

# **SURVEILIA**

**Final Year Project**

**Session 2017-2021**

A project submitted in fulfilment of the  
COMSATS University Degree  
of  
BS in Computer Science (CUI)






Department of Computer Science

COMSATS University Islamabad, Lahore Campus

28 December 2020

## Project Detail

Type (Nature of project)	<input type="checkbox"/> Development <input type="checkbox"/> Research <input checked="" type="checkbox"/> <b>R&amp;D</b>			
Area of specialization	Deep Learning, Activity Recognition			
<b>Project Group Members</b>				
Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	SP17-BCS-109	IFRAH TEHLEEL	<a href="mailto:ifrahteh@gmail.com">ifrahteh@gmail.com</a>	
(ii)	SP17-BCS-028	JAN MUHAMMAD MIRZA	janmuhammadmirza@gmail.com	
(iii)	SP17-BCS-145	NAUMAN AKRAM	iemnauman.akram@gmail.com	

\*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

## Plagiarism Free Certificate

This is to certify that, I am IFRAH TEHLEEL D/o MUHAMMAD ASLAM RANJHA, group leader of FYP under registration no CIIT/SP17-BCS-109/LHR at Computer Science Department, COMSATS Institute of Information Technology, Lahore. I declare that my FYP proposal is checked by my supervisor and the similarity index is 6% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: 28-12-2020

Name of Group Leader: IFRAH TEHLEEL

Signature:



Name of Supervisor: DR. USAMA IJAZ BAJWA Co-Supervisor (if any): \_\_\_\_\_

Designation: ASSISTANT PROFESSOR Designation: \_\_\_\_\_

Signature: \_\_\_\_\_ Signature: \_\_\_\_\_

HoD: \_\_\_\_\_

Signature: \_\_\_\_\_

## Abstract

Surveillance cameras are gradually being used at every place that catches the anomalous event, yet the checking capacity of security agencies has not met the level. Traditional CCTV monitoring technologies are highly reliant on human attention to detect, infer, and monitor irregular, criminal behaviour. Moreover, the deep neural networks have limitations of resources as they are computation-intensive and training them requires a lot of training data and updated systems. To overcome this problem, our project Surveilialia; automatically detects the anomaly in real-time using low resources and instantly generates the alarm in case of any suspicious behaviour by using deep learning through activity recognition algorithms and video analysis.

We have trained our Deep Neural Network with two different architectures but the architecture we opted for application development is TSM+Mobilenetv2 in which we achieve training accuracy of 88% with the loss of 0.3233. and test accuracy of 83% in which Abnormal class had 86% Precision and 83% Re-call whereas 86% and 88% was the Precision and Re-call of our Normal class. In the end, based on these statistics, we were able to calculate the F1-score which was an abnormal class of 85% and a normal class having 87%. We calculated FLOPs to be a performance metric for our case. The complexity for ResNet50 was 8.24 GFLOPs and for MobileNetV2 was 0.64 GFLOPs. This clearly shows that MobileNetV2 is less computationally expensive, hence, we chose it. We used the UCF-Crime dataset [1] (a video dataset) for training to extract tsezZzhe features through Temporal Shift Module (TSM) [2] which aims to achieve 3D-CNN accuracy by maintaining 2D-CNN complexity. It would be able to differentiate between the abnormal and normal events in the live stream.

Mostly, such computations are complex and require intense computational resources. Hence, our target is to provide a deep learning model that is resource-efficient and works on simple machines such as a CPU. Furthermore, we have demonstrated our framework by porting our application (Surveilialia) on different platforms i.e., Windows and Linux, and to achieve better, faster inference results we deployed Surveilialia on NVIDIA Jetson Nano [3] (a small, powerful computer).

The proposed project is a python-based application developed using OpenCV, PyQt5, PyTorch, NumPy, SQLite3, and NVIDIA's CUDA.

## **Acknowledgment**

SURVEILIA has been funded by the National Grassroots ICT Research Initiative (NGIRI) at IGNITE.

# Table of Contents

SURVEILIA .....	1
Chapter 1: Introduction .....	14
1 Introduction .....	15
1.1 Introduction.....	15
1.2 Objectives .....	16
1.3 Problem statement.....	17
1.4 Assumptions & constraints .....	17
1.5 Project Scope .....	18
1.6 Product Features.....	19
Chapter 2: Requirements Analysis .....	20
2 Requirements Analysis.....	21
2.1 Literature review .....	21
2.1.1 Application-Based Related work.....	21
2.1.2 Research-Based Related work .....	22
2.2 Stakeholders list .....	27
2.3 Users of Surveilias.....	27
2.4 Requirements elicitation .....	28
2.4.1 Functional requirements .....	28
2.4.2 Non-functional requirements .....	31
2.5 Requirements traceability matric .....	33
2.6 Use case descriptions .....	34

2.6.1	Use case-01: Login .....	34
2.6.2	Use case-02: Logout .....	35
2.6.3	Use case-03: Anomaly Detection .....	35
2.6.4	Use case-04: To View Profile.....	36
2.6.5	Use case-06: View anomaly detected clips.....	37
2.6.6	Use case-07: View Users .....	38
2.7	Use case design .....	39
2.7.1	Use case-01: User and Admin Login.....	39
2.7.2	Use case-02: Logout .....	40
2.7.3	Use case-03: Anomaly Detection .....	41
2.7.4	Use case-04: View Profile .....	42
2.7.5	Use case-05: View Anomaly Detected Clip .....	43
2.7.6	Use case-06: View User .....	44
2.7.7	Use case: SURVEILIA (Complete System) .....	45
2.8	Software development life cycle model .....	46
2.8.1	Model Used in our project: .....	46
2.8.2	Why? .....	46
Chapter 3: System Design .....		48
3	System Design .....	49
3.1	Work breakdown structure (WBS).....	49
3.2	Activity diagram .....	50
3.2.1	Login .....	50

3.2.2	Anomaly Detection.....	51
3.2.3	Passing through the model.....	52
3.3	Sequence diagram .....	53
3.3.1	Sequence Diagram: Login .....	53
3.3.2	Sequence Diagram: Create a New User.....	54
3.3.3	Sequence Diagram: Anomaly Detection.....	55
3.4	Software architecture .....	56
3.5	Network diagram (Gantt chart) .....	57
3.6	Collaboration diagram.....	58
Chapter 4: System Testing.....		59
4	System Testing .....	60
4.1	Test Cases .....	60
4.1.1	Test Case-01 .....	60
4.1.2	Test Case-02 .....	61
4.1.3	Test Case-03 .....	61
4.1.4	Test Case-04 .....	62
4.2	Unit Testing .....	62
4.3	Integration testing .....	63
4.4	Acceptance testing .....	63
Chapter 5: Application .....		64
5	Application .....	65
5.1	Inference Engine .....	65



5.1.1	Dataset .....	65
5.1.2	Architecture .....	65
5.1.3	Comparison of the results achieved using ResNet50 and MobileNetV2 Architecture	
65		
5.1.4	Accuracy .....	66
5.1.5	Confusion Matrix.....	67
5.1.6	Classification Report .....	69
5.1.7	Loss .....	70
5.2	Application Frontend .....	71
5.2.1	Login Screen.....	71
5.2.2	Main/Home Screen .....	72
5.2.3	Choose Between Language Screens .....	72
5.2.4	Camera Screen .....	74
5.2.5	Camera Options Screen .....	75
5.2.6	Anomaly History Screen .....	76
5.3	Users Information Screen.....	77
5.4	Account Information Screen .....	78
Chapter 6: Project Code.....		79
6	Project Code .....	80
6.1	TSM_Model:.....	80
6.2	Recording Anomalous Events to the folder .....	86
6.3	Reading CSV file to display data .....	86
Chapter 7: Conclusion .....		88

7	Conclusion.....	89
7.1	Problems faced and lessons learned.....	89
7.2	Project summary .....	89
7.3	Future work.....	90
	Chapter 8: References .....	92
8	References .....	93

## List of Tables

Table 1: Characteristics of Surveilina Users .....	27
Table 2: Functional Requirement-01: Login .....	29
Table 3: Functional Requirement-02: Dashboard .....	29
Table 4: Functional Requirement-03: Anomaly Detection .....	30
Table 5: Functional Requirement-04: Video Storage.....	30
Table 6: Functional Requirement-05: Register New User .....	30
Table 7: Non-Functional Requirement-01: Security .....	31
Table 8: Non-Functional Requirement-02: Performance .....	31
Table 9: Non-Functional Requirement-03: Usability .....	32
Table 10: Non-Functional Requirement-04: Completeness .....	32
Table 11: Non-Functional Requirement-05: Robustness .....	32
Table 12: Non-Functional Requirement-03: Usability .....	33
Table 13: Requirement Traceability Matrix .....	33
Table 14: Use case description-01: Login .....	34
Table 15: Use case description-02: Logout .....	35
Table 16: Use case description-03: Anomaly Detection .....	35
Table 17: Use case description-04: View Profile .....	36
Table 18: Use case description-06: View Anomaly Detected Clip .....	37
Table 19: Use case description-07: View Users.....	38
Table 20: Gantt Chart .....	57
Table 21: Test Case-01 .....	60
Table 22: Test Case-02.....	61
Table 23: Test Case-03 .....	61
Table 24: Test Case-04.....	62

## List of Figures

Figure 1: Real-time Anomaly Detection using Trajectory-level Crowd Behaviour Learning .....	22
Figure 2: Architecture of AIVSS [16] .....	24
Figure 3: Performance of AIVSS [16].....	25
Figure 4: Localization of Anomaly [17].....	26
Figure 5: Dense Sampling [18].....	26
Figure 6: Use case-01: Login .....	39
Figure 7: Use case 02- Logout.....	40
Figure 8: Use case-03: Anomaly Detection.....	41
Figure 9: Use case-04: View Profile .....	42
Figure 10: Use case-05: View Anomaly Detected Clip .....	43
Figure 11: Use case-06: View User.....	44
Figure 12: Use case: Surveilina.....	45
Figure 13: Incremental Model .....	47
Figure 14: Work Breakdown Structure .....	49
Figure 15: Activity Diagram-01: Login .....	50
Figure 16: Activity Diagram-02: Anomaly Detection.....	51
Figure 17: Activity Diagram-03: Passing through the model.....	52
Figure 18: Sequence Diagram: Login.....	53
Figure 19: Sequence Diagram: To Create New User .....	54
Figure 20: Sequence Diagram: Anomaly Detection.....	55
Figure 21: System Architecture.....	56
Figure 22: Collaboration Diagram.....	58
Figure 23: Accuracy Plot for MobileNetV2 .....	66
Figure 24: Accuracy Plot for ResNet50 .....	66
Figure 25: Confusion Matrix for MobileNetV2 .....	67

Figure 26: Confusion Matrix for ResNet50.....	68
Figure 27: Classification Report for MobileNetV2.....	69
Figure 28: Classification Report for ResNet50 .....	69
Figure 29: Loss Plot for MobileNetV2.....	70
Figure 30: Loss Plot for ResNet50 .....	70
Figure 31: Login Screen .....	71
Figure 32: Main/Home Screen .....	72
Figure 33: Language Screen.....	73
Figure 34: Screen with the Urdu Language.....	73
Figure 35: Camera Screen .....	74
Figure 36: Camera Options Screen.....	75
Figure 37: Anomaly History Screen.....	76
Figure 38: Users Information Screen .....	77
Figure 39: Account Information Screen .....	78

# **Chapter 1: Introduction**

# 1 Introduction

## 1.1 Introduction

Security has become a very important and necessary factor in our daily life. From small houses to massive industries surveillance is a basic aspect for everyone. With the advent of artificial intelligence, the work in surveillance systems is reaching its maximum limits.

Surveillance cameras play a significant role in preventing or at least recognizing the crime. Due to cheap technology, surveillance cameras are installed everywhere such as streets, shopping malls, hospitals, schools, banks, and even residential areas to observe human activities. The basic purpose of surveillance cameras is to keep track of human activity to handle sudden events that may require the attention of the public or maybe only of authorities. Of late, such cameras are used keeping in mind the ultimate objective to build a secure environment for people. But these traditional surveillance cameras are human-based, i.e., it requires a person/s to constantly monitor the video feed. Hence, full of errors, biases, and more importantly exhaustive. In the past, there were a lot of anomalous events that could have been avoided but the traditional surveillance system failed to do the job such as loot of 10 mobile shops in Karachi [4], armed robbery at Westfield mall [5], and many others of similar nature. A solution to this problem is needed, that is cost-effective and simple to use.

A lot of work has been done to automate the security in the field of artificial intelligence. In the past classical deep neural networks were used which were not accurate and did not have good quality results then the deep neural networks overcame them and provided better accuracy, but the main issue is most of them are computationally expensive and require the users to change their system architecture or system hardware, due to which it turns out to be very costly. Our proposed project performs the task of identifying unexpected incidents, cost-effectively using advanced deep learning-based methods like taking a large amount of data. It provides the solution by providing resource efficiency thus not requiring any update in the system or usage of any new, advanced, and expensive technology.

As our goal is to implement the deep neural network to detect anomalies in real-time camera feed, which is a video streaming hence we require a video dataset to train and test our deep neural network. For this reason, we have trained and tested using the UCF Crime dataset [1]. In our case, the dataset consists of around 1120 videos with a total runtime of 70 hours. It consists of 14

categories, which include 13 types of anomalous events i.e. (accident, stealing, robbery, shoplifting, arson, explosion, shooting, arrest, vandalism, burglary, fighting, abuse, and assault,) and one for normal events. The dataset is divided with an 80:10:10 ratio for training, testing, and validation, respectively. The system is trained on two different architectures using Temporal Shift Module [2] i.e., Resnet50 [6] and MobilenetV2 [7]. The frontend of the main system is developed in Python using the PyQt5 library. As the user of our application would be a security guard who might not be very familiar with technology, we have kept low key, simple, and easy to use interface with minimum clicks required to move between the options and perform actions. It provides the user with the option to either add a camera or input a video file. Additionally, a multi-language (English and Urdu) option is also included. Furthermore, we have distinguished our work by porting our application (Surveilialia) to different platforms i.e., Windows and Linux, and to achieve better faster inference results we deployed Surveilialia on NVIDIA Jetson Nano [3] (a small, powerful computer).

Hence, with the development of **Surveilialia**, any abnormal activity in a live video stream or a video file is instantly reported. And the system clips and stores the part where an abnormal activity (if any) occurs, making the process of a surveillance system reliable, efficient, effective, and reducing the constant human dependency.

## 1.2 Objectives

The objectives for the proposed project are as follows:

- To develop a product that can detect the actions of humans and their interpretation using deep learning knowledge through automated activity analysis.
- To provide a resource-efficient system that works on a simple machine such as a CPU with real-time streaming.
- To build a system that can run and detect activities on multiple screens, simultaneously.
- The product that supports the security workers to enhance the method of handling abnormal activities in real-time
- To observe suspicious/anomalous behavior and reduce constant human dependence.
- To automate the anomaly detection process likely to be as human
- Clipping and storing the part where the anomaly has occurred
- To build the application GUI, user-friendly, and interactive so that a common man can use it such as a security guard.



### 1.3 Problem statement

The present world has evolved in the terms of advanced technology but still one does not feel secure at the malls and other public places. Every day, we hear stories of robbery, vandalism, and other street crimes. Though the surveillance cameras are increasingly installed in almost all public areas, some of the incidents are either unnoticed or are detected when it is too late as the law enforcement monitoring abilities have not kept up with the pace. The eyes of law enforcement miss several anomalous incidents and crimes still exist. The surveillance is limited to human operators. It is almost impossible for a human brain to thoroughly focus on multiple CCTV screens when multiple activities are taking place in parallel across each screen. Some have improved their systems by using DVR (digital video recorder) or applications built through deep learning techniques, but they require an update in systems or CCTVs with integrated chips. Hence, there is a need for a deep learning-based application that utilizes fewer resources, unlike the others.

To solve this problem, there is a need for a low resource-efficient automated system that differentiates between normal and abnormal events in multiple streams, simultaneously, and in case of any abnormal event, the security team is alerted to take appropriate actions immediately. The proposed project is a solution to this problem, it is a low resource-efficient automated system that detects abnormal activities when the multiple video streams of CCTV are passed through the trained Neural Network in real-time and alerts the security team against any unusual activity detected to take required measures.

### 1.4 Assumptions & constraints

The proposed project is assumed to be a user-friendly application that helps to handle surveillance tasks by automatically detecting anomalies and improving the security system as a result. It is assumed to be a low resource-efficient application that runs on any PC. It is believed that the product benefits through:

- Reducing human efforts to save the exhaustive 24/7 watch duty
- Real-time detection of anomalies in multiple screens, simultaneously
- Building a resource-efficient system that works on a simple machine such as a CPU with real-time streaming.
- Reducing crime rates by the implementation of strict controls everywhere
- Identification of the suspects and real victims

Although there are a lot of pros in the proposed application, there are some small but technical cons; the detection of abnormal and normal activities as separate is doubtful, especially in crowded areas. Moreover, the unavailability of labeled datasets makes the system to be slightly inefficient as its learning is not enough. Another major issue is maintaining the privacy of the public during anomaly detection. The application does not deal with the multiple types of abnormal activities as it is intended to perform binary classification, hence, it is only limited to two classes i.e., '*normal*' and '*abnormal*' (activities involving abuse, shoplifting, assault, stealing, vandalism, arson, explosion, arrest, fighting, burglary, accident, shooting and robbery).

## 1.5 Project Scope

Security is one of the key concerns for any person, either at home or in public places. Surveillance cameras are now being used in almost every area, be it home or public place. Security guards sit behind those CCTV screens, inspecting the behavior of people, and monitoring their activities. But unfortunately, the human brain cannot focus on multiple screens at a single time. Several applications are being developed to automatically detect abnormal activities therefore the security can be ensured, and people can move without safety risks and concerns. We have developed an application that automatically detects abnormal activity in surveillance CCTV streams and generates an alarm.

Presently, much work is done in the field of activity recognition but most of them require high computational costs, expensive and advanced systems with the best GPUs. Hence, they are computation-intensive, and training them requires a lot of training data. Resultantly, the user must update his/her system to achieve the desired result. To overcome this problem, we have proposed a low resource-efficient system that detects an anomaly in multiple streams in real-time.

The proposed application includes,

- Real-time identification of suspicious activities like robbery, theft, etc.
- Automation and facilitation of intensive Surveillance workload for single to multiple camera streams (up to 6 simultaneously)
- A resource-efficient application requiring low computation costs.
- Real-time detection of anomalous/suspicious human activities over live CCTV/ IP Camera input feed using deep learning approach.
- Detection of activities that require the attention of security personnel so that such activities can be dealt with on time and be minimized.

- Additionally, we have embedded AI on the edge, with Inference Engine, core application deployed on NVIDIA's Jetson Nano I-GPU powered Platform

## 1.6 Product Features

The following are the features of Surveilias:

- **Usability:** The application is easy and convenient to use. It also works on any system installed at the security office.
- **Portability:** The application is portable as it can be deployed on Linux, Windows, and NVIDIA Jetson Nano platform.
- **Functionality:** It is functional on CCTV Live Stream or stored video.
- **Flexibility:** It can adapt to possible or future changes in the requirements.

## **Chapter 2: Requirements Analysis**

## **2 Requirements Analysis**

### **2.1 Literature review**

A considerable amount of work has been done and published in the field of anomaly detection through deep learning techniques. Many companies have already installed systems for automatic detection of abnormal activity to maintain their security level.

Here, we have mentioned a few practical applications and research works published in this field and how our proposed system differs from them.

#### **2.1.1 Application-Based Related work**

##### **1) Surveillance App**

Surveillance App [8] is a real-time wireless surveillance CCTV home system app developed by Reservoir Dev, which detects noise or any movement in the live stream and notifies the viewer. It allows the user to monitor his/her home and speak through the user's device microphone to scare off the thief/intruder. But it is limited to the indoor or specifically, home of the user.

