AHMED MUSTAFA
01-134192-092
SULMAN AHMED SATTI
01-134192-079

Crowd Behavior Analysis based Automated Surveillance System

Bachelor of Science in Computer Science

Supervisor: Dr. Sumaira Kausar

Co-Supervisor: Dr. Momina Moetesum

Department of Computer Science
Bahria University, Islamabad

January 2023

Certificate

We accept the work contained in the report titled "Crowd Behavior Analysis based Automated Surveillance System", written by Mr. Ahmed Mustafa AND Mr. Sulman Ahmed Satti as a confirmation to the required standard for the partial fulfillment of the degree of Bachelor of Science in Computer Science.

Approved by . . . :

Supervisor: Dr. Sumaira kausar (Senior Associate Professor)

_____

Internal Examiner: Name of the Internal Examiner (Title)

_____

External Examiner: Name of the External Examiner (Title)

_____

Project Coordinator: Ms. Maryam Khalid Multani (Assistant Professor)

_____

Head of the Department: Dr. Arif Ur Rehman (HOD Computer Science)

_____

June , 2023

Abstract

Surveillance systems based on CCTV cameras and other sensors are quite common now but they remain quite limited in their effectiveness and efficiency. Due to all judgement and decision-making being left to human observers, it is quite common to see the observer misread the situation and by the time they take an action, the situation has already gotten quite worse. The proposed system provides surveillance systems a much needed deep-learning based monitoring system that can detect violence outbreaks quickly and effectively, hence allowing human observers to diffuse the situation before things get any worse. The architecture of the proposed system has four components:

- A 3D-Convolutional Neural Network trained on a dataset of violence outbreaks in.
- A Django-based frontend that allows users to detect violence and aggression in their CCTV camera/ live feed,
- as well as by uploading videos.
- The system also shows logs of recorded videos and the current time of video in which violence was detected.

In order to successfully create the proposed system, initially a deep learning architecture(3D-DenseNet) is selected. The chosen model is also compared with existing solutions which include but are not limited to:

- Conv-
- BiConv-LSTM
- Conv2D systems

and the results have been recorded to display the effectiveness of each architecture. Then all the data in the dataset is perprocessed. Ater preprocessing, the data is converted into tensors and model is trained. When model accuracy is satisfactory,the states are saved and then used in the front-end application.

Acknowledgments

In The name of Allah, Most Gracious, Most Merciful. All praises be to Allah Almighty as no matter how much we thank Him to give us the ability to complete this project and endure all difficulties in the process would not be enough. We would also like to take this opportunity to thank Dr. Momina Moetesum, our initial supervisor for providing me with the proper guideline on in the initial phases of the project. We would also like to thank Dr. Sumaira Kausar, our current supervisor who accepted the supervision for this project after the departure of the initial supervisor from the university and answered all our queries enabling the conclusion of this project. We are also deeply in gratitude of our respected parents who supported us throughout our lives and prayed for our success.

AHMED MUSTAFA, SULMAN AHMED SATTI
Islamabad, Pakistan

May 2023

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

LEA	Law Enforcement Agencies
CNN	Convolution Neural Network
3D-CNN	3 Dimensional Convolution Neural Network
2D-CNN	2 Dimensional Convolution Neural Network
CV	Computer Vision
DL	Deep Learning
ML	Machine Learning
LSTM	Long Short Term memory
CONV-LSTM	Convolutional - Long Short Term memory
Bi-CONV-LSTM	Convolutional - Bidirectional Long Short Term memory
CSRF	Cross Site Request Forgery
Resnet	Residual Neural Network
SVM	support vector machine
KDE	Kernel Density Estimation
Vif	Violent flows
OVif	Oriented Violent Flows
SIFT	Scale-Invariant Feature Transform
Mo-SIFT	Motion Scale-Invariant Feature Transform
DenseNet	Densely Connected Convolutional Networks
3D-DenseNet	3 Dimensional Densely Connected Convolutional Networks
Df	Data frame
MVT	Model View Template
XML	Extensible Markup Language
url	uniform Resource Locator
HTML	Hyper Text Markup Language
js	javascript
json	javascript object notation
GUI	Graphical User Interface
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ssl	secure socket layer
smtp	simple mail transfer protocol

Chapter 1

Introduction

1.1 Project Background/Overview

It was a night before my cousin's wedding and gentlemen were sitting in courtyard of house. Topic under discussion was extrajudicial killing over alleged blasphemy. My father was citing one of his own incidents when he barely escaped becoming victim from a mob. These were the days when he was in the autos business. One morning while coming to his workplace, he saw a driver, in traditional religious attire parked his vehicle in the middle of busy street and left. As expected, that led to a chaos while driver returned seemingly careless of the mess he just created. When he was about to leave the place, my father confronted him and admonished him over his incautious attitude. My father also gave an additional statement which, little did he knew, would be bringing grave consequences.
In afternoon, whence returning from mosque after praying Asr, he saw numerous people on bikes blocking the entry and exit points to place where incident took place in the morning. They were chanting slogans "Ran Away, Blasphemer. . . Ran away". Now he knew he has got himself into serious trouble.
Guessing what would have happened, running away from the spot would have made matters worse. So, he turned himself in, voluntarily, to crowd and asked about the charges. He was told that he has issued blasphemous statements over religious institutes: Madrassas and Mosques. He demanded the person(driver), who made the allegations. He revealed to them that he holds a master's degree himself in Islamic Law. Then pointing to the driver, he narrated what happened in the morning and his statement. The statement was,
"Irresponsible attitude of guys like you in religious attire has tarnished value of Mosques and Madrassas. Observing you has led people to think that manners aren't taught in these institutes."
It became evident to everyone that driver caused all trouble. Crowd, which was about to be

Figure 1.1: Sialkot lynching: Sri Lanka minister slams extremist mobs in Pakistan, Source: Republic World

turned into a mob, made apologies to my father and soon dispersed.
Similar patterns have been observed in incidents.

1. with Srilankan citizen in Sialkot (Das and Kumar, 2021)[1] shown in figure 1.1
2. with Junaid Jamsheed at Karachi Airport (Diaries, 2016) [2] shown in figure 1.2

with exception of a peaceful resolve.
These are just a few reported ones. But many such incidents happen every now and then, around different corners of Pakistan, over different reasons. Regardless of the reason, human behavior in such scenarios has been observed the same especially in larger social gatherings (Power and Morton, 2022).

1.2 Problem Description

Here in Pakistan, post 9/11 brought wave of terrorism. Counter security measures were taken like antiterrorism operations and enhancing monitoring through CCTV cameras were installed in sensitive areas. Now they are common even in little shops, streets etc. Every now and then, CCTV footages are circulating on media after some incident has happened.

There is a four-step process which repeats itself in most reported incidents.

1. Firstly, an incident takes place under the observation of a CCTV Camera/s.

Figure 1.2: Unknown men attack Junaid Jamshed at Islamabad airport, Source: Geo TV

2. Then it is reported on electronic/social media.

3. CCTV camera/s of the scene are checked for footage.

4. Law enforcement agencies act according to information obtained from the footage.

1.3 Problem Statement

Similarly, Crowd violence incidents by mobs have been caught on footage. Based upon which LEA bring people into custody and further legal procedures are put into place. But damage has been done.
"*Prevention is better than Cure.*"
If LEA can be informed at the very 1st step, then they would be able to intercept the situation from occurring.

1.4 Project Description

This can be resolved through a surveillance system which is based on CV. When such mobs are into shaping, surveillance system can smartly identify alarm LEA on real time. Currently, most video streams from CCTVs are feed forward to screens/computers from where it is monitored by a person or multiple persons. But there are key issues with this approach.

