



MOBEEN TAHIR

01-134182-025

MEHRUN-NISA RAJA

01-134182-024

# Gun Detection System

Bachelors of Science in Computer Science

Supervisor: Jawwad Ijaz

Department of Computer Science  
Bahria University, Islamabad

June 2022



## Certificate

We accept the work contained in the report titled "Gun Detection System", written by Mobeen Tahir & Mehrun-nisa Raja as a confirmation to the required standard for the partial fulfillment of the degree of Bachelors of Computer Science.

**Approved by...:**

Supervisor: Jawwad Ijaz

---

Internal Examiner: Usman Shaffique (Senior Lecturer)

---

External Examiner (Title):

---

Project Coordinator: Dr Moazzam Ali (Assistant Professor)

---

Head of the Department: Dr. Arif ur Rahman (Head of Department)

---

June 23<sup>rd</sup>, 2022



# Abstract

Consistently, a lot of populace accommodates firearm related brutality everywhere. In this work, we foster a PC based completely computerized framework to recognize fundamental deadly implements, especially handguns. Late work in the field of profound learning and move learning has shown huge advancement in the space of item identification and acknowledgment. We have executed YOLOv4 "You Only Look Once" object location model via preparing it on our custom data set. The preparation results affirm that YOLO V4 beats YOLO V3 and traditional convolutional neural network. Moreover, concentrated GPUs or high calculation assets were not needed in that frame of mind as we utilized transfer learning for preparing our model. Applying this model in our observation framework, we can endeavor to save human existence and achieve decrease in the pace of murder or mass killing. Moreover, our proposed system can likewise be carried out in top of the line observation and security robots to distinguish a weapon or dangerous resources to stay away from any sort of attack or endanger to human existence.

# Acknowledgments

In the name of Allah, the most beneficent and the most merciful. We are highly grateful to the One who created us and blessed us with a privileged life. We would like to thank our parents who have always been a pillar of strength and support. We are thankful to our parents who taught us that how hard work, devotion, and working towards your goal with pure intention can take you to places.

Furthermore, we would like to thank and appreciate our supervisor Sir, Jawwad Ijaz who has given us a chance to work on a project that can help in creating value for our beloved country. His professional guidance, time, and effort will always be remembered. Last but not the least, we would like to appreciate our friends who make studying and hard times less challenging. May ALLAH (SWT) guide us to the right path.

MOBEEN TAHIR & MEHRUN-NISA RAJA  
Islamabad, Pakistan

June, 2022

*“Hard work beats talent when talent doesn’t work hard.”*

Tim Notke





# Table of Contents

<b>Abstract</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Problem Description . . . . .	2
1.3 Objective . . . . .	2
1.4 Project Scope . . . . .	2
1.5 Limitations . . . . .	3
<b>2 Literature Review</b>	<b>4</b>
2.1 A traditional Computer Vision Technique . . . . .	4
2.2 Deep Learning . . . . .	6
2.2.1 Convolutional Neural Network (CNN): . . . . .	6
2.2.2 Single Shot Dectector . . . . .	7
2.3 VGG16 . . . . .	8
2.4 Transfer Learning . . . . .	9
2.4.1 YOLO . . . . .	9
2.4.2 Machine Learning VS Deep Learning . . . . .	10
2.4.3 Comparison of Discussed Techniques . . . . .	12
<b>3 Requirements Specifications</b>	<b>15</b>
3.1 Existing System . . . . .	15
3.2 Proposed System . . . . .	15
3.3 Functional Requirements . . . . .	17
3.4 Non-Functional Requirements . . . . .	17
3.5 Hardware Requirements . . . . .	17
3.6 Software Requirements . . . . .	18
3.7 Use Cases . . . . .	19

<b>4</b>	<b>System Design</b>	<b>25</b>
4.1	System Architecture . . . . .	25
4.2	System Architecture Diagram . . . . .	26
4.3	Design Constraints . . . . .	27
4.4	Design Methodology . . . . .	27
4.5	High Level Design . . . . .	27
4.5.1	Sequence Diagram . . . . .	27
4.5.2	Activity Diagram . . . . .	29
4.6	Low Level Design . . . . .	30
4.6.1	Web Application . . . . .	30
4.6.2	Client-Side App . . . . .	34
4.7	GUI Design . . . . .	38
4.7.1	Usability Principles . . . . .	38
4.8	External Interfaces . . . . .	39
<b>5</b>	<b>System Implementation</b>	<b>41</b>
5.1	Introduction . . . . .	41
5.2	System Architecture . . . . .	41
5.2.1	Object Detection . . . . .	42
5.2.2	YOLO . . . . .	42
5.3	Tools And Technologies . . . . .	43
5.3.1	AWS S3 . . . . .	43
5.3.2	Twilio . . . . .	43
5.3.3	Roboflow . . . . .	44
5.3.4	Google Colab Notebook . . . . .	44
5.3.5	Visual Studio Code . . . . .	44
5.3.6	Adobe XD . . . . .	44
5.3.7	QT Designer . . . . .	44
5.4	Experimental Setup . . . . .	45
5.4.1	Dataset Collection . . . . .	45
5.4.2	Labelling Dataset . . . . .	46
5.4.3	LabelImg . . . . .	46
5.4.4	TXT File . . . . .	47
5.4.5	Class Labels . . . . .	48
5.5	Methodology . . . . .	49
5.6	Model Training . . . . .	50
<b>6</b>	<b>Testing</b>	<b>52</b>

6.1	System Testing . . . . .	52
6.2	Functional testing . . . . .	52
6.3	Interface testing . . . . .	53
6.4	Usability testing . . . . .	54
6.5	Compatibility testing . . . . .	54
6.6	Performance testing . . . . .	54
6.7	Testing Strategies . . . . .	54
6.7.1	Black Box Testing . . . . .	54
6.7.2	Specification Testing . . . . .	55
6.7.3	White Box Testing . . . . .	55
6.8	Testing Performance Test Cases . . . . .	55
6.9	Testing Usability Test Cases . . . . .	56
6.10	Test Cases . . . . .	56
6.10.1	Test Case 1 : Registration . . . . .	56
6.10.2	Test Case 2 : Log In . . . . .	57
6.10.3	Test Case 3 : Gun Detection . . . . .	57
6.10.4	Test Case 4 : Saving Snapshot . . . . .	58
6.10.5	Test Case 5 : Alert Generation . . . . .	58
6.10.6	Test Case 6 : Notification . . . . .	59
6.11	Limitations . . . . .	60
<b>7</b>	<b>Conclusion</b>	<b>61</b>
7.1	Future Works . . . . .	62
<b>A</b>	<b>User Manual</b>	<b>63</b>
	<b>References</b>	<b>72</b>

# List of Figures

2.1	Single Shot Detector Multi-Box Detector . . . . .	8
2.2	YOLOv3 . . . . .	10
3.1	Guns . . . . .	16
3.2	Water Guns . . . . .	16
3.3	Use Case . . . . .	19
3.4	Registration . . . . .	20
3.5	Log In Process . . . . .	21
3.6	Monitoring Process . . . . .	22
3.7	Saving Snapshot . . . . .	23
3.8	Notification Process . . . . .	24
4.1	Client-Server Architecture . . . . .	25
4.2	System Architecture . . . . .	26
4.3	Sequence Diagram . . . . .	28
4.4	Activity Diagram . . . . .	29
4.5	Sign Up . . . . .	30
4.6	Log In . . . . .	31
4.7	Password Reset . . . . .	31
4.8	Password Reset Sent . . . . .	32
4.9	Enter New Password . . . . .	32
4.10	Password Reset Complete . . . . .	33
4.11	Alert Dashboard . . . . .	33
4.12	Alerts . . . . .	34
4.13	Log In Window . . . . .	35
4.14	Settings Window . . . . .	36
4.15	Detection Window . . . . .	37
5.1	System Architecture . . . . .	42

5.2	Guns . . . . .	45
5.3	Guns . . . . .	46
5.4	Labellmg . . . . .	46
5.5	TXT File . . . . .	47
5.6	Class Labels . . . . .	48
5.7	Accuracy of our Model . . . . .	51
6.1	Interface Testing . . . . .	53
A.1	Client Side Login . . . . .	63
A.2	Client Side Login . . . . .	64
A.3	Client-Side Settings Window . . . . .	64
A.4	Client-Side Settings Window . . . . .	65
A.5	Client-Side Detection Window . . . . .	65
A.6	Client-Side Detection Window . . . . .	66
A.7	Client-Side Detection Window . . . . .	66
A.8	Client-Side Detection Window . . . . .	67
A.9	Client-Side Detection Window . . . . .	67
A.10	Client-Side Detection Window . . . . .	68
A.11	Watergun Detection . . . . .	68
A.12	Server-Side Sign Up . . . . .	69
A.13	Server-Side Login . . . . .	69
A.14	Alerts Dashboard . . . . .	70
A.15	Alerts . . . . .	70
A.16	SMS Notification Alert . . . . .	71

# List of Tables

2.1	Machine Learning Vs Deep Learning . . . . .	11
2.2	Comparison of Discussed Techniques . . . . .	14
3.1	Registration Process . . . . .	20
3.2	Log In . . . . .	21
3.3	Monitor . . . . .	22
3.4	Save Snapshot . . . . .	23
3.5	Notify . . . . .	24
6.1	Test Case: Register . . . . .	57
6.2	Test Case: Login . . . . .	57
6.3	Test Case: Gun Detection . . . . .	58
6.4	Test Case: Saving Snapshot . . . . .	58
6.5	Test Case: Alert Generation . . . . .	59
6.6	Test Case: Notification . . . . .	59

# Acronyms and Abbreviations

<b>API</b>	Application Programming Interface
<b>CNN</b>	Convolutional Neural Networks
<b>FPS</b>	Frames Per Second
<b>GPU</b>	Graphics Processing Unit
<b>GUI</b>	Graphical User Interface
<b>ML</b>	Machine Learning
<b>PMMW</b>	Passive Millimeter Wave
<b>SPP</b>	Special Pyramid Pooling
<b>SURF</b>	Speed-up Robust Feature
<b>SIFT</b>	Scale Invariant Feature Transform
<b>UI</b>	User Interface
<b>VGG</b>	Visual Geometry Group
<b>YOLO</b>	You Only Look Once

# Chapter 1

## Introduction

### 1.1 Overview

From many statistics, it can be assumed that the violence rate concerning guns and harmful weapons is increasing every year, becoming a challenge for law enforcement agencies to deal with this issue on time. There are many places where the crime rate caused by guns is very high, especially in places where there are no gun control laws. The early detection of violent crime is of immense importance for citizens' security is one of the main concerns, Pakistan encounters. No doubt that despite having the latest tools and technologies, there are still some places where we lack in security due to which the criminals manage to get away with their crimes committed. Young individuals are being more involved in such criminal activities. Many people die each year from gun-related violence whether it is an individual attack or a massive attack. Research shows that gun is the main weapon used for various crimes like robbery, theft, and rape. The Crime rate in Pakistan is high as we know, Karachi is the hub for such criminals' activities, people live in fear there that at any time of the day, they may be shot or robbed by gunpoint. Karachi being the hub does not mean such activities do not take place in other parts of the country. It is considered a norm to be robbed. Criminals due to this take advantage of the weak security and mostly remain successful in their mission, knowing that they won't be either identified or caught and the loss of the people is never recovered. Our final year project deals with these issues by working on the security system by building a Gun Detection System. The weapon detection system will detect the weapon that the criminal or any individual possesses in a closed environment using machine learning. Once detection of the weapon is successful, a web application will be built to notify the administration of that particular area of the unusual possession of a weapon by



an individual so that an action can be taken immediately. Further more, our project differentiates between a toy gun/water gun and the other usual gun.

## 1.2 Problem Description

The problem identification is the ability of any mishap to happen and not being able to report to the incident or identify any suspicious activity is due to the security loopholes we still have. Criminals get away with robbery, theft, breaking through and, rape. In Pakistan, few individuals also tend to possess a weapon/gun without any license, which they tend to use in rage for personal rivalries which results in loss of life. Attacks in educational institutes have become even much more common, terrorist attack and attacks on institutional faculty over rage and many students also tends to carry a weapon and do not fear the consequences of possession and usage of a weapon. Our project thus aims to tackle all these issues, by on-spot identification of the gun possessed by the criminal. It will notify the administration of the organization on their specific security by building a web application. Along with this, also for future reference, a snapshot of the scene will be saved.

## 1.3 Objective

“Our objective is to develop a prototype of a gun detection system identifying guns. The main concern is to make sure to identify a gun and differentiate it from a toy gun/water gun, being possessed by any individual who is on the selected premises for now. For this system we will be building a web application that will serve the purpose of detecting the gun and an alert will be generated on it only if the gun is detected.No alerts to be generated if the gun is a water gun/toy gun. The criminal activity will be further investigated based on the snapshot of the incident that will be saved along with the alert.”

## 1.4 Project Scope

The scope of this project is to provide a better security in the premises of the selected area and attempt to design and develop a system that can detect the guns in no time with less computational resources. It is evident from technological advancements that most of the human-assisted applications are now automated, and are computer-based. Our project aim's to detect the guns in no time with less computational resources. Additionally, the security officials will also get access to the

system through an application. The training and testing will be based on the input data set that is the images of the gun and the storage of data will be in the form snapshot of a video clip of the entire scenario.

## 1.5 Limitations

- Internet Connection
- Camera Result
- Light Factor

## Chapter 2

# Literature Review

Being able to provide high security and reduce criminal activities which tend to be life-threatening becomes challenging at any place. Due to this, criminal activity and its identification have become the main concern for all the organizations to be able to control the harmful activities being carried out by the criminals. This is the reason that for over many years researchers have been working and finding out different ways and techniques to be able to detect these activities with the help of the advancements in the computer field. In doing several researchers have contributed to the idea of observing the behavior and activities using object detection. Our project is also based on object detection, in which the object will be identified whether the object in the image is a gun, this will ultimately inform us about the possession of a weapon by an individual which will result in evaluating the criminal activity that is being performed. To develop the framework of a system, the process is broken down into three main levels: firstly, to extract low level information, secondly to detect the object or track or identify some unusual activity and thirdly, the high level involves making decision on the disturbance or unusual activity behavior. Over the years there have been several methods and techniques used in detection of a gun such as:

- A traditional Computer Vision Technique
- Deep learning

### 2.1 A traditional Computer Vision Technique

As the technology evolves, methods to solve a solution evolve too. Before there were any advancement in ML, we used a traditional computer vision, image processing

techniques to detect the objects. Researchers have a remarkable work when it comes to following the image processing approach. Author Rohit kumar's [1] contribution to gun detection was in such a way that they proposed a framework which exploited color-based segmentation (Color based segmentation is performed to extract the color related to gun, i.e. black if the gun is of black color.) to eliminate un related object from an image using k-means clustering. Harris interest point detector and Fast Retina Keypoint (FREAK) is used to locate the object (gun) in the segmented images. These two techniques and their similarity to the prototype was used to identify if they detected blob is a gun or not. The authors had used their own created data set that consisted of 65 positive and 24 negative images which had a true positivity rate of 84.26%. Apart from the above mentioned, techniques and processes, there were some other main processes that lead to the gun detection mentioned by author Rohit such as preprocessing which involved removal of different noises from the image which arises during the acquisition or transmission of image and then the next step is to move on to the blob extraction to extract a blob from a segmented image. Moving onwards to morphological closing and boundary extraction, it is used to perform to eliminate the presence of small gaps in the blob and then boundary of blob is extracted in order to obtain the shapes of the gun or the object specified. Here the author introduced a SURF Feature extraction methodology which is faster than other interest point detector like SIFT, to extract features from identified different objects and hence compared with the basic standard features of the gun(object), which identifies if the detected thing is a object or not . The authors had used their own created data-set that consisted of 15 positive and 10 negative images which had a true positivity rate of 86.67%. Concealed Weapon Detection [2] by Bhafna Khajone follows several step and techniques that were used to detect a concealed weapon. Firstly, Infrared sensor images were used as input, secondly the process begins with the image being denoised meaning that noise to be removed from the image. After the noise has been removed, the authors used a wavelet technique to enhance/clarify the image. Once enhancement is done, the fusion of the images has to take place, as multiple images of one object exist. Next comes the image decomposition by creating image pyramids. Lastly segmentation takes place that is where the objects are further extracted from decomposed images. Neither the data set nor the positivity rate has been mentioned. All the above-mentioned techniques followed the image processing methods, that did not have better accuracy or efficiency. To bring improvements in the already used methods Machine Learning and Deep Learning became highly famous in order to solve problems related to object detection.

## 2.2 Deep Learning

### 2.2.1 Convolutional Neural Network (CNN):

Deep learning is a subset of machine learning where artificial neural networks, algorithms inspired by the human brain, learn from large amounts of data. Deep learning allows machines to solve complex problems even when using a data set that is very diverse, unstructured and inter-connected. Authors after image processing, used different techniques to solve the issues in detection of the gun. Different approach used by the author Gyanendra K. Verma [3] was of We have used Deep Convolutional Network (DCN), a state-of-the-art Faster Region-based CNN model, through transfer learning, for automatic gun detection from the clustered scene. During training the authors performed the preprocessing of data set images different approach to tackle the same issue. They used Deep Learning based Convolutional Neural Network (CNN), primarily Faster R-CNN (Region based CNN) which uses a combination of region proposals along with convolutional neural networks to extract the features of an image based on the regions, along with labeling those extracted features and then training the entire model using this process by going through the entire training data set. While training the model, pre processing steps were performed by abstracting the mean value of RGB. which is computed on the training data by the authors. Then, they moved along the stack of convolutional layers whose filters are of the size  $3 \times 3$ , which have a convolutional stalk of 1 pixel. Next, by the use of five Max pooling layers, spatial pooling was implemented. Max pooling was performed on a  $2 \times 2$  window with the assistance of 2 strides. Three fully connected layers follow the stack of those convolutional layers, which have varied depth in varied architectures. Soft-max is the final layer and with the rectification non-linearity, the entire hidden layer can be provisioned. For classification multi layered architecture was required for which the author used a technique called gradient decent learning for weight changes in the classification network of CNN. On the basis of the data set used by the authors which had sixty-five positive images (gun present) and twenty-four negative images (gun is not present). The data set was set up such that it comprises of images of various kind of handheld gun with various scale, revolution and orientation. The overall accuracy achieved by us using proposed system before with the traditional image processing was 84.26% which was transformed to 93% by using deep learning. This technique proved to be very effective as the results show. Another approach implemented by author Jose L Salazar [4] who presented a new data set obtained from a real CCTV and synthetic images, to which Faster R-CNN was applied using Feature Pyramid Network with ResNet-50

resulting in a weapon detection model able to be used in quasi real-time CCTV (90 ms of inference time with an NVIDIA GeForce GTX-1080Ti card) improving the state of the art on weapon detection in a two stages training. In this work, an exhaustive experimental study of the detector with these datasets was performed, showing the impact of synthetic datasets on the training of weapons detection systems, as well as the main limitations that these systems present nowadays. The generated synthetic data set and the real CCTV data set are available to the whole research community. The accuracy level achieved was 91.43%. Another approach implemented by author Gulzar Ahmed [5] was that he designed a new Intelligent Ammunition Detection and Classification (IADC) system using Convolutional Neural Network (CNN). The proposed system is designed to identify persons carrying weapons and ammunition using CCTV cameras. When weapons are identified, the cameras sound an alarm. In the proposed IADC system, CNN was used to detect firearms and ammunition. The CNN model which is a Deep Learning technique consists of neural networks, most commonly applied to analyzing visual imagery has gained popularity for unstructured data classification. Additionally, this system generates an early warning through detection of ammunition before conditions become critical. Hence the faster and earlier the prediction, the lower the response time, loses and potential victims. The proposed IADC system provides better results than earlier published models like VGGNet, OverFeat-1, OverFeat-2, and OverFeat-3. Another approach implemented by author Muhammad Tahir Bhatti [6] were sliding window/classification and region proposal/object detection. Some of the algorithms used are VGG16, Inception-V3, Inception-ResnetV2, SSDMobileNetV1, Faster-RCNN Inception-ResnetV2 (FRIRv2), YOLOv3, and YOLOv4. Precision and recall count the most rather than accuracy when object detection is performed so these entire algorithms were tested in terms of them. Yolov4 stands out best amongst all other algorithms and gave a F1-score of 91% along with a mean average precision of 91.73% higher than previously.