##### **2) iCentana**

iCentana [9] is AI video analytics for automated real-time identification of critical events. It monitors cameras and automatically detects any abnormal activity. It identifies precursor events, learns, and adapts automatically, rapidly reviews recorder video, and detects abnormal precursor events. They are costly and lack in localization of anomalies.

##### **3) Dahua Security**

A Chinese CCTV company [10] launched cameras to detect anomalies in public areas. Their mission is "Enabling a safer society and smarter living". They have cameras with integrated chips to be able to run deep neural networks right on board. Key technologies used are HDCVI Technology, Predictive Focus Algorithm, EPOE, and ANPR. Though, it is much accurate but requires special CCTV cameras with integrated chips, which are very costly.

#### 4) Hikvision

Hikvision [11] provides digital surveillance and is also Chinese based, CCTV cameras embedded with intelligent video analytics, company. But people report that it is poor efficiency, does not train properly and causes much delay in alarming the security person which is a huge problem.

#### 5) Mobotix

Mobotix [12] provides a surveillance system for both indoor and outdoor. It is secure to use. But it is very much complex in terms of its user interface. Most of the important features are hidden behind the toolbars.

### 2.1.2 Research-Based Related work

#### 1) Real-time Anomaly Detection using Trajectory-level Crowd Behaviour Learning

The paper [13] in the topic presents anomaly detection involving low to medium populated crowded areas in real-time, using trajectory level behavioral learning. It suggests a system that automatically detects the anomaly in the crowd. It divides the output into two distinct classes: normal and abnormal. In this case, an abnormal activity is classified as anyone violating the uniform movement pattern of the crowd.

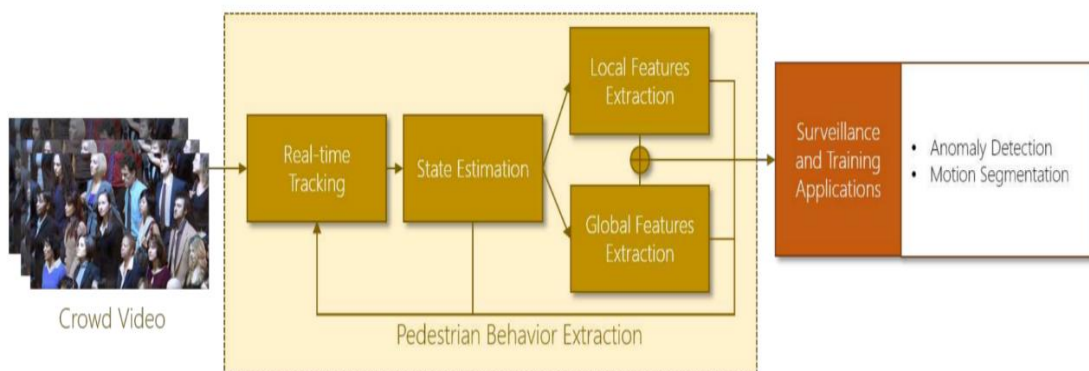


Figure 1: Real-time Anomaly Detection using Trajectory-level Crowd Behaviour Learning

Figure 1 demonstrates an overview of the approach they used. The different stages of their algorithm are real-time tracking, state estimation, and feature extractions. In Figure 1, local and global features are used for individual and overall crowd movement. Crowd video is passed to the system and by the application of the algorithm, any local or global abnormal activity is detected. They have demonstrated the performance on PETS 2016 ARENA dataset.

However, this paper only limits to the abnormality in the direction of movement of the crowd. Other abnormal events such as accidents, theft, vandalism, burglary are not dealt with in this proposition.

## **2) IoT based Security System using Raspberry Pi**

The mentioned study [14] depicts a security system that performs its actions using a Raspberry Pi. Where, Raspberry Pi [15] is a card-sized, low-cost device that plugs with the computer monitor or another screen. The system works when any movement in the area is detected by the PIR sensors and IR sensors attached to the raspberry pi. the user is instantly notified in the case of abnormal activity, which for this is the movement of people.

This paper shows that many devices like raspberry pi, sensors, etc. are required to provide complete security whereas our proposed system is resource-efficient which does not require any external hardware or other update and works on the CPU of a normal PC.

## **3) TSM: Temporal Shift Module for Efficient Video Understanding**

Reference [2] describes that the explosive growth of video distribution creates challenges for high-precision and low computing cost output of video interpretation. Conventional 2D CNNs are computationally inexpensive, but temporal relationships cannot be captured; 3D CNN based approaches can achieve good efficiency, but are computationally expensive, making it costly to deploy.

They proposed a generic and efficient Temporal Shift Module (TSM) [2] in this paper that enjoys high productivity as well as high performance. 3D CNN output can be attained, but 2D CNN performance can be maintained. TSM transfers part of the channels along the temporal dimension, thereby facilitating the exchange of knowledge between neighboring frames. Their

architecture is both powerful and precise, allowing video recognition on edge devices with low latency.

#### 4) Advance intelligent video surveillance system (AIVSS)

AIVSS [16] is a research-based article describing the need for intelligent video surveillance systems as surveillance cameras are increasing day by day. The paper proposes the idea to develop surveillance cameras based on internet protocol (IP).

The author explains that the IP videos are much better in terms of processing power and improved detection. Figure 2 below depicts the architecture of AIVSS. It shows that all the elements of the system are connected using IP cameras.



*Figure 2: Architecture of AIVSS [16]*

Figure 3 below depicts the system's performance. It not only depends on the hardware and software but the privacy, too. The author explains that the privacy issue is a crucial factor in many countries. On the other side, the author explains that network performance is also important.



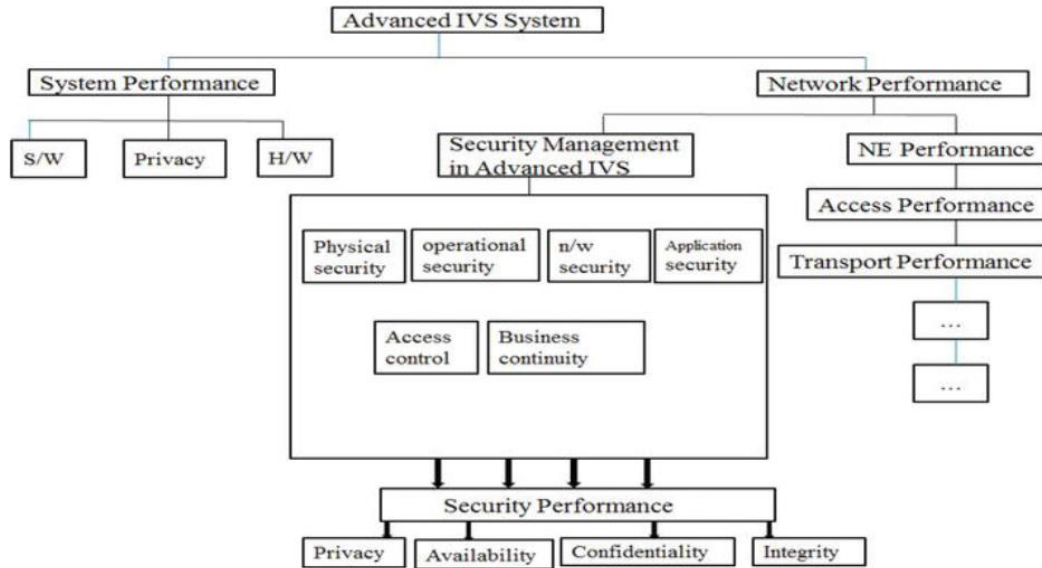


Figure 3: Performance of AIVSS [16]

## 5) Anomaly Localization in Topic-based Analysis of Surveillance Videos

The paper [17] in the subject here recommends a technique to spot the location of an anomaly in the video with many agents actively pursuing different tasks, and the size information by integrating Spatio-temporal gradient descriptors. The author has discussed the framework to be three-tier for the analysis of abnormal videos. The flow of the application is modelling, detection, and then localization of anomaly. Figure 4 below depicts the localization of anomalies using different techniques.

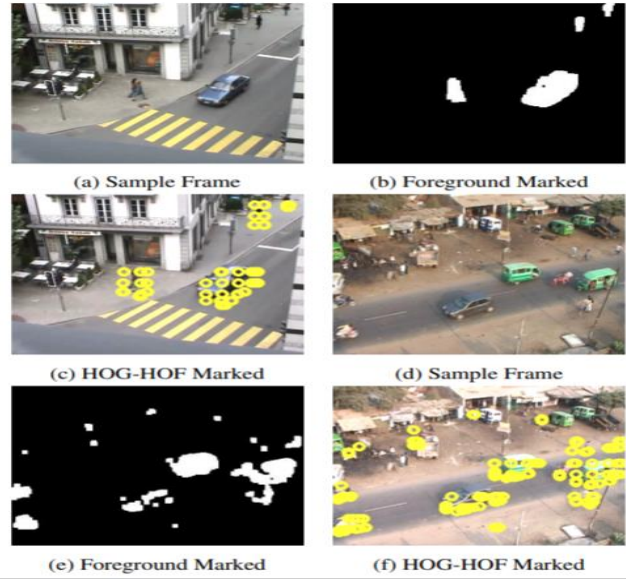


Figure 4: Localization of Anomaly [17]

## 6) Video Anomaly Detection in Confined Areas

The paper [18] focuses on the detection of an anomaly in confined areas. It proposes a new technique and algorithm for detection at such places. According to this research paper, the event is said to be abnormal when the speed of the object, the motion of the object in forbidden time, or the path of the object is different from what it is trained with. The system converts the video into grayscale first and then it samples the frames into Spatio-temporal volumes using dense sampling as depicted in Figure 5. The events with relatively lesser occurrence are detected as anomalous.

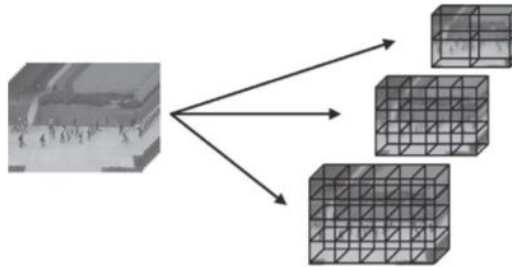


Figure 5: Dense Sampling [18]

## 7) Survey Paper on Anomaly Detection in Surveillance Videos

In paper [19], the author describes the need for automated anomaly detection in surveillance videos. The method used in this paper is to train the neural network using deep multiple instance learning. But it only detects the presence of the anomaly. The location and time cannot be detected. This paper motivated us to create an application that not only detects the anomaly but also the location and time.

## 2.2 Stakeholders list

Stakeholders of our project are,

- Security Firm
- Anomaly Detection Researchers
- Data Analysts
- Law Enforcement
- Project team
  - Ifrah Tehleel
  - Jan Muhammad Mirza
  - Nauman Akram
- Project Supervisor
  - Dr. Usama Ijaz Bajwa
- Final Year Project Jury
- CUI Lahore

## 2.3 Users of Surveilila

Below Table 1 are the types of authorized users, along with their detailed characteristics who may use this application:

*Table 1: Characteristics of Surveilila Users*

User Type	Computer Knowledge Level	Business Knowledge Level	Access and Permissions	Frequency of Use
-----------	--------------------------	--------------------------	------------------------	------------------

<b>Admin</b>	Basic to Advanced level	Can easily obey the instructions of the system	All permissions and access are granted.	Uses whenever the detail of other users is to be updated or a new user is to be added. Uses over an extended period.
<b>Security Guard or another User</b>	Basic to Advanced level	Can easily obey the instructions of the system	Limited access and permissions.  Not allowed to view or edit users' details.	Uses daily to monitor the video streams or to check the alarm.

## 2.4 Requirements elicitation

A significant part of our specifications is compiled from observation and brainstorming about reality. We read a large variety of academic papers and visited tech forums and programmer groups that are important. We also met with the specialists in the areas of Computer Vision, Deep Learning, and NVIDIA's CUDA Platform patented toolchain.

We addressed our project with individuals who had already done work in a related field before us and their limitations, precision problems, network architecture techniques, etc. were addressed. We have also been focusing on defining the best interface methods for the implementation of our application, with functionalities already installed.

### 2.4.1 Functional requirements

A functional requirement is a description of the facility that the software should perform. It defines a system or its components, all the tasks that the machine executes, and the processes. It benefits you to capture the planned performance of the system. Functional requirements for our application '*Surveilia*' are:

### 2.4.1.1 Functional Requirement-01: Login

In Table 2 below, the functional requirements of the login page are mentioned.

*Table 2: Functional Requirement-01: Login*

ID	Functional Requirements
FR11	The application shall permit the user to log in.
FR12	The application shall maintain a separate account for each user.
FR13	The application shall grant the admin access after verification of login credentials.
FR14	The application shall permit the admin to update his/her username and password.
FR15	The application shall only permit the admin to change the login credentials of other users.
FR16	The application shall maintain a database of all the user accounts.

### 2.4.1.2 Functional Requirement-02: Dashboard

In Table 3 below, the functional requirements of the dashboard are mentioned.

*Table 3: Functional Requirement-02: Dashboard*

ID	Functional Requirements
FR21	The application shall permit the user to choose between options displayed on the dashboard.

### 2.4.1.3 Functional Requirement-03: Anomaly Detection

In Table 4 below, the functional requirements of anomaly detection are mentioned.

*Table 4: Functional Requirement-03: Anomaly Detection*

ID	Functional Requirements
FR31	The application shall alert the user/admin when an anomaly is detected.
FR32	The application shall permit the admin and the user to view the clipped videos.

#### **2.4.1.4 Functional Requirement-04: Video Storage**

In Table 5 below, the functional requirements of the video storage are mentioned.

*Table 5: Functional Requirement-04: Video Storage*

ID	Functional Requirements
FR41	The application shall store the anomaly.
FR42	The application shall allocate a unique ID to each clip in anomaly history.
FR43	The application shall permit the admin/user to view any video clip from the anomaly history.

#### **2.4.1.5 Functional Requirement-05: Register New User**

In Table 6 below, the functional requirements of adding the new user are mentioned.

*Table 6: Functional Requirement-05: Register New User*

ID	Functional Requirements
FR51	The application shall only permit the admin to add a new user.

FR52	The admin must add the information of the new user.
------	---

## 2.4.2 Non-functional requirements

Non-functional requirements describe the performance quality of the software system. It permits you to impose limitations on the design of the system. It sets the system's criteria and constraints. Apart from its simple features, it describes what is required from the system. Non-functional requirements of our application 'Surveilias' are:

### 2.4.2.1 Non-Functional Requirement-01: Security

The non-functional requirements regarding security are mentioned in Table 7.

*Table 7: Non-Functional Requirement-01: Security*

ID	Non-Functional Requirement
NFR11	Only the Logged in/Signed up the user(s) shall be allowed to access the functionalities. The application must only allow authorized users with correct login credentials to access the system.
NFR12	The application shall be secure.

### 2.4.2.2 Non-Functional Requirement-02: Performance

The non-functional requirements regarding the performance of the application are mentioned in Table 8.

*Table 8: Non-Functional Requirement-02: Performance*

ID	Non-Functional Requirement
NFR21	The start-up time of the application must not be more than 20 seconds.
NFR22	The application must alert the user when an anomaly has been detected, and the time required for it must be no more than 5 seconds.
NFR23	The response time between click and reaction must be less than two

	seconds.
--	----------

#### 2.4.2.3 Non-Functional Requirement-03: Usability

The non-functional requirements regarding usability are mentioned in Table 9.

*Table 9: Non-Functional Requirement-03: Usability*

ID	Non-Functional Requirement
NFR31	The interface of the application must be easy to understand.
NFR32	The user must get familiar with it in no more than 20 seconds.

#### 2.4.2.4 Non-Functional Requirement-04: Completeness

The non-functional requirements regarding the completeness are mentioned in Table 10.

*Table 10: Non-Functional Requirement-04: Completeness*

ID	Non-Functional Requirement
NFR41	The application must always be consistent and efficient in detecting the anomaly.
NFR42	The application must always generate the same result for a user's task.

#### 2.4.2.5 Non-Functional Requirement-05: Robustness

The non-functional requirements regarding the robustness are mentioned in Table 11.

*Table 11: Non-Functional Requirement-05: Robustness*

ID	Non-Functional Requirement
NFR51	The application's average time to crash should not be more than 10 times within a month.



### 2.4.2.6 Non-Functional Requirement-06: Reusability

The non-functional requirements regarding usability are mentioned in Table 12.

*Table 12: Non-Functional Requirement-03: Usability*

ID	Non-Functional Requirement
NFR61	The application must permit the reuse of the functions in the system in a different environment.

## 2.5 Requirements traceability matrix

Table 13 displays the requirements traceability matrix.

*Table 13: Requirement Traceability Matrix*

Test Case ID	FR_ID	Description of Requirement	Objective	Priority
01	FR-05	The system will facilitate the users with 2 types of accounts i.e., Admin and User.	Register a new user.	High
02	FR-01	The system will facilitate the user with a login system.	Logging user account.	High
03	FR-02	The system allows the user to choose between the items on the dashboard.	Choosing options on Dashboard.	High
04	FR-04	The system shall clip and store the anomaly frames (image and video).	Anomaly Clipping	High
05	FR-03	The system shall detect the	Anomaly Detection	High

		anomaly.		
--	--	----------	--	--

## 2.6 Use case descriptions

### 2.6.1 Use case-01: Login

The use case description for the login process is shown in Table 14.

*Table 14: Use case description-01: Login*

<b>ID:</b>	1
<b>Name of Use Case:</b>	Login
<b>Actors:</b>	Admin, User
<b>Description:</b>	The admin/user wants to log in to the application.
<b>Pre-Condition:</b>	Admin/User should already have an account. Users must know his/her login credentials of his/her account.
<b>Post-Condition:</b>	Admin/User has been signed into the application, successfully.
<b>Events:</b>	<ol style="list-style-type: none"> <li>1. Admin/user opens the application.</li> <li>2. Admin/user fills in the username and password fields. And selects the type of user.</li> <li>3. Admin/user clicks on the login button.</li> <li>4. On authentication from the database, the user has been signed in.</li> </ol>
<b>Alternatives Flow:</b>	<ul style="list-style-type: none"> <li>• Admin/user enters an incorrect username or password or user type.</li> <li>• Access denied</li> <li>• In case of forgetting the password, the user must contact the admin.</li> </ul>

<b>Exceptions:</b>	None
--------------------	------

### 2.6.2 Use case-02: Logout

The description of the use case of the logout process is shown in Table 15Table 15.

*Table 15: Use case description-02: Logout*

<b>ID:</b>	2
<b>Name of Use Case:</b>	Logout
<b>Actors:</b>	Admin, User
<b>Description:</b>	The user can logout from the application.
<b>Pre-Condition:</b>	The user must be signed in to his/her account.
<b>Post-Condition:</b>	Admin/user has been successfully logged out from the application.
<b>Events:</b>	<ol style="list-style-type: none"> <li>1. The user presses the logout button.</li> <li>2. The user logs out from the application.</li> </ol>
<b>Alternatives Flow:</b>	The user has already logged out.
<b>Exceptions:</b>	None

### 2.6.3 Use case-03: Anomaly Detection

The use case description for the detection of anomaly is shown in Table 16.

*Table 16: Use case description-03: Anomaly Detection*

<b>ID:</b>	3
<b>Name of Use Case:</b>	Anomaly Detection
<b>Actors:</b>	Admin, User
<b>Description:</b>	The application shall detect the anomaly, clip the

	video, store it, and alert the user via the desktop application.
<b>Pre-Condition:</b>	There should be a live feed.
<b>Post-Condition:</b>	An anomaly has been detected.
<b>Events:</b>	<ol style="list-style-type: none"> <li>1. The camera captures the live stream.</li> <li>2. The application pre-processes the stream.</li> <li>3. The application detects the anomaly.</li> <li>4. The application clips the relevant part.</li> <li>5. The application notifies the user.</li> </ol>
<b>Alternatives Flow:</b>	No anomaly has been detected.
<b>Exceptions:</b>	None

#### 2.6.4 Use case-04: To View Profile

The description of the use case to view the profile is shown in Table 17.