1. It is highly subjected to human bias and other limitations like laziness, attention span, focus etc.
2. Even if an incident is identified on time, further human involvement especially in communications slows down the whole process.
3. Problems with the footage can't be resolved by human eye like blur, dim light etc. It can be fixated with CV techniques.

Hence, there is a requirement to develop a smart system, which addresses these issues.

1.5 Project Objectives

Our main objective is to develop a smart solution to address the above problem by developing a system that classifies violence effectively and alerts the user of such activity.Lastly, Django framework is used to introduce,

1. a user friendly front-end.
2. a secure system that restricts access to invalid users.

1.6 Project Constraints

Though Project sounds optimistic with huge impacts in future, but it comes up with some constraints.

- First and foremost, is coverage of zones through CCTVs. As mentioned in problem description, counter measures against terrorism led to installation of CCTV cameras at large. But it included majorly crowded, busy and high stakes/risk zones. Areas which aren't under eye of CCTV or some other continuous video source then service would be unavailable there.
- The dataset used for training is the only publicly available dataset at the time which fulfills requirements.
- Other major constraint is good internet connectivity which is basic requirement of live feed of video to the 3-S. Hence quality of service will be compromised in regions with internet connectivity problem. Such regions are prevalent throughout Pakistan.
- Lastly, CCTV camera footage especially live feed etc. is a matter of high security concern which normally is not allowed to students.
- A stable internet connection is required in order to send the auto-generated email at high speeds to the user. If not available it may hinder the functionality of the system.

1.7 Project Assumptions

Considering constraints and other factors, presuppositions and assumptions are made. This may fade out some reality but will help in arriving at a precise and feasible solution in limited amount of time.

Firstly, users of the service are supposed to have sound internet connectivity till the service is under usage. Also, CCTV Cameras are assumed to be fully functional all the time.

Lastly, to address the accessibility of CCTV cameras, there is a major consideration to be made. For now, product will be demonstrated on publicly available CCTV/Camera/Mobile etc. videos and streaming from the webcam of the pc. It would be considered that they are streaming live from regions when incident was taking place.

1.8 Project Scope

Though complete solution mentions of video stream from CCTV, but project working will be focused on

1. Classification of violence through incoming video clip.
2. Detection of both individual and mob violence.
3. Saving logs of video name and video time in which violence was detected.
4. Live classification using webcam of users PC.

Other thing is, flagging violence is main functionality of the system and raising alarm automatically which is Smartness of the System, as it sends an auto-generated email to the user and creates an audio alarm. However, it won't be suggesting what course of action needs to be taken. Since it is clearly not a recommendation system.
Lastly deployment through web-framework is though part of the project and security has also been mentioned in the project objectives. But scope is limited to basic security features which includes user authentication via username and password and measures taken to stop csrf(Cross Site request Forgery). Deeper concepts of cyber security of the system is, for now out of scope of Project. The deployment of the system on a web hosting and handling multiple users is also not covered in the project.

Chapter 2

Literature Review

In recent years the demand for smart surveillance systems that can detect threats such as violence has increased exponentially. This is mainly due to an increase in violent activity throughout the world, either in protests, rallies or even random incidents at different places.This put the safety of people present at the premise at a higher risk. As a result, security systems were developed and concurrently research was also conducted in which new methods/ techniques were developed and existing ones were tested to see how they can meet this demand.

2.1 Existing Systems

2.1.1 Abto Software

The Abto software[3] is a technology designed to help ensure public safety by detecting and alerting on violent activity in real-time. It can be integrated into any security system, making it a versatile and useful tool for a variety of settings.
The software uses visual crowd surveillance to monitor for signs of violent activity, such as fights or other physical altercations. When it detects such activity, it automatically generates an alert, enabling authorities or other responsible parties to respond as quickly as possible
. To make this possible, the software uses intelligent video analytics, which are designed to analyze CCTV footage and extract relevant information from it. These analytics use image processing and machine learning techniques to analyze the footage quickly and accurately, enabling them to detect violent activity much more efficiently than a human observer could.
Additionally, the various components of the Abto software are designed to work together in a flexible environment, allowing them to be customized to serve a specific purpose.

This means that the system can be tailored to meet the needs of a particular setting or application, making it even more useful and effective.
The frameworks and libraries used in the development include:

- OpenCV v3-v4.
- TensorFlow PyTorch.
- Keras.
- NumPy, Scikit-learn, Pandas.

The Programming Languages used are:

- Python
- C++
- Java/Android
- Objective-C
- MATLAB/Octave
- R

Areas of application include:

- Law enforcement.
- Public safety ensuring.
- Crowd monitoring and behavioral analysis.

2.1.2 Lightspeed Alert

Lightspeed[4] Alert is a threat detection application designed to help schools prevent suicides, bullying, and school violence. It does this by using a combination of cloud integrations and smart agents to automatically scan and detect concerning online indicators. These indicators could include things like threatening language or behavior on social media or other online platforms.
When these indicators are detected, Lightspeed Alert sends detailed reports to designated school staff for timely intervention. The reports include information about the potential threat and any relevant context.
In addition to the automated scanning and reporting, Lightspeed Alert also uses patented artificial intelligence technology and human review to analyze alerts and evaluate imminent threats in real-time. This helps to ensure that only genuine threats are acted upon, and that

appropriate interventions are taken.
If a threat is deemed high level, safety specialists will immediately report the issue to chosen members of school staff and/or law enforcement. This helps to ensure that the necessary steps are taken to protect students and prevent an incident from occurring.

2.1.3 Oddity AI Systems

Oddity[5] is developing a violence detection system that can help in the detection of high impact crime while it is being streamed on live footage. The algorithms that will be used to monitor the live feed will first detect the abnormality that is occurring in the footage and then send an alert to personnel of security.
The algorithms can detect:

- Violent incidents.
- Accidents such as: car crashes.
- Perimeter breaches.

The algorithms are designed to analyze subjects anonymously, and they operate independently while being deployed on premise. Additionally the system has the capacity to process 1000 videos at a time.

2.2 Related Works

2.2.0.1 Deep Learning Approaches:

The approach of Sudhakaran et al.(2017)[6] network consists of a series of convolutional layers followed by max pooling operations for extracting discriminant features and convolutional long short memory (conv-LSTM) for encoding the frame level changes, that characterizes violent scenes, existing in the video.
The convolutional layers are trained to extract hierarchical features from the video frames and are then aggregated using the conv-LSTM layer. The network functions as follows:

- The frames of the video under consideration are applied sequentially to the model.
- Once all the frames are applied, the hidden state of the conv-LSTM layer in this final time step contains the representation of the input video frames applied.
- This video representation, in the hidden state of the conv-LSTM, is then applied to a series of fully-connected layers for classification.

The proposed model used for the CNN part was the AlexNet[7] model which was trained on the ImageNet database[8]. This model extracts frame level features. In the conv-lstm layers 256 filters are used which results in 256 feature maps.
The network stores the difference between adjacent frames as inputs which enables it to model changes taking place between adjacent frames.

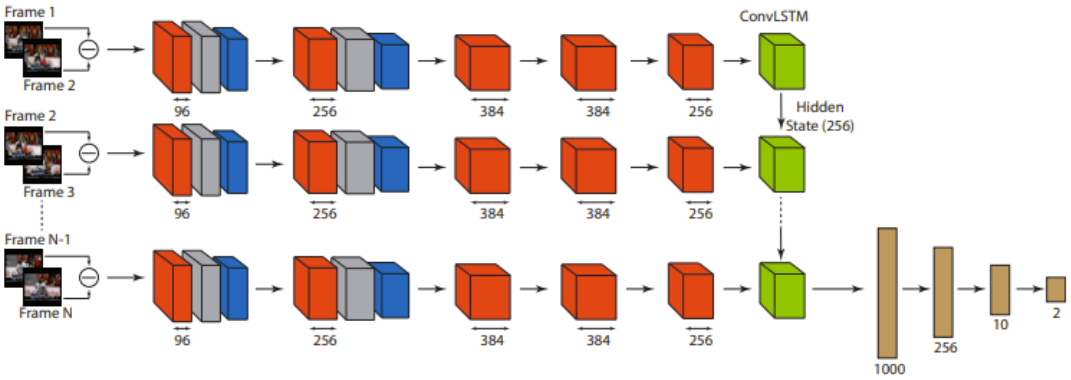


Figure 2.1: Architecture of convolutional LSTM

Next is the approach of Gall et al.(2011)[9].Frames from the sequence of images are accumulated in order to build a representative image for each sequence.
For this method there are 2 important steps:

- Feature extraction from image.
- A 2D convolution network to classify the representative image and obtain the result.

