Figure 33: Pond Footage Page actions when there is an alert

7 Requirements Matrix

Requirement ID	Requirement Type	Requirement Name	Module
FR2	Required	Enhance Video	Preprocessing
FR3	Required	Enhance Image	Preprocessing
FR4	Required	Detect fish	Classification
FR6	Required	Clustering	Clustering
FR7	Required	Detect colors	Classification
FR20	Required	Compress Frame	Preprocessing
FR17	Required	Get features	Classification
FR1	Required	Show Pond	Data Acquisiton
FR5	Required	Capture Image	Data Acquisiton
N/A	New Requirement	get Pond footage	Data Acquisiton
N/A	New Requirement	get Ammonia Image	Data Acquisiton

Figure 34: Requirement Matrix Table

References

[1] C. Beyan and R. B. Fisher, "Detecting abnormal fish trajectories using clustered and labeled data," in *2013 IEEE International Conference on Image Processing*, pp. 1476–1480, IEEE, 2013.

[2] A. Nady, A. Atia, and A. Abutabl, "Real-time abnormal event detection in crowded scenes," *Journal of Theoretical and Applied Information Technology*, vol. 96, pp. 6064–6075, 09 2018.

[3] S. Duggal, S. Manik, and M. Ghai, "Amalgamation of video description and multiple object localization using single deep learning model," in *Proceedings of the 9th International Conference on Signal Processing Systems*, ICSPS 2017, (New York, NY, USA), pp. 109–115, ACM, 2017.

[4] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," 2018.

[5] Y. Toh, T. Ng, and B. Liew, "Automated fish counting using image processing," in *2009 International Conference on Computational Intelligence and Software Engineering*, pp. 1–5, IEEE, 2009.

[6] B. J. Boom, P. X. Huang, C. Beyan, C. Spampinato, S. Palazzo, J. He, E. Beauxis-Aussalet, S.-I. Lin, H.-M. Chou, G. Nadarajan, *et al.*, "Long-term underwater camera surveillance for monitoring and analysis of fish populations," *VAIB12*, 2012.

[7] A. Rodriguez, A. J. Rico-Diaz, J. R. Rabuñal, J. Puertas, and L. Pena, "Fish monitoring and sizing using computer vision," in *International Work-Conference on the Interplay Between Natural and Artificial Computation*, pp. 419–428, Springer, 2015.

[8] H. Lu, Y. Li, and S. Serikawa, "Underwater image enhancement using guided trigonometric bilateral filter and fast automatic color correction," in *2013 IEEE International Conference on Image Processing*, pp. 3412–3416, IEEE, 2013.

[9] R. Lumauag and M. Nava, "Fish tracking and counting using image processing," in *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM)*, pp. 1–4, IEEE.

[10] Z. Chen, J. Cao, Y. Tang, and L. Tang, "Tracking of moving object based on optical flow detection," in *Proceedings of 2011 International Conference on Computer Science and Network Technology*, vol. 2, pp. 1096–1099, IEEE, 2011.

[11] N. D. Nguyen, K. N. Huynh, N. N. Vo, and T. Van Pham, "Fish detection and movement tracking," in *2015 International Conference on Advanced Technologies for Communications (ATC)*, pp. 484–489, IEEE, 2015.

[12] C. Tang, U. F. von Lukas, M. Vahl, S. Wang, Y. Wang, and M. Tan, "Efficient underwater image and video enhancement based on retinex," *Signal, Image and Video Processing*, pp. 1–8, 2019.