*Table 17: Use case description-04: View Profile*

<b>ID:</b>	4
<b>Name of Use Case:</b>	View Profile
<b>Actors:</b>	Admin, User
<b>Description:</b>	The application shall permit the user/admin to view the profile and update it.
<b>Pre-Condition:</b>	The user/admin must be signed in with the correct sign-in credentials.
<b>Post-Condition:</b>	The user has viewed and updated his profile.
<b>Events:</b>	<ol style="list-style-type: none"> <li>1. The user/admin presses the account button.</li> <li>2. The user/admin clicks the view profile button.</li> </ol>

	<ol style="list-style-type: none"> <li>3. The user/admin updates the profile by entering updated information.</li> <li>4. The user/admin changes his/her password if required.</li> </ol>
<b>Alternatives Flow:</b>	The user/admin does not update his profile and returns.
<b>Exceptions:</b>	None

### 2.6.5 Use case-06: View anomaly detected clips

The use case description to view the anomaly detected clip is shown in Table 18.

*Table 18: Use case description-06: View Anomaly Detected Clip*

<b>ID:</b>	6
<b>Name of Use Case:</b>	View anomaly detected clip
<b>Actors:</b>	User and Admin
<b>Description:</b>	The application shall permit the admin to view the anomaly detected clip log.
<b>Pre-Condition:</b>	The anomaly detected clip must be present in the log.
<b>Post-Condition:</b>	The clip is found and successfully played.
<b>Events:</b>	<ol style="list-style-type: none"> <li>1. The admin/user logs in to the application.</li> <li>2. The admin/user clicks the button of anomaly clips.</li> <li>3. The admin/user searches a particular event from the clip history.</li> <li>4. The admin clicks the URL.</li> <li>5. The video starts to play.</li> </ol>
<b>Alternatives Flow:</b>	There is no video in the anomaly detected clip log.
<b>Exceptions:</b>	None

### 2.6.6 Use case-07: View Users

The description of the use case to view users are shown in Table 19.

*Table 19: Use case description-07: View Users*

<b>ID:</b>	7
<b>Name of Use Case:</b>	View Users
<b>Actors:</b>	Admin
<b>Description:</b>	The application shall permit the admin to view, edit, and update the information of the other users.
<b>Pre-Condition:</b>	The admin must be signed into the system with the correct sign-in credentials.
<b>Post-Condition:</b>	The admin has viewed, updated, deleted, or added users.
<b>Events:</b>	<ol style="list-style-type: none"><li>1. The admin clicks the user's button.</li><li>2. The admin views, updates, deletes, or adds new users.</li></ol>
<b>Alternatives Flow:</b>	The user does not exist.
<b>Exceptions:</b>	None

## 2.7 Use case design

### 2.7.1 Use case-01: User and Admin Login

The use case for the login process for the user and admin is shown in Figure 6 **Error! Reference source not found.**

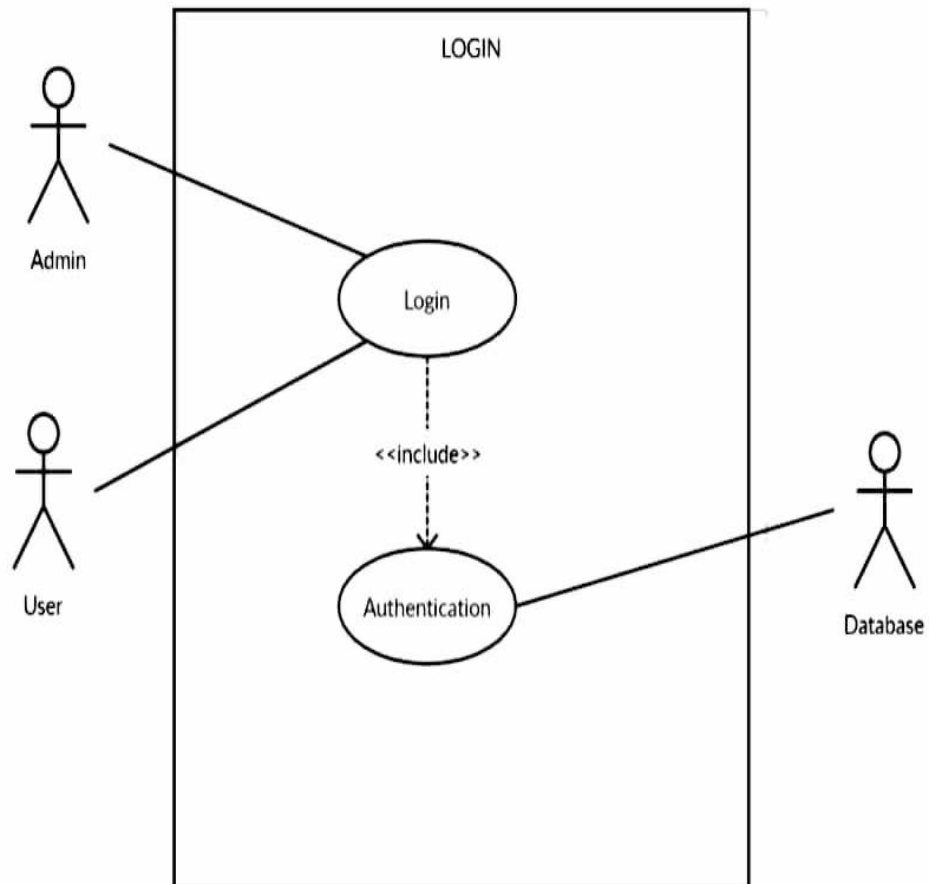


Figure 6: Use case-01: Login

### 2.7.2 Use case-02: Logout

The options for the logout process of the admin and the user are shown in Figure 7.

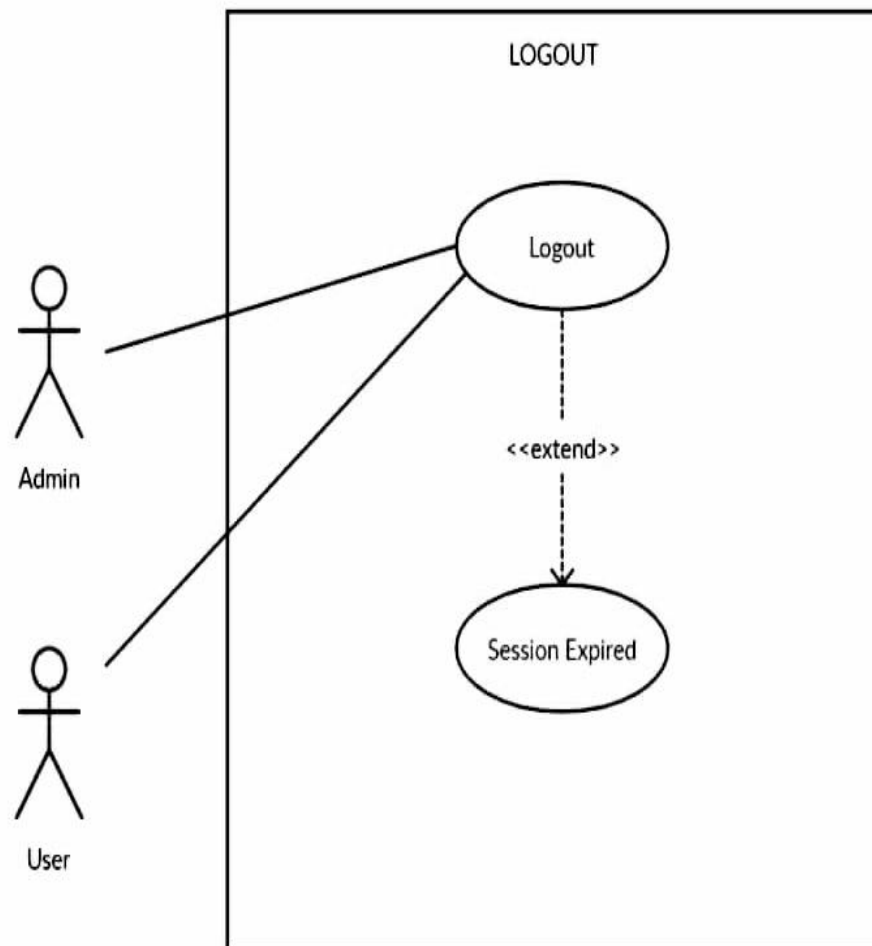


Figure 7: Use case 02- Logout



### 2.7.3 Use case-03: Anomaly Detection

The options for anomaly detection are represented in Figure 8 in form of a use case diagram.

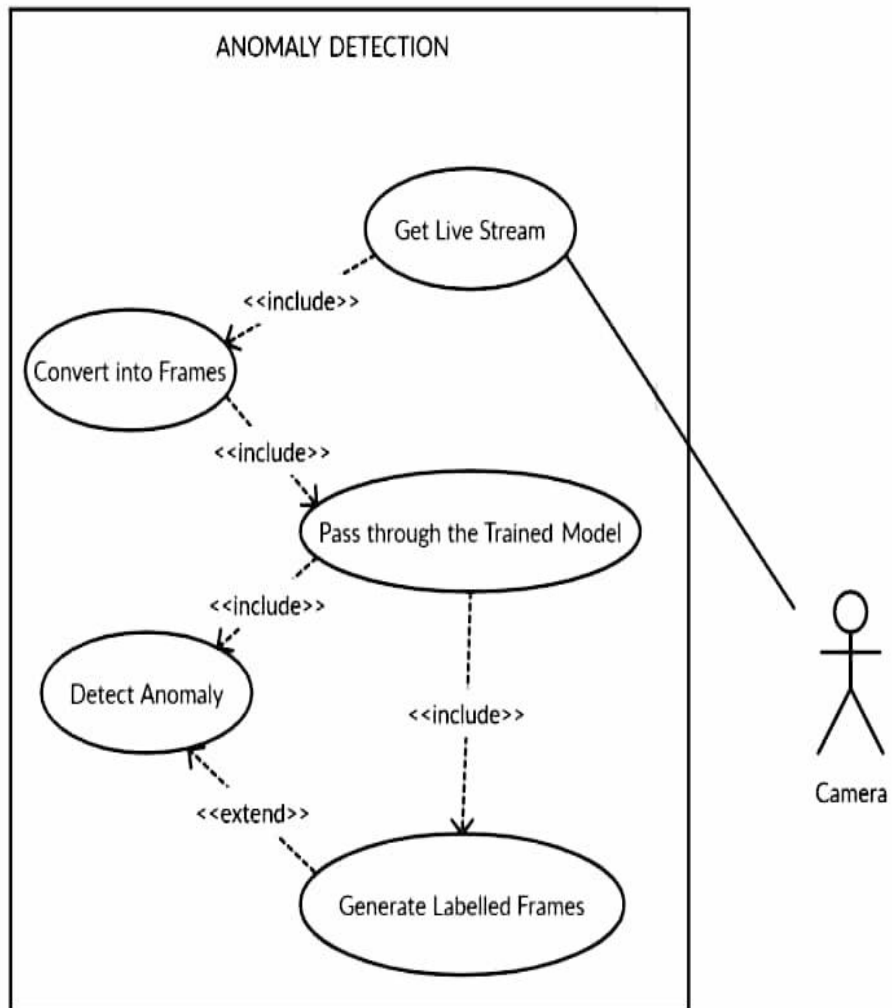


Figure 8: Use case-03: Anomaly Detection

#### 2.7.4 Use case-04: View Profile

The option to view the profile is shown in Figure 9.

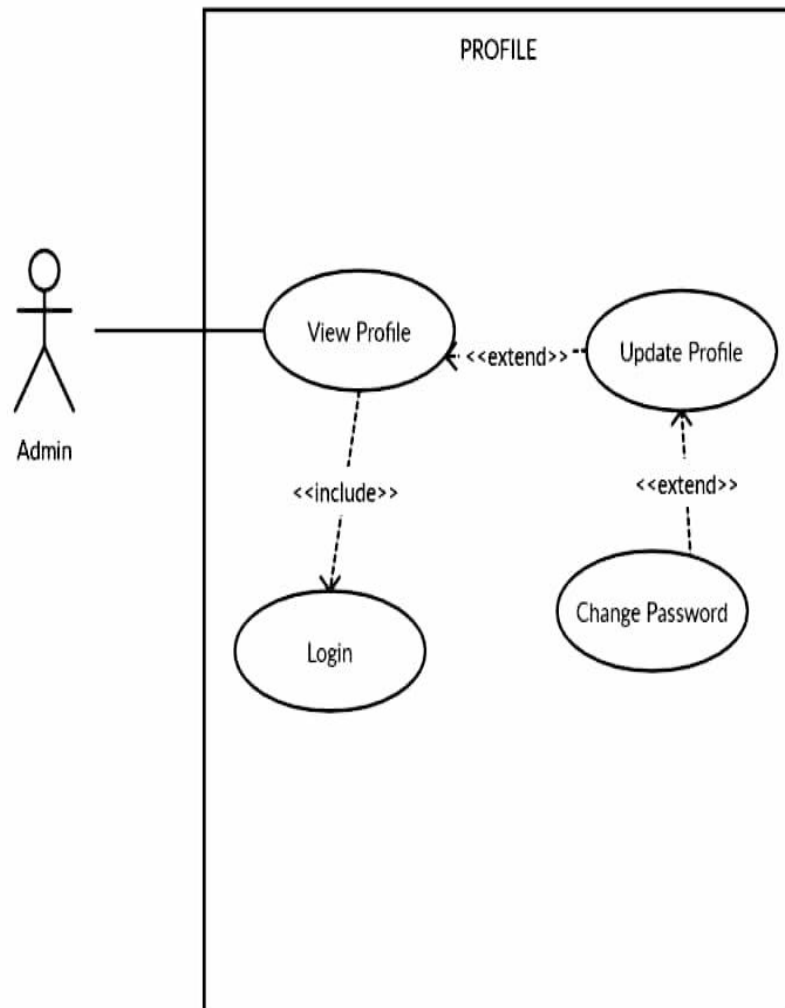


Figure 9: Use case-04: View Profile

### 2.7.5 Use case-05: View Anomaly Detected Clip

The use case to view the anomaly detected clip is represented in Figure 10.

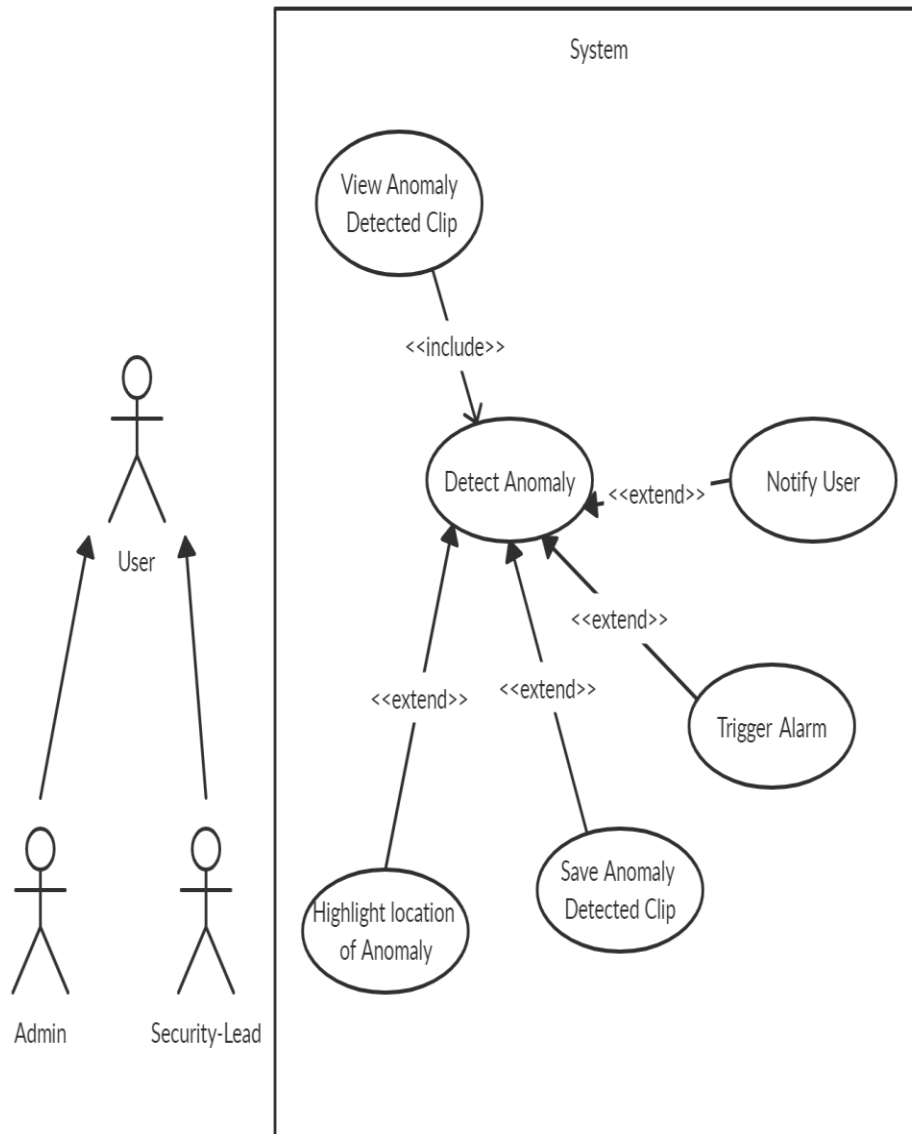


Figure 10: Use case-05: View Anomaly Detected Clip

### 2.7.6 Use case-06: View User

The options for viewing the user and its details are shown in Figure 11.

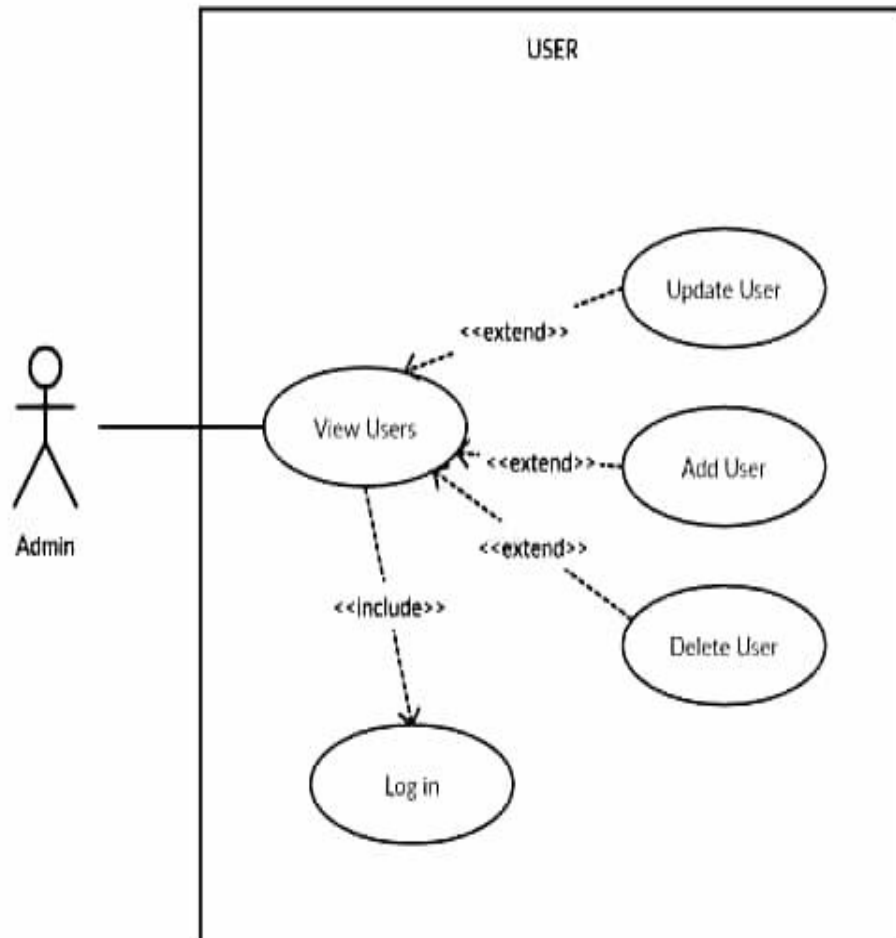


Figure 11: Use case-06: View User

### 2.7.7 Use case: SURVEILIA (Complete System)

The use case in Figure 12 depicts the overview of the entire system by combining all modules in one diagram for a better understanding of the reader.