The main step of feature extraction id performed by first leveraging the relevant motion parts and reduce the irrelevant parts of the image. Then Hough voting[9] is used to obtain weighted frames. Then the image is rebuilt using the weighted frames obtained and after that a 2D convolution network is used to give final decision for the sequence of the image.

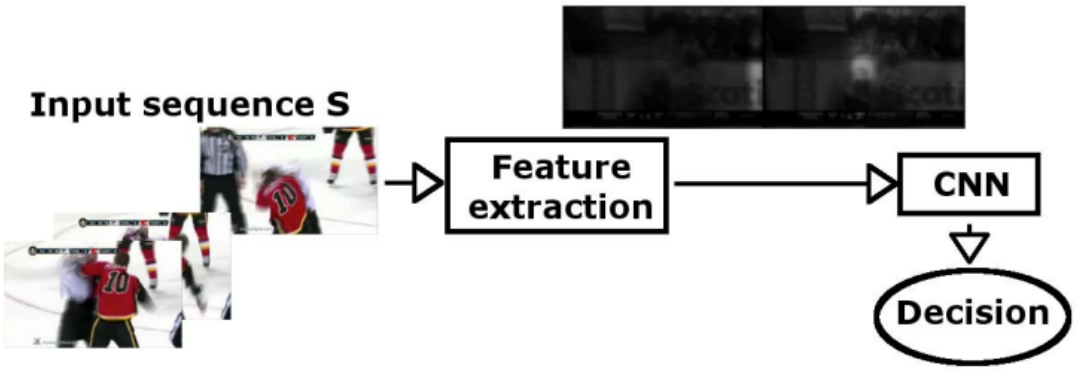


Figure 2.2: Architecture of Hough-Forest + 2D CNN

In Hason et al.(2018)[10] the architecture used for this approach builds on the traditional convolutional LSTM architecture to detect violence in videos.
This architecture includes bidirectional temporal encodings and elementwise max pooling. Each video frame is encoded as a collection of feature maps via a forward pass through a VGG13 network[11].

After encoding the feature maps are passed on to the BiConvLSTM to further encode them along the temporal direction of the video. This will enable the model to perform both forward and backward pass. Then the elementwise maximization is performed on each of the encodings to create a representation of the entire video. After that the representations are passed to a classifier in order to detect violence.

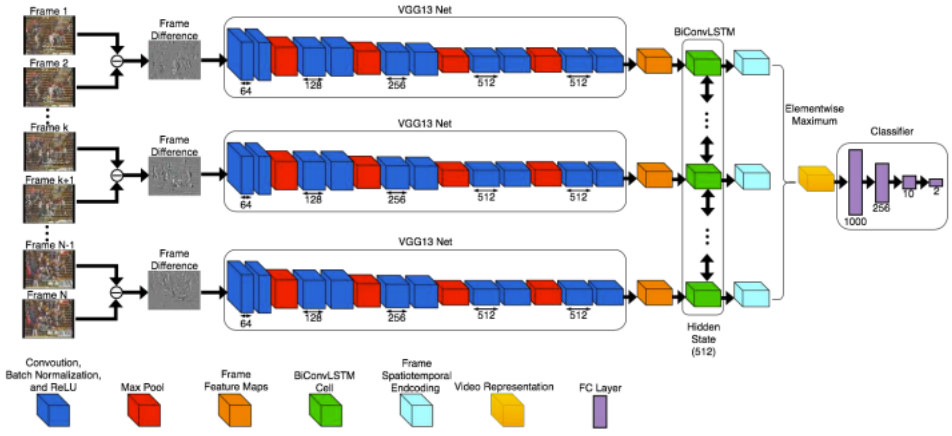


Figure 2.3: Architecture of Bi-Convolutional LSTM

2.2.0.2 Hand Crafted Approaches:

In the approach Gao et al.(2016)[12] the feature of Oriented violent flows in used for violence detection. This makes full use of the oriental information of optical flows. Feature selection is done using Adaboost method and those features are then trained on an SVM classifier.
In this paper, the combination of both Oriented violent flows + the violent flows features is also done and then ADAboost[13] and SVM[14] are used on those features

The Violent flows(Vif)[15] is a descriptor which creates a feature vector from a video. This is done by computing optical flow between consecutive frames and then calculates a magnitude of flow vector. Then the magnitudes of flow vectors are used to calculate magnitude-change maps, which are used to calculate mean-magnitude map.
This mean magnitude map is divided into M × N non-overlapping regions, in each of which the frequencies of magnitude changes are collected and represented as a fixed-size histogram. And the final ViF vector is the concatenation of these histograms.

The Oriented Violent flows(OVif)[12] descriptor is the proposed descriptor used in the approach, which performs in the same way as the Vif descriptor, however the Vif descriptor looses some information for example when the flow vectors of the same pixel in two sequential frames have the same magnitude but only differ in their directions, the effect of ViF seems to be restricted. This is because that ViF thinks that there is no difference between these two flow vectors but actually they differ a lot. This feature depicts information related to both motion magnitudes as well as direction.

In the approach of Xu et al.(2014)[16] the Mo-SIFT[17] is used as a feature extraction technique. Then The KDE[18] method is used to select the most representative features from the extracted features. After selecting the features sparce coding is applied to transform reduced low level descriptors into compact mod-level features.
Then max pooling is done to obtain a highly discriminative representation of the entire video. Finally SVM[14] classifier is trained using these features.

The **Mo-SIFT**[17] descriptor consists of a 256-dimensional vector that combines standard SIFT[19] features with a histogram of optical flow. The process for creating this descriptor begins by applying the standard SIFT algorithm to identify visually distinctive points in an image. Next, points with insufficient optical flow around them are eliminated, leaving only those with strong motion. The resulting descriptor is made up of the standard SIFT features in the first 128 dimensions and the aggregated histogram of optical flow in the remaining 128 dimensions.

Kernel density estimation (KDE)[18] is a method for estimating the probability density function (PDF) of a random variable. It can be used to analyze the distribution of Mo-SIFT features extracted from an image or video. In KDE, a kernel function is applied to each Mo-SIFT feature to assign a weight based on its proximity to a given point. The resulting weights are summed to estimate the PDF at that point.

Sparse coding is a machine learning technique that represents data using a small number of features, or "atoms," from a fixed dictionary, with the majority of the weights set to zero. It is used to compactly and efficiently represent data, and can be implemented using optimization algorithms such as gradient descent or iterative shrinkage-thresholding.

2.3 Comparison

The table below (Table 2.1) compares the accuracy of the approaches discussed in the literature review whith the proposed approach and dataset(Violence-flows).

Approach	**Architecture**	**Accuracy**
Sudhakaran et al.(2017)[6]	Conv-LSTM	94.57
Hason et al.(2018)[10]	Bi-Conv-LSTM	93.87
Gao et al.(2016)[12]	Vif + Ovif	88
Xu et al.(2014)[16]	Violent video detection based on MoSIFT feature and sparse coding	89.05

Table 2.1: Table of comparison between approaches

Chapter 3

Requirement Specifications

For any system to be successful there are a certain number of requirements that must be met. These requirements can be primarily based on the functionality of the system, which means that what tasks the system will do are specified. The requirements also include other factors such as hardware requirements and also other non-functional requirements such as security and/or user friendliness of the system. In this chapter these requirements are specified.