### 2.2.2 Single Shot Dectector

Single Shot detector like YOLO takes only one shot to detect multiple objects present in an image using multi-box. It is significantly faster in speed and high-accuracy object detection algorithm. SSD approach had been taken by the authors Aditya Vikram [7] to bring in more accuracy and efficiency. Their paper implemented weapon detection using a convolution neural network (CNN) based SSD and Faster RCNN algorithms. They used two types of datasets. One dataset, which had pre-labelled images and the other one is a set of images, which were labelled manually.

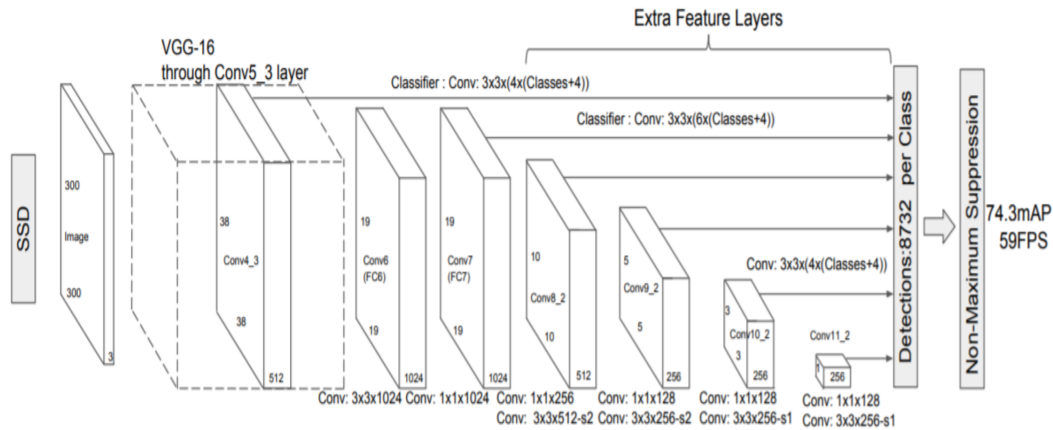


Figure 2.1: Single Shot Detector Multi-Box Detector

SSD speeds up the process by eliminating the need of region proposal network. To increase the efficiency SSD brings a few technologies including default boxes and multi-scale features. These improvements allow SSD to match the Faster R-CNN's accuracy using lower resolution images, which further pushes speed higher. SSD and Faster RCNN algorithms are simulated for pre-labeled and self-created image dataset for weapon detection. Both the algorithms are efficient and give good results but their application in real time is based on a trade off between speed and accuracy. In terms of speed, SSD algorithm gives better speed with 0.736 s/frame. Whereas Faster RCNN gives speed 1.606s/frame, which is poor compared to SSD. With respect to accuracy, Faster RCNN gives better accuracy of 84.6%. Whereas SSD gives an accuracy of 73.8%, which is poor compared to faster RCNN. SSD provided real time detection due to faster speed but Faster RCNN provided superior accuracy.

## 2.3 VGG16

We have authors Justin Lai et al [8] who have tapped into other techniques within deep learning field. They resized the images first to make it compatible with the tensor-box framework. Then by using the scripts they set a bounding-boxes around each gun and after that they reformatted the output scripts to be used as json. They have used VGG-16 classifier and tested on the 200 images of different guns and achieved accuracy of 58% for revolver and 46% on rifle. After that they decided to implement Over-feat through tensor-box and trained by using 20% confidence threshold and a learning rate of 0.0003. This resulted in train accuracy of 93% and test accuracy of 89%. Another approach was implemented by the author Javed Iqbal et al [9] in which they used VG-16 with the combination of Region Proposal

Network (RPN) and Faster R-CNN for object classification. They used data set from multiple sources and achieved an accuracy of 88.3%.

## 2.4 Transfer Learning

Transfer learning is a famous method in computer vision because it allows us to build accurate models in a time saving way. With transfer learning, rather than beginning learning process from zero, you start from images or pattern that exist before by learning when solving a different problem. This way you save time. In computer vision, transfer learning is usually expressed through the use of pre-trained models. A pre-trained model is a model that was trained on a large benchmark data set to solve a problem similar to the one that we want to solve. The approach had been implemented by the authors Mehmet Tevfik Agda et al [10] where they implemented multiple architectures such as Alex-Net, VGG16 and VGG19 with transfer learning to achieve their desired results. The authors used data set from various open access sources. The total images in their data set were 16000 images containing 9500 knives, 3500 guns, and 3000 normal pictures. The authors achieved a greater accuracy value using transfer learning as compared to learning from scratch. They had the following accuracies:

- Alex-Net with 97.74%,
- VGGI 6 with 99.38%
- VGGI 9 with 99.27% which is higher than the highest accuracy they obtained from Alex-Net with 84.58% when learning from scratch.

### 2.4.1 YOLO

We have author Arif Warsi et al [11] who used a deep learning model called Yolov3 on their own dataset of guns with different orientation and position by using imageNet. By using transfer learning for training of YOLOv3 model and weight trained on ImageNet by YOLOv3 team instead of starting from zero. YOLOv3 is an object detection algorithm widely used for real time processing. Input images were divided into  $M \times M$  grids. A single object is then predicted by this grid cell. Logistic regression is used to predict an object scores for each bounding-box by YOLOv3 and changes the method to compute the cost function.

The detection results are examined frame by frame in the videos the highest accuracy they achieved was 98.64%. Another author Lei Pang [12]proposes using a real-time detection method for detecting concealed metallic weapons on the human



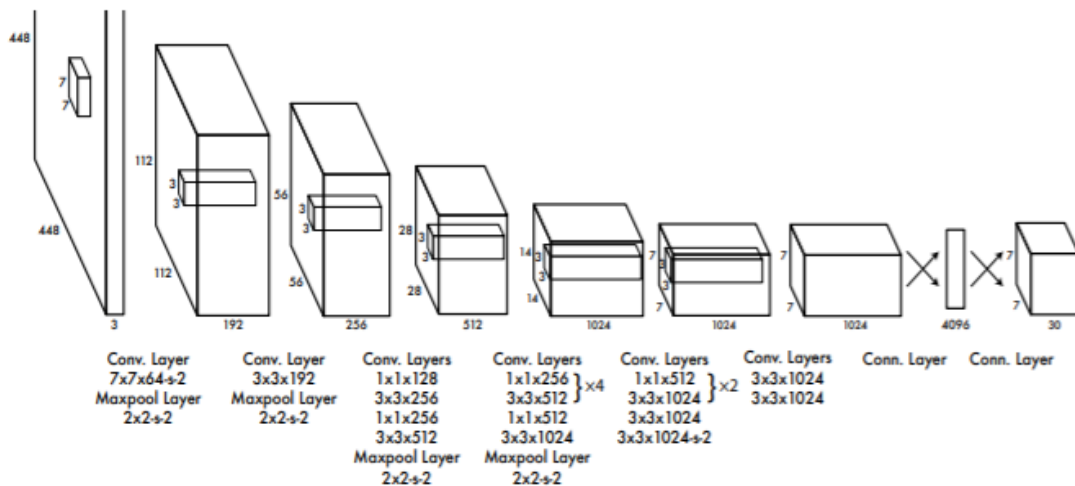


Figure 2.2: YOLOv3

body applied to passive millimeter wave (PMMW) imagery based on the YOLOv3, and a sample dataset. It uses the darknet-53 and darknet-13 networks for feature extraction. Similar to the Visual Geometry Group (VGG) network, it is mainly composed of a set of  $3 \times 3$  and  $1 \times 1$  convolutional layers. The YOLOv3 deepens the convolutional network to extract a greater number of deep features. A short-cut is added to YOLOv3 to build a residual module, thus avoiding gradient disappearance from deep network training. Secondly, it is introduced into the Faster R-CNN, with the idea of inputting the a priori anchor box during model training, but the selection of the anchor box is automatically obtained from the training data by means of k-means clustering. The experiments shown that the YOLOv3-based contraband detection method for the PMMW images and can meet the real-time detection requirements during large passenger flows. In terms of trade of detection accuracy, detection speed, and computation resource, the YOLOv3-53 model is more advantageous and effective, even with an insufficient sample data. Due to the equipment limitations, the available multi-source PMMW image data are limited, and the data needs to be detected and improved. Furthermore, more training tests of various types of contraband should be developed for further PMMW security requirements.

## 2.4.2 Machine Learning VS Deep Learning

The key difference between deep learning vs machine learning stems from the way data is presented to the system. Machine learning algorithms almost always require structured data, whereas deep learning networks rely on layers of the ANN (artificial neural networks). Machine learning algorithms are built to “learn” to do things by

understanding labeled data, then use it to produce further outputs with more sets of data. However, they need to be retrained through human intervention when the actual output is not the desired one. Deep learning networks do not require human intervention as the nested layers in the neural networks put data through hierarchies of different concepts, which eventually learn through their own errors. However, even these are subject to flawed outputs if the quality of data is not good enough.

<b>Machine Learning</b>	<b>Deep Learning</b>
<b>Optimal Data Volumes</b>	
Thousands of Data Points	Bid Data:Millions of Data Points
<b>Outputs</b>	
Numerical Value, like classification or score	Anything from numerical values to free-form element, such as free text and sounds
<b>How it works</b>	
Uses various types automated algorithms that learn to model functions and predict future actions from data	Uses neural networks that pass through many processing layers to interpret data features and relations
<b>How it's managed</b>	
Algorithms are detected by data analyst to examine specific variable in data sets	Algorithms are largely self-directed on data analysis once they're put into production

Table 2.1: Machine Learning Vs Deep Learning.

### 2.4.3 Comparison of Discussed Techniques

Category	Research Paper	Authors	Methods Used	Dataset	Accuracy
Image Processing	A Computer Vision based Framework for Visual Gun Detection using SURF	Rohit Kumar Tiwari	SURF feature extraction	Self-made 15 Positive and 10 Negative images	86.67%
	A Computer Vision based Framework for Visual Gun Detection using Harris Interest Point Detector	Rohit Kumar Tiwari * and Gyanendra K. Verma	Harris interest point detector and Fast Retina Keypoint (FREAK)	Self- made 65 Positive and 24 Negative images	84.26%
	Concealed Weapon Detection Using Image Processing	Bhavna Khajone, Prof. V. K. Shandilya	Image Processing	Not mentioned	None
Deep Learning	A Handheld Gun Detection using Faster R-CNN Deep Learning	Gyanendra K. Verma	R-CNN Deep Learning	Self -made 65 Positive Images and 24 Negative Images	93%
	Real-time gun detection in CCTV	Jose-L-Salazar	CNN Deep Learning Feature Pyramid Network with ResNet-50 Data augmentation	Multiple Datasets (33,500-gun images)	91.43%

	Intelligent Ammunition Detection and Classification (IADC) system	Gulzar Ahmed	CNN	Not specified	96.74%
	Weapon Detection in Real-Time CCTV Videos Using Deep Learning	Muhammad Tahir Bhatti	VGG16, YOLOv3, FRCNN	Different Sources	93%
	Weapon Detection using Artificial Intelligence and Deep Learning for Security Applications	Aditya Vikram	(CNN) based SSD and Faster RCNN	Self-made Database	SSD 73.8% RCNN 84.6%
	Developing a Real-time gun detection in Classifier	Justin Lai et al	VGG16	Self-made Dataset (200 Images)	89%
	Leveraging Orientation for Weakly Supervised Object Detection with Application to Firearm Localization	Javed Iqbal et al	VGG16, RFCNN, RPN	Different Sources (10973 + 13647 Images)	88.3%

	Deep Neural Networks based on Transfer learning	Memet tevfik Agdas et al	Transfer learning	16000 images by different resources	VGG16 99.38% VGG19 99.27% Alex-Net 97.74%
	Gun Detection System using YOLOv3	Arif Warsi et al	YOLOv3	Data-set from ImageNet	98.64%
	Real Time Concealed Object Detection From PMW	Le Pang et al	YOLOv3, SSD, VGG16	Limited Dataset	95%
	Gun Detection in Surveillance Videos using Deep Neural Networks	JunYo Lim et al	M2Det	UCF Crime Dataset (3000 gun images)	80%
	Automatic gun detection Approach for video surveillance	Mai kamal el den Mohamed et al	CNN in combination with AlexNet and GoogleNet	Multiple Sources (13743 Images)	GoogleNet 97.9% AlexNet 99.2%
	Weapon Detection Using YOLO V3 for Smart Surveillance System [13]	Sanam Narejo et al	YOLOv3	Custom Dataset	98.89%

Table 2.2: Comparison of Discussed Techniques

## Chapter 3

# Requirements Specifications

### 3.1 Existing System

In this section we will analyze the existing system. Through our research, we found out that quite a lot of work has been done on weapon detection, but still the accuracy was not high and the application was lagging behind. For the system the previously used techniques such as CNN or Faster R-CNN were not up to today's standard, thus needed a better approach for object detection, due to which YOLO (You Only Look Once) was introduced, which changed the computer vision techniques and set a standard for object detection.

### 3.2 Proposed System

Our proposed system is used to detect a gun from the video and identify if a person possess a gun or not, and differentiate between a toy gun and gun. This procedure/detection will be carried out on the basis of YOLO. Our system will comprise of a web application, that will be accessible to the security officials. The app would be user friendly and to do so an easy and understandable user interface would be built. The system uses few compute vision methods and deep learning for identification of a gun from captured image. For object detection and classification, we trained the classifier model of YOLOv4, i.e, "You Only Look Once" on our customized dataset. Our dataset comprises of guns and waterguns. For our gun detection system, we used our own customized dataset. We collected our dataset from google images, which were about 3000+ images in which specifically choose the weapon of single type (pistol) and of water gun for development purpose, which we will later increase the types of weapon. Following are the some of the images from

our dataset:



(a)



(b)



(c)

Figure 3.1: Guns



(a)



(b)



(c)

Figure 3.2: Water Guns

### 3.3 Functional Requirements

- User should be able to Sign Up or Log In to the system
- User must be able to enter a phone number in the dialogue box
- User should enter location in the dialogue box
- The admin panel should be available to the user for alerts and information
- User must be notified through sms on mobile number
- Camera must be also in a working condition
- Internet also must available
- System should be able to read frames and analyze video.
- System should be able to identify the gun and water gun.

### 3.4 Non-Functional Requirements

- **Availability:** After installation of the application on computer, it will be available for use.
- **Re-usability:** Different modules of the system will be able to be reusable in some other systems.
- **Reliability:** The system should be reliable in detecting a gun from a video.
- **Maintainability:** Developers will be responsible for system maintainability.

### 3.5 Hardware Requirements

The hardware requirements for our system:

- Laptop
- Workstation.
- Internet Connection
- Camera

This system does not need any assistance of extra hardware.



### 3.6 Software Requirements

The development of our project uses the python. The prerequisites software and libraries for this project are:

- **Labelling Directory:** Labelling Directory, a graphic image annotation tool. Qt and Python are used to build its GUI. Annotations can be saved as per our required format (XML, PASCAL, VOC). It is an open source and available to all.
- **Google Colab:** For the training and testing of our model we used google colab. The code on Colab's notebook is executed Google's Cloud Server. This helps you use Google's Hardware regardless of the power of your computer.
- **Visual Studio Code:** We used VS Code for the development of our web application, it is a code editor with support for development operations like debugging, task running, and version control.
- **Adobe XD:** We used Adobe XD for designing the user interface of our web application.
- **QT Designer:** Qt Designer is a tool that provides us to create GUI of client side of our web application productively and efficiently.

### 3.7 Use Cases

**Note :** The use cases may change during SDLC

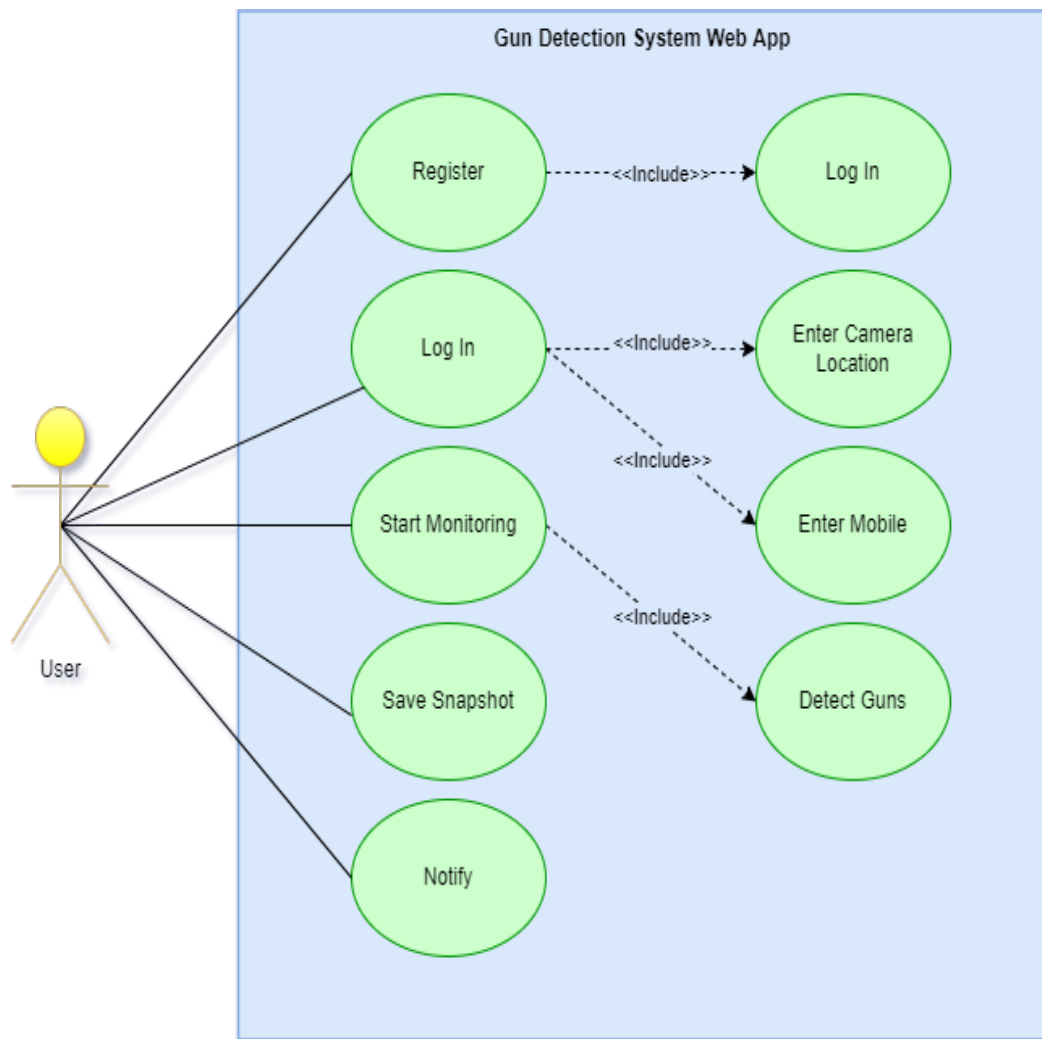


Figure 3.3: Use Case

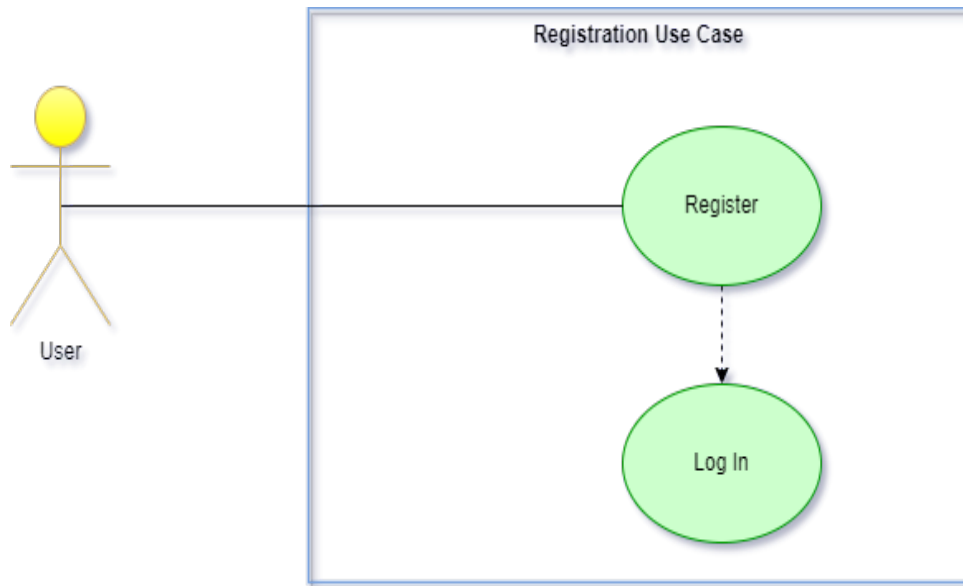


Figure 3.4: Registration

Registration Process	
Actor	User
Description	Use case specifies registration process
Main Success factor	User must be able to register and add in Log in details
Pre-Condition	The application is in running state
Post-Condition	User has been successfully registered