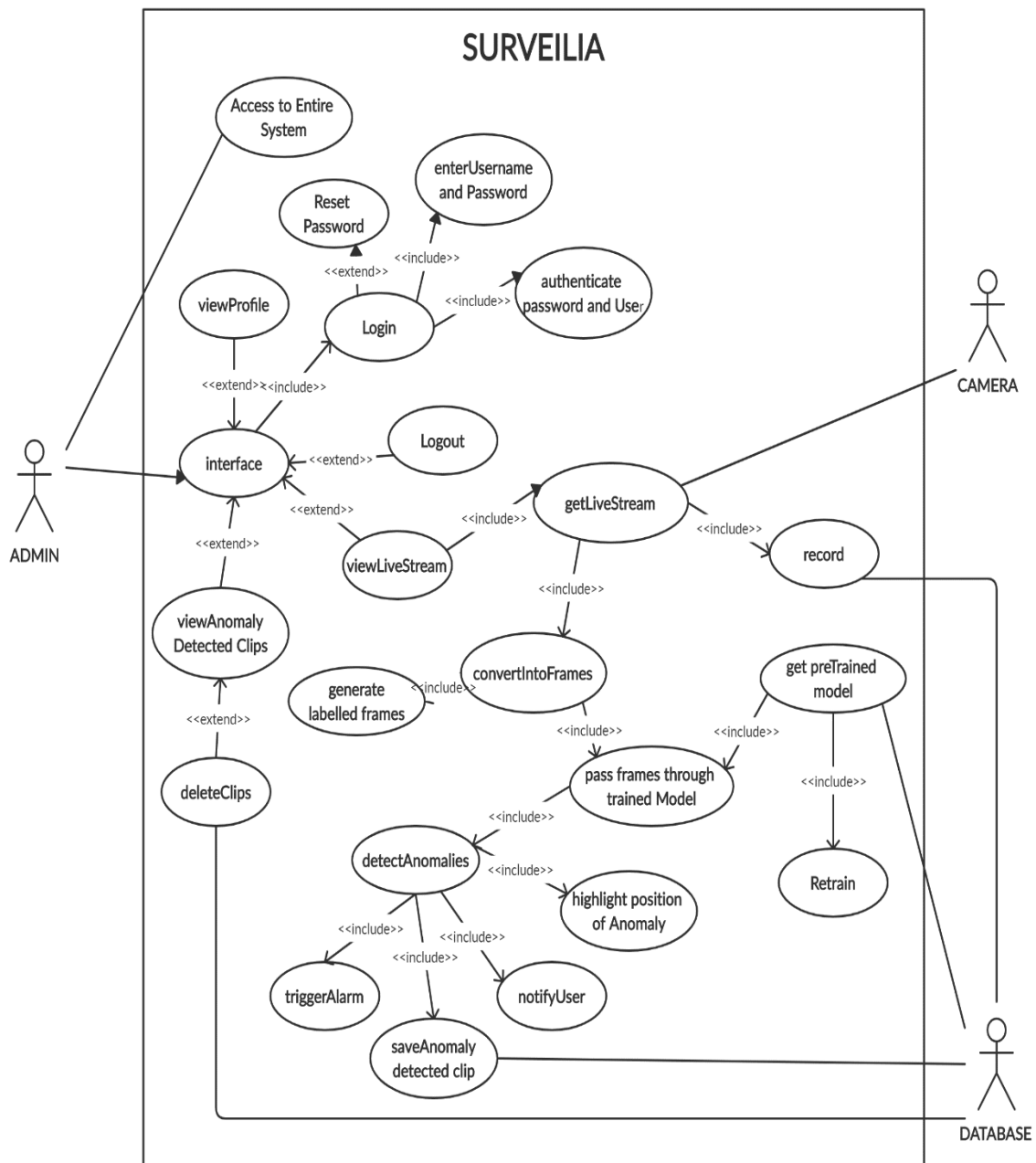


Figure 12: Use case: Surveilila

## 2.8 Software development life cycle model

The software development life cycle is a conceptual outline that describes all the events in the development of a project from planning and creating to maintenance and testing of the system. The basic stages are initiation, planning, designing, building, coding, testing, and deployment. Some of the most important models of this cycle:

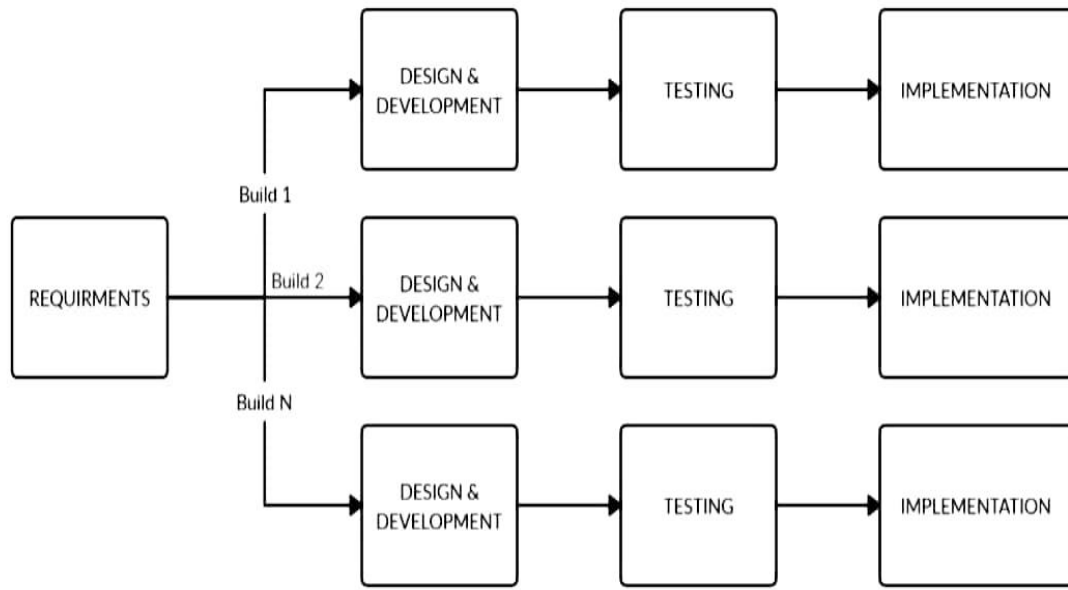
- Incremental model
- Waterfall model
- Agile model
- Spiral model
- Iterative model
- RAD

### 2.8.1 Model Used in our project:

We are choosing an **Incremental Model** for the development process of our system. The incremental model is one of the most important models of SDLC. In this process, requirements are broken down into multiple separate modules of the cycle. And it is iterative.

### 2.8.2 Why?

- This model is feasible for us if there is any alternation required throughout the project. So, it would be more reasonable.
- It is easier to identify the risks and handle them separately in the iterations.
- Our project required revisions until we receive our final project.
- Testing and debugging during the smaller iterations is a better option. Each iteration is an easily managed milestone.



*Figure 13: Incremental Model*

Figure 13 shows how incremental works in different builds. Each build is a little more progressed at the beginning which makes the work increment in small patches and provides flexibility to the model for risk analysis.

## **Chapter 3: System Design**



## 3 System Design

### 3.1 Work breakdown structure (WBS)

The overall work breakdown structure is shown in Figure 14. It describes different phases such as project management, design documents, and so on until the testing and deployment phase.

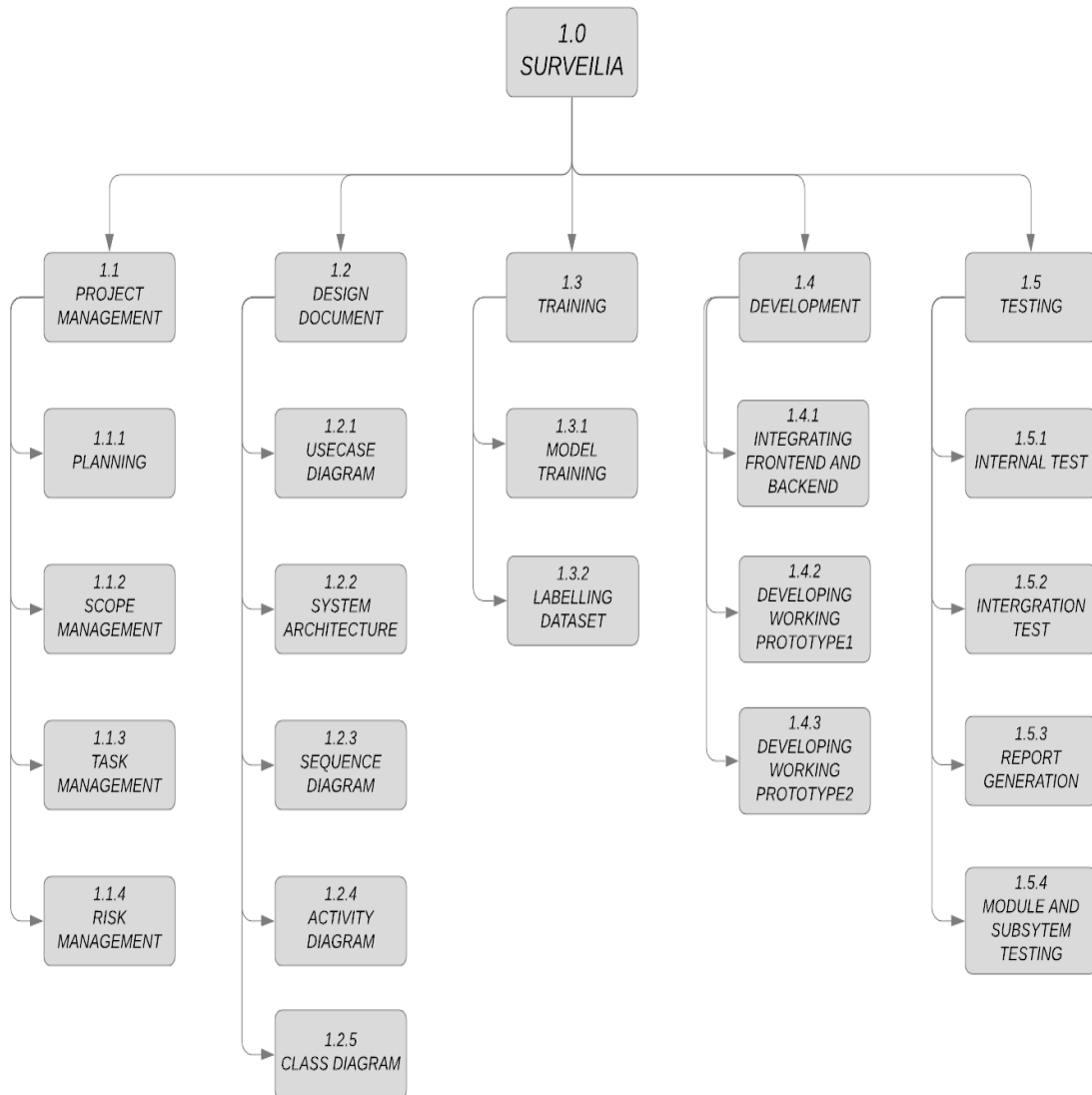
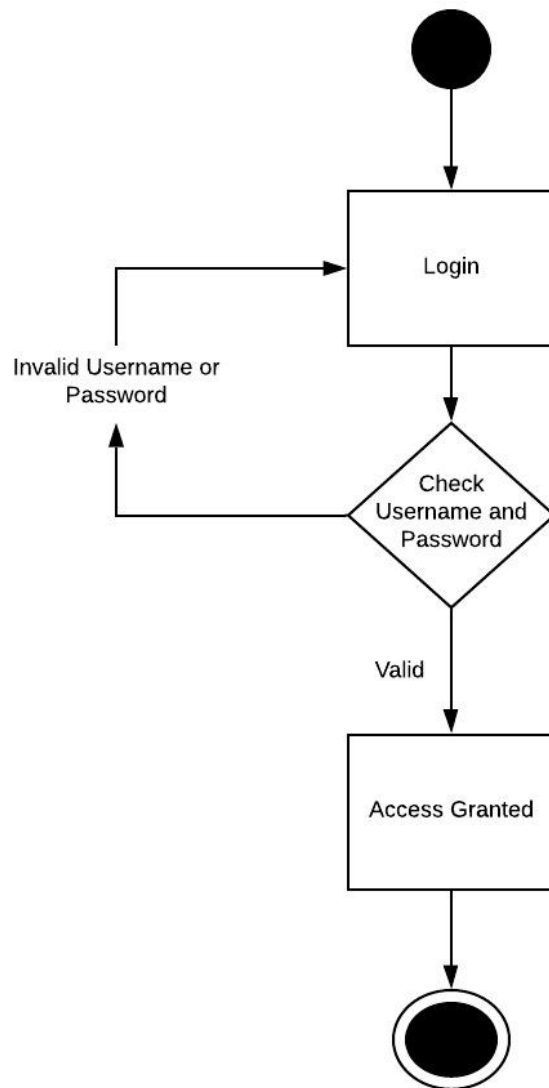


Figure 14: Work Breakdown Structure

## 3.2 Activity diagram

### 3.2.1 Login

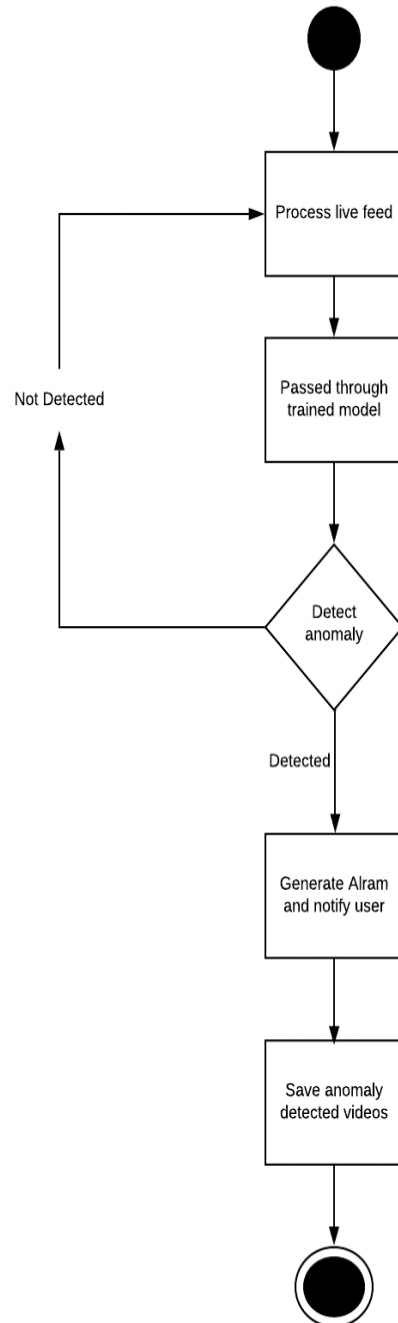
The flow of activities of login is shown in Figure 15. The user enters the credentials which are authenticated with SQLite3 Database. If correct credentials are provided the user is logged in to the main page of the application. If the login credentials provided are not correct, then the message is generated: “invalid username or password”.



*Figure 15: Activity Diagram-01: Login*

### 3.2.2 Anomaly Detection

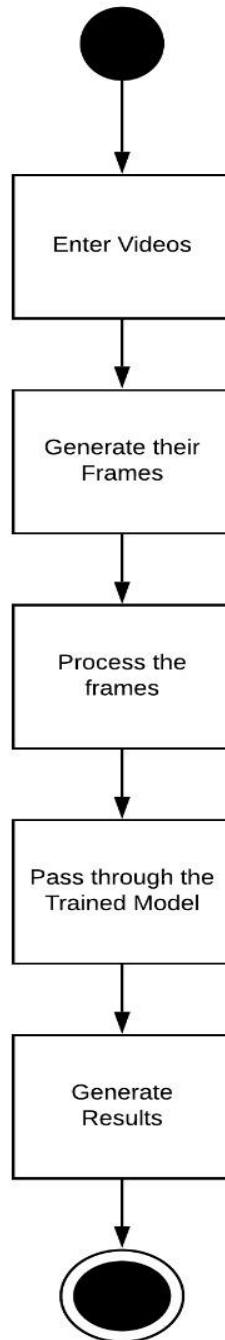
The activity diagram of the flow of activities anomaly detection is shown in Figure 16.



*Figure 16: Activity Diagram-02: Anomaly Detection*

### 3.2.3 Passing through the model

The activity diagram of the flow of activities of passing videos through the model is shown in **Error! Reference source not found.**Figure 17.



*Figure 17: Activity Diagram-03: Passing through the model*

### 3.3 Sequence diagram

#### 3.3.1 Sequence Diagram: Login

Figure 18 shows the sequence diagram for the login. It displays the sequence of modules and their actions for login. The user enters the credentials which are authenticated with SQLite3 Database. If correct credentials are provided the user is logged in to the main page of the application. If the login credentials provided are not correct then the message is generated, “invalid username or password”.

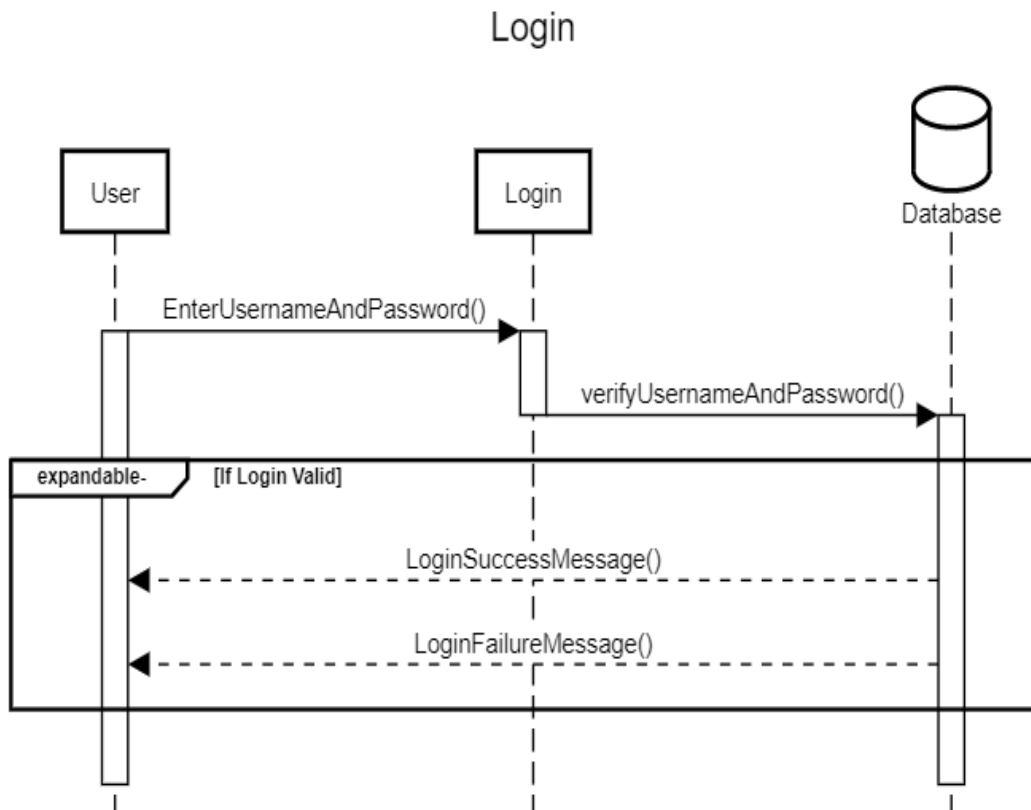


Figure 18: Sequence Diagram: Login

### 3.3.2 Sequence Diagram: Create a New User

Figure 19 shows a sequence diagram to create a new user. It displays the sequence of modules and their actions to create a new user.