3.1 Existing System

In recent times there has been high demand for surveillance systems that can help the security personnel in performing their duties. So that the risk of any mishap or dangerous activity can be drastically reduced resulting in an increase in safety for the people present on the premise. Hence there are some systems that have been developed to cater this requirement. However the need is not yet satisfied as there are some issues regarding the existing systems. Some of these issues include:

- Generation of false positives.
- High computation power required in order for the system to work effectively.

When false positives are generated, the user is required to give higher attention in order to monitor the system because there is wrong classification meaning that genuine incidents may be ignored. When high computation power is required to operate a system it may cause the host device to hang, making it to detect and report any incident.

3.2 Proposed System

The system we propose is a web based application witch will consist of an end-to-end deep learning model that will both detect and recognize violence occurring in the incoming

video clip deployed with it. The model will be a 3D-CNN model. The proposed model has the architecture of DenseNet consisting of dense blocks and transition layers. The front end or user interface will be developed using Django framework. The user will upload a video and then the system will predict whether there is violent or non violent behavior in the video.
The system will be power efficient as it is a web based application meaning, the computation will not be done on the users machine but on a web server.

3.3 Requirement Specifications

3.3.1 Functional Requirements

Functional requirements of a system are an important factor that should be taken into consideration. This is because these requirements define the steps/ conditions that should be fulfilled in order to make the system function successfully.
The functional requirements of our system are:

- The system will be able to take video clip as input.
- the system will be classify violent or non-violent activity for both recorded video and live streaming via webcam.
- The user interface will be available for the user to upload the video clip.
- The model present will successfully classify violence or/and non-violence the incoming clip based on the features extracted.
- In the case of live classification, the system will generate an alarm, which includes creating audio alert along with sending the user an auto-generated email and make the borders red when violence is detected.
- In the case of recorded videos when violence is detected, the system will save video logs in which video name and the time at which violence is detected will be saved.
- The system will be able to successfully register users.
- Registered users can successfully log in and access the Recorded, live and logs pages.

3.3.2 Non-functional Requirements

- The system is user friendly and easy to navigate.
- The website should be available to user at all time.
- The time taken for the system to classify the clip is expected to take from 2-4 seconds.

- The input clip should not be corrupted in any manner.
- Only verified users should be able to access the system.

3.3.3 Hardware Requirements

The system requires a computer with a strong computer with graphic card. This is because video processing will be performed, which means that the model has to be first trained on the video data set and then classify the behavior accurately.

Hardware Component	Requirement
RAM	8GB
Hard Disk	100GB
Graphics card	NVIDIA Ge-Force GTX 1080 4GB
Processor	Intel Core i5

Table 3.1: Hardware requirements

3.4 Use Cases

3.4.1 Use case 1

Figure 3.1 shows the use case of how the user is registered to the platform and Table 3.2 shows the description of what actions the user is to perform during his/her registration,along with the brief description of the steps leading to success and failure.

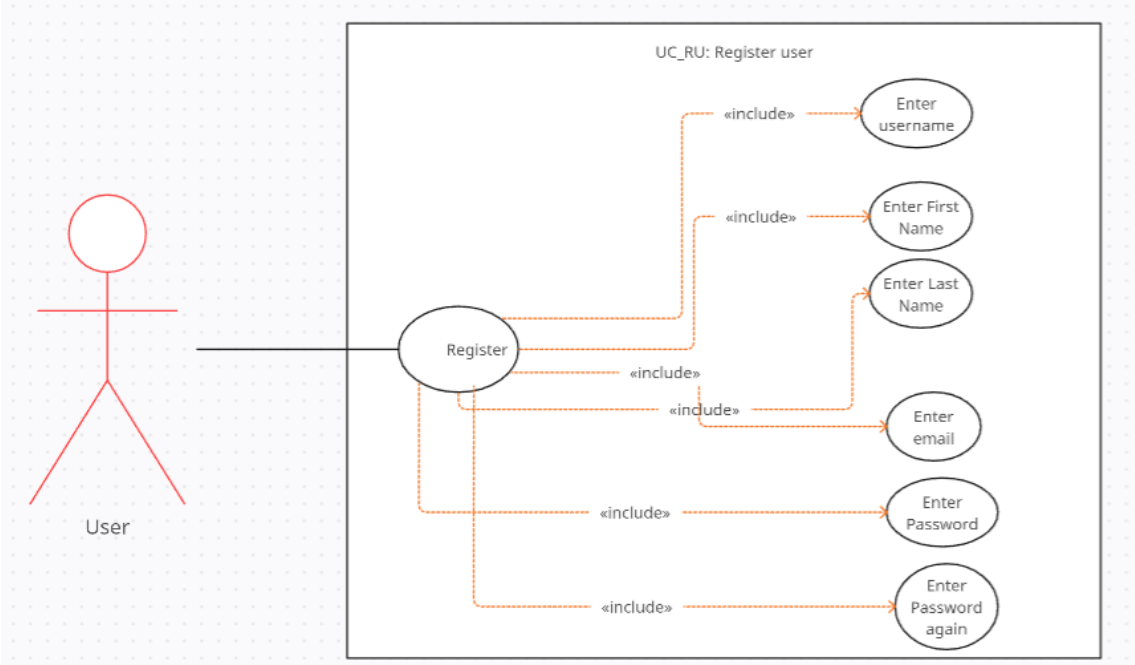


Figure 3.1: Use Case Diagram for User registration

Use Case id	UC-RU
Use case name	Register User
Actors	User
Data	Name, First Name, Last Name, Email, Password, Repeat password
Trigger Stimulus	User presses register button
Pre-condition	Registration page is open
Assumptions	All data entered by user is correct and valid.
Normal flow of events	• User Enters details. • User is successfully registered.
Alternate flow of events	• User enters details. • One or more of the details entered by the user is invalid/incorrect. • User registration is unsuccessful.
Main success scenario	User registration is successful.
Post Condition	User can proceed to login to access the main dashboard.

Table 3.2: Use case description for user registration.

3.4.2 Use case 2

Figure 3.2 shows the use case of how the user is logged in and Table 3.3 contains the use case description. This description contains the actions user is to perform while he/she logs in along with brief description of the steps leading to success and failure.