Table 3.1: Registration Process

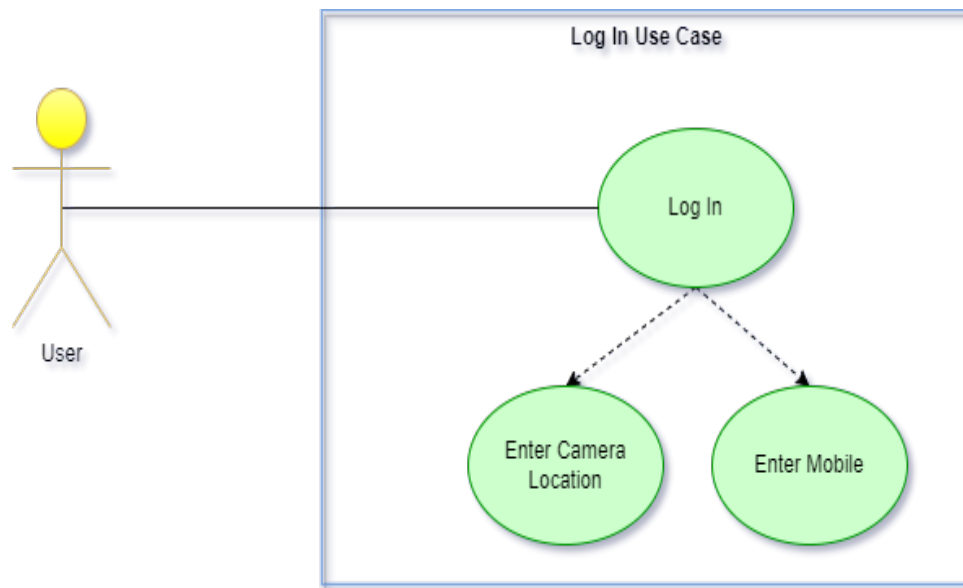


Figure 3.5: Log In Process

<b>Log In</b>	
Actor	User
Description	This use case specifies Log In procedure
Main Success factor	The user must be able to Log In and enter specified details
Pre-Condition	The User should be registered
Post-Condition	User has been successfully Logged In

Table 3.2: Log In

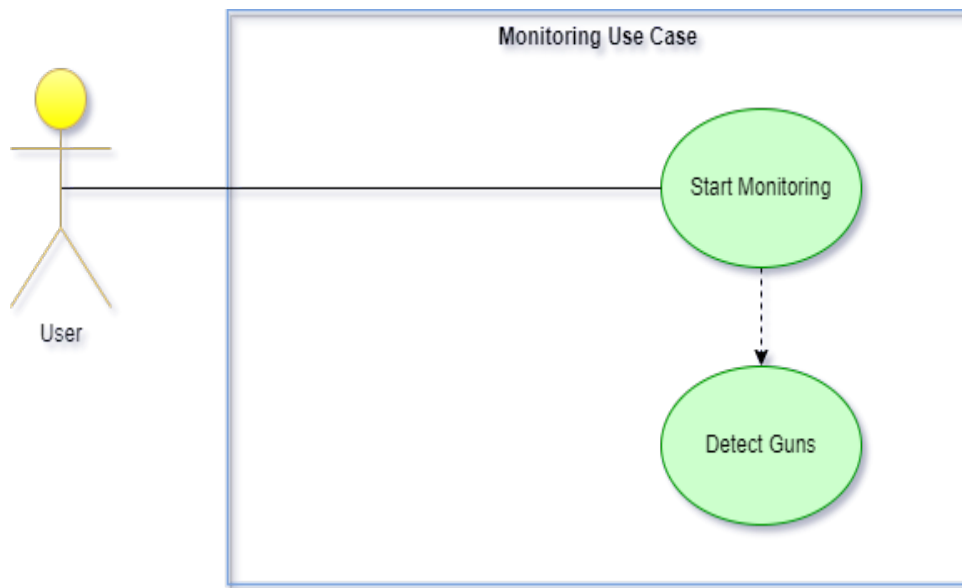


Figure 3.6: Monitoring Process

<b>Monitor</b>	
Actor	User
Description	This use case specifies monitoring procedure
Main Success factor	User must be able to monitor the situation
Pre-Condition	The camera and application should be running
Post-Condition	Gun has been detected

Table 3.3: Monitor

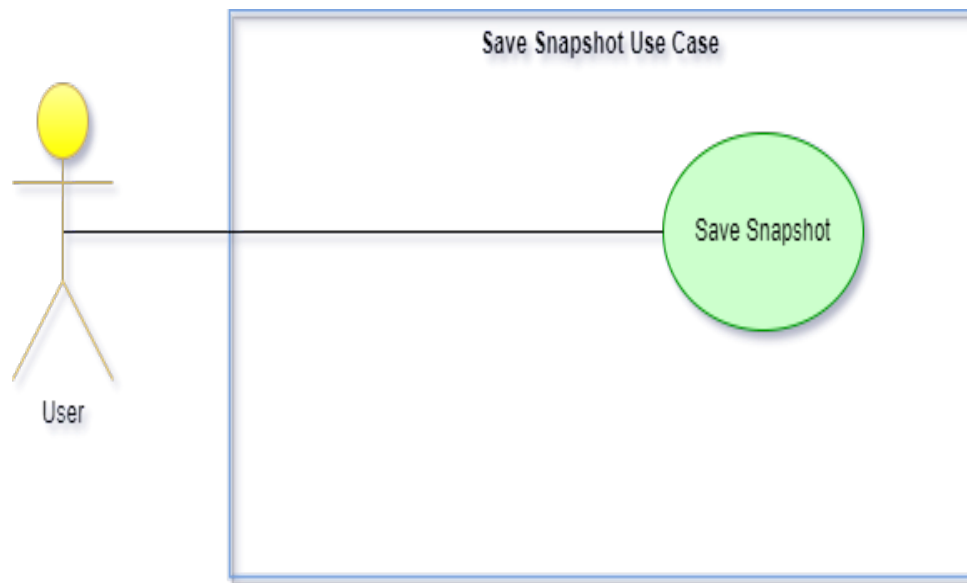


Figure 3.7: Saving Snapshot

<b>Save Snapshot</b>	
Actor	User
Description	This use case specifies saving the frame
Main Success factor	The screenshot to be saved
Pre-Condition	The gun should be detected in frame
Post-Condition	Saved Snapshot

Table 3.4: Save Snapshot

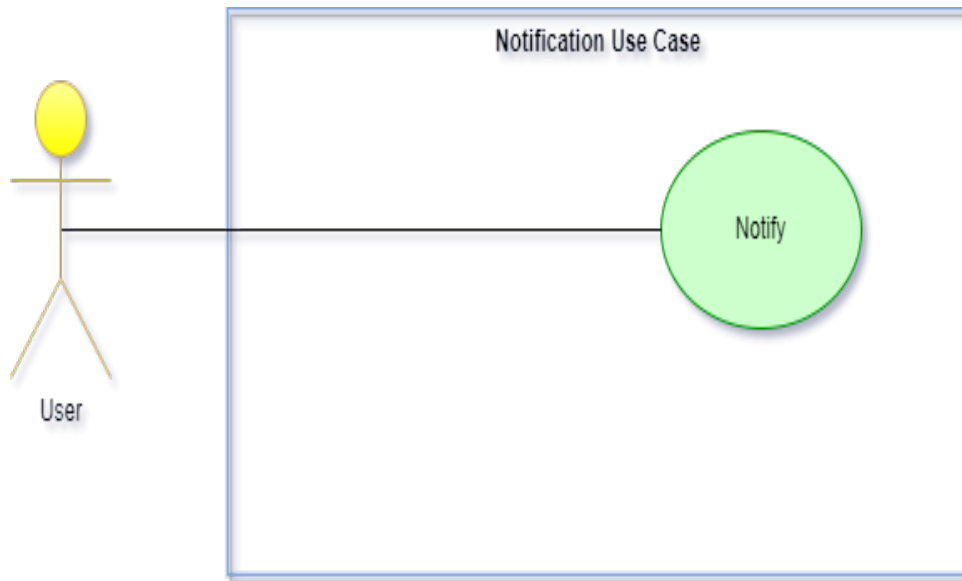


Figure 3.8: Notification Process

<b>Notify</b>	
Actor	User
Description	This use case specifies alert generation
Main Success factor	The notification received through sms
Pre-Condition	The gun should be detected in the scene
Post-Condition	Notified

Table 3.5: Notify

## Chapter 4

# System Design

In this chapter of System Design, which is all about defining component, interfaces and data to reach to our specified requirements, we have a deeper look into the development phases of our gun detection system. This chapter will discuss the system architecture and provide details of the design methodology, constraints, models, interaction between the system and the user.

### 4.1 System Architecture

Our developed system based on 3-tier application architecture. The gun detection is done at the client-side application by each input frame, of both the pistol and water gun. After successful detection of the guns, the saved frame that consists only of the pistol will be alerted to the server side, which stores the information of the detected gun(pistol) in database which is accessed by the server.

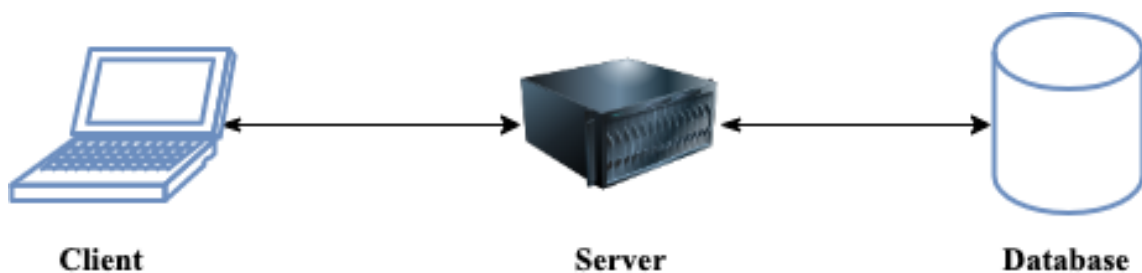


Figure 4.1: Client-Server Architecture



## 4.2 System Architecture Diagram

An architectural diagram is a diagram of a system that is used to deduce the overview of the software system. It is an important tool as it provides an overall view of the physical application of the software system. Our developed system's overall architectural overview is illustrated in the diagram below:

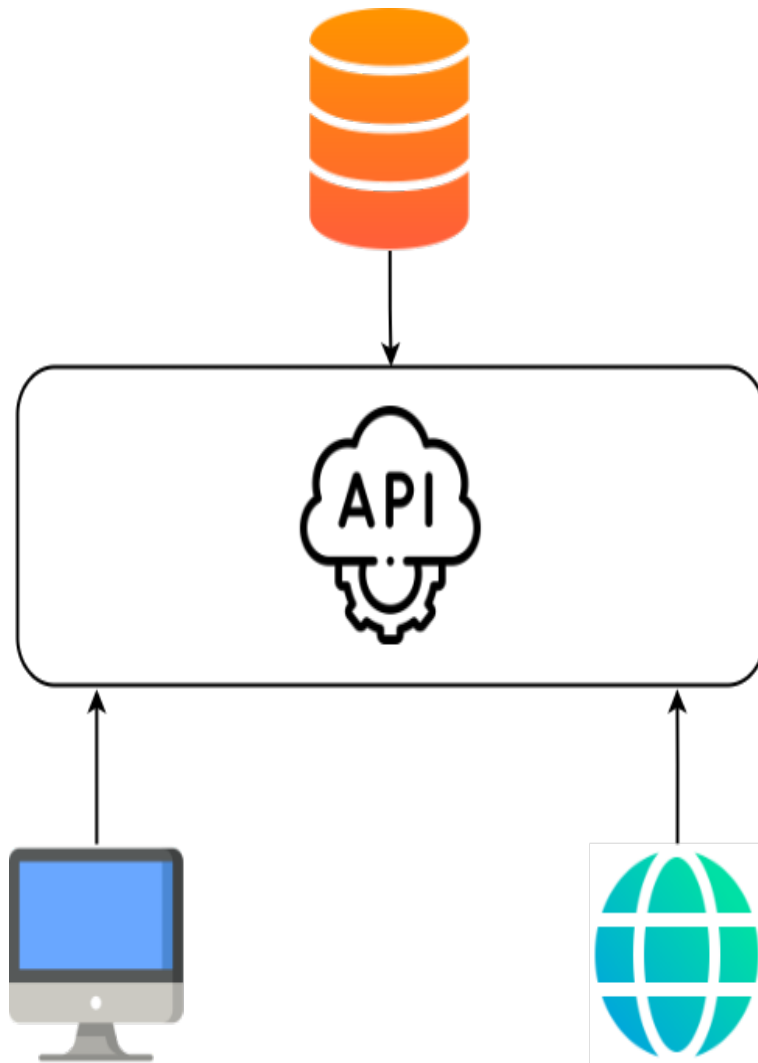


Figure 4.2: System Architecture

## 4.3 Design Constraints

There are some design constraints in our project.

- Our system's user interface must be operational
- Application to be user-friendly
- Response time to be minimum
- Application should be responsive

A constraint that we face on the system is the non-availability of multiple guns' dataset.

## 4.4 Design Methodology

In design methodology we define the procedures and techniques which will help us in designing our system. As we have to develop a web application, this is why we needed some Python (Django) knowledge. The design process always requires iterations and improvements on each level in order to minimize errors in the final product due to which we brainstormed and encouraged new ideas and collaborative thinking to work through each idea and arrive at the best and optimal solution.

## 4.5 High Level Design

High level design provides us a detailed overview of how different functionalities and features coordinate and interact with each other. This section highlights the basic workflow along with alternative paths just in case a failure occurs. Following diagrams are used to demonstrate how different functions of this app will work.

### 4.5.1 Sequence Diagram

A sequence diagram is a type of interaction diagram which describes how and in what order a group of objects works together. They illustrate how different parts of a system will interact with each other to carry out a function, and the order in which the interactions occur when a particular use case is executed.

In our first step, the user will launch the application, then the system will ask the user to enter login credentials or register. After the user has successfully registered, the user will then enter login details. On those login details user will further be asked to enter Camera Location and Phone Number. After entering the specified

details, the system will launch the monitoring window. Once our application starts monitoring, the detection algorithm detects the types of guns, and forms a bounding box around it. After successful detection of the gun, our application saves the detected frame only of the detected pistol/handgun and generates alert including the frame/ snapshot to the server to save it in the database.

The figure below is the sequence diagram of our proposed system, which is showing the interactions between the objects, and is describing the order in which the interaction is taking place.

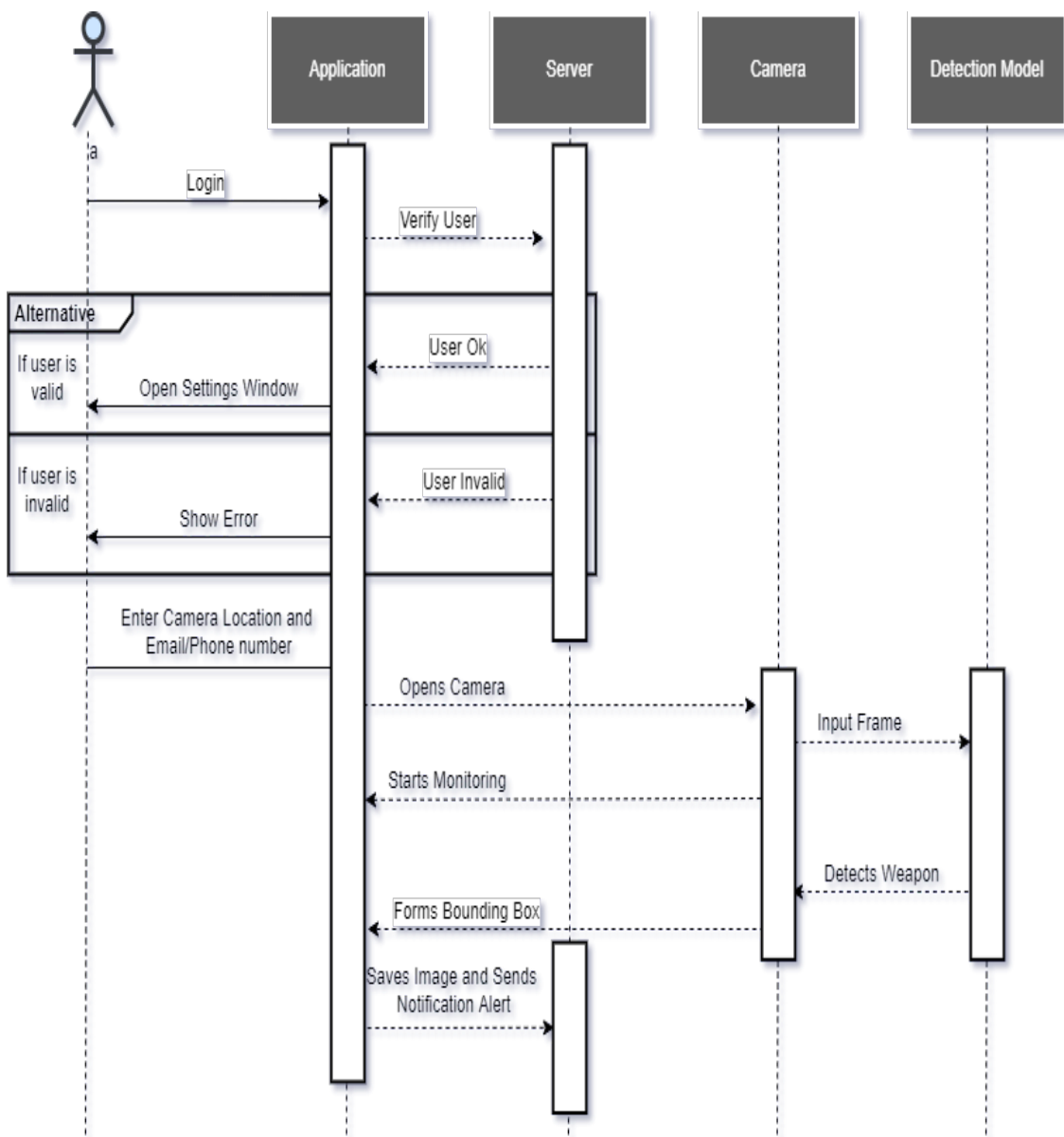


Figure 4.3: Sequence Diagram

### 4.5.2 Activity Diagram

This activity diagram explains the behavior of the system from the starting point to the ending point in the form of flowchart. It shows that the user will feed input to application then the application will detect both the guns by using the trained model. Then, if it succeeds in detecting the gun, it will form a bounding box around the guns and will display the labels. Followed by that, the system saves the frame/snapshot of the detected pistol and then generates alert of that pistol, that includes the frame/ snapshot sent to the server to save it in the database. The figure below is the activity diagram of our proposed system, which is showing the behaviour between the objects, from the beginning till the end.