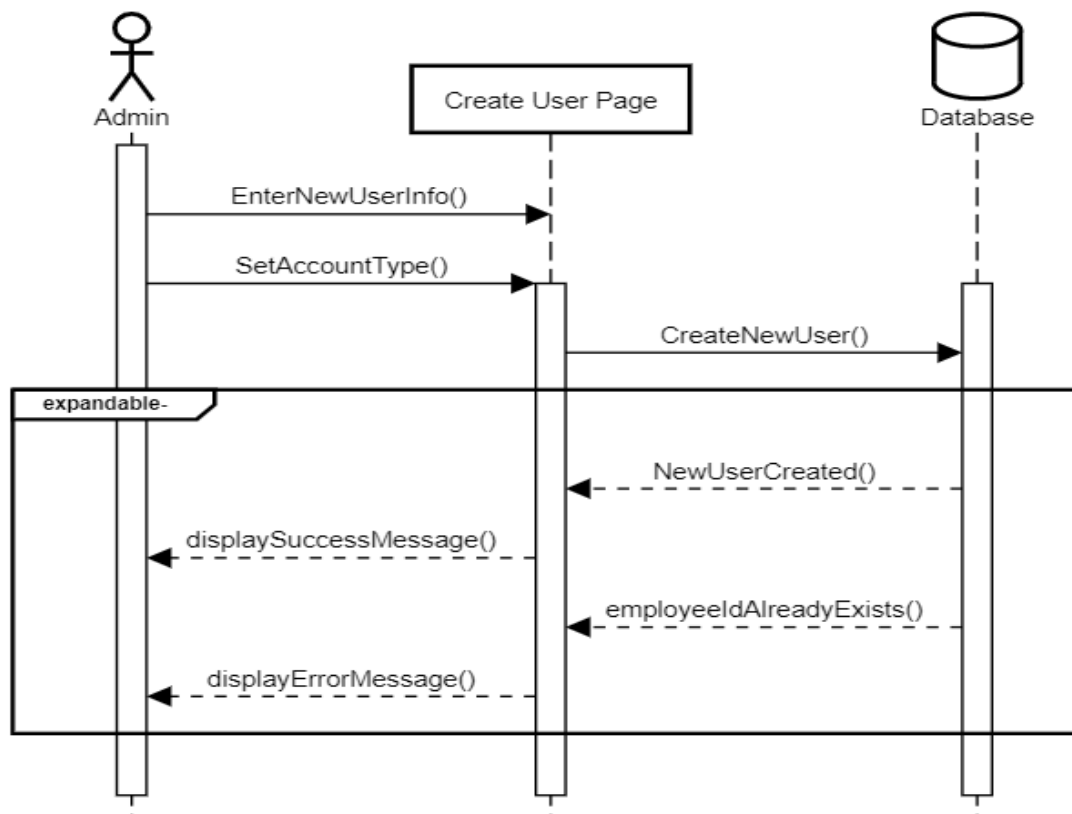
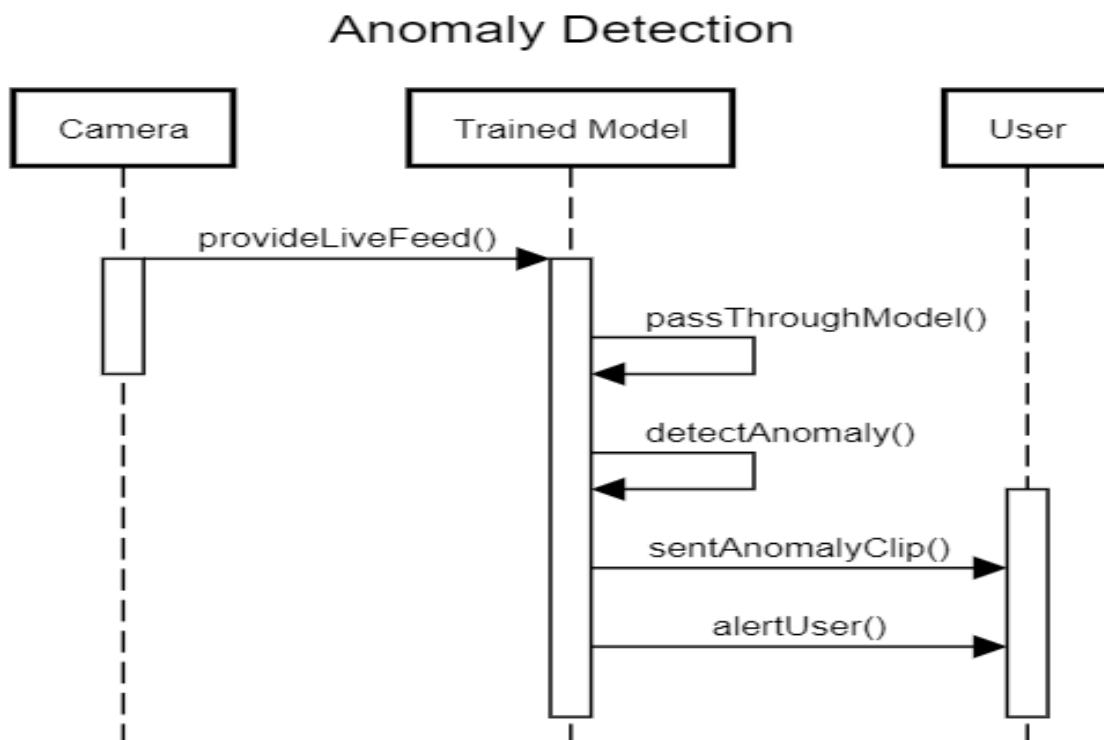


Figure 19: Sequence Diagram: To Create New User

### 3.3.3 Sequence Diagram: Anomaly Detection

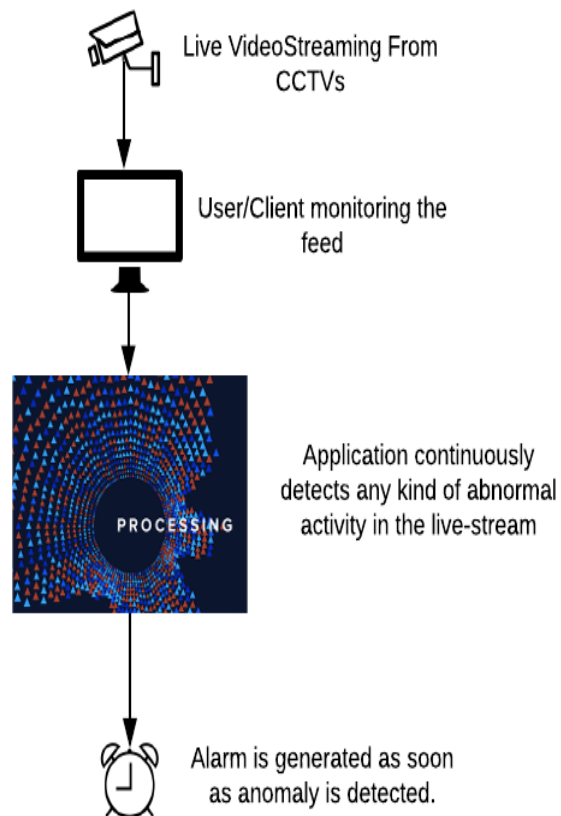
Figure 20 shows a sequence diagram for anomaly detection. It displays the sequence of modules and their actions for anomaly detection. The live feed is fed to the inference engine of the trained neural network. It gets the probability of event occurrence and displays if it is a normal or abnormal event.



*Figure 20: Sequence Diagram: Anomaly Detection*

### 3.4 Software architecture

The flow of Surveilias is live Camera feed or video footage is fed to the system or NVIDIA Jetson Nano. It applies data processing and feeds frames through an inference engine. Then the frames are fed to a less computation-intensive model i.e., trained temporal shift module [2] based inference engine to obtain probability output/results. Anomaly is detected based on probability and it extracts the key-event anomaly snippets, and the user is alerted as shown in Figure 21. We used models of python like PyTorch, NumPy, etc.



*Figure 21: System Architecture*



### 3.5 Network diagram (Gantt chart)

Table 20 is the Gantt chart, it graphically represents which tasks would be done in which duration, what tasks were done in parallel, and which tasks were done in series.

*Table 20: Gantt Chart*

ACTIVITIES	Feb 2020	Mar 2020	April -May 2020	June -July 2020	Aug 2020	Sept 2020	Oct 2020	Nov 2020	Dec 2020
Proposal submission									
Project Planning									
Training the model with the dataset									
Prototype1									
Prototype 2									
Finalizing the prototype									
Creating the application									
Testing & bug fixing									
Beta release									

### 3.6 Collaboration diagram

Figure 22 is the collaboration diagram for Surveilila. It shows flow from the user to the backend processes of the application.

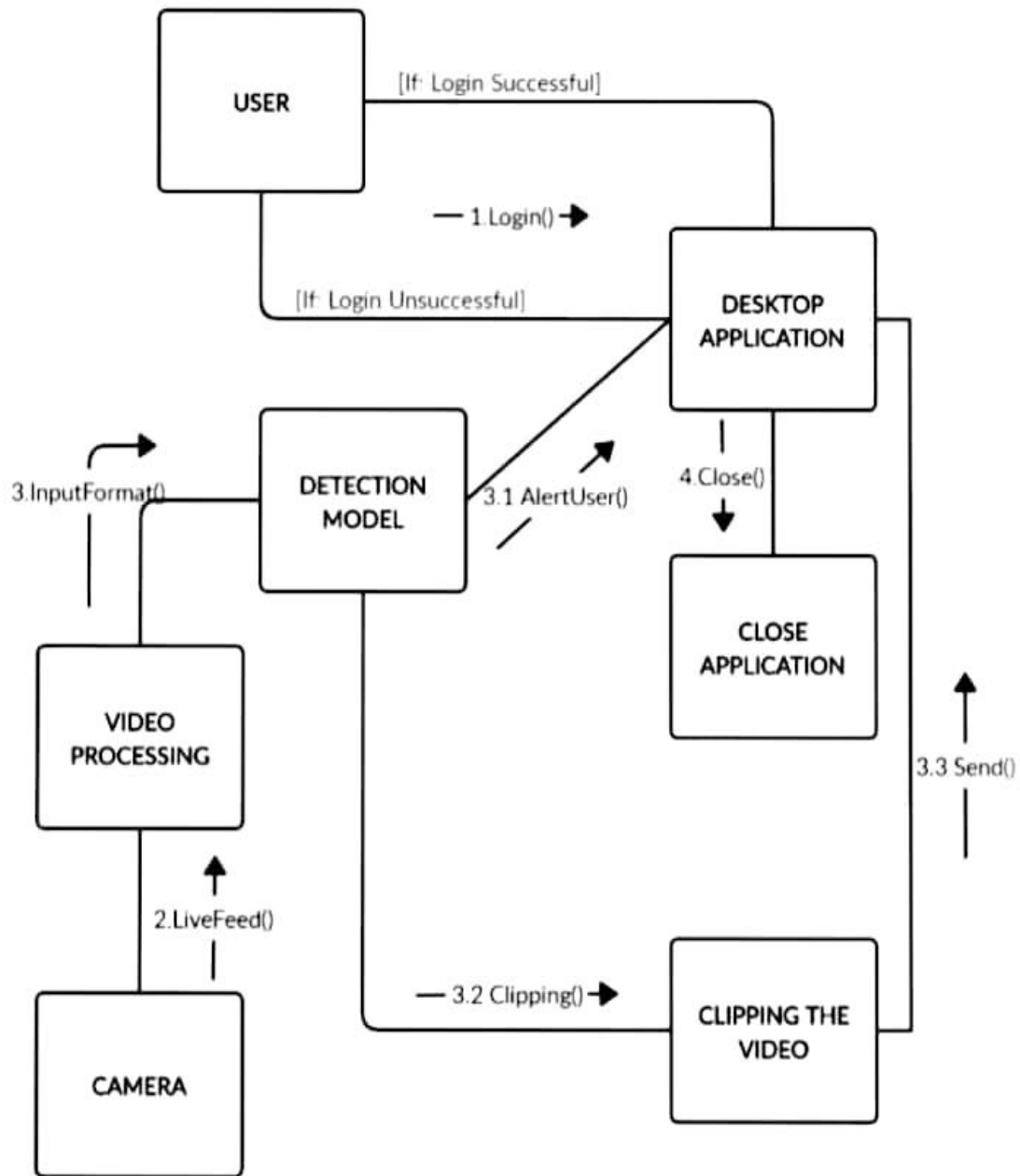


Figure 22: Collaboration Diagram

## **Chapter 4: System Testing**

## 4 System Testing

### 4.1 Test Cases

#### 4.1.1 Test Case-01

This test case Table 21 is for the Functional Requirement FR-01: Login. It explains what users can do and what can go wrong if not tested. This test case aims to log in with the correct login credentials.

*Table 21: Test Case-01*

<b>Test case Name</b>	TC-01
<b>Application Name</b>	SURVEILIA
<b>Use case</b>	Login
<b>Input Summary</b>	The user provides valid login credentials and clicks the login button.
<b>Output Summary</b>	SUCCESS: The user has been logged in.  FAILURE: The access to the user is denied and an error message stating "invalid username or password" is displayed.
<b>Pre-Conditions</b>	Users must enter correct login credentials.
<b>Post-Conditions</b>	The user logs in and the main screen appears.

#### 4.1.2 Test Case-02

This test case Table 22 is for the Functional Requirement FR-03: Anomaly detection.

*Table 22: Test Case-02*

<b>Test case Name</b>	TC-02
<b>Application Name</b>	SURVEILIA
<b>Use Case</b>	Anomaly Detection
<b>Input Summary</b>	The live feed coming from the camera.
<b>Output Summary</b>	SUCCESS:  The anomaly is detected, the user is notified, and the anomaly image(s) and video clip are generated.
<b>Pre-Conditions</b>	Live Feed coming from the camera.
<b>Post-Conditions</b>	The video clip is sent from the database to the user interface.

#### 4.1.3 Test Case-03

This test case Table 23 is for the Functional Requirement FR-04: Video Storage

*Table 23: Test Case-03*

<b>Test case Name</b>	TC-03
<b>Application Name</b>	SURVEILIA
<b>Input Summary</b>	The live feed coming from the camera.
<b>Output Summary</b>	SUCCESS:  The live feed and all the anomaly detected clip(s) are stored on disk.
<b>Pre-Conditions</b>	Live Feed from the camera.

<b>Post-Conditions</b>	Videos are stored on disk.
------------------------	----------------------------

#### 4.1.4 Test Case-04

This test case Table 24 is a test case for adding a camera to the application.

*Table 24: Test Case-04*

<b>Test case Name</b>	TC-04
<b>Application Name</b>	SURVEILIA
<b>Input Summary</b>	The IP address of the camera to be attached is provided to the application.
<b>Output Summary</b>	SUCCESS:  The camera is added, and its feed starts to display on the display screens of the application. Also, the inferencing is activated.
<b>Pre-Conditions</b>	The application is running, and the camera is on.
<b>Post-Conditions</b>	The camera is viewed and detected in the application.

## 4.2 Unit Testing

In unit testing, all the components of the application are individually tested. It is the first and most fundamental part of testing. In this step, we checked the individual elements of our project that execute unit tasks and are components of the project's whole workflow.

### **4.3 Integration testing**

Integration testing is the second step of the software testing procedure. At this level of testing, the system is tested after combining the distinct units into clusters. It is to test the faults and errors in the interaction between the combined units.

The following functionalities were checked:

- ✓ Adding an IP camera to the application.
- ✓ Adding a Video Clip from the system to detect if there is any anomaly or not.

The application was tested and checked. Our application passed the integration testing phase.

### **4.4 Acceptance testing**

Acceptance testing is defined as the final step for the software testing procedure. It is to govern whether the required specifications of the system are met. At this step, we estimate whether the system under test is complete with the basics and necessities for final processing.

We tested our application against the requirements stated in the proposal.

- ✓ Authorization of user to login.
- ✓ Real-time detection of an anomaly.
- ✓ Extraction of image and video clips (of the whole abnormal event) where the anomaly has occurred.
- ✓ Storing the image and video clips on-disk.

Our application fulfills the listed requirements.

## **Chapter 5: Application**



## 5 Application

The application consists of three core modules: the inference engine, python UI, and NVIDIA Jetson nano.

### 5.1 Inference Engine

#### 5.1.1 Dataset

The UCF Crime dataset [1], a video dataset is used to train and test the Deep Neural Network. In our scenario, the dataset consists of about 1120 videos with a total runtime of 70 hours. It consists of 14 categories, including 13 types of anomalous incidents, i.e. (accident, stealing, robbery, shoplifting, arson, explosion, shooting, arrest, vandalism, burglary, fighting, abuse, and assault) and one for the normal events. For training, testing, and validation, the dataset is broken down into an 80:10:10 ratio.

#### 5.1.2 Architecture

We trained the deep neural network on Colab using Tesla T4/K80 GPU. It was trained on two different architectures using TSM i.e., Resnet50 and MobileNetV2. The training strategy included disk-resident frame extraction using python-based directory/file parser and OpenCV for automated labeling of the dataset (normal vs abnormal).

#### 5.1.3 Comparison of the results achieved using ResNet50 and MobileNetV2 Architecture

<b>Temporal Shift Module (TSM) + MobileNetV2 Architecture</b>	<b>Temporal Shift Module (TSM) + ResNetV2 Architecture</b>
Validation accuracy of 83%.	Validation accuracy of 87%.
2,226,434 params	23,515,130 params
23.5M having computer complexity of 4.12GMAC or 8.24 GFLOPs	2.2M having computed complexity of 0.32GMAC or 0.64GFLOPs.

#### 5.1.4 Accuracy

We achieved a validation accuracy of 83% using MobileNetV2 as shown in Figure 23 and a validation accuracy of 87% as shown in Figure 24.