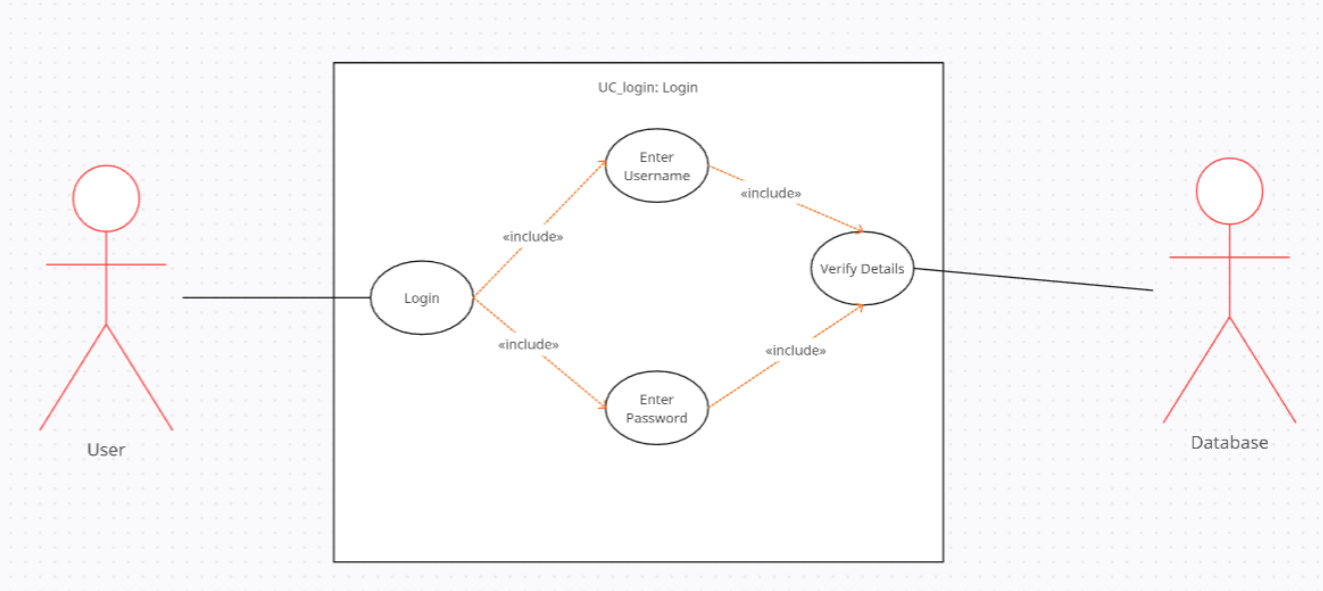


Figure 3.2: Use case diagram for User login

Use Case id	UC-Log
Use case name	Login
Actors	User
Data	Email and password.
Trigger Stimulus	User clicks on the login button
Pre-condition	• User has registered successfully to the platform. • Login page is open.
Assumptions	User has registered successfully to the platform.
Normal flow of events	• User enters his/her username and password. • User logs in successfully.
Alternate flow of events	• User enters his/her email and password. • The email and/or password entered are incorrect. • User login is unsuccessful.
Main success scenario	User has successfully logged in.
Post Condition	User enters the main dashboard.

Table 3.3: Use case description for user login

3.4.3 Use case 3

Figure 3.3 shows the use case in which series of steps are taken in order to classify a video are shown and tables 3.4 - 3.6 give the description of the entire process in which the user uploads the video to prompting the system to start pre-processing of that video then finally the prediction the trained model will give to the user.

Also the user can view more details regarding the video for recorded videos through the video player button as described in Table 3.7. Table 3.8 shows the detailed description of the process of creating logs of recorded videos. The tables also describe the steps that lead to success and/or failure.

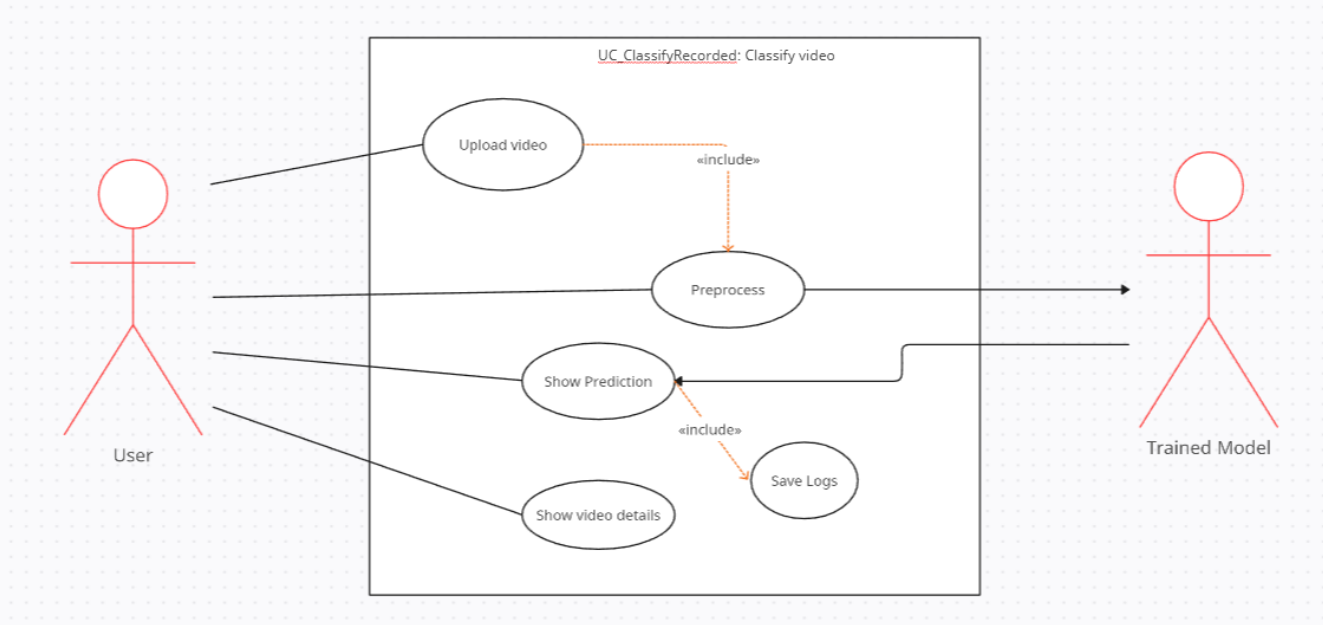


Figure 3.3: Use case diagram of recorded video classification.

Use Case id	UC-ClassifyRecoreded-001
Use case name	Load Video
Actors	User
Data	Video Clip
Trigger Stimulus	choose file and then submit button is pushed
Pre-condition	User is logged in successfully
Assumptions	User is logged in successfully
Normal flow of events	• Pop-up is opened, and user accesses the video/ webcam is turned on. • Video load is successful/ camera opens successfully.
Alternate flow of events	• Pop-up is opened, and user accesses the video. • Video is missing/corrupt. • Video load failed.
Main success scenario	Video is loaded successfully
Post Condition	Video starts playing and preporcessing begins

Table 3.4: Use case description for load video.

The table above describes the process of loading the video on the web-page. loading the video is only possible if the user has logged in successfully. The user then clicks the load button and then a pop-up is opened in which the user browses to get the video file. If the file is found, it gets uploaded and starts playing. If the file is corrupt or missing the upload is failed.

Use Case id	UC-ClassifyRecoreded-002
Use case name	Preprocess video
Actors	User, Trained Model
Data	Video clip
Trigger Stimulus	Connect button is pressed
Pre-condition	Video has been successfully loaded
Assumptions	Model is well trained and can give accurate classification.
Normal flow of events	• Video is loaded. • Features are extracted. • Data augmentation is done.
Alternate flow of events	• Video is loaded however features are unable to be extracted. • Video is missing/corrupt. • Pre-processing failed.
Main success scenario	Pre-processing is successful
Post Condition	Pre-processed data is received by the trained model.

Table 3.5: Use case description for pre-process video

The above table describes the sequence of events that occur during the pre-processing of the video. The video clips are transformed in such a manner that it can be acceptable to the deep learning model. This includes changing in shape and size of the clip and converting them into tensors and then passes that data to the model. however if the video clip is corrupt or missing the pre-processing will fail.

Use Case id	UC-ClassifyRecoreded-003
Use case name	Show prediction
Actors	User, Trained Model
Data	Label returned by trained model
Trigger Stimulus	• Video has been successfully pre-processed and data is fed into model. • Model returns the classified label.
Pre-condition	Video has been successfully pre-processed.
Assumptions	Model is well trained and can give accurate classification.
Normal flow of events	• Pre-processed data is fed to the model. • Model returns label. • Classified result is shown to the user and background is changed.
Alternate flow of events	• Pre-processed data is fed to the model. • Model returns wrong label. • Wrong result is shown to the user and background is modified accordingly.
Main success scenario	User is shown correct label.
Post Condition	Log of video is created

Table 3.6: Use case description for show prediction

The above table describes the process of showing the correct label to the user. For this to happen the video must be pre-processed and sent to the model. By assuming that the model shows accurate predictions the pre-processed data is passed to the model and the model returns the accurate label.