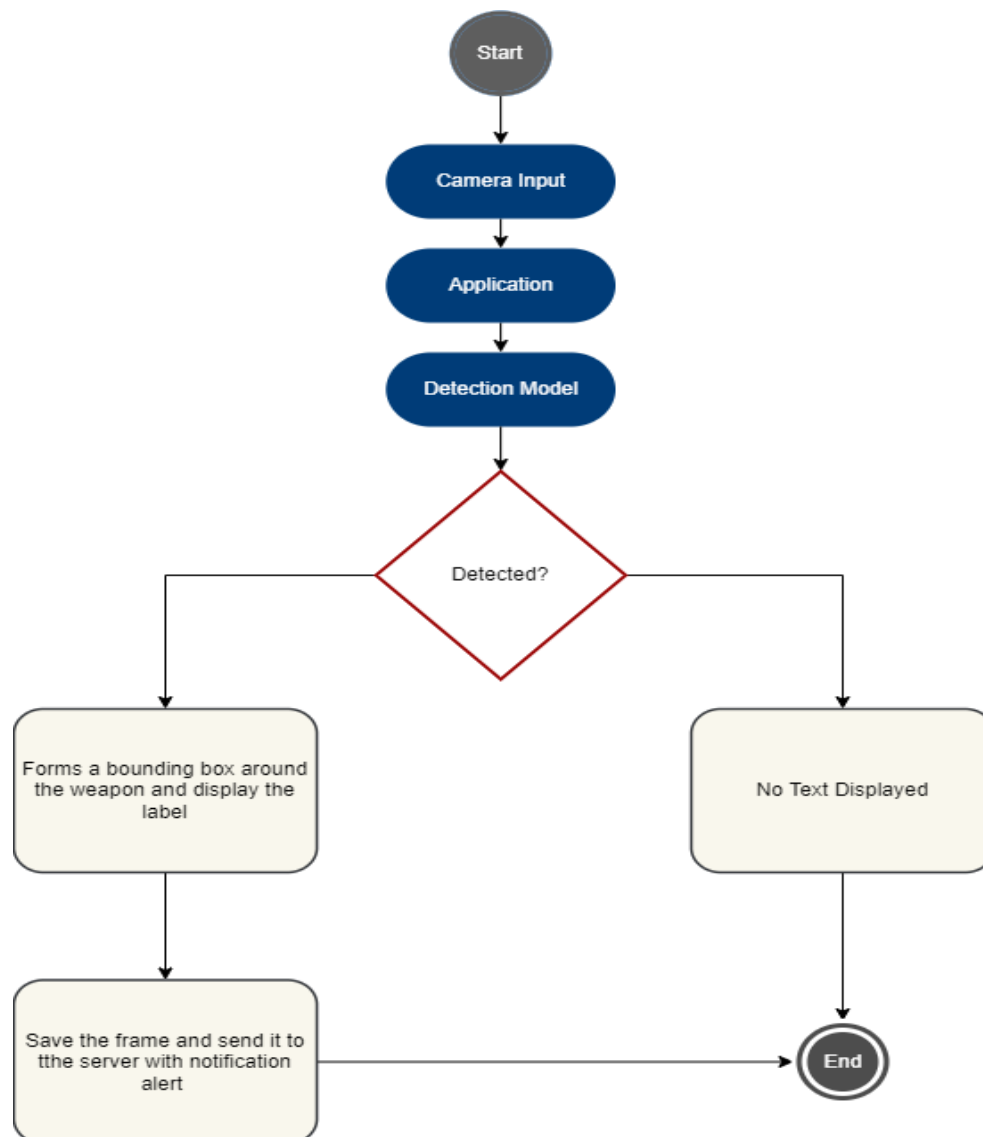


Figure 4.4: Activity Diagram

## 4.6 Low Level Design

The purpose of LLD or a low-level design document (LLDD) is to give the internal logical design of the actual LLD describes the class diagrams with the methods and relations between classes and program specs. It describes the modules so that the programmer can directly code the program from the document.

### 4.6.1 Web Application

Our Web-Application consists of two main pages. The first page consists of the registration process which includes:

- Sign up
- Login
- Forgot password

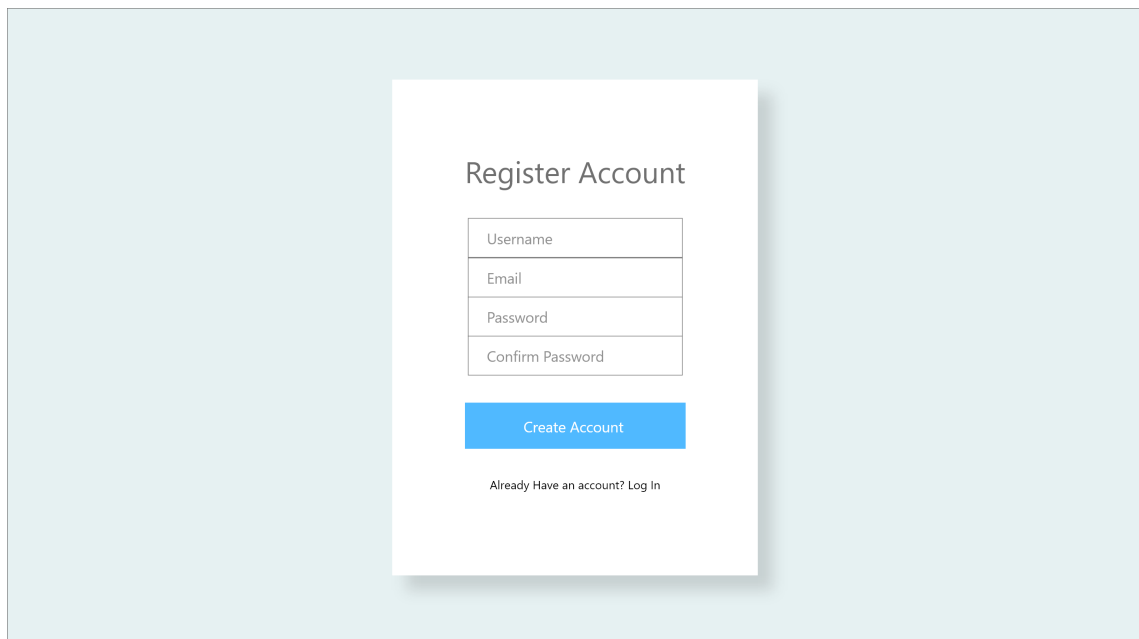
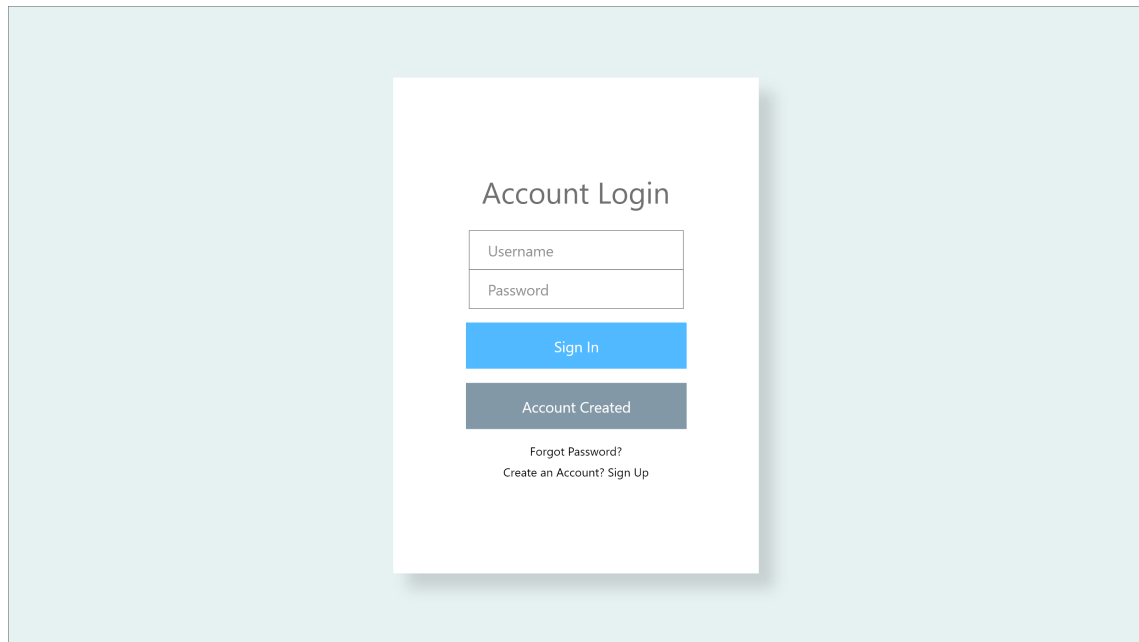
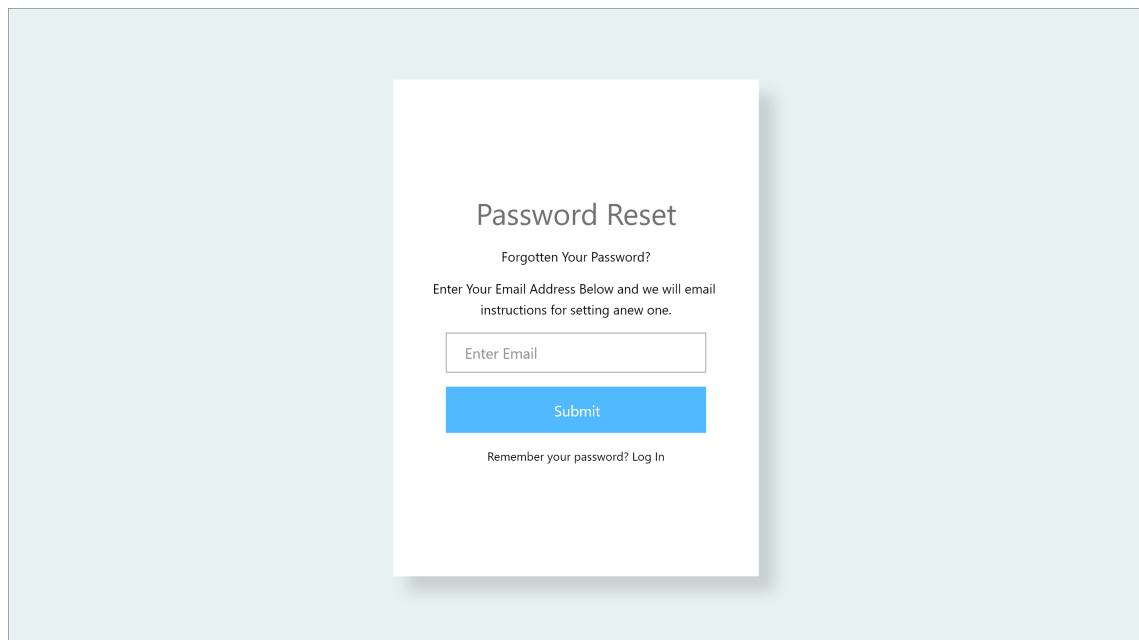


Figure 4.5: Sign Up



The image shows a login form titled "Account Login" centered on a light blue background. The form is white and contains the following elements from top to bottom: a title "Account Login", two input fields labeled "Username" and "Password", a blue "Sign In" button, a grey "Account Created" button, and two links: "Forgot Password?" and "Create an Account? Sign Up".

Figure 4.6: Log In



The image shows a password reset form titled "Password Reset" centered on a light blue background. The form is white and contains the following elements from top to bottom: a title "Password Reset", a sub-header "Forgotten Your Password?", a paragraph "Enter Your Email Address Below and we will email instructions for setting anew one.", an input field labeled "Enter Email", a blue "Submit" button, and a link "Remember your password? Log In".

Figure 4.7: Password Reset

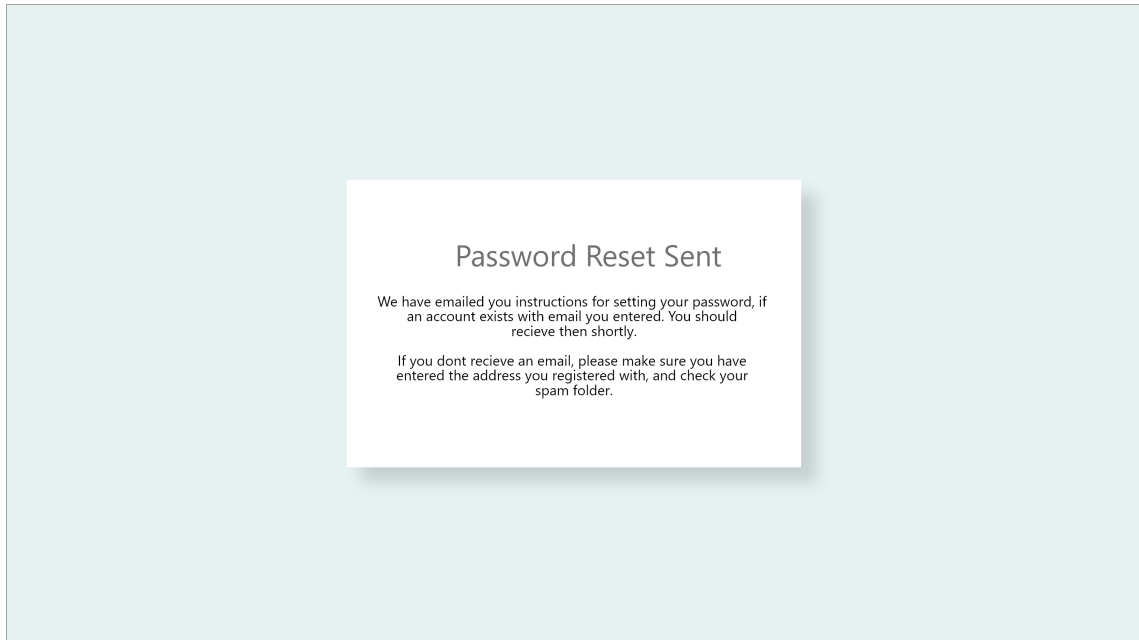


Figure 4.8: Password Reset Sent

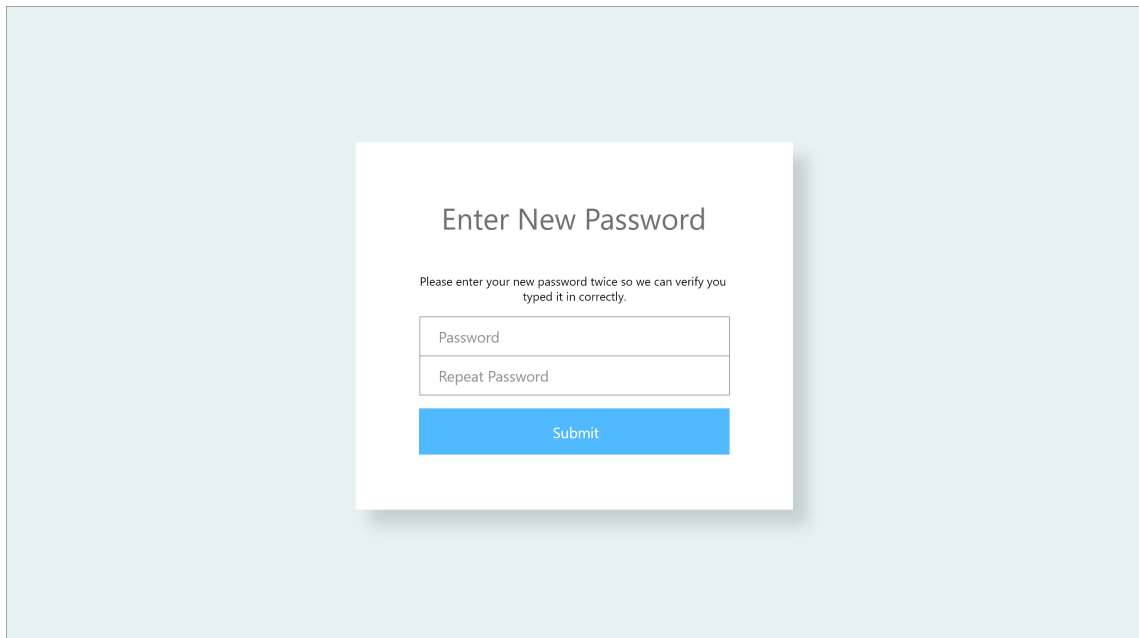


Figure 4.9: Enter New Password

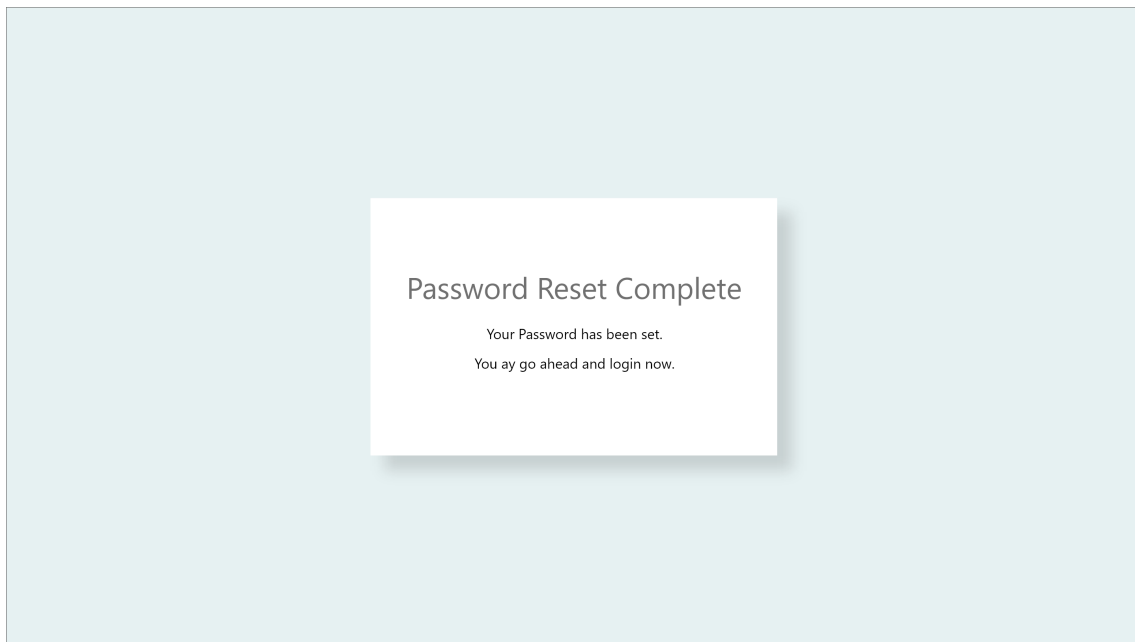
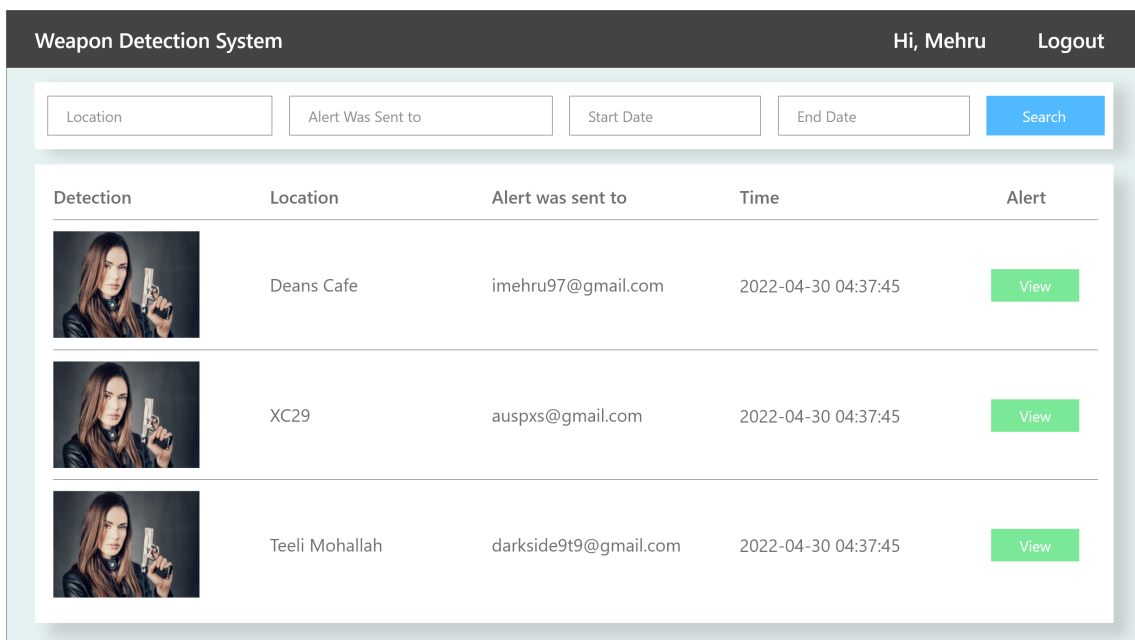


Figure 4.10: Password Reset Complete

Then the second page consists of the following:

- Alert dashboard
- Alert Details Page



The screenshot shows the 'Weapon Detection System' interface. At the top, it says 'Hi, Mehru' and 'Logout'. Below this is a search bar with fields for 'Location', 'Alert Was Sent to', 'Start Date', and 'End Date', and a 'Search' button. The main content is a table with the following data:




Detection	Location	Alert was sent to	Time	Alert
	Deans Cafe	imehru97@gmail.com	2022-04-30 04:37:45	<a href="#">View</a>
	XC29	auspxs@gmail.com	2022-04-30 04:37:45	<a href="#">View</a>
	Teeli Mohallah	darkside9t9@gmail.com	2022-04-30 04:37:45	<a href="#">View</a>