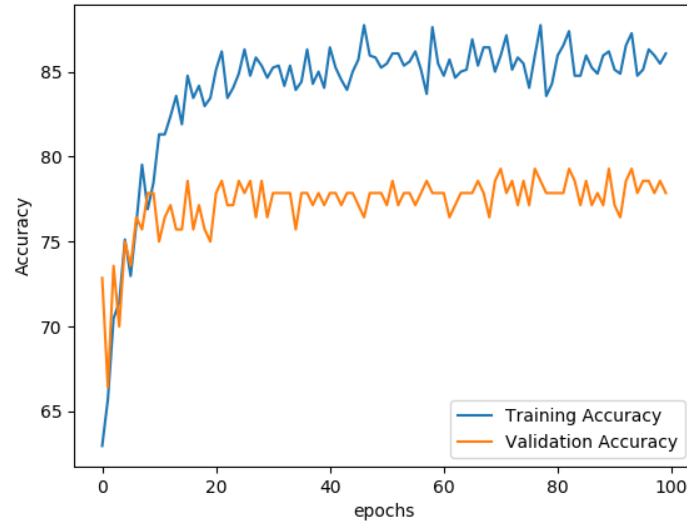


Figure 23: Accuracy Plot for MobileNetV2

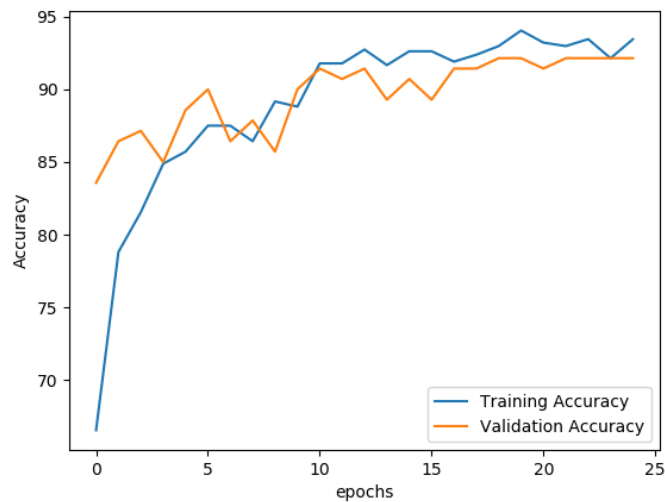


Figure 24: Accuracy Plot for ResNet50

### 5.1.5 Confusion Matrix

As we can see in Figure 25 out of 77 Normal Videos, our model predicted 65 of them correctly and 9 were wrongly classified. And out of 64 total Abnormal Videos, our model predicted 55 of them correctly and 11 were wrongly classified. From these statistics, we came to know that our model has a test accuracy of 85%.

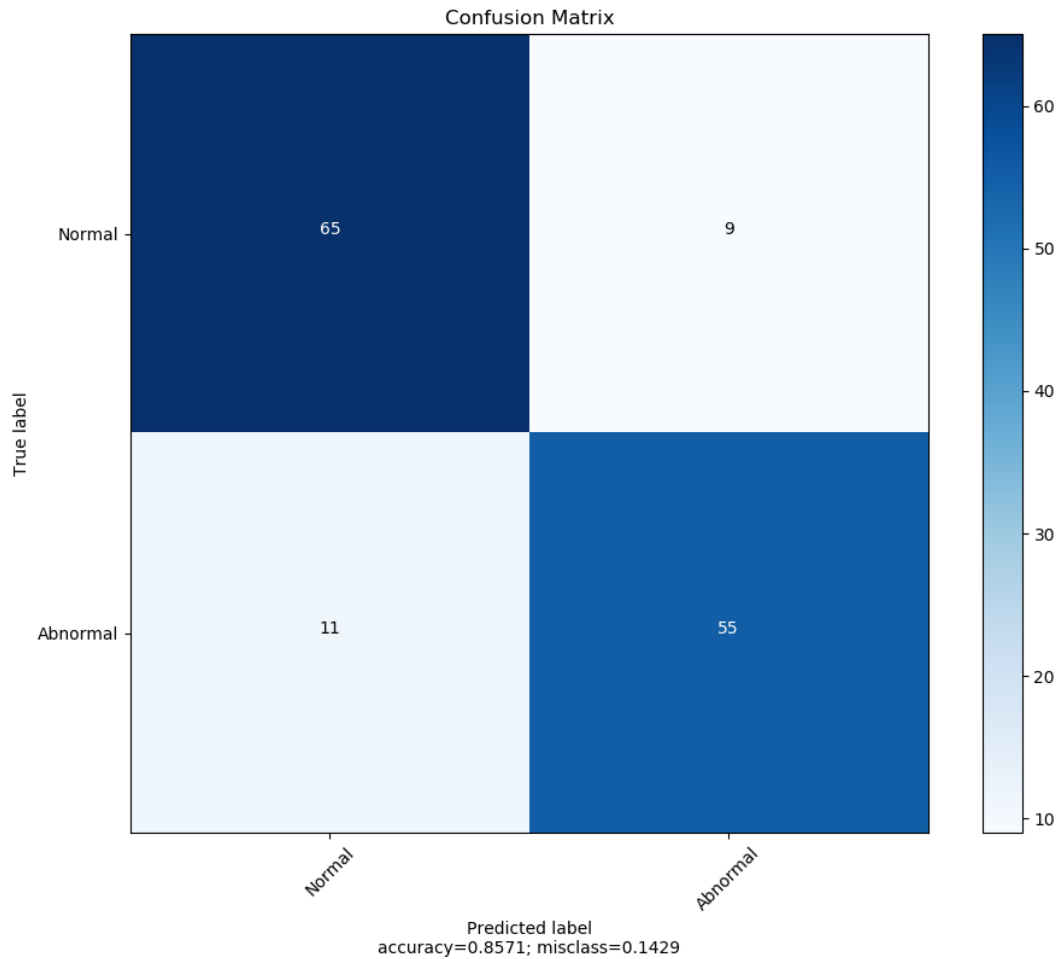


Figure 25: Confusion Matrix for MobileNetV2

As we can see in Figure 26 out of 77 Normal Videos, our model predicted 60 of them correctly and 6 were wrongly classified. And out of 64 total Abnormal Videos, our model predicted 64 of them correctly and 10 were wrongly classified. From these statistics, we came to know that our model has a test accuracy of 85%.

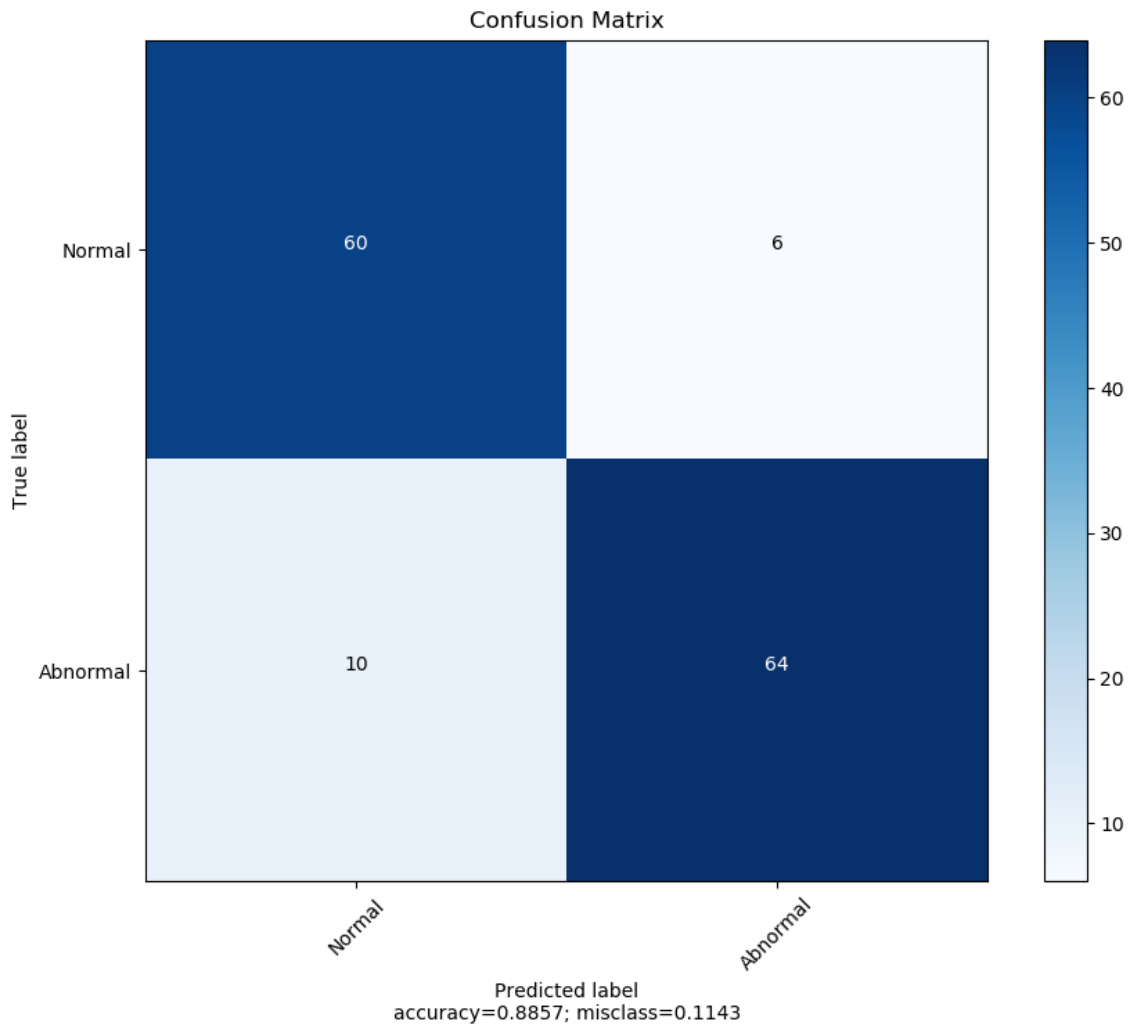


Figure 26: Confusion Matrix for ResNet50

### 5.1.6 Classification Report

A classification report tells us about the quality of our proposed model, it also tells class-wise information about our results, so that an appropriate decision can be made. In our case, the abnormal class had a precision of 81% and the normal also had a precision of 81%. This is represented in Figure 27 and Figure 28.

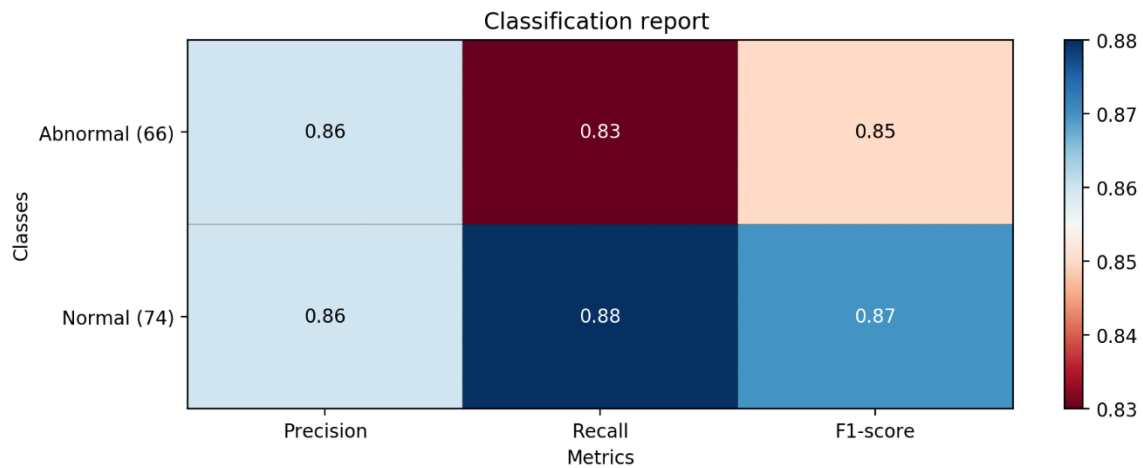


Figure 27: Classification Report for MobileNetV2

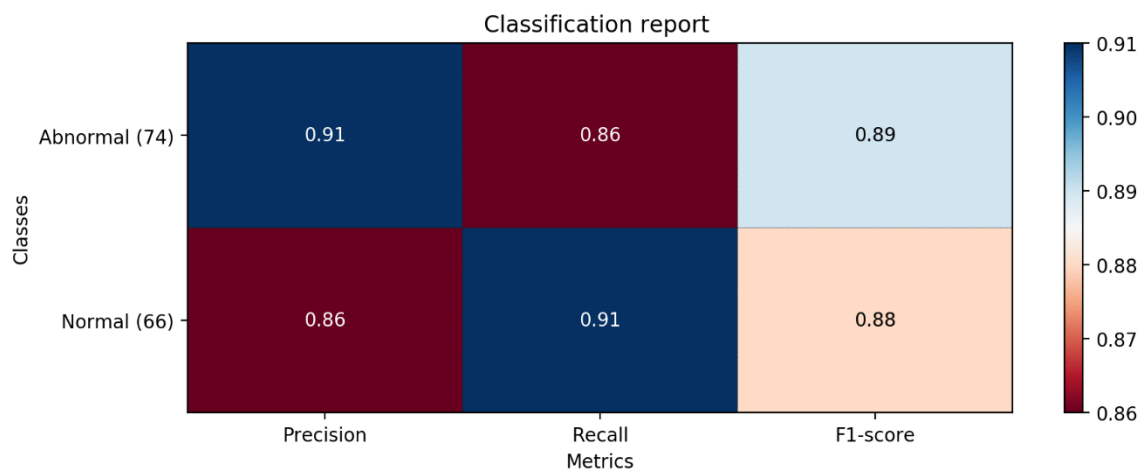


Figure 28: Classification Report for ResNet50

### 5.1.7 Loss

The plot for loss is shown in Figure 29 and Figure 30 for MobileNetV2 and ResNet50, respectively.

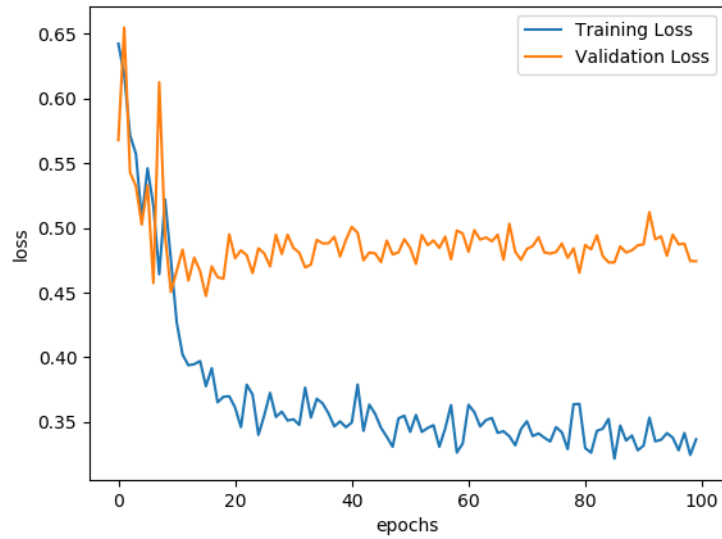


Figure 29: Loss Plot for MobileNetV2

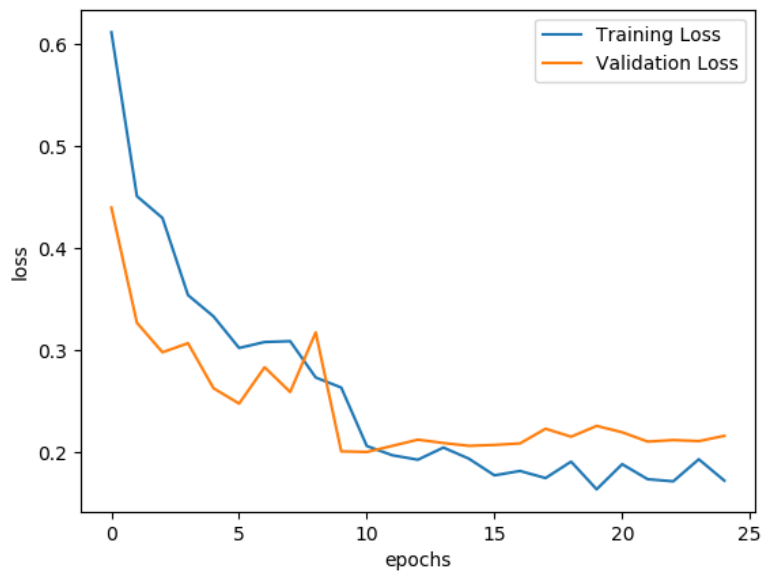


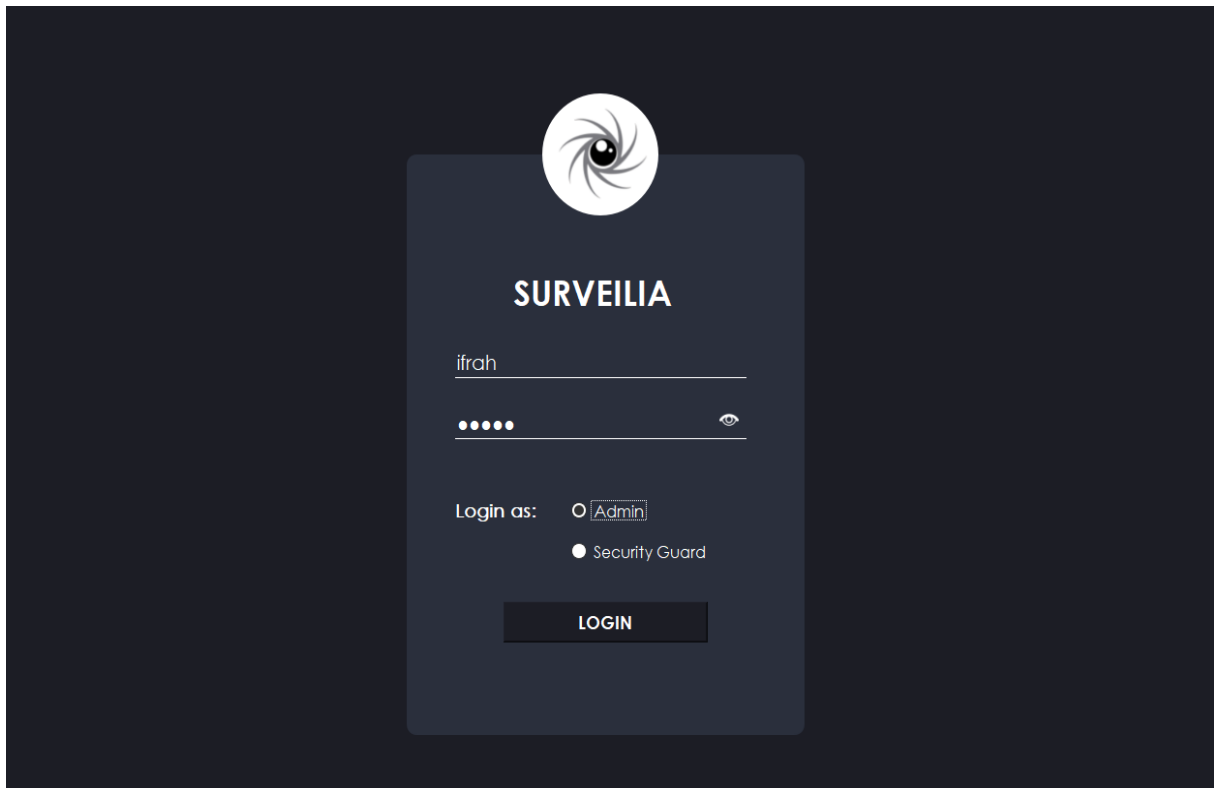
Figure 30: Loss Plot for ResNet50

## 5.2 Application Frontend

The screenshots of the Surveilias desktop application interface are shown below.