Use Case id	UC-ClassifyRecoreded-004
Use case name	Show Details
Actors	User
Data	video clip.
Trigger Stimulus	Video is played.
Pre-condition	Video has been successfully uploaded.
Assumptions	video is successfully loaded.
Normal flow of events	• User Loads video. • user gets to see the details of video such as current time, duration etc .
Alternate flow of events	• User loads video. • Video is loaded but error occurred. **Or** • nothing happens.
Main success scenario	User is shown details of video.
Post Condition	None

Table 3.7: Use case description for show details

The table above shows the sequence of events that happen in order to show video details. Assuming that the video has been loaded successfully and started playing, when the user hovers on the video player a pop-up opens and details including, length, duration of the video will be shown.

Use Case id	UC-ClassifyRecoreded-005
Use case name	Save Logs.
Actors	trained model
Data	video clip.
Trigger Stimulus	Violence is classified.
Pre-condition	Violence is classified.
Assumptions	video is successfully loaded and is playing.
Normal flow of events	• Preprocessed frames are passed to the model. • Model returns the correct prediction. • If violence is detected the video name, user name and timestamp are saved to logs file.
Alternate flow of events	• Preprocessed frames are passed to the model. • Model returns the wrong prediction. • Due to wrong prediction wrong details are saved to logs file.
Main success scenario	Correct classification details are saved.
Post Condition	None

Table 3.8: Use case description for Creating video logs

The table above shows the sequence of events that happen in order to create logs of a video. Assuming that the video is being classified, when the violence is detected, the username of current user along with video name and timestamp of the video is saved in the logs file.

3.4.4 Use case 4

The Figure 3.4shows how the system retrieves the logs data from the logs file and Table 3.9 shows the description. the description contains the actions the system performs while retrieving and displaying video logs.

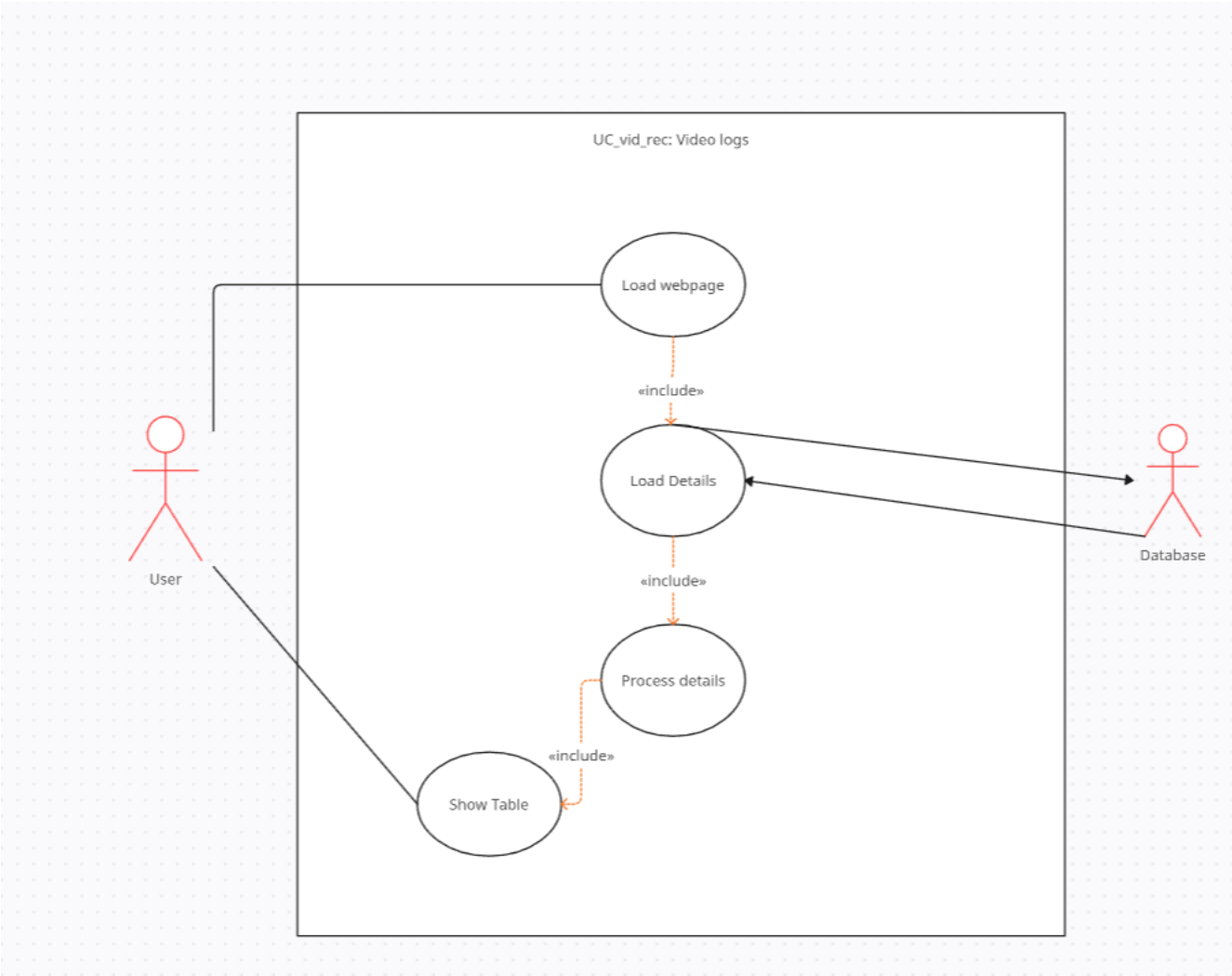


Figure 3.4: Use case for displaying logs

The table 3.9 shows the sequence of events that happen in order to load logs of a video. Assuming that the video is being classified, when the violence is detected, the username of current user along with video name and timestamp of the video is saved in the logs file.

Use Case id	UC-vid-rec
Use case name	Video Logs.
Actors	user
Data	video clip.
Trigger Stimulus	User clicks the logs tab
Pre-condition	user has logged in.
Assumptions	user has logged in and has played recorded videos.
Normal flow of events	• User clicks the logs tab • system loads and filters the data. • If violence was detected in any of users video then name and timestamp of that video will be displayed in tabular form.
Alternate flow of events	• User clicks the logs tab • Error in loading and/or filtering data. • Empty page is displayed.
Main success scenario	Logs are displayed.
Post Condition	None

Table 3.9: Use case description for loading video logs

3.4.5 Use Case 5

Figure 3.5 shows the use case in which series of steps are taken in order to classify a video live stream are shown and tables 3.11-3.12 give the description of the entire process in which the user opens the tab prompting the system to start pre-processing of that video stream then finally the prediction the trained model will raise an alarm.