Figure 4.11: Alert Dashboard



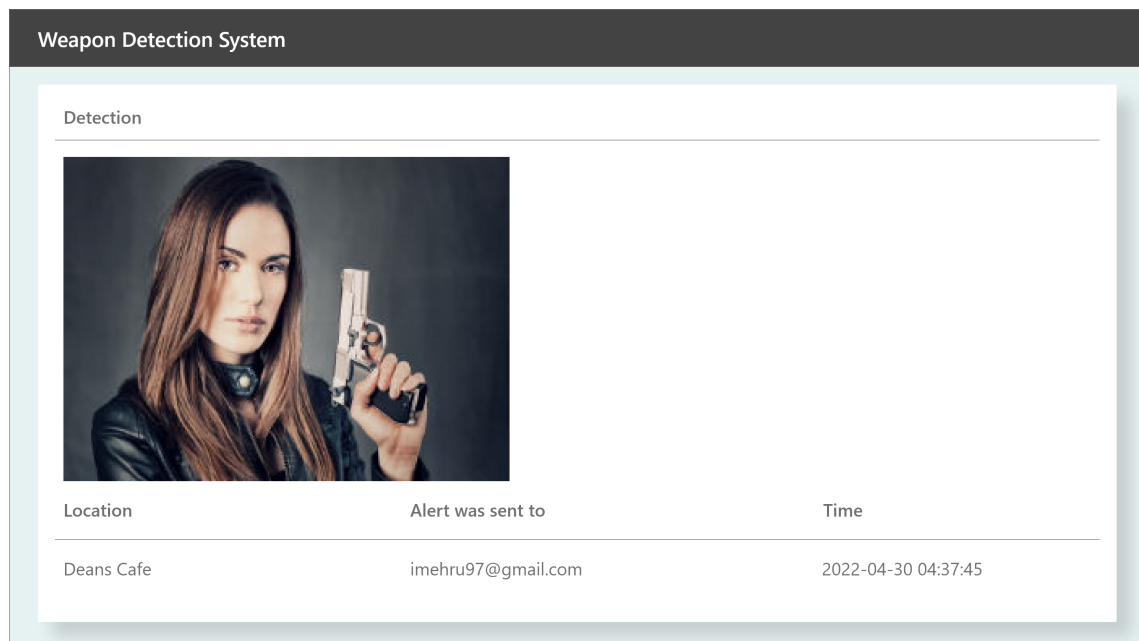
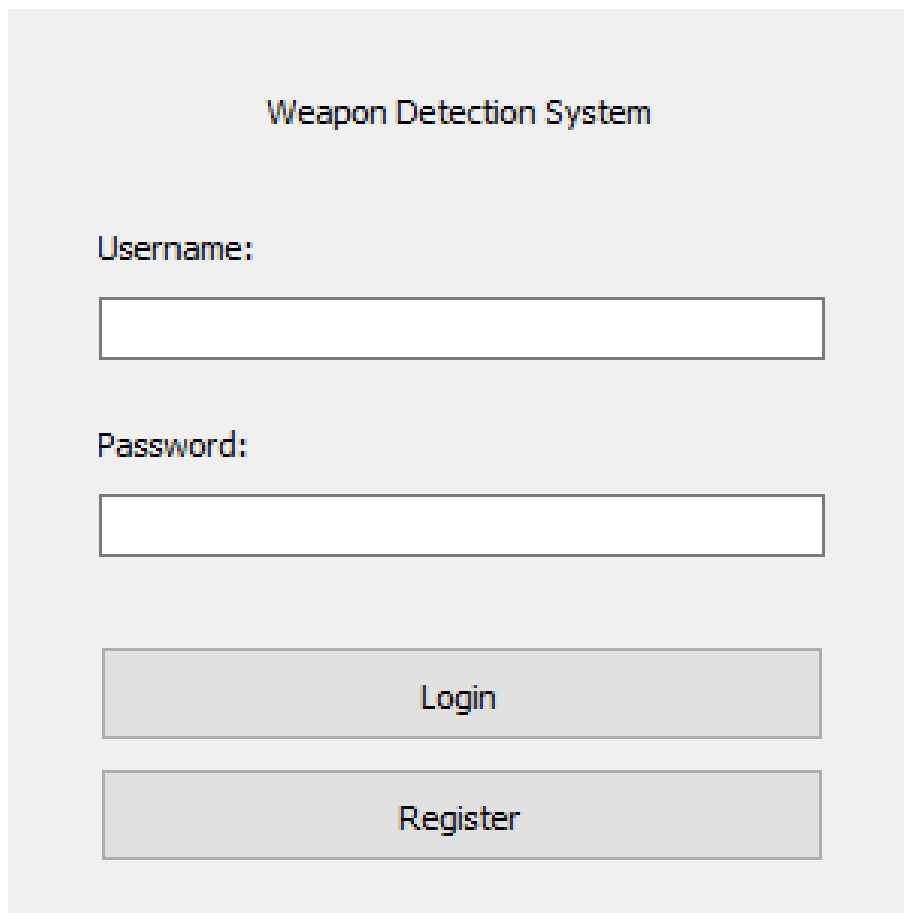


Figure 4.12: Alerts

#### 4.6.2 Client-Side App

- Login Window
- Settings Window
- Monitoring Window



The image shows a login window for a 'Weapon Detection System'. It features a title 'Weapon Detection System' at the top. Below the title are two input fields: one for 'Username:' and one for 'Password:'. At the bottom of the window are two buttons: 'Login' and 'Register'.

Weapon Detection System

Username:

Password:

Login

Register

Figure 4.13: Log In Window

Enter camera's location and a mobile number or email address.

Camera's location:

Mobile number or email address:

Figure 4.14: Settings Window

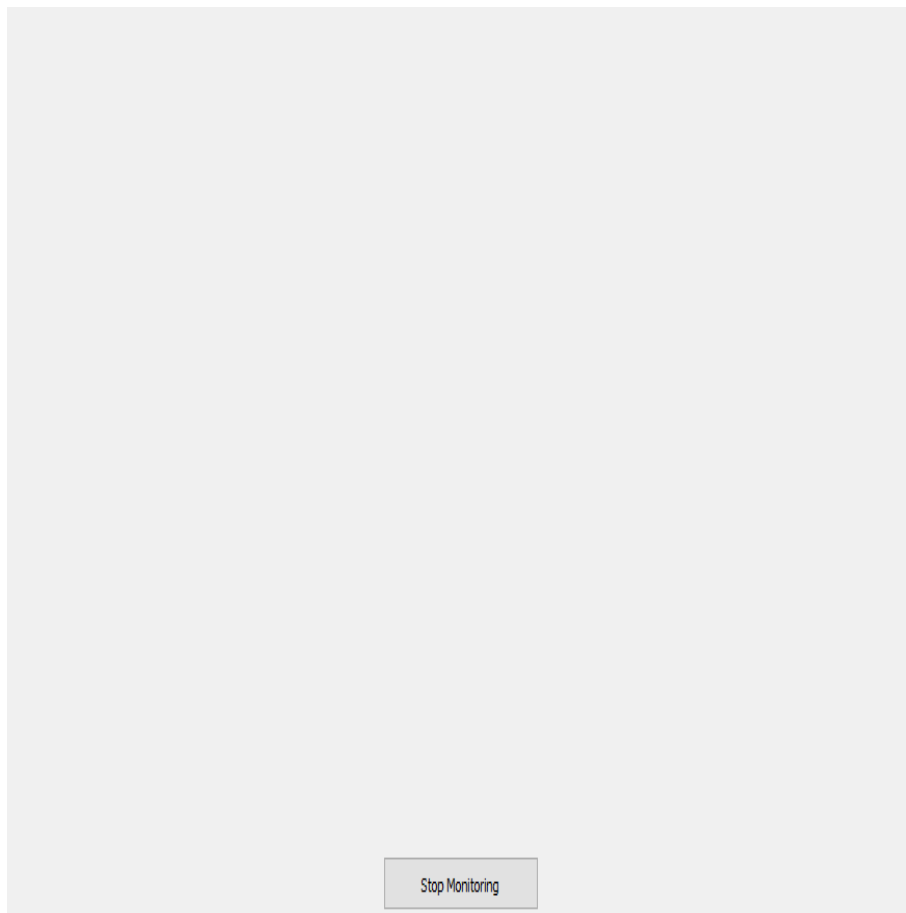


Figure 4.15: Detection Window

## 4.7 GUI Design

The user interface of our proposed application is quite simple, understandable and easy to use. We have kept few buttons and input fields to get it to work. For our web application after launching and opening, it starts off by asking the user to register into the system, after registration, the client-side is requested to enter login credentials, and after completing the basic login and registration procedure. All the attributes, pages, input fields, dialogue box, format and layout of our GUI has been kept simple and uniform rather than complicated so that the user faces no problem or difficulty in using it.

### 4.7.1 Usability Principles

We have followed the Usability Principles given by Dr. Donald Norman for designing the Graphical User Interface of our proposed system. Usability means user-centered design. Both the design and development process are focused on the user, to make sure their goals, models, and requirements are met.

#### Visibility

Users need to know what all the options are and know straight away how to use them. This means that the user should know what's going on inside the application. For example in our system, it is obvious that the user has to show a gun, which on detection will generate an alert, which will be clearly visible to the user. The information regarding the detected gun is also kept clear so that the user faces no difficulty in reading it.

#### Feedback

Like they say, every action needs a reaction. Meaning any functionality performed should give a response that could be in form of sound, highlighting or animations. The feedback should also be given in a reasonable time frame. In our case, we are giving feedback to the user, in the form of alert and also by detection of the gun, whenever the system is exposed to a gun.

#### Constraints

Constraints are the limitation of an interface. This prevents the users from choosing the incorrect options. In our system, we are only using the inbuilt camera of a laptop. This is the one constraint. Secondly, the quality of a camera is another

constraint. If the quality is not good enough, the resolution is not proper it can result either in wrong detection or no detection. Another constraint of our system is that we are at the moment only covering the small area, not a huge space. The detection area is limited. Another constraint is that we have to show at least 2 feet away for the system to be able to detect the weapon. Internet connection is also another constraint of our system, if we have a poor connection, this might result in delayed or no alerts at all. We have made this application for national as well as for international use and we know that English is the most common language in the world. So that in future it can be widely used. Another constraint in our system, is that the alerts cannot be deleted once sent to the alert's panel.

### **Mapping**

Mapping is the relationship between controls and their effect. The control buttons are mapped better onto the sequence of actions. We have a camera screen where users feed gun to the system and upon feeding the corresponding details are shown which shows where and what actions are being performed and tells us at what stage are we in the system.

### **Consistency**

Consistency, how consistent our system is. On using it every time, the same action should produce the same result. We have kept the appearance of both the web app and client-side app consistent. Our buttons, fonts and labels are kept uniform. Color scheme is also kept consistent to avoid any confusion.

### **Affordance**

Affordance is the basically, how we see something to how it is used. In our application the camera display opens in order to have a gun being shown to it. After that the application simply displays a registration form, which indicates the user to register into the system. Secondly from the client-side end, a login dialogue box pops up which also indicates the user to log in into to the system. This procedure then lands us on to the dashboard, showing all the details such as the alerts and images regarding the system which gives user a complete idea of system.

## **4.8 External Interfaces**

- The laptop must have a working camera in order to clearly stream the video.

- The system that is being used to build this project must have 8GB RAM & 256GB + SSD Hard Drive for smooth working experience.
- The internet connection must be good
- There must be significant space for the data-set training

# Chapter 5

## System Implementation

### 5.1 Introduction

The Implementation requires the translation of the design into programs that work successfully. In the implementation phase, the system is installed, all the processes are completed, and the documentation is provided to the user. Once this phase is completed, the application will be in static production, when the system enters static production, It will verify to ensure that all the requirements that we have planned are met and that we have obtained an acceptable result.

### 5.2 System Architecture

System architecture defines the basic structure of our system that highlights both the internal and external components of the project. The proposed system is made by using the YOLOv4 object detection model and Python. The Architecture of the system is categorized into three main steps/processes.

1. **Camera Input:** The input takes in the form of a video frames or image.
2. **Processing:** Processing is the key process, as the main functionality is performed here such as extracting frames from the video, resizing the frame as needed, and then entering it to our trained model. After the results are found.
3. **Output:** After the successful detection of the weapon, the frame will be saved and sent to the server with a notification alert.



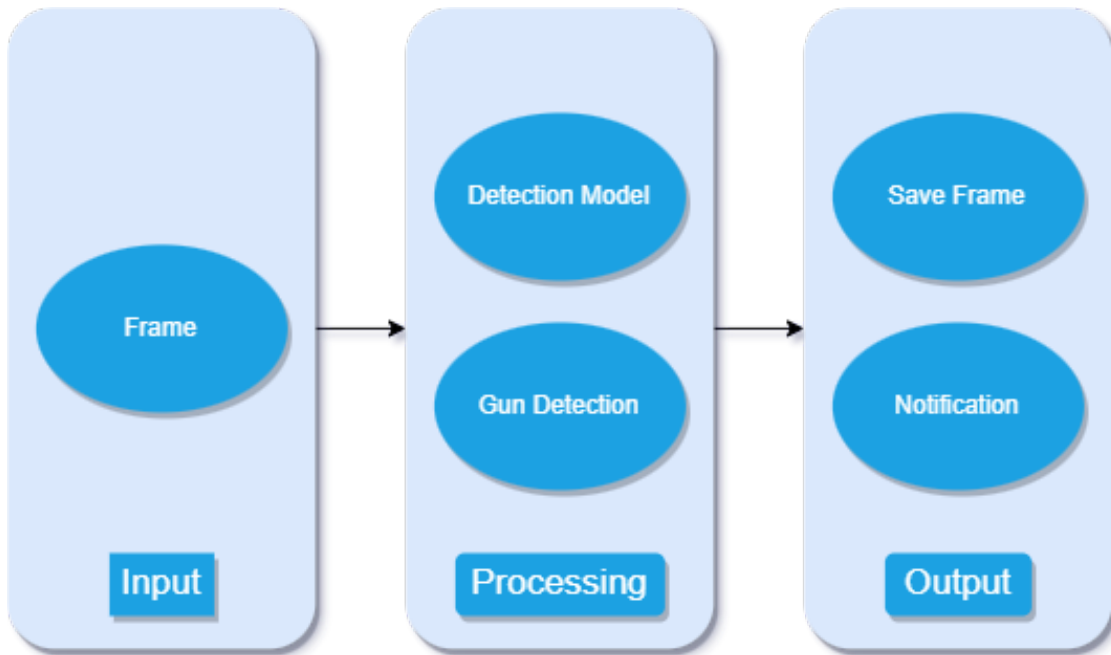


Figure 5.1: System Architecture

### 5.2.1 Object Detection

In object detection, the input is given as an image with the ground truth values of the bounding box. The input image is divided into a grid cell containing the center point which is primarily responsible for object detection and a square grid. Each grid cell will predict bounding boxes and the confidence scores linked with the boxes. The score will indicate the accuracy of the box and how confident is the model, that the box contains the specified object. The confidence score is IoU of predicted values and the ground truth values of the bounding box.

### 5.2.2 YOLO

YOLOv4 is a real-time object detection that achieved state-of-the-art performance on the COCO dataset [14]. YOLOv4 functions by splitting the object detection task into two pieces. Firstly, regression to identify object positioning via bounding boxes, and secondly, the classification to identify the object's class. YOLOv4 implementation is done by using the Darknet framework. YOLOv4 has enables us with a better object detection network architecture and new data augmentation techniques. As compared to the previous models, YOLOv4 has amazing high performance for a great high FPS. Every YOLO model is an object detection model. An Object detection model is trained to look at an image and look for a subset of object classes. On finding, these object classes are enclosed in a bounding box and their corresponding

class is identified. The COCO dataset contains approximately 80 object classes, and our models are trained and evaluated on the COCO dataset. From here now on, it's assumed that object detection models will generalize to new object detection tasks if they are exposed to new training data. The object detector takes an image in for input and compresses its features down through a convolutional neural network backbone. The combination of backbone feature layers takes place in the neck. The detection takes place in the head. YOLO is a one-stage detector, that makes the predictions for object localization and classification concurrently. The backbone network is pre-trained on ImageNet classification. By pretraining, we mean that the network's weights have beforehand been adapted to recognize relevant features in an image. We considered the CSPDarknet53 backbone for our YOLOv4 model. CSPDarknet53 is based on DenseNet, which was made to connect the layers in the CNN. The DenseNet has been modified to be able to separate the feature map of the base layer by copying it and sending one copy through the dense block and sending the other directly to the next stage. This is to remove the computational bottlenecks and improve the learning ability by sending the raw version of the feature map. Now, we combine the features that have been formed in the backbone to move toward the detection phase. Among all the options, YOLOv4 opts for PANet for the feature aggregation and adds an SPP block following the CSPDarknet53 to elevate the receptive field and filter out the important features from the backbone. Lastly, in the YOLOv4 Detection step, the same head has been used as the one used in YOLOv3, with anchor-based detection steps, and three tiers of detection granularity.

## 5.3 Tools And Technologies

### 5.3.1 AWS S3

We used AWS services to store our images in the backend. The images that we are saving, on the alert generation, are the ones being stored on AWS S3. Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance.

### 5.3.2 Twilio

We used Twilio for mobile alerts. Twilio's APIs power its platform for communications. Behind these APIs is a software layer connecting and optimizing communications networks around the world to allow your users to call and message anyone.

### 5.3.3 Roboflow

Roboflow is an online platform for improved data collection preprocessing, and model training techniques. Roboflow provides flexibility to the users in terms that users can also upload their own data set. We used Roboflow to upload our own dataset and applied some pre-processing techniques like augmentation. Roboflow also supports team collaboration, any changes made to the data, will be shown to anyone that is part of the team.

### 5.3.4 Google Colab Notebook

Google Colab is a free Jupyter notebook environment running completely in the cloud. Colab needs no setup to be required also any notebooks you create can be shared and edited by your team members. Colab supports the most popular machine learning libraries which can be easily loaded into your notebook. We used Google Colab to train our model. Firstly, we created a notebook, then executed the code in python by cloning the YOLO repository, then we imported our dataset from Roboflow through API, after that we configured our model and completed our model training. [15]

### 5.3.5 Visual Studio Code

We used VS Code for the development of our web application, it is a code editor with support for development operations like debugging, task running, and version control. We developed a client-side and server-side application on vs code which we will later deploy on Heroku for real-time alerts. We build the server-side application in the Django framework of python and used the REST framework for API. [16]

### 5.3.6 Adobe XD

We used Adobe XD for prototyping our weapon detection system. Adobe XD is the Adobe prototyping tool for user experience and interaction designers. Adobe XD features are used for creating wireframes, prototypes, and screen designs for digital products such as websites and mobile apps. [17]

### 5.3.7 QT Designer

We created our application GUI by using the QT Designer which helped us to create productively and efficiently. Qt Designer is a tool that provides us user interface to create GUIs for our PyQt applications. With this tool, we can create GUIs by

dragging and dropping QWidget objects on an empty form. After this, we can arrange them into a coherent GUI using different layout managers. Qt Designer is platform and programming language independent. It doesn't produce code in any particular programming language, but it creates a file with UI extensions which are XML files.

## 5.4 Experimental Setup

Python was chosen as the programming language because it is a high-level programming language that is simple to learn and code, making it one of the most extensively used programming languages for constructing machine learning and deep learning algorithms.

### 5.4.1 Dataset Collection

We collected our dataset from google images, which were about 3000+ images which consisted of a single type of gun(pistol/hand gun) and water gun for development purposes, we will later increase the types of weapons in our detection system. There were several limitations while making our dataset, it goes without saying that we could not possibly include all types of guns in our dataset, so we choose specific guns to make our customised dataset for development purposes. Following are the certain pictures from our data set.

#### Dataset pictures



Figure 5.2: Guns

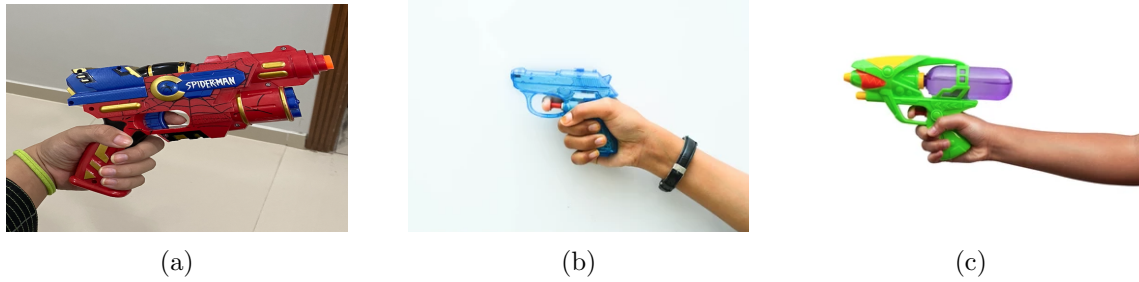


Figure 5.3: Guns

### 5.4.2 Labelling Dataset

The images in the dataset were manually labeled with the use of a system called 'LabelImg.' each image is labeled by drawing bounding boxes around the desired signs in the image and selecting their relevant classes.