### 5.2.1 Login Screen

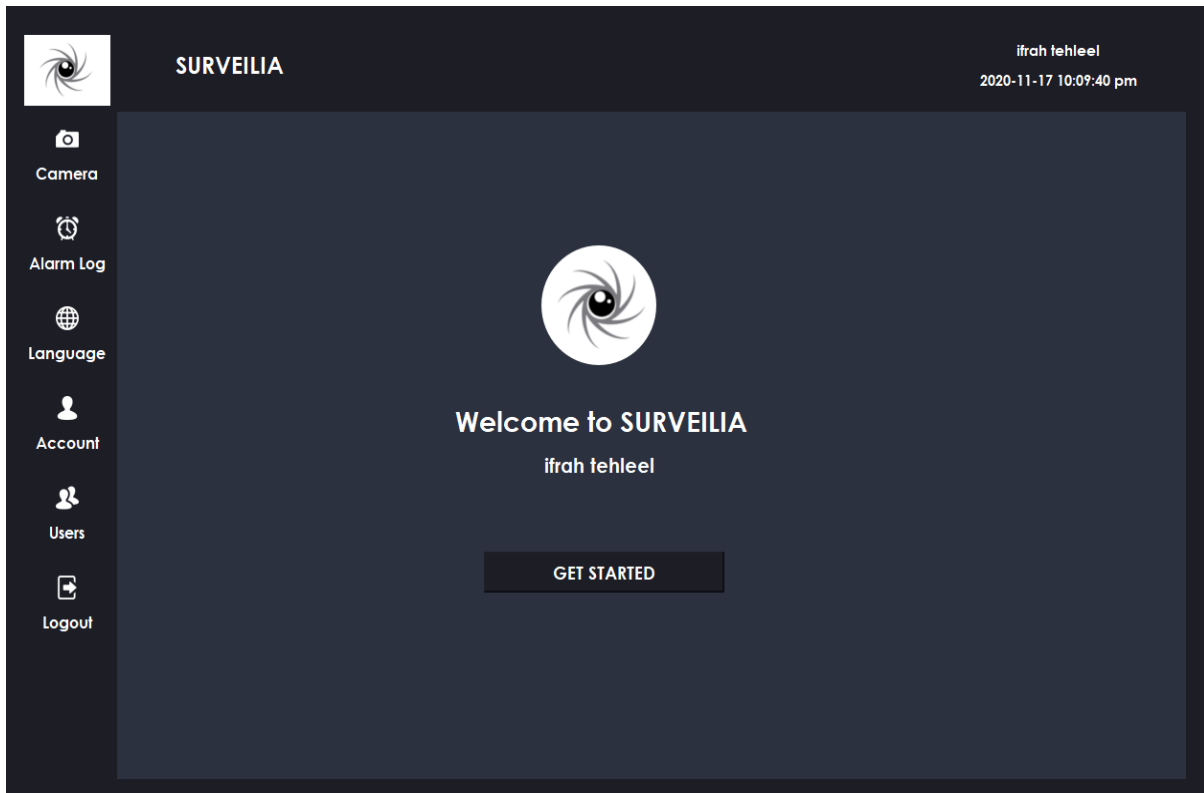
Figure 31 depicts the login screen of Desktop Application which is developed in python using PyQt5. The user will enter username, password, and user type to log in, successfully.



*Figure 31: Login Screen*

### 5.2.2 Main/Home Screen

Figure 32 depicts the main screen of Surveilias. On the left side, we have a menu bar that consists of multiple options. Clicking the ‘Get started’ button directs to the camera screen. On the top right, the logged-in username is displayed along with the current time and date.

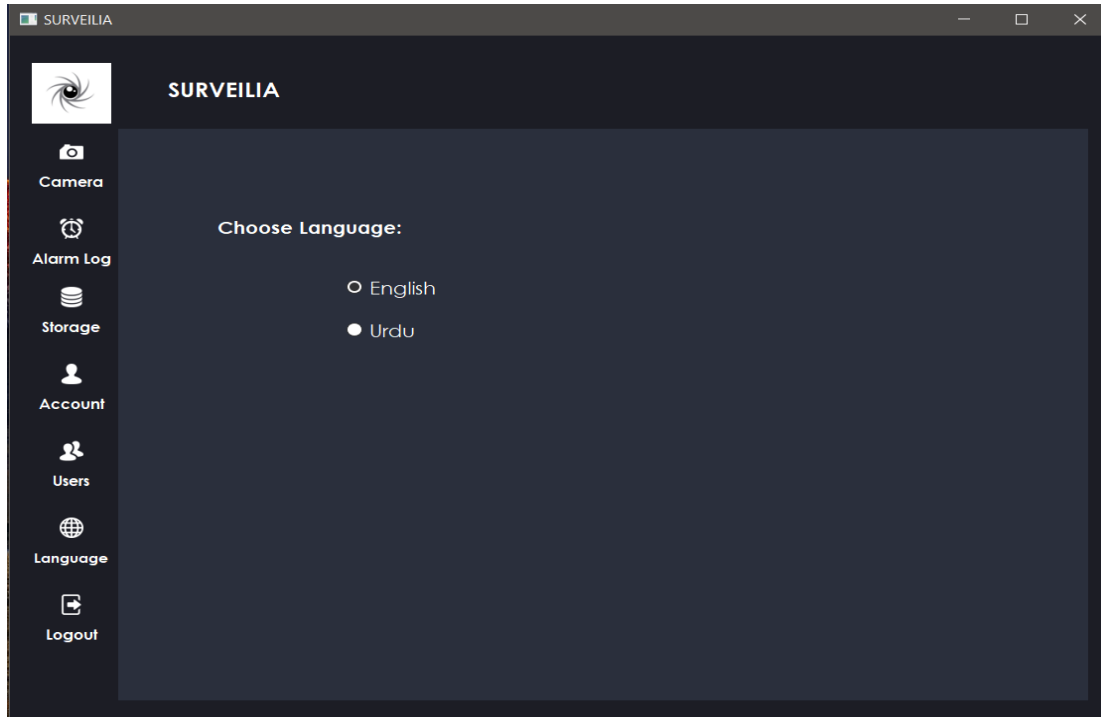


*Figure 32: Main/Home Screen*

### 5.2.3 Choose Between Language Screens

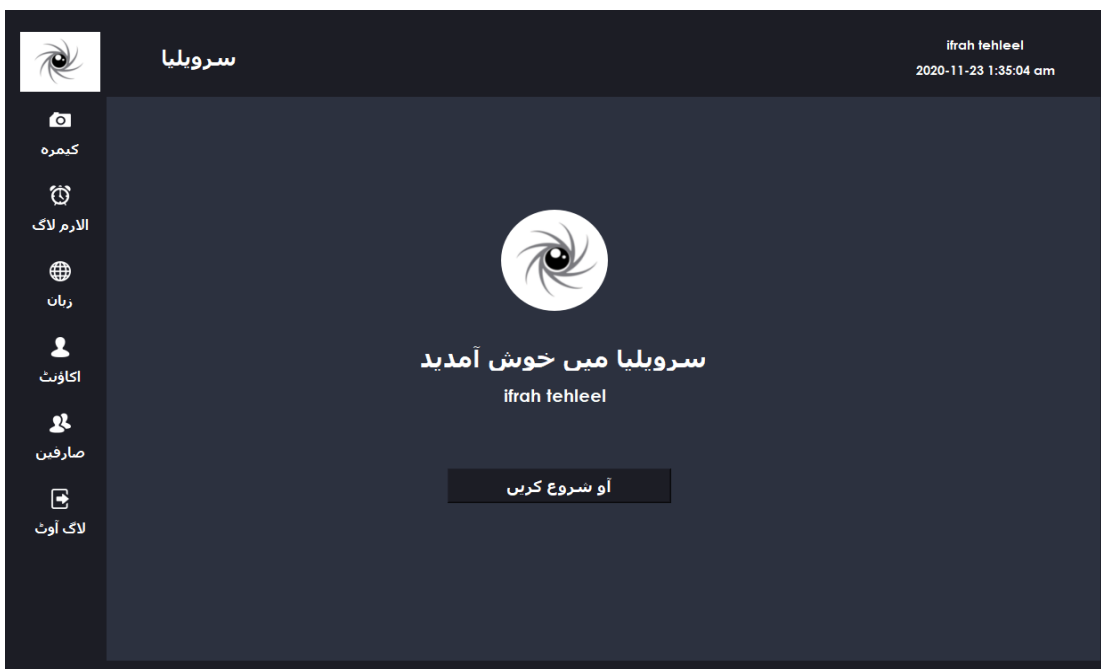
Our application supports multi-languages i.e., English and Urdu. We have added the option of Urdu language so the user who is a security guard can easily understand and use the application. Figure 33 shows the screen for choosing between languages.





*Figure 33: Language Screen*

And Figure 34 depicts the translated GUI.



*Figure 34: Screen with the Urdu Language*

## 5.2.4 Camera Screen

Figure 35 shows 6 cameras added to the application and red-green text displays if there is any normal or abnormal activity. On the right side 'add camera' button is placed which gives the option to add a camera to the displays, under its cameras list is present which lets the user edit or close the camera being displayed.

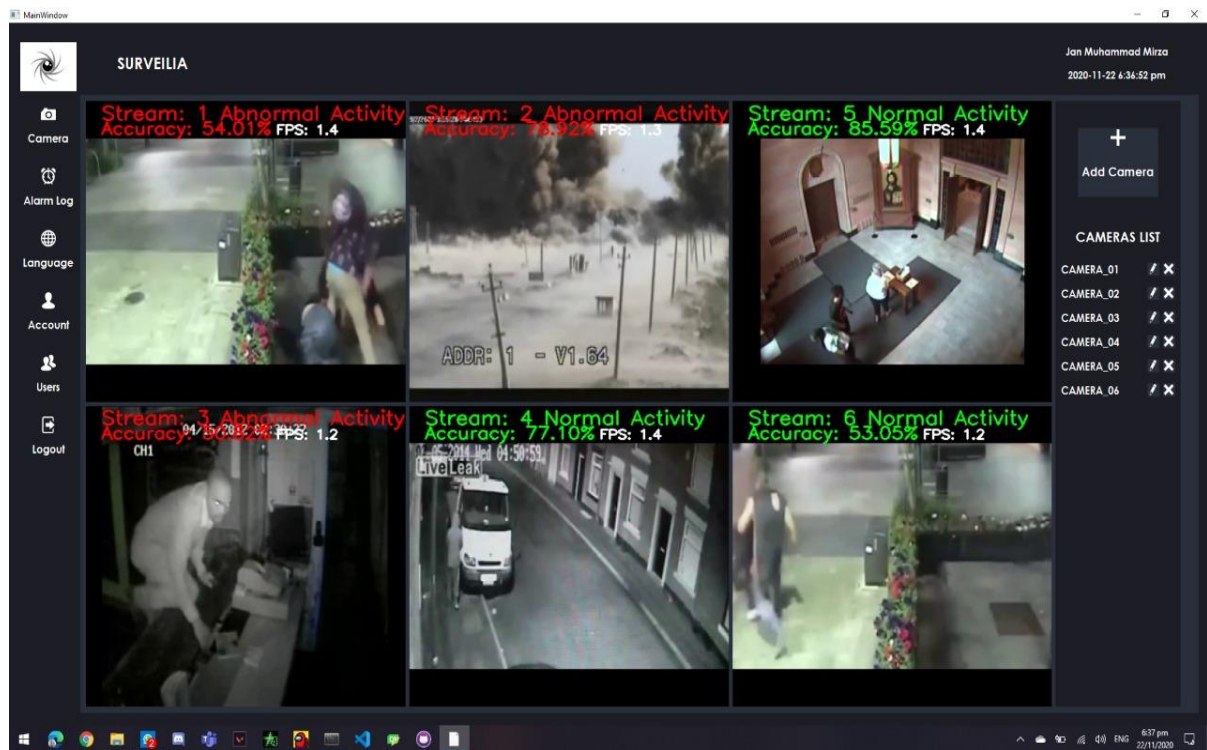
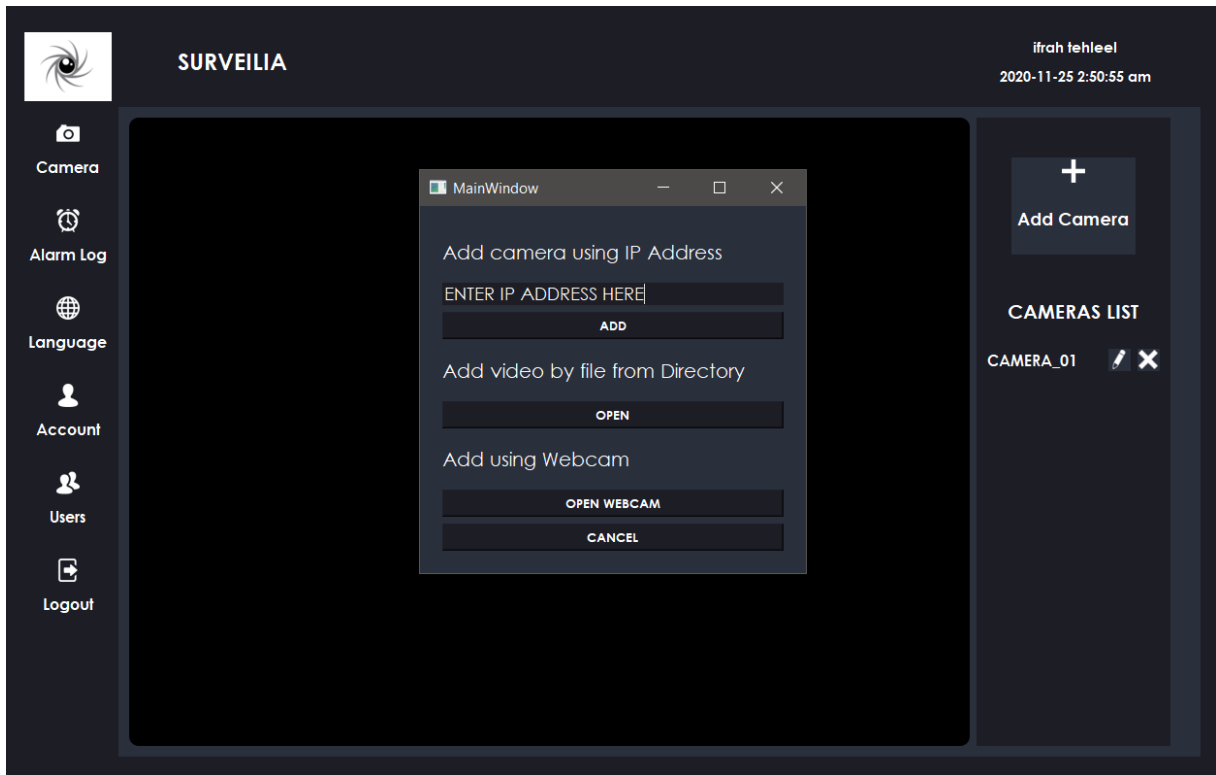


Figure 35: Camera Screen

### 5.2.5 Camera Options Screen

Figure 36 represents the camera options dialog box. It consists of three options i.e., to add the camera using IP address, add video from the computer, or to open the webcam.



*Figure 36: Camera Options Screen*

## 5.2.6 Anomaly History Screen

Figure 37 shows the anomaly history screen, where the user can view all the abnormal events detected in a tabular form. And the user can play the video by clicking the URL. It consists of camera ID (where the anomaly has been detected), date, time, and video URL.

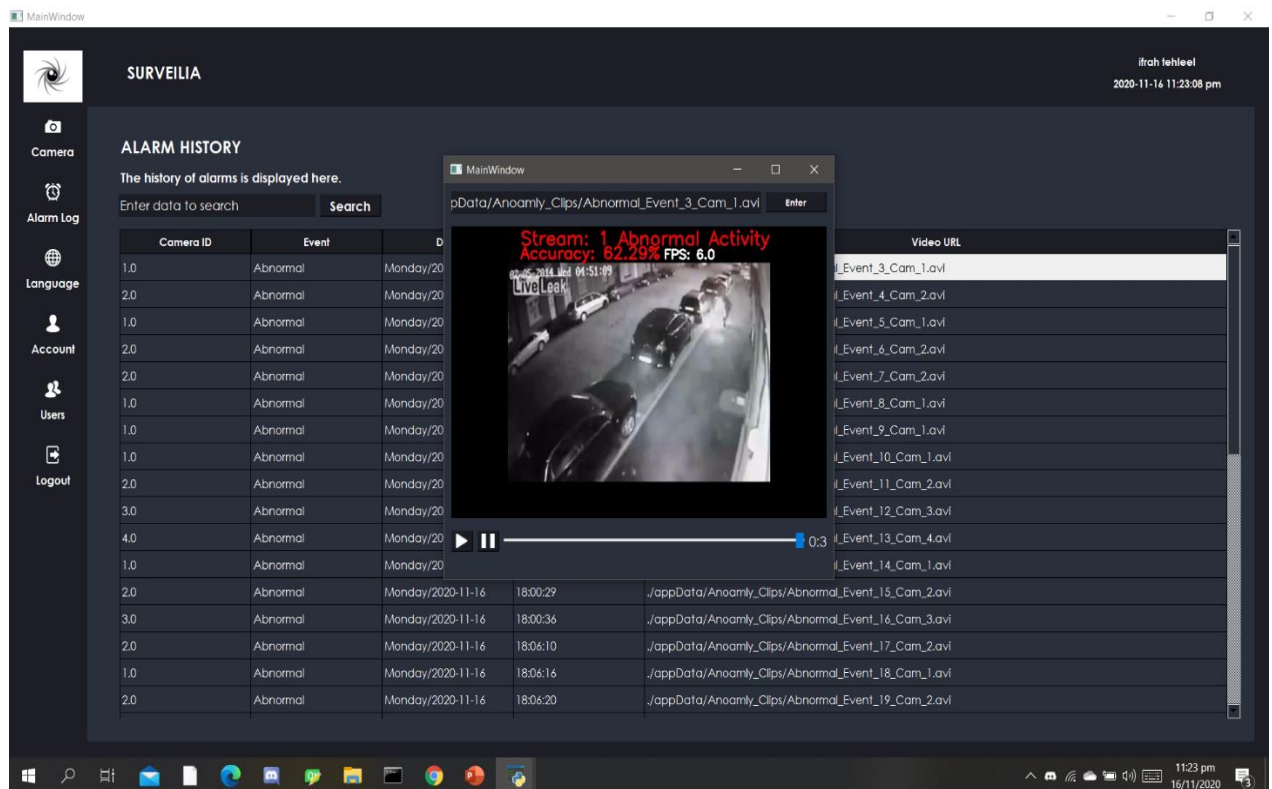



Figure 37: Anomaly History Screen

### 5.3 Users Information Screen

Figure 38 shows the admins and users' lists and details. It is only accessed by the admin. The details of users are displayed in tabular form which consists of an ID, first name, last name, username, password, contact number, and address.



**SURVEILIA**

ifrah tehleel  
2020-12-23 2:21:02 am

**USERS INFORMATION**

View: ☒ Admins ☐ Security Guards

**ADD** **DELETE**

	Admin ID	First Name	Last Name	Username	Password	Contact No.	Address
1	1	ifrah	tehleel	ifrah	ifrah	923014474797	57b
2	2	Jan Muhammad	Mirza	jm	jm	9231023123	lahore
3	3	Nauman	Akram	nauman	nauman	9231023123	kasur

*Figure 38: Users Information Screen*

## 5.4 Account Information Screen

Figure 39 shows the details of the logged-in user.

**SURVEILIA**

ifrah tehleel  
2020-12-23 2:21:02 am

**ACCOUNT INFORMATION**

First Name:	ifrah
Last Name:	tehleel
Username:	ifrah
Password:	•••••
Contact Info:	923014474797
Address:	57b

*Figure 39: Account Information Screen*

## **Chapter 6: Project Code**

## 6 Project Code

Our project's code is written in Python (version 3.7.6) which can be divided into the following sections:

- TSM\_Model
- Recording Anomalous Events to the folder
- Reading CSV file to display data

And many other sections that are vital for the project.