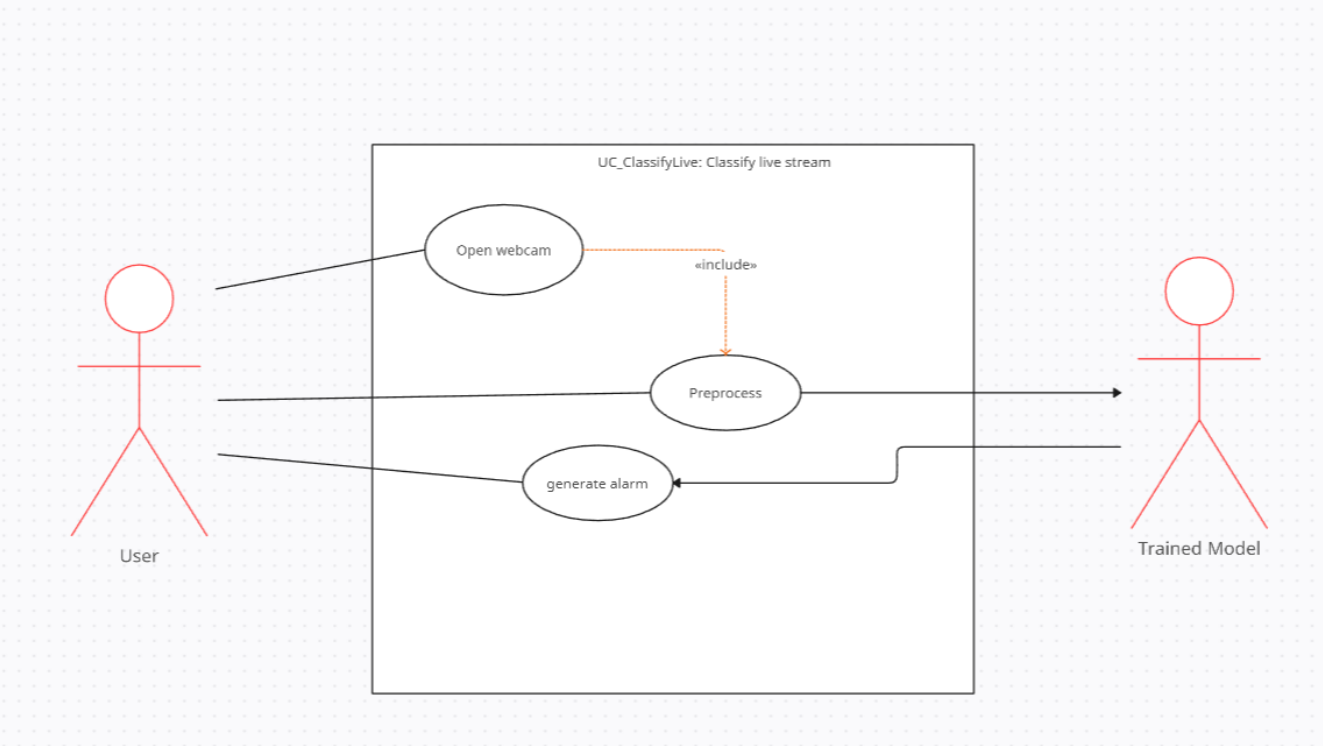


Figure 3.5: Use case diagram of Live classification.

Use Case id	UC-UC-ClassifyLive-001
Use case name	Open live tab
Actors	User
Data	Video Clip
Trigger Stimulus	User clicks on live tab
Pre-condition	User is logged in successfully
Assumptions	User is logged in successfully
Normal flow of events	• webcam is turned on. • camera opens successfully.
Alternate flow of events	• Video is missing/corrupt. • Video stream failed.
Main success scenario	Video is loaded successfully
Post Condition	Video starts playing and preporcessing begins

Table 3.10: Use case description for Open live tab.

The table above describes the process of loading the video on the web-page. loading the video is only possible if the user has logged in successfully. The user then clicks the

load button and then a pop-up is opened in which the user browses to get the video file. If the file is found, it gets uploaded and starts playing. If the file is corrupt or missing the upload is failed.

Use Case id	UC-UC-ClassifyLive-002
Use case name	Preprocess video
Actors	User, Trained Model
Data	Video clip
Trigger Stimulus	Connect button is pressed
Pre-condition	Video has been successfully loaded
Assumptions	Model is well trained and can give accurate classification.
Normal flow of events	• Video is loaded. • Features are extracted. • Data augmentation is done.
Alternate flow of events	• Video is loaded however features are unable to be extracted. • Video is missing/corrupt. • Pre-processing failed.
Main success scenario	Pre-processing is successful
Post Condition	Pre-processed data is received by the trained model.

Table 3.11: Use case description for pre-process Live-stream

The above table describes the sequence of events that occur during the pre-processing of the video. The video clips are transformed in such a manner that it can be acceptable to the deep learning model. This includes changing in shape and size of the clip and converting them into tensors and then passes that data to the model. however if the video clip is corrupt or missing the pre-processing will fail.

Use Case id	UC-ClassifyLive-003
Use case name	generate alarm
Actors	User, Trained Model
Data	Label returned by trained model
Trigger Stimulus	• Video has been successfully pre-processed and data is fed into model. • Model returns the classified label.
Pre-condition	Video has been successfully pre-processed.
Assumptions	Model is well trained and can give accurate classification.
Normal flow of events	• Pre-processed data is fed to the model. • Model returns label. • Label is checked and alarm is generated.
Alternate flow of events	• Pre-processed data is fed to the model. • Model returns wrong label. • alarm generated for wrong classification or not generated when violence detected.
Main success scenario	Alarm is generated successfully.
Post Condition	None

Table 3.12: Use case description for generate alarm

The above table describes the process of showing the correct label to the user. For this to happen the video must be pre-processed and sent to the model. By assuming that the model shows accurate predictions the pre-processed data is passed to the model and the model returns the accurate label.

3.4.6 Use Case 6

The figure 3.6 shows the set of tasks a system administrator performs. These tasks are admin only and include, adding user, removing user and modifying user details. The tables 3.13 - 3.15 provide description of how those tasks are performed along with conditions for success and failure.