### 5.4.3 LabelImg

For each image, an txt file, also known as a 'Annotation file,' is created and saved into a specified space. Annotations are saved as txt files in YOLO format. [18]

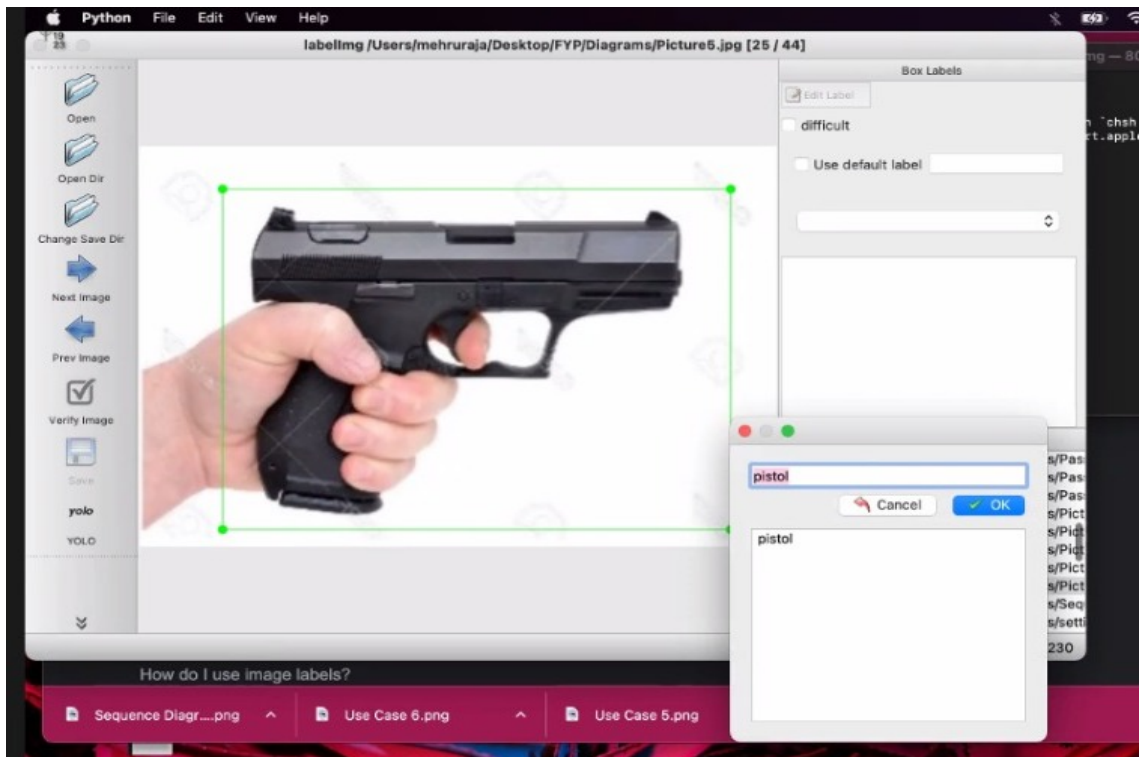


Figure 5.4: LabelImg

#### 5.4.4 TXT File

The annotation files contain details about the weapon in the image such as name of image, class and coordinates of the annotation. These files are further used to train and enable the algorithm to detect the weapon. We labelled our dataset in YOLO format which was to be used in training of Yolov4 model.

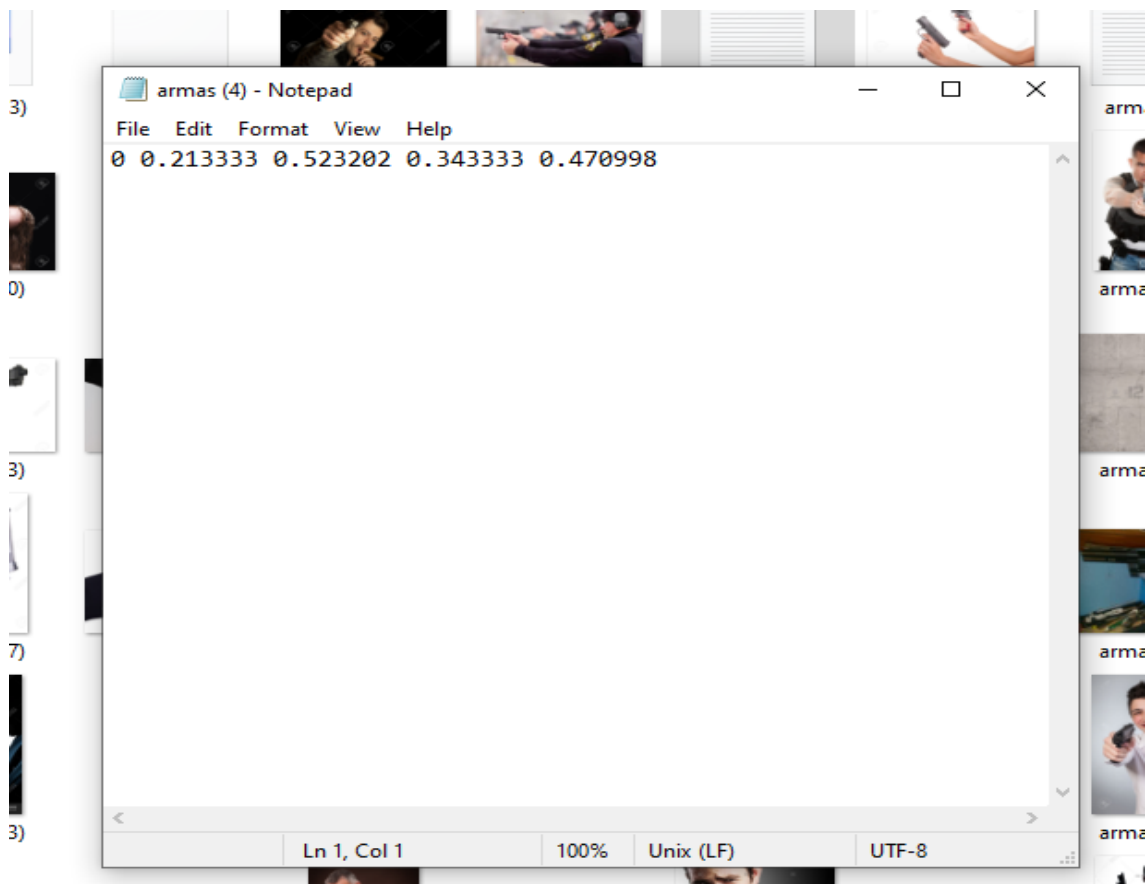


Figure 5.5: TXT File

### 5.4.5 Class Labels

A weapon in an image can be detected and identified by humans. The human visual system is quick and accurate, and it can handle complicated tasks like detecting multiple things. We can quickly train computers to identify and analyze multiple weapons inside an image with high accuracy, due to the availability of large amounts of data, faster GPUs, and better algorithms. Assigning a class label to an image is a part of image labeling; all labels are kept in the class file shown below.

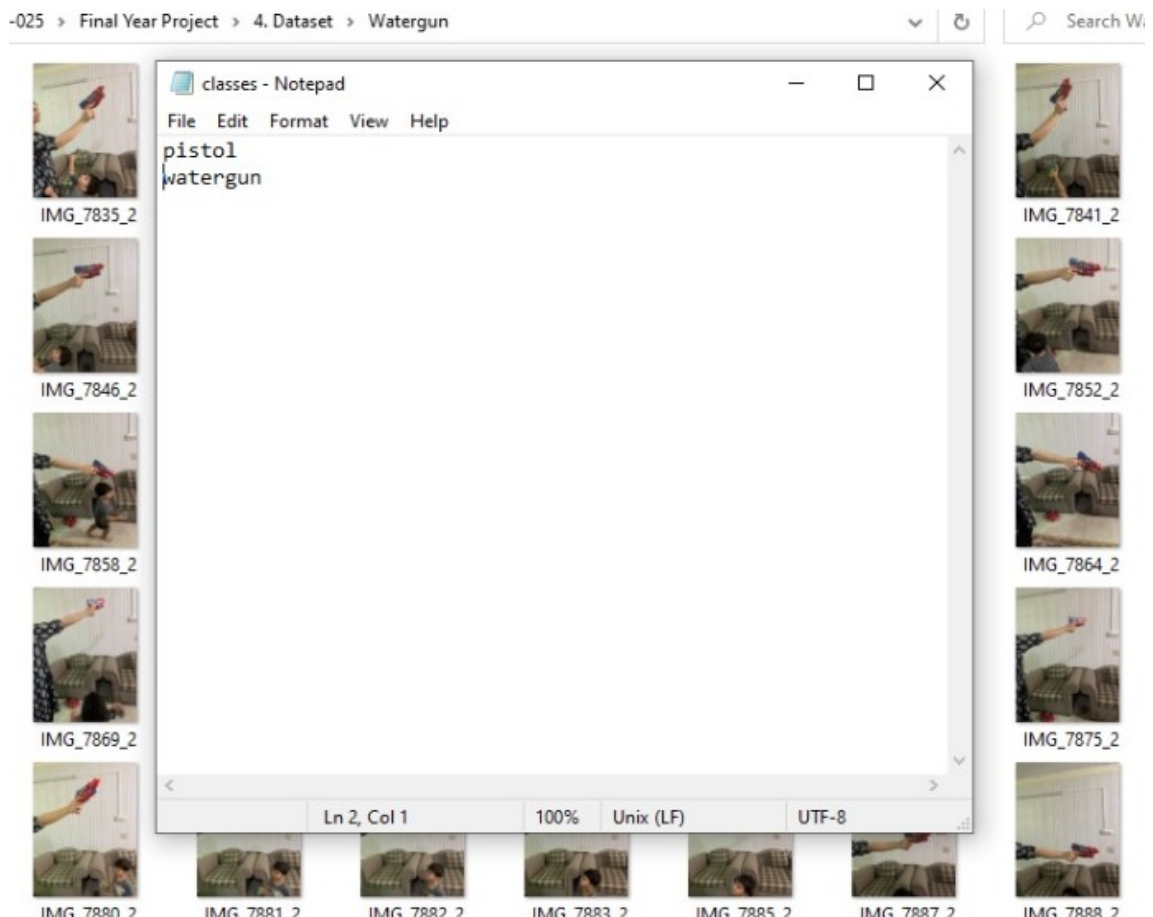


Figure 5.6: Class Labels

## 5.5 Methodology

Our application is developed using a Water-Fall Model, keeping in mind the enhancements that might be made in the future. The application was developed in different phases.

### • Research

In the first phase of application development, we were gathering the required essential information regarding the development of the application. We read various research papers and also tried to find any existing dataset for guns but we could not find any, therefore we decided to make our own dataset. We trained our yolov5(pytorch) model by Ultralytics, but there were several issues deploying on client-side.

### • Development

In the second phase, we planned and initiated the development of the application. So we started by making our custom dataset, firstly just to get an idea we made a small dataset consisting of just 1 weapon we used labelImg directory which is an open-source platform for labeling datasets. We first labeled our dataset in Yolo format and trained it by using YOLO v4 on google Colab notebook. For the development of our detection system interface, we used VS Code and QT Designer.

### • Web-Application Development

1. Server-Side Application
2. Client-Side Application

For the development of our gun detection system, we used the following:

- Python
- OpenCv
- Django
- QT Designer

The server-side application will be hosted on heroku later. To store saved snapshots Amazon S3 will be used. YOLO v4 object detection model will be used for the detection algorithm. For the Server-side application, we will create a new account to show the system's functionality. Once the account is created, then we can log in within the client-side application. After successfully logging in, we will need



to provide the camera's location and mobile number, which later will be used to send the alert. After entering the specified details we'll need to press the start monitoring button and the detection window will be opened within the detection window a camera's output is displayed. Once a gun is detected a bounding box is drawn around it and the percentage of confidence is shown within the console, and also the frame will be saved and sent to the server. When we open the alert link we can see the detection image the location, the receiver, and the time. Also, we can log in to the account that we created earlier and see the list of alerts with all the information and the filtering option will also be available. We can also click on the view button to see the larger image. We can click the link and the alert page will be opened within your phone and there you can see the new alert and as we press the view button a larger image is displayed in the system. Our system will also have a password reset functionality where we can provide a registered email address and the password reset email will be sent including text and the link for the password reset. And over there we can provide a new password for the account and then a new password will be set automatically.

## 5.6 Model Training

We at last succeeded in training our model by using the object detection algorithms to help us implement the YOLOv4 model by using our own custom dataset. It used transfer learning techniques to reduce the amount of training data required and shortens the data training time. We used YOLOv4 and got some reasonably good results by training our custom dataset on Google Colab. The average accuracy was over **80.65%**. We used the dataset approximately of more than 3000 images, on which we applied the 95/1/4 rule which means that 95% of the data is for training, 1% for validation and 4% for testing.

```
calculation mAP (mean average precision)...
104
detections_count = 259, unique_truth_count = 102
class_id = 0, name = pistol, ap = 80.65%      (TP = 38, FP = 12)
class_id = 1, name = watergun, ap = 98.09%   (TP = 56, FP = 0)

for conf_thresh = 0.25, precision = 0.89, recall = 0.92, F1-score = 0.90
for conf_thresh = 0.25, TP = 94, FP = 12, FN = 8, average IoU = 69.88 %

IoU threshold = 50 %, used Area-Under-Curve for each unique Recall
mean average precision (mAP@0.50) = 0.893703, or 89.37 %
Total Detection Time: 3 Seconds

Set -points flag:
`-points 101` for MS COCO
`-points 11` for PascalVOC 2007 (uncomment `difficult` in voc.data)
`-points 0` (AUC) for ImageNet, PascalVOC 2010-2012, your custom dataset

mean average precision (mAP@0.5) = 0.893703
```

Figure 5.7: Accuracy of our Model

# Chapter 6

## Testing

### 6.1 System Testing

System testing is a testing procedure carried out on your system, to evaluate the performance of your product, if it is working as planned and if the system complies with its specified requirements. System testing is one of the most essential parts or processes in the building of your system or application. It is confirmed that many systems face a failure when they're testing, and the evaluation is poor. Evaluation of our work is extremely important, not only of our own system but also evaluating it with the existing systems, software, hardware, and methods to have better knowledge and idea of our system. Testing can be of two types, qualitative and quantitative. In qualitative, we look at the minor details, understanding more in detail what the user wants and whereas the quantitative is all about objectivity and group behavior. We must know where our system excels and where our system lacks, there is no such thing as development that is perfect. Flaws and imperfections are a part of every process or system. The following are different types of testing that should be considered during the system testing.

### 6.2 Functional testing

Functional testing is basically in which through our team of testers a quality assurance is determined whether our application is acting the way it is supposed to as our defined or mentioned requirements. In our system, the functional testing would be testing the functionalities of the web application such as the user interaction, and as well as the tasks or functionalities are being performed correctly and logically. Our main purpose is to make sure that the quality is being met according to our

expectations of the system and to also reduce errors so that in return there is a complete customer satisfaction. The functional testing, we performed as follows:

- To check If all the required and mandatory fields are present and being displayed correctly on the screen.
- To check, if all our fields are working fine, such as enter password or log in details.
- To check, how the application responds to closing or reopening of the web application.

### 6.3 Interface testing

In the Interface testing, the system's GUI functionality is tested. We check whether the system is meeting the requirements of the application or not. The main testing of our system mainly comprised of the system being able to detect the gun. In our web application, there is a dashboard that contains alert information, so we see if the panel can scroll properly or not. As the main purpose of Interface testing is to verify the functionality of an interface, therefore we tested our app's interface from all the aspects.

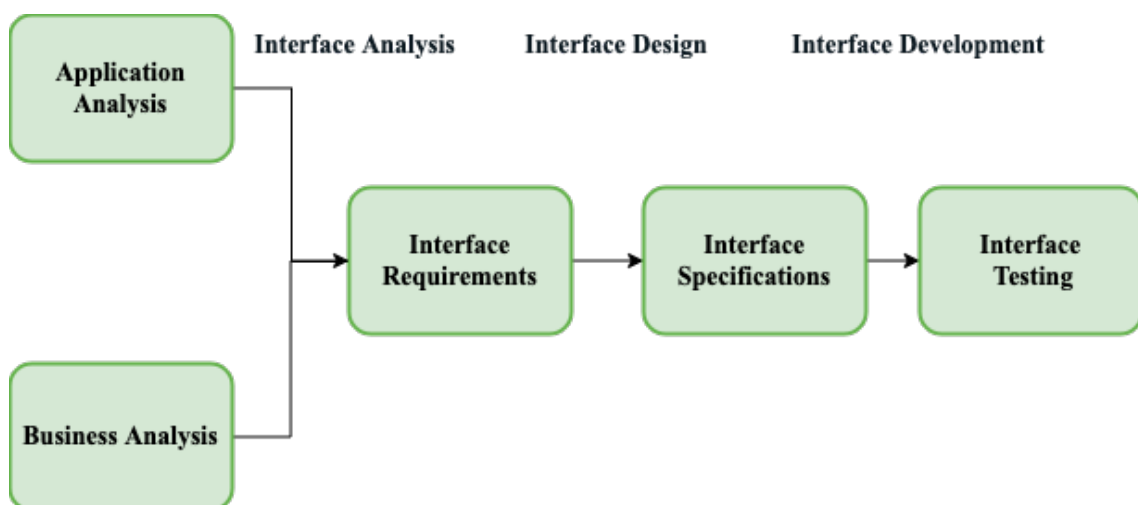


Figure 6.1: Interface Testing

## 6.4 Usability testing

Usability in simpler words is testing out how easy it is to use your system or application. How practical and user-friendly system have you built. Keeping in mind the user experience comes with the user being satisfied and that is achieved by building a user-friendly app, thus considering this factor, we have tried to keep our design quite simple and minimalistic so that the user faces no difficulty in using our application. Usability testing requires representative users to test your application to get better feedback rather than just your own developer testing the app. We have kept our application quite simple; all the user has to do is register and expose a gun to the camera so that it can detect it.

## 6.5 Compatibility testing

Compatibility testing is used to check the compatibility of our system with different computing models, meaning which platform supports the working of our application. Our web application is compatible with all browsers and also our client-side app is compatible with MacOS and windows.

## 6.6 Performance testing

Performance testing is to test how well our application runs under some load. It tests the efficiency of the system. Our system mainly consists of how reactive the application is to detection. How quickly does it respond when a gun is exposed to it. So our system works quite fast as it detects guns in an average time of 120ms, and the average accuracy is about 88%.

## 6.7 Testing Strategies

- Black Box Testing
- Specification Testing
- White Box testing

### 6.7.1 Black Box Testing

The black box testing, as the name suggests keeps the core functionalities hidden but the testing is done from the user experience. In the black box, we test the interface and the requirements of the application. For our application, we perform

black box testing in such a way that the camera is exposed to the gun and then on our system, the bounded box around the gun is shown indicating the detection of the gun and the dashboard shows the alert and the snapshots. We randomly ran our application on different volunteers to test the application on them, assuring that the distance between the user using the application and the gun is at least 2 feet to get the best results.

### 6.7.2 Specification Testing

In Specification testing, we trained our custom model and tested it on dataset images. Our dataset is divided in such a way that 95% is kept for training, and 4% for testing, whereas 1% for validation. Once the Model detects the gun, it forms the bounding box and returns the confidence score for the gun that match the training data.

### 6.7.3 White Box Testing

White Box Testing is basically testing the core functionalities of the system. The internal structure and the code are tested. For white box testing, the code should be easily understandable to both the team members. For white box testing, these areas are to be considered.

- Code functionalities
- Detection of the weapon
- Testing each function
- Response time

## 6.8 Testing Performance Test Cases

To evaluate the performance of our model, we performed several tests to analyze the performance of our model. Following are test cases we applied to our system.

- To ensure that the response time and confidence score of our detection are according to our requirements.
- To check whether our client-side application and server-side application remain connected during all time
- To ensure the camera quality is also satisfactory

- To ensure detection of both the guns.
- To ensure that upon detection of the gun, the alerts are generated
- To ensure that the snapshot of the gun is also saved

## 6.9 Testing Usability Test Cases

As discussed earlier, usability testing involves how user- friendly the application is. The main purpose is to have an easy-to-use app rather than a complicated one that will be difficult to understand and operate. Therefore, we intend to build an application that is easy to use and acceptable as well to be sold in the market. For this purpose, we ensured the following:

- Font size to be visible and easy to read
- Buttons to be of a standard size
- Buttons to be placed on the same screen
- Logo to be consistent with the application
- Color schemes not to be too sharp

## 6.10 Test Cases

Various test cases were performed and run through our detection system to check the system's performance & effectiveness. We have listed them below:

### 6.10.1 Test Case 1 : Registration

In test case 1, we will be testing our web application's registration process. We tested this process and our web application successfully passed this test as our web-app was signed up successfully several times without any bugs and errors.