The code is built with the following libraries:

- PyQt version 5.9.2
- PyQt5 version 5.15.1
- PyTorch version 1.6
- OpenCV version 3.4.2.16
- torchvision version 0.7.0
- NumPy version 1.18.1
- Pandas version 1.0.1
- SQLite version 3.31.1

### 6.1 TSM\_Model:

TSM\_Model consists of the definition of the TSM model that is used for frame extraction, feature extraction, categorizing normal and abnormal activities, and then storing the abnormal clips.

```
def tsmmodel (self, f, check):

    # os.environ[""] = "0"

    emptyString = ""

    print()
    print("=====>>>> Loading model ... Please wait ...")

    # just adding some comments to check git
    def parse_shift_option_from_log_name(log_name):
        if "shift" in log_name:
            strings = log_name.split("_")
            for i, s in enumerate(strings):
                if "shift" in s:
                    break
            return True, int(strings[i].replace("shift", "")),
        strings[i + 1]
    else:
```



```

        return False, None, None

    # args = parser.parse_args()

    this_weights =
    "tsm_model/checkpoint/TSM_ucfcrime_RGB_mobilenetv2_shift8_blockres_avg_se
gment8_e50/ckpt.best.pth.tar"
    # this_weights =
    'checkpoint/TSM_ucfcrime_RGB_resnet50_shift8_blockres_avg_segment8_e25/ck
pt.best.pth.tar'
    is_shift, shift_div, shift_place =
    parse_shift_option_from_log_name(
        this_weights
    )
    modality = "RGB"
    if "RGB" in this_weights:
        modality = "RGB"

    # Get dataset categories.
    categories = ["Normal Activity", "Abnormal Activity"]
    num_class = len(categories)
    # this_arch = 'resnet50'
    this_arch = "mobilenetv2"

    net = TSN(
        num_class,
        1,
        modality,
        base_model=this_arch,
        consensus_type="avg",
        img_feature_dim="225",
        # pretrain=args.pretrain,
        is_shift=is_shift,
        shift_div=shift_div,
        shift_place=shift_place,
        non_local="_nl" in this_weights,
    )

    checkpoint = torch.load(this_weights,
map_location=torch.device("cpu"))
    # checkpoint = torch.load(this_weights)

    checkpoint = checkpoint["state_dict"]

    # base_dict = {'base_model.' + k).replace('base_model.fc',
'new_fc'): v for k, v in list(checkpoint.items())}
    base_dict = {".".join(k.split(".")[1:]): v for k, v in
list(checkpoint.items())}
    replace_dict = {
        "base_model.classifier.weight": "new_fc.weight",
        "base_model.classifier.bias": "new_fc.bias",
    }
    for k, v in replace_dict.items():
        if k in base_dict:
            base_dict[v] = base_dict.pop(k)

    net.load_state_dict(base_dict)
    # net.cuda().eval()
    net.eval()
    transform = torchvision.transforms.Compose(
        [
            Stack(roll=(this_arch in ["BNInception", "InceptionV3"])),
            ToTensorFormatTensor(

```

```

        div=(this_arch not in ["BNInception", "InceptionV3"])
    ),
    GroupNormalize(net.input_mean, net.input_std),
]
)
try:
    os.makedirs("./appData/Anoamly_Clips")
    os.makedirs("./appData/Anoamly_Images")
except OSError as e:
    pass

print("loading Video...")
if f == emptyString:

    cap = cv2.VideoCapture(cv2.CAP_DSHOW)
else:

    cap = cv2.VideoCapture(f)

i_frame = -1
count = 0
print("Ready!")
writer = None
c = 0

while cap.isOpened():
    count += 1
    i_frame += 1
    ret, img = cap.read() # (480, 640, 3) 0 ~ 255

    # release everything when the job is finished

    if ret:
        if (
            i_frame % 3 == 0
        ): # skip every other frame to obtain a suitable frame
rate
            t1 = time.time()

            img_tran =
transform([Image.fromarray(img).convert("RGB")])

            input = img_tran.view(
                -1, 3, img_tran.size(1), img_tran.size(2)
            ).unsqueeze(0)

            with torch.no_grad():
                logits = net(input)

                h_x = torch.mean(F.softmax(logits, 1), dim=0).data

                print(h_x)

                pr, li = h_x.sort(0, True)

                probs = pr.tolist()

                idx = li.tolist()

                t2 = time.time()
                print(count, "-", categories[idx[0]], "Prob: ",
probs[0])

```

```

        current_time = t2 - t1
dim = (420, 420)
img = cv2.resize(img, dim, interpolation=cv2.INTER_AREA)
height, width, _ = img.shape
# height, width, _ = img.shape

# label = np.ones([height // 10, width, 3]).astype("uint8")

+ 255

if categories[idx[0]] == "Abnormal Activity":
    R = 255
    G = 0
    print("\007")
    Abnormality = True
    c += 1
else:
    R = 0
    G = 255
    Abnormality = False

cv2.putText(
    img,
    "Stream: " + str(check) + " " + categories[idx[0]],
    (20, int(height / 16)),
    cv2.FONT_HERSHEY_SIMPLEX,
    0.9,
    (0, int(G), int(R)),
    2,
)

cv2.putText(
    img,
    "Accuracy: {:.2f}%".format(probs[0] * 100, "%"),
    (20, int(height / 9)),
    cv2.FONT_HERSHEY_SIMPLEX,
    0.8,
    (0, int(G), int(R)),
    2,
)

cv2.putText(
    img,
    "FPS: {:.1f} ".format(1 / current_time),
    (250, int(height / 9)),
    cv2.FONT_HERSHEY_SIMPLEX,
    0.6,
    (255, 255, 255),
    2,
)

if writer is None:
    if c % 60 == 0:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        (H, W) = img.shape[:2]
        path = "./appData/Anoamly_Clips/"
        name = len(glob.glob(path + "*.avi"))

        getVidName = path +
"Abnormal_Event_{}_Cam_{}.avi".format(
    name + 1, check
)
        self.getStatsOfAbnormalActivity(check,
getVidName)

        writer = cv2.VideoWriter(getVidName, fourcc, 30.0,
(W, H), True)

```

```

# Saving Anaomalous Event Image and Clip
if Abnormality:
    writer.write(img)
    # record stat every two seconds if exists
    # if c % 60 == 0:
    #     self.getStatsOfAbnormalActivity(check)
    # if tempThres > 0.75:

    path = "./appData/Anoamly_Images/"
    index = len(glob.glob(path + "*.jpg"))
    # imageName = getFileName(path+'.jpg')
    imageName = path +
"Abnormal_Event_{}_Cam_{}.jpg".format(
    index + 1, check
)
    cv2.imwrite(imageName, img)

    # image_frame = np.concatenate((img, label), axis=0)

    if ret == True:
        if check == 1:
            dim = (self.display_1.width(),
self.display_1.height())
            img1 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
            height, width, _ = img1.shape
            bytesPerLine = 3 * width
            img1 = qtg.QImage(
                img1.data,
                width,
                height,
                bytesPerLine,
                qtg.QImage.Format_RGB888,
            ).rgbSwapped()

            self.display_1.setPixmap(qtg.QPixmap.fromImage(img1))

            elif check == 2:
                dim = (self.display_1.width(),
self.display_1.height())
                img2 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
                height, width, _ = img2.shape
                bytesPerLine = 3 * width
                img2 = qtg.QImage(
                    img2.data,
                    width,
                    height,
                    bytesPerLine,
                    qtg.QImage.Format_RGB888,
                ).rgbSwapped()

                self.display_2.setPixmap(qtg.QPixmap.fromImage(img2))
                # self.camSignal = 0

            elif check == 3:
                dim = (self.display_1.width(),
self.display_1.height())
                img3 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
                height, width, _ = img3.shape
                bytesPerLine = 3 * width

```

```

        img3 = qtg.QImage(
            img3.data,
            width,
            height,
            bytesPerLine,
            qtg.QImage.Format_RGB888,
        ).rgbSwapped()

self.display_3.setPixmap(qtg.QPixmap.fromImage(img3))
    # self.camSignal = 0

        elif check == 4:
            dim = (self.display_1.width(),
self.display_1.height())
            img4 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
            height, width, _ = img4.shape
            bytesPerLine = 3 * width
            img4 = qtg.QImage(
                img4.data,
                width,
                height,
                bytesPerLine,
                qtg.QImage.Format_RGB888,
            ).rgbSwapped()

self.display_4.setPixmap(qtg.QPixmap.fromImage(img4))
    # self.camSignal = 0

        elif check == 5:
            dim = (self.display_1.width(),
self.display_1.height())
            img5 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
            height, width, _ = img5.shape
            bytesPerLine = 3 * width
            img5 = qtg.QImage(
                img5.data,
                width,
                height,
                bytesPerLine,
                qtg.QImage.Format_RGB888,
            ).rgbSwapped()

self.display_5.setPixmap(qtg.QPixmap.fromImage(img5))
    # self.camSignal = 0

        elif check == 6:
            dim = (self.display_1.width(),
self.display_1.height())
            img6 = cv2.resize(img, dim,
interpolation=cv2.INTER_AREA)
            height, width, _ = img6.shape
            bytesPerLine = 3 * width
            img6 = qtg.QImage(
                img6.data,
                width,
                height,
                bytesPerLine,
                qtg.QImage.Format_RGB888,
            ).rgbSwapped()

self.display_6.setPixmap(qtg.QPixmap.fromImage(img6))

```

```

        # self.camSignal = 0
    else:
        print("Else not found")

    else:
        cap.release()
        cv2.destroyAllWindows()

```

## 6.2 Recording Anomalous Events to the folder

In case of detection of anomalous event, the video feed is clipped and stored into the disk. The following script represents the process:

```

# Record Anaomalous event to a file
def getStatsOfAbnormalActivity(self, cameraID, videoPath):
    x = datetime.datetime.now()
    with open("./appData/Details.csv", mode="a") as csv_file:
        fieldnames = ["CameraID", "Event", "Date", "Time", "Video
Path"]

        writer = csv.DictWriter(csv_file, fieldnames=fieldnames)
        if csv_file.tell() == 0:
            writer.writeheader()
        date = str(x.strftime("%A") + "/" + str(x.date()))
        time = str(x.strftime("%H:%M:%S"))

        writer.writerow(
            {
                "CameraID": cameraID,
                "Event": "Abnormal",
                "Date": date,
                "Time": time,
                "Video Path": videoPath,
            }
        )

```

## 6.3 Reading CSV file to display data

To display the details of anomaly, the pre-stored anomaly details are read from the CSV file and are displayed. The script for this section is:

```

#####READ CSV FILE TO DISPLAY DATA IN
QTABLEWIDGET#####

def anomaly_tableDetail(self):
    self.menuStackedWidget.setCurrentIndex(2)
    read_file = pd.read_csv(r"./appData/Details.csv")
    read_file.to_excel(r"./appData/Details.xlsx", index=None,
header=True)

    self.anomaly_details =
xlrd.open_workbook("./appData/Details.xlsx")
    self.sheet = self.anomaly_details.sheet_by_index(0)
    self.data = [
        [self.sheet.cell_value(r, c) for c in range(self.sheet.ncols)]
        for r in range(self.sheet.nrows)
    ]
    # print(self.data)

```

```

self.alarm_tableWidget.setColumnCount(5)
self.alarm_tableWidget.setRowCount(
    self.sheet.nrows - 1
) # same no.of rows as of csv file

for row, columnvalues in enumerate(self.data):
    for column, value in enumerate(columnvalues):
        self.item = QtWidgets.QTableWidgetItem(
            str(value)
        ) # str is to also display the integer values
        self.alarm_tableWidget.setItem(row - 1, column, self.item)
        # to set the elements read only
        self.item.setFlags(QtCore.Qt.ItemIsEnabled)

```

The complete code is available at the GitHub Repository:

1. **Application Code Backend (Training + Core Inferencing Engine):**  
<https://github.com/Nauman007/Surveilia-Backend>
2. **Application Code Complete (GUI Integration with backend):**  
<https://github.com/ifrahtehleel/Surveilia>

## **Chapter 7: Conclusion**



## 7 Conclusion

### 7.1 Problems faced and lessons learned

The task of anomaly detection is similar to the outlier detector problem from normal events and thus it is difficult due to the low occurrence of anomalous events. The selection of a suitable dataset is always challenging for such a critical task as model performance is directly dependent on the type of dataset used for training. Moreover, selection of efficient deep learning model also involves expertise which someone can learn with time so an extensive literature review was done to select a suitable model i.e. TSM, it was a new and recently published work so there were issues that had to be resolved by extensively studying deeper concepts and we had to resolve them on our own.

Secondly, we had limited resources for training the model as Deep learning models are computationally expensive, it was a big challenge but was resolved through utilizing online services and we were able to train our model and achieved satisfactory results.

The third problem we encountered was to create a user-friendly application such that the application can be used by a layperson. We have been able to design our ideal interface by implementing a variety of UX approaches and iterative UI prototype design.

We contacted the security head to get a better knowledge of the practical implementation of a surveillance system. And added the asked requirements in the application.

### 7.2 Project summary

Due to cheaper technology, surveillance cameras are installed everywhere such as streets, shopping malls, hospitals, schools, banks, and even residential areas to observe human activities. The basic objective of maintaining track of human behavior is to cope with unusual incidents that might need the public's attention, or maybe just the authorities' attention. The conventional watch method is focused on people; therefore, it is full of flaws, prejudices, and, most significantly, exhaustive. There have been numerous anomalous incidents in the past that could have been prevented, but conventional security technologies have struggled to do the job, such as the robbery of the Puyallup South Hill Mall and the theft of the Westfield Century City Mall [5] and many others of a similar type.

Surveilialia is a resource-efficient application that automates CCTV monitoring by analyzing multiple CCTV live video streams and to produce instant alerts against any suspicious/abnormal behavior detected. This project is intended to provide a cost-effective, less computationally expensive solution by using a trained Deep Neural Network.

We trained the model with two different architectures i.e., TSM+ResNet20 and TSM+MobileNetV2 but the architecture we opted for application development was TSM+Mobilenetv2 in which we were able to achieve training accuracy of 88% with the loss of 0.3233. Moreover, we were able to achieve a test accuracy of 83% in which the Abnormal class had 86% Precision and 83% Re-call whereas 86% and 88% were the Precision and Re-call of our Normal class. In the end based on these statistics, we were able to calculate the F1-score which was an abnormal class of 85% and a normal class having 87%. As we trained TSM with two different architectures we also calculated FLOPS and based on this performance metric we selected which model to deploy. As ResNet50 had a complexity of 8.24 GFLOPs and MobileNetV2 had a complexity of 0.64 GFLOPs based on these stats we opted for MobileNetV2 as it was less computationally expensive.

### **7.3 Future work**

Discussed below are the ideas for expansion of the project in the future:

- In the future, this project can be extended by adding more classes of anomaly detection such as detection of camera tampering.
- Surveilialia as a mobile and web-based application
- Another extension would be, providing the user to choose between the classes of the anomaly which he/she wants to detect at the camera instead of all the classes to be detected at the same time. Such as, if the camera is placed in the meeting room; the user may only want to detect any missing object from the room.



## **Chapter 8: References**

## 8 References

- [1] “UCF Crime Dataset,” [Online]. Available: <https://www.kaggle.com/mission-ai/crimeucfdataset>. [Accessed December 2020].
- [2] “Temporal Shift Module,” [Online]. Available: <https://github.com/mit-han-lab/temporal-shift-module>. [Accessed 30 July 2020].
- [3] “NVIDIA,” [Online]. Available: <https://developer.nvidia.com/embedded/jetson-nano-developer-kit>. [Accessed 22 12 2020].
- [4] “Robbers loot 10 mobile shops in Karachi’s Gulshan-e-Iqbal,” [Online]. Available: <https://www.samaa.tv/news/pakistan/2020/07/robbers-loot-10-mobile-shops-in-karachis-gulshan-e-iqbal/>. [Accessed December 2020].
- [5] “ARMED ROBBERY WESTFIELD MALL,” [Online]. Available: <https://www.mymcmedia.org/tag/armed-robbery-westfield-mall/>. [Accessed December 2020].
- [6] “ResNet50 Architecture,” [Online]. Available: <https://iq.opengenus.org/resnet50-architecture/#:~:text=ResNet50%20is%20a%20variant%20of,explored%20ResNet50%20architecture%20in%20depth..> [Accessed December 2020].
- [7] “MobileNetV2,” [Online]. Available: <https://arxiv.org/abs/1801.04381>. [Accessed December 2020].
- [8] R. Dev, “Surveillance App,” [Online]. Available: <https://play.google.com/store/apps/details?id=com.reservoirdev.surveillance&hl=en>. [Accessed 30 July 2020].
- [9] “iCetana,” [Online]. Available: <https://icetana.com/>. [Accessed 30 July 2020].

- [10] D. Security. [Online]. Available: <https://www.dahuasecurity.com/>. [Accessed 31 July 2020].
- [11] “Hikvision,” Hikvision, [Online]. Available: <https://www.hikvision.com/>. [Accessed 27 July 2020].
- [12] “Mobotix,” [Online]. Available: <https://www.mobotix.com/en/unique-quality>. [Accessed 27 July 2020].
- [13] “Realtime Anomaly Detection using Trajectory-level Crowd Behaviour Learning,” December 2020. [Online]. Available: [https://www.cv-foundation.org/openaccess/content\\_cvpr\\_2016\\_workshops/w20/papers/Bera\\_Realtime\\_Anomaly\\_Detection\\_CVPR\\_2016\\_paper.pdf](https://www.cv-foundation.org/openaccess/content_cvpr_2016_workshops/w20/papers/Bera_Realtime_Anomaly_Detection_CVPR_2016_paper.pdf).
- [14] “IoT based Security System using Raspberry Pi,” [Online]. Available: <https://www.ijert.org/iot-based-security-system-using-raspberry-pi>. [Accessed December 2020].
- [15] “RaspBerry Pi,” [Online]. Available: <https://www.raspberrypi.org/>. [Accessed December 2020].
- [16] “Advance Intelligent video surveillance system,” [Online]. Available: <https://www.intechopen.com/books/intelligent-video-surveillance/advance-intelligent-video-surveillance-system-aivss-a-future-aspect>. [Accessed 31 July 2020].
- [17] “Anomaly Localization in Topic-based analysis of Surveillance videos,” [Online]. Available: <https://www.cs.cmu.edu/~dpathak/papers/wacv15.pdf>. [Accessed 29 July 2020].
- [18] “Video Anomaly Detection in Confined Areas,” [Online]. Available: <https://pdf.sciencedirectassets.com/280203/1-s2.0-S1877050917X00148/1-s2.0-S1877050917319294/main.pdf?X-Amz-Security-Token=IQoJb3JpZ2luX2VjEGQaCXVzLVVhc3QtMSJHMEUCIQDqWRiDU9zfneyjU41BFQ8eUwWlBFoZGnA4UCRrwRtZXwIgpQ63Vw2oNx5DBjX5NoJLv%2B%2FXED1GRIuvirKGIaZ0>. [Accessed 30 July 2020].

- [19] “Anomaly Detection,” [Online]. Available: <https://www.irjet.net/archives/V7/i1/IRJET-V7I1207.pdf>. [Accessed 30 July 2020].
  
- [20] C. G. S. H. Ji Lin, “TSM: Temporal Shift Module for Efficient Video Understanding,” [Online]. Available: <https://hanlab.mit.edu/projects/tsm/>. [Accessed December 2020].

## 2020\_FYP II

### ORIGINALITY REPORT

6%

SIMILARITY INDEX

4%

INTERNET SOURCES

2%

PUBLICATIONS

3%

STUDENT PAPERS

### PRIMARY SOURCES

1	<b>emresahin.net</b> Internet Source	1%
2	<b>repozitorij.mathos.hr</b> Internet Source	1%
3	<b>Submitted to Higher Education Commission Pakistan</b> Student Paper	<1%
4	<b>hanlab.mit.edu</b> Internet Source	<1%
5	<b>Submitted to Universiti Teknologi Petronas</b> Student Paper	<1%
6	<b>lutpub.lut.fi</b> Internet Source	<1%
7	<b>www.samaa.tv</b> Internet Source	<1%
8	<b>Ji Lin, Chuang Gan, Song Han. "TSM: Temporal Shift Module for Efficient Video Understanding", 2019 IEEE/CVF International Conference on Computer Vision (ICCV), 2019</b>	<1%