<b>TestID</b>	1
<b>Test Case Description</b>	Testing of Registration Process
<b>Initial State</b>	Application Should be Running
<b>Input</b>	User will enter details
<b>Expected Output</b>	Successful Sign Up
<b>Output</b>	User Input is valid and account is created
<b>Status</b>	Pass

Table 6.1: Test Case: Register

### 6.10.2 Test Case 2 : Log In

In test case 2, we will be testing our web application's registration process. We tested this process and we were able to login multiple times and our web app passed this test successfully without any errors.

<b>TestID</b>	2
<b>Test Case Description</b>	Testing of login Process
<b>Initial State</b>	Application Should be Running
<b>Input</b>	User will enter details
<b>Expected Output</b>	Successful Sign In
<b>Output</b>	User log in details are valid and signed in
<b>Status</b>	Pass

Table 6.2: Test Case: Login

### 6.10.3 Test Case 3 : Gun Detection

In test case 3, we will be testing our client-side application core functionality of detection. We tested this process and we were able to detect gun and water gun multiple times and our server side received alerts and snapshot of detection successfully without any errors.



<b>TestID</b>	3
<b>Test Case Description</b>	Testing of Gun Detection Process
<b>Initial State</b>	User must have a camera/webcam running with machine
<b>Input</b>	User holds gun in front of camera
<b>Expected Output</b>	Bounding box is formed around gun
<b>Output</b>	Our Application is detecting the gun
<b>Status</b>	Pass

Table 6.3: Test Case: Gun Detection

#### 6.10.4 Test Case 4 : Saving Snapshot

In test case 4, we will be testing our client-side application where we are testing that our application saves the snapshot after successful detection of gun. We tested this process and we were able to detect gun multiple times and passed this test.

<b>TestID</b>	4
<b>Test Case Description</b>	Testing of Snapshot Saving Process
<b>Initial State</b>	Application must be running and client must be logged in
<b>Input</b>	Gun should be detected and bounding box must be formed around gun
<b>Expected Output</b>	Saved Snapshot of detection
<b>Output</b>	Successfully saved snapshot of detection
<b>Status</b>	Pass

Table 6.4: Test Case: Saving Snapshot

#### 6.10.5 Test Case 5 : Alert Generation

In test case 5, we will test our alert generation, where we are testing that detected gun with saved snapshot is sent to the server after detection and server side is receiving that alert successfully. We tested this process and were able to receive alerts multiple times and passed this test.

<b>TestID</b>	5
<b>Test Case Description</b>	Testing of Alert Generation
<b>Initial State</b>	Client-Side application must have internet connectivity
<b>Input</b>	Snapshot must be saved and alert including snapshot should be sent to server over the internet.
<b>Expected Output</b>	Received Alert
<b>Output</b>	Successfully received alert of detection
<b>Status</b>	Pass

Table 6.5: Test Case: Alert Generation

#### 6.10.6 Test Case 6 : Notification

In test case 6, we will be testing our alert notification by receiving mobile SMS. We tested this process and we were able to receive alerts multiple times and passed this test.

<b>TestID</b>	6
<b>Test Case Description</b>	Testing of mobile notifications
<b>Initial State</b>	Server-Side application must be running
<b>Input</b>	Alert must be received.
<b>Expected Output</b>	Receiving mobile notification alert
<b>Output</b>	Received SMS including alert link
<b>Status</b>	Pass

Table 6.6: Test Case: Notification

## 6.11 Limitations

There were several limitations that we faced just like any other application. Such as, the convolutional neural network has its drawbacks like the classification of images with different certain different positions. Due, to this our model, also experiences these limitations. Our model is also limited; it can only detect one gun at the moment. Another limitation was of a GPU and the computational resources were also limited so thus training takes a lot of time through neural networks. Another limitation, we faced is that of light, if the light is unusual, the system might not be able to detect the weapon. Also, the camera quality proved another limitation, if the quality is not good enough to record thus the weapon detection would not be accurate. Along with this is the internet connectivity as alerts have to be generated for which the internet connection is mandatory. Due to all these reasons our system faces certain limitations. Hopefully, in the future we will try to overcome these limitations.

## Chapter 7

# Conclusion

Our Gun Detection System has been successfully designed and developed. Our application, aims to develop a system for the betterment of security, to help the system recognize criminal activity being done with improved technology. Considering especially Pakistan, despite the physical appearance and nonstop monitoring by the security officials, criminals always manage to get away and this always raises a question about the system's capability or competence.

This System will drastically help in detecting the gun possessed by anyone, which might get a chance to escape from the naked eye and improve the current situation of the security concerns at the moment.

Our Gun Detection system would completely replace current infrastructure with the growing availability of low-cost storage, video infrastructure, and better video processing technologies.

In this study, the YOLO v4 object detection model was implemented and trained over our customized collected data set for gun detection. We have proposed a model that provides a sense to a machine to be able to identify a gun and a water gun separately and differentiate between them. Upon identification of the gun, it should be able to generate an alert only when a gun(pistol) has been detected and not on the detection of the water gun during this period. We have also tried to improve the system in such a way that the incident's recording has also been saved so that for any future investigations there is proof of the entire incident. There is a dire need to improve and update the current surveillance capabilities with better resources to aid in monitoring the effectiveness of human operators.

We additionally examined the difficulties and restrictions that were faced by gun recognition, also performed several test cases to be able to identify our strengths and weaknesses so that in the future we can reduce our weaknesses and be able to

work more on our strengths so that our end product given to the user should be free of any loopholes.

## 7.1 Future Works

- In the future, we can train our model using YOLOv5 to even have better accuracy, than we are experiencing right now.
- We can extend our dataset by including a variety of guns and along with that variety of other weapons such as knives.
- We know any organization's security officials are allowed a possession of the gun, so in the future, we can train the model in such a way that it does not generate alerts when the gun is being detected by the guard.

# Appendix A

## User Manual

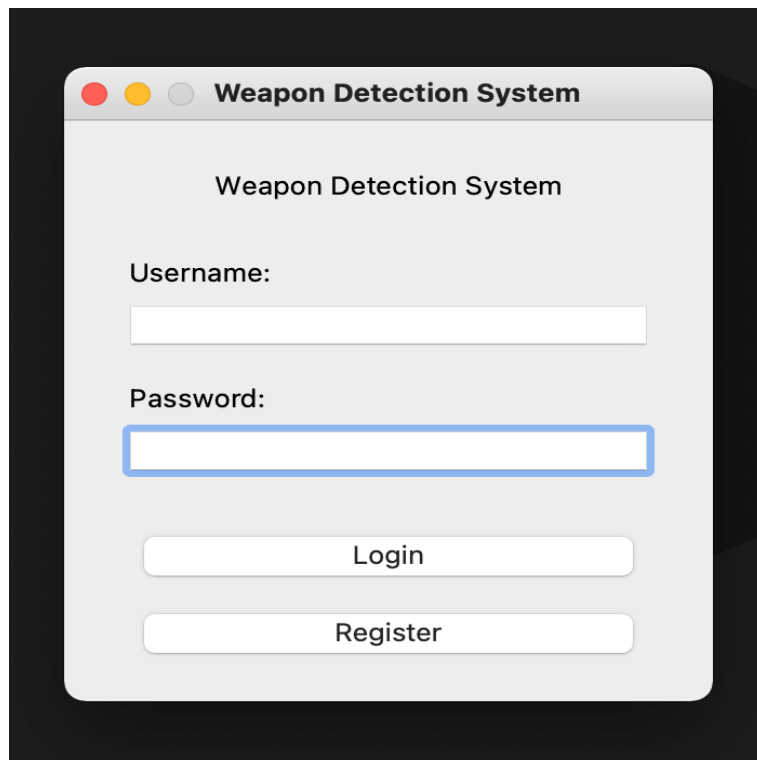
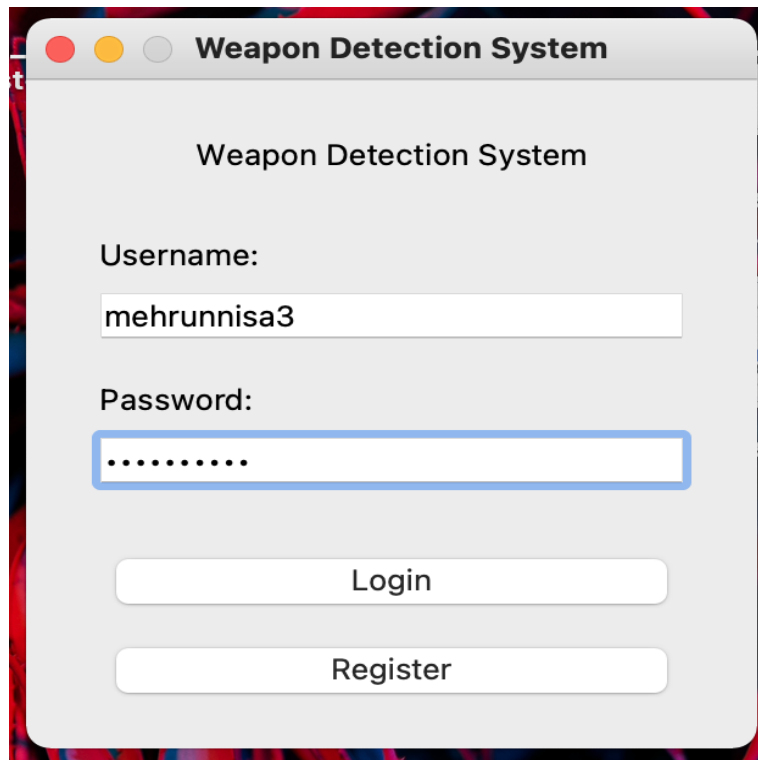
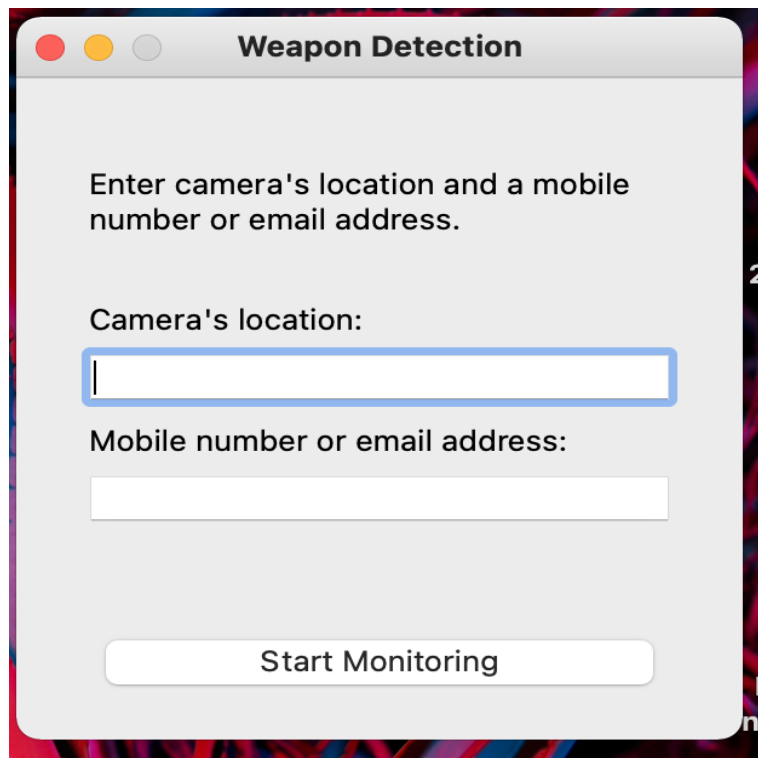


Figure A.1: Client Side Login



The screenshot shows a window titled "Weapon Detection System". Inside the window, the title "Weapon Detection System" is centered at the top. Below the title, there are two labels: "Username:" and "Password:". The "Username:" label is followed by a text input field containing the text "mehrunnisa3". The "Password:" label is followed by a password input field containing ten dots. Below the input fields, there are two buttons: "Login" and "Register".

Figure A.2: Client Side Login



The screenshot shows a window titled "Weapon Detection". Inside the window, there is a text prompt: "Enter camera's location and a mobile number or email address." Below the prompt, there are two labels: "Camera's location:" and "Mobile number or email address:". The "Camera's location:" label is followed by a text input field. The "Mobile number or email address:" label is followed by a text input field. Below the input fields, there is a button labeled "Start Monitoring".

Figure A.3: Client-Side Settings Window

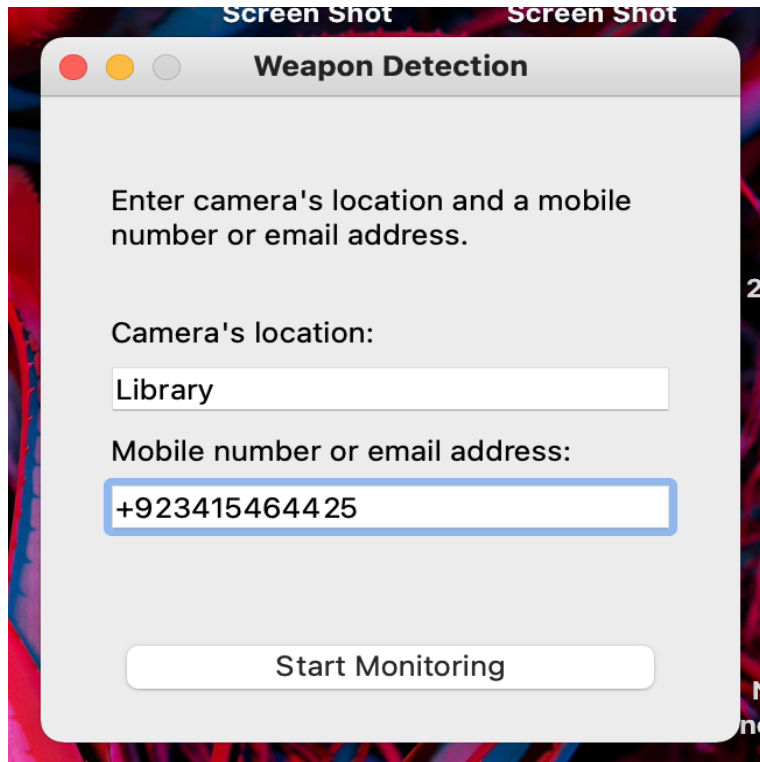


Figure A.4: Client-Side Settings Window

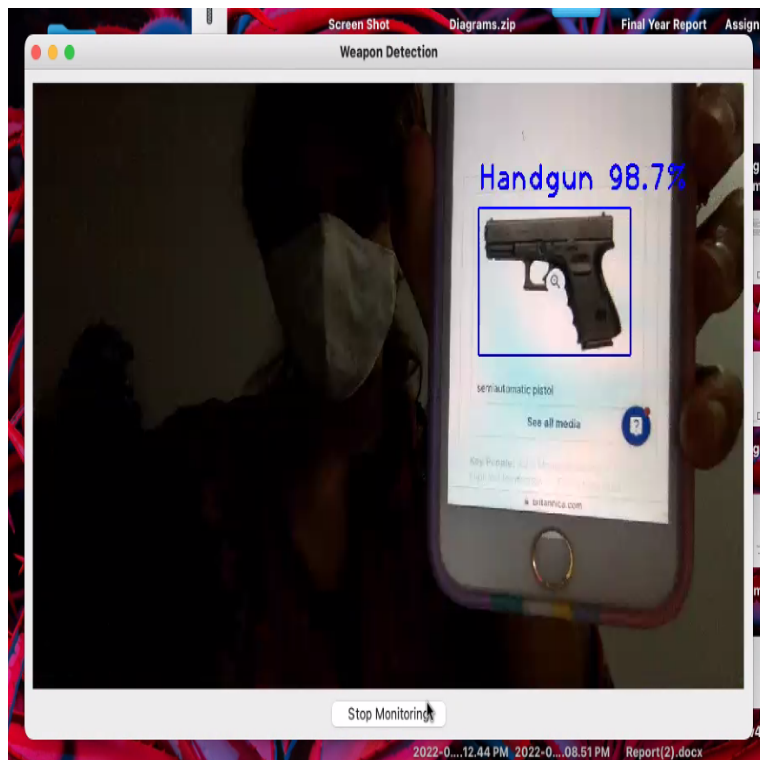


Figure A.5: Client-Side Detection Window



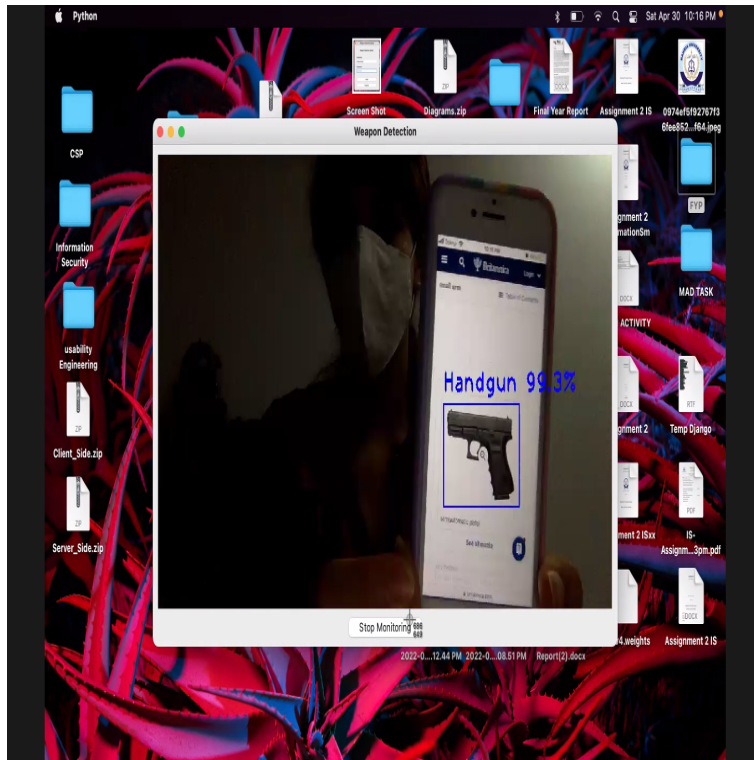


Figure A.6: Client-Side Detection Window

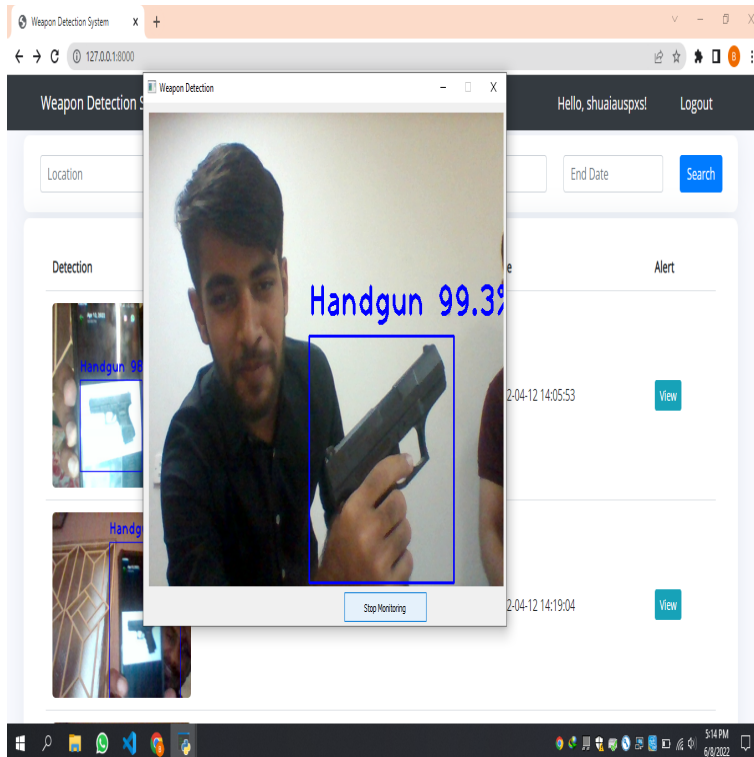


Figure A.7: Client-Side Detection Window

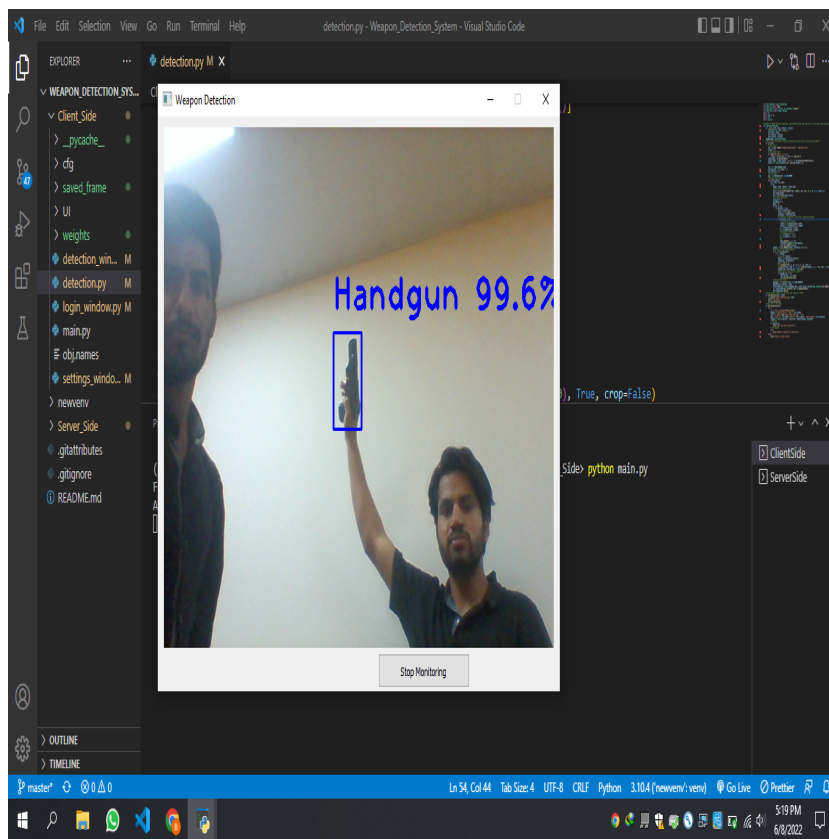


Figure A.8: Client-Side Detection Window

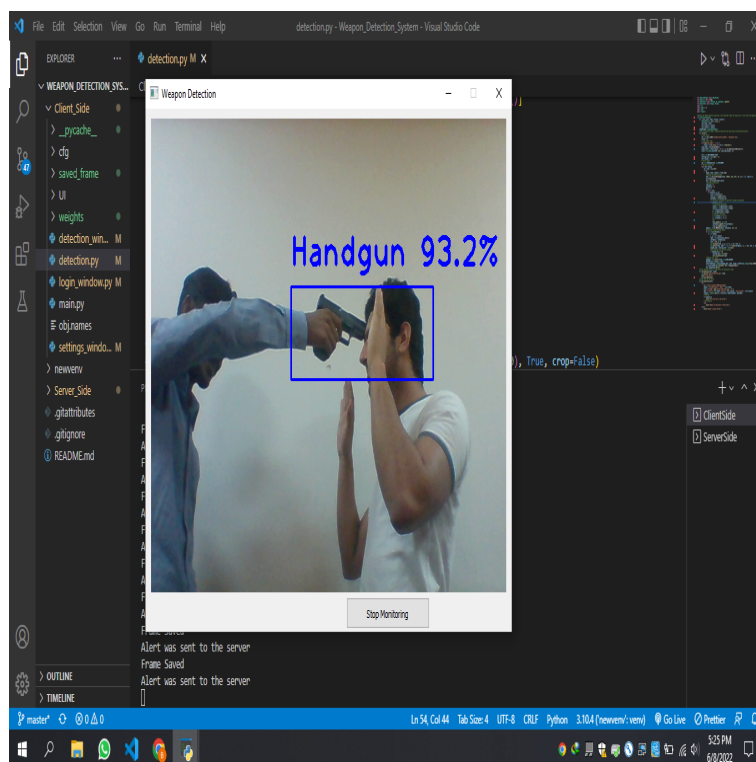


Figure A.9: Client-Side Detection Window

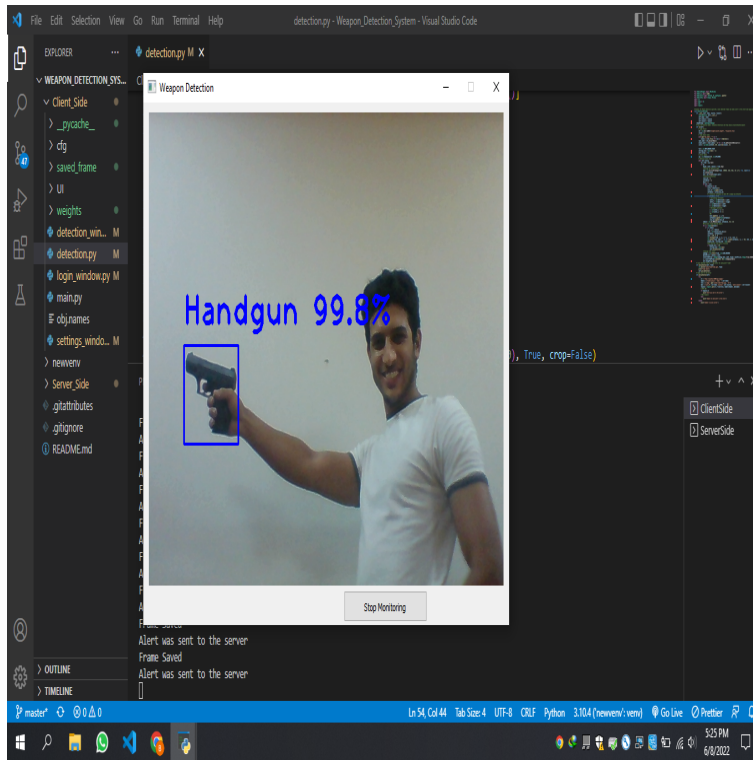


Figure A.10: Client-Side Detection Window

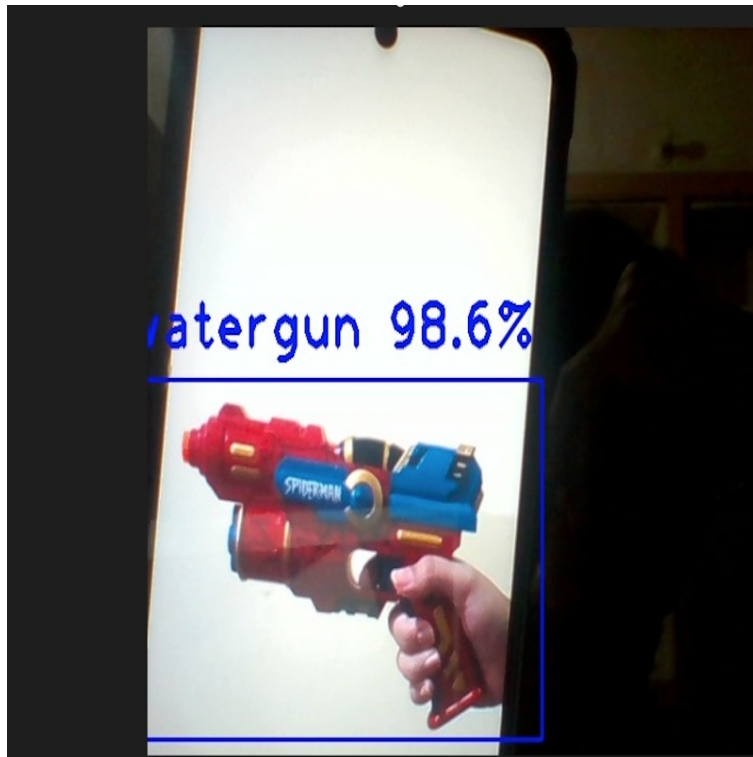


Figure A.11: Watergun Detection

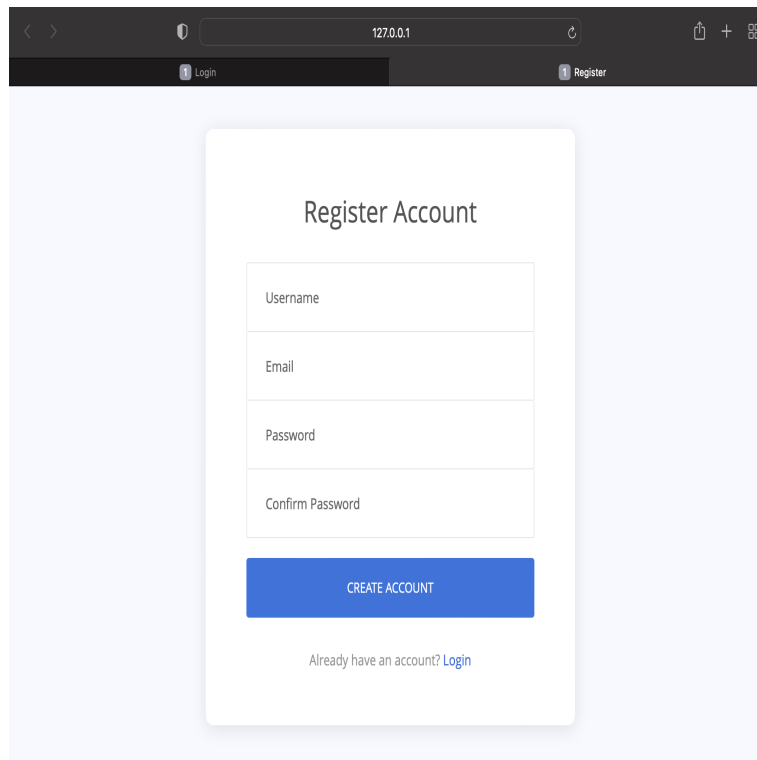


Figure A.12: Server-Side Sign Up

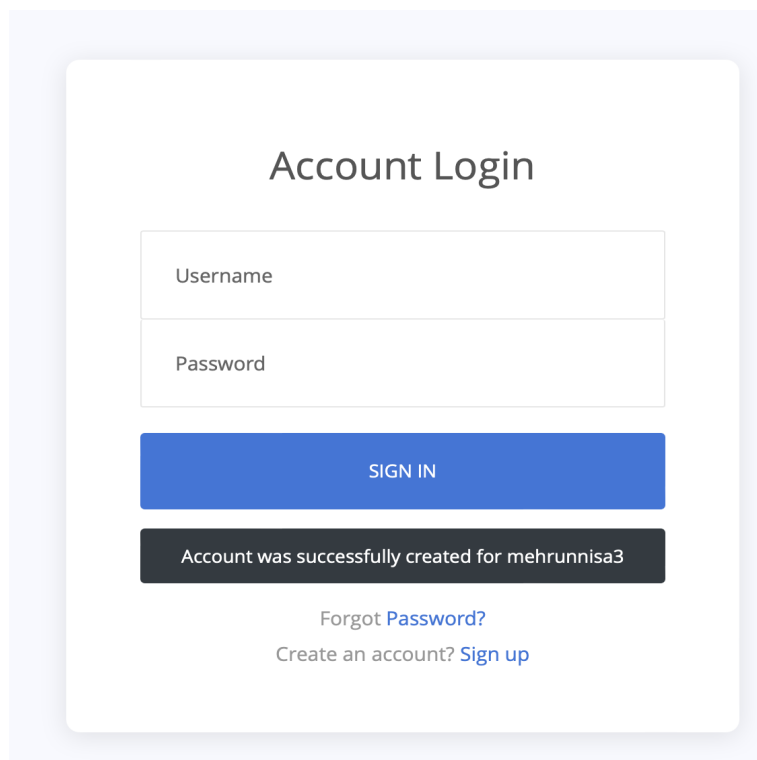


Figure A.13: Server-Side Login

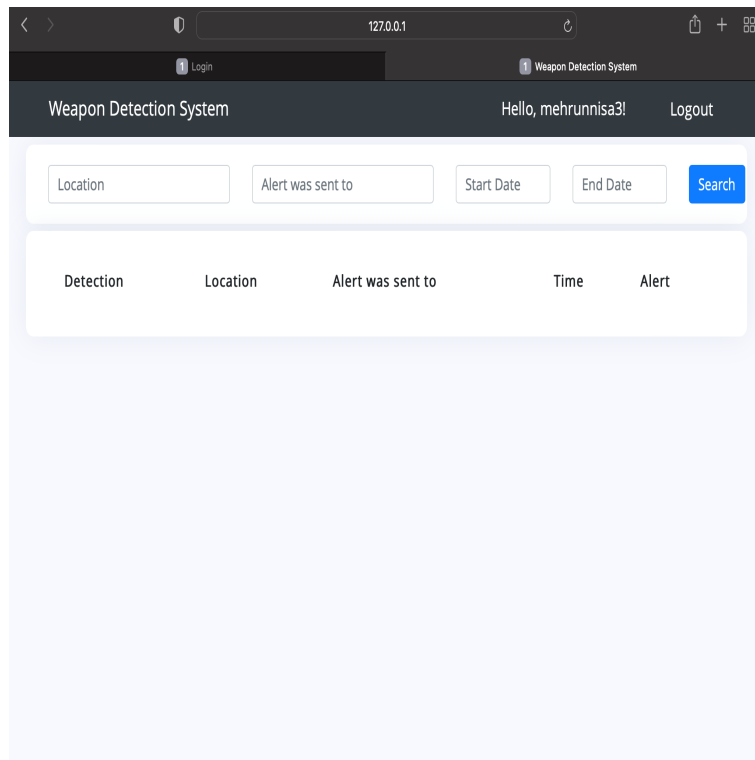


Figure A.14: Alerts Dashboard

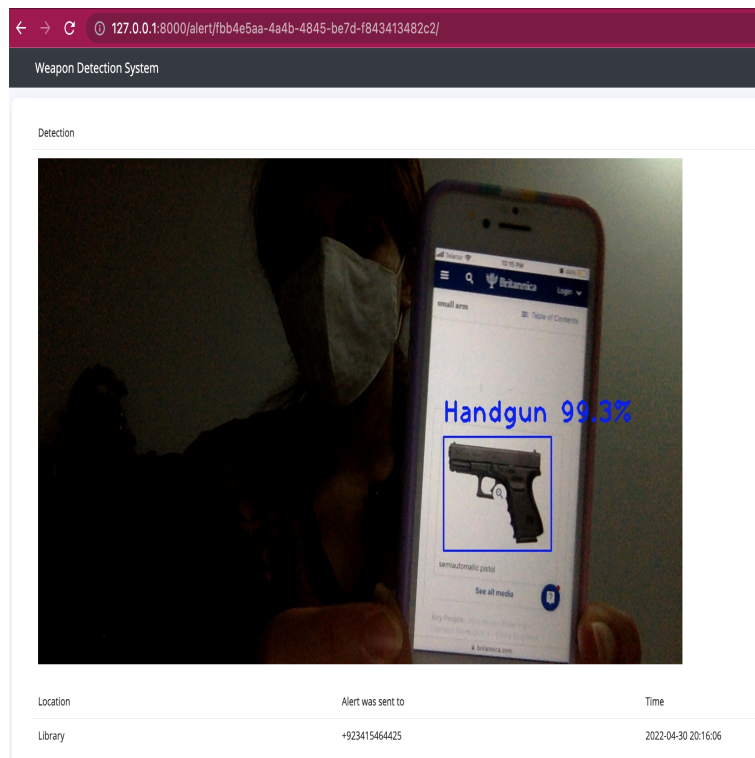


Figure A.15: Alerts

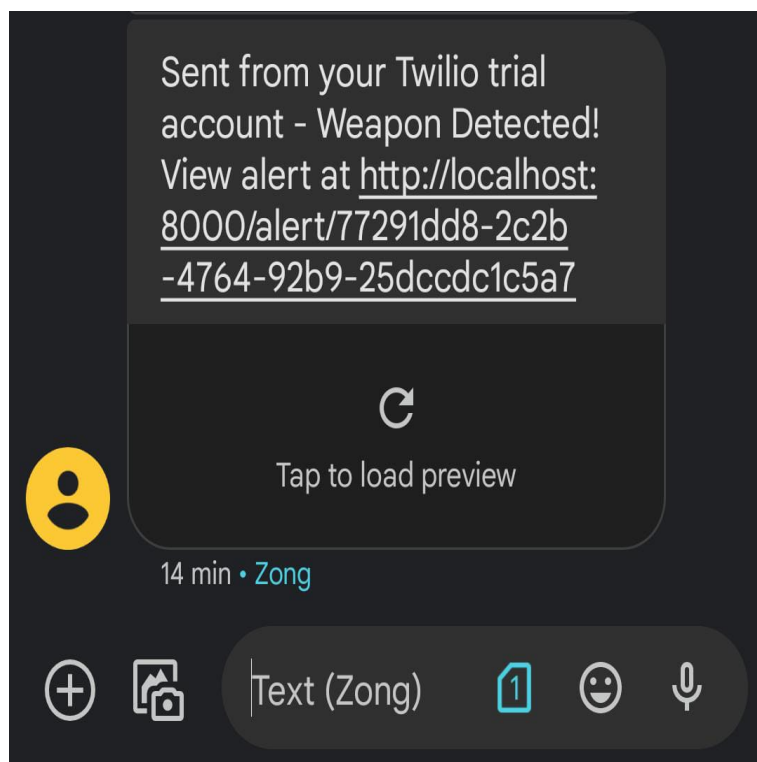


Figure A.16: SMS Notification Alert

# References

- [1] Rohit Kumar. Tiwari et al. A computer vision based framework for visual gun detection using harris interest point detector. *Procedia Computer Science*, 54:703–712, 2015.
- [2] Bhavna Khajone and VK Shandilya. Concealed weapon detection using image processing. *Int. J. Sci. Eng. Res*, 3:1–4, 2012.
- [3] Gyanendra K Verma and Anamika Dhillon. A handheld gun detection using faster r-cnn deep learning. In *Proceedings of the 7th international conference on computer and communication technology*, pages 84–88, 2017.
- [4] Jose L Salazar González, Carlos Zaccaro, Juan A Álvarez-García, Luis M Soria Morillo, and Fernando Sancho Caparrini. Real-time gun detection in cctv: An open problem. *Neural networks*, 132:297–308, 2020.
- [5] Gulzar Ahmad, Saad Alanazi, Madallah Alruwaili, Fahad Ahmad, Muhammad Adnan Khan, Sagheer Abbas, and Nadia Tabassum. Intelligent ammunition detection and classification system using convolutional neural network. 2021.
- [6] Muhammad Tahir Bhatti, Muhammad Gufran Khan, Masood Aslam, and Muhammad Junaid Fiaz. Weapon detection in real-time cctv videos using deep learning. *IEEE Access*, 9:34366–34382, 2021.
- [7] Harsh Jain, Aditya Vikram, Ankit Kashyap, Ayush Jain, et al. Weapon detection using artificial intelligence and deep learning for security applications. In *2020 International Conference on Electronics and Sustainable Communication Systems (ICESC)*, pages 193–198. IEEE, 2020.
- [8] Justin Lai and Sydney Maples. Developing a real-time gun detection classifier. *Course: CS231n, Stanford University*, 2017.

- [9] Javed Iqbal, Muhammad Akhtar Munir, Arif Mahmood, Afsheen Razaqat Ali, and Mohsen Ali. Leveraging orientation for weakly supervised object detection with application to firearm localization. *Neurocomputing*, 440:310–320, 2021.
- [10] Mehmet Tevfik Ağdaş, Muammer Türkoğlu, and Sevinç Gülseçen. Deep neural networks based on transfer learning approaches to classification of gun and knife images. *Sakarya University journal of computer and information sciences*, 4(1):131–141, 2021.
- [11] Arif Warsi, Munaisyah Abdullah, Mohd Nizam Husen, Muhammad Yahya, Sheroz Khan, and Nasreen Jawaid. Gun detection system using yolov3. In *2019 IEEE International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, pages 1–4. IEEE, 2019.
- [12] Lei Pang, Hui Liu, Yang Chen, and Jungang Miao. Real-time concealed object detection from passive millimeter wave images based on the yolov3 algorithm. *Sensors*, 20(6):1678, 2020.
- [13] Sanam Narejo, Bishwajeet Pandey, Ciro Rodriguez, M Rizwan Anjum, et al. Weapon detection using yolo v3 for smart surveillance system. *Mathematical Problems in Engineering*, 2021, 2021.
- [14] Coco dataset. <https://towardsdatascience.com/getting-started-with-coco-dataset-82def99fa0b8>.
- [15] Google colab. [https://colab.research.google.com/?utm\\_source=scs-index](https://colab.research.google.com/?utm_source=scs-index).
- [16] Visual studio code information. <https://code.visualstudio.com/>.
- [17] All about adobe xd. <https://www.adobe.com/products/xd.html>.
- [18] Labelimg. <https://github.com/tzutalin/labelImg>.