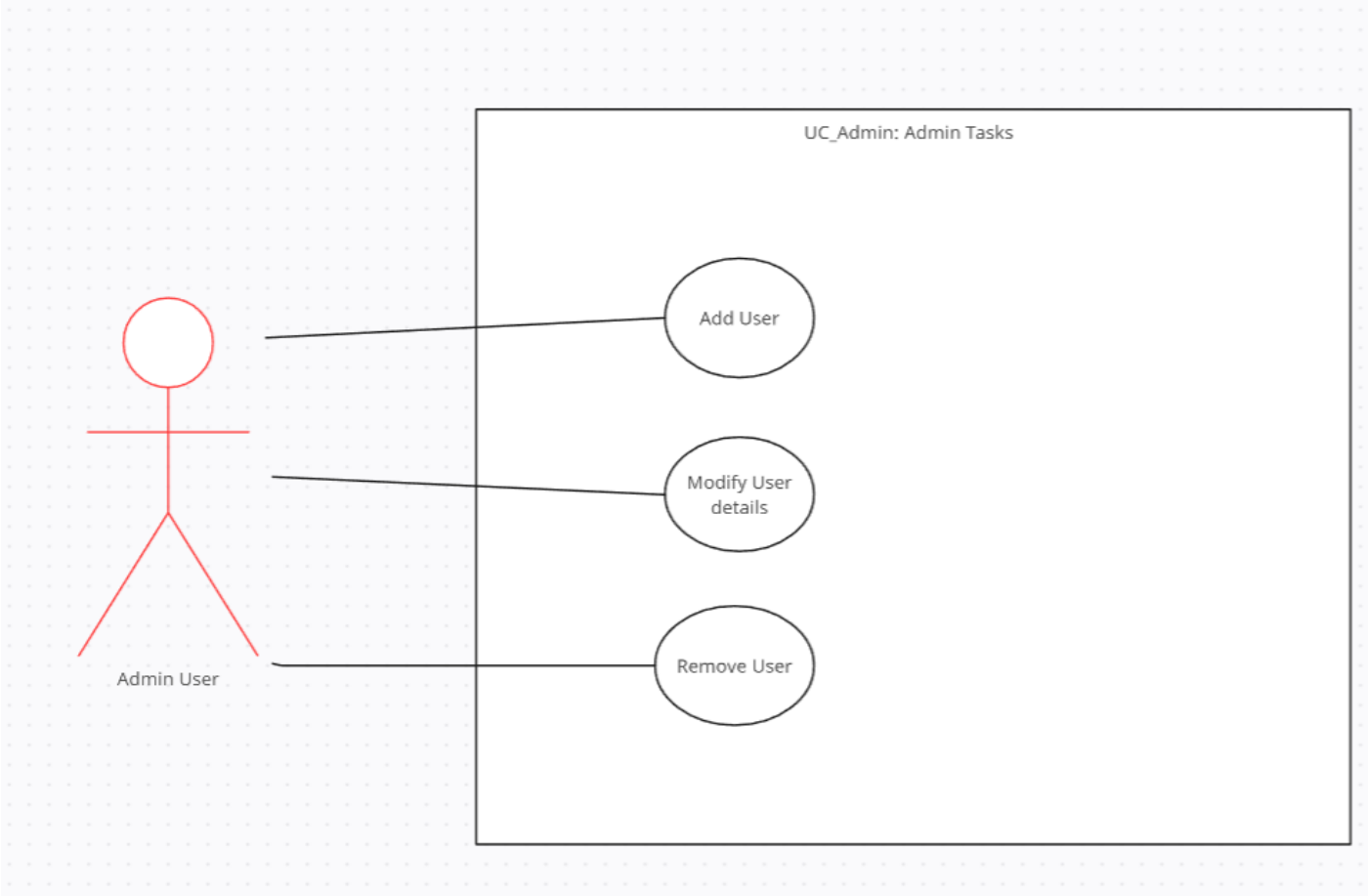


Figure 3.6: Use case for admin only tasks

Use Case id	UC-Admin-001
Use case name	Add user
Actors	User
Data	User details
Trigger Stimulus	User clicks on add user button
Pre-condition	user must be logged in to the admin pannel.
Assumptions	User is logged into the admin pannel.
Normal flow of events	• Admin clicks on add user button. • Admin enters user details and submits them. • User is created successfully.
Alternate flow of events	• Admin clicks on add user button. • Admin enters user details in wrong format and submits them. • User creation failed.
Main success scenario	Modified user table is shown.
Post Condition	None

Table 3.13: Use case description for admin add user

The table above describes the process of how the admin can create a user. For this to happen the admin must be logged in the admin panel. Along with that, the table also describes the set of steps to be taken to ensure success of the process.

Use Case id	UC-Admin-002
Use case name	Remove user
Actors	User
Data	None
Trigger Stimulus	User clicks on delete user button
Pre-condition	user must be logged in to the admin panel.
Assumptions	User is logged into the admin panel.
Normal flow of events	• Admin selects user. • Admin clicks on delete user button. • Admin confirms deletion of user • User is deleted successfully.
Alternate flow of events	• Admin selects user. • Admin clicks on delete user button. • Admin does not confirm deletion of user. • User deletion failed.
Main success scenario	Modified user table is shown.
Post Condition	None

Table 3.14: Use case description for admin remove user

The table above describes the process of how the admin can delete a user. For this to happen the admin must be logged in the admin panel. Along with that, the table also describes the set of steps to be taken to ensure success of the process.

Use Case id	UC-Admin-003
Use case name	Modify user details
Actors	User
Data	User details
Trigger Stimulus	User clicks on add user button
Pre-condition	user must be logged in to the admin panel.
Assumptions	User is logged into the admin panel.
Normal flow of events	• Admin selects user • Admin modifies user details and submits them. • User is modified successfully.
Alternate flow of events	• Admin selects user. • Admin modifies user details in wrong format and submits them. • User modification failed.
Main success scenario	Modified user table is shown.
Post Condition	None

Table 3.15: Use case description for admin modify user

The table above describes the process of how the admin can modify a user's details. For this to happen the admin must be logged in the admin panel. Along with that, the table also describes the set of steps to be taken to ensure success of the process.

Chapter 4

Design

Designing a system is important because it allows tasks to be efficiently and effectively organized and executed. A well-designed system can improve productivity, reduce errors, and make it easier to adapt to changes or updates. It also allows for better scalability and maintainability, and can help ensure that the system meets its intended purpose and user needs. Additionally, a well-designed system can also improve user experience and satisfaction.

4.1 System Architecture

Our system mainly comprises of 2 modules. First one is the back end module, in which the core functionality of the system resides. In the case of our project the core functionality is to detect violence in an incoming stream of video. The deep learning module is present in the back end of the system. The second module is the front end module. This module is the one in which user interaction occurs. The user will in the case of our project the front end will be a website. the user will send in video data to the front end and that data will be sent to the back end of the system to perform classification.

4.1.1 Deep Learning model design

Figure 4.1 describes how the DL(Deep Learning) model functions. First the model is trained on the data set and then the states of that model is saved. After that the saved states are loaded and then the video input is given to the model. The model then classifies the video as violent or non-violent.

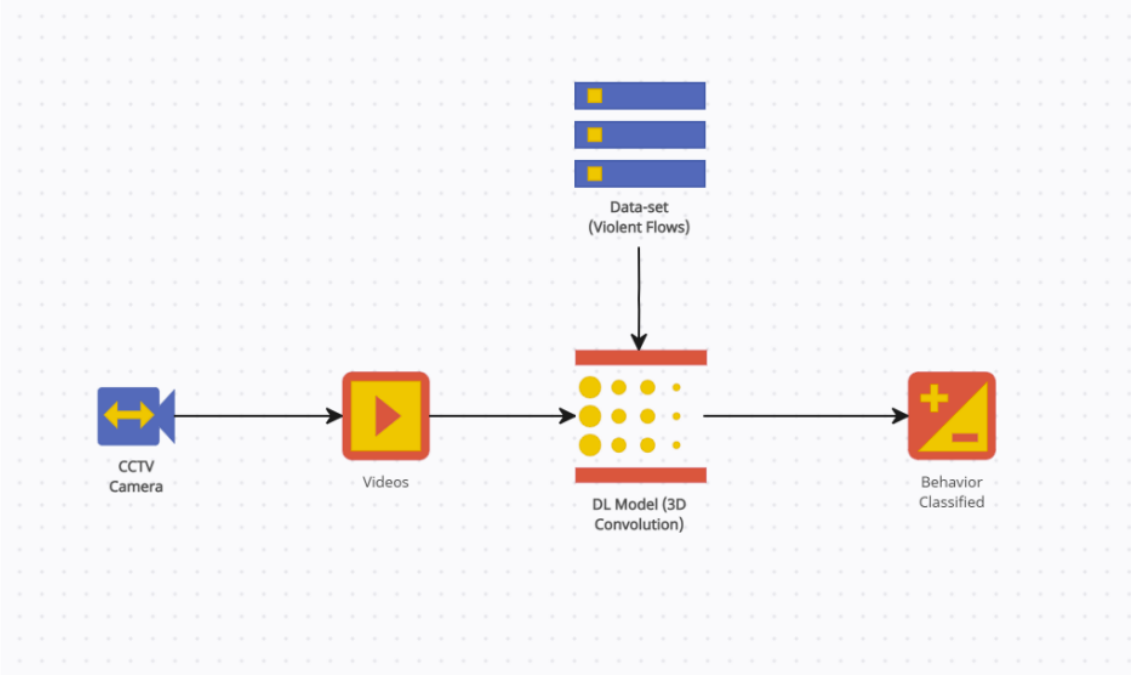


Figure 4.1: Deep learning model design

4.1.2 Front end/Web Design

The web based application will be developed using the Django framework. In figure 4.1 the basic architecture of the Django framework is highlighted and a brief explanation of each component is given below.

Model:

Model is a python object that provide a structure to the data of the application. It enables the user to add, modify, delete and query records in the database.

View:

View is a request handler function which receives http requests and http responses. The user will access the application via the view section by sending a request. After receiving the request, the views will decide what model or template will be sent to the user depending on the request.

Template:

In this section the front end of the web framework is created. When the web-page is loaded the user interacts with the template and sends the input to the views section. Then the view section displays the data via the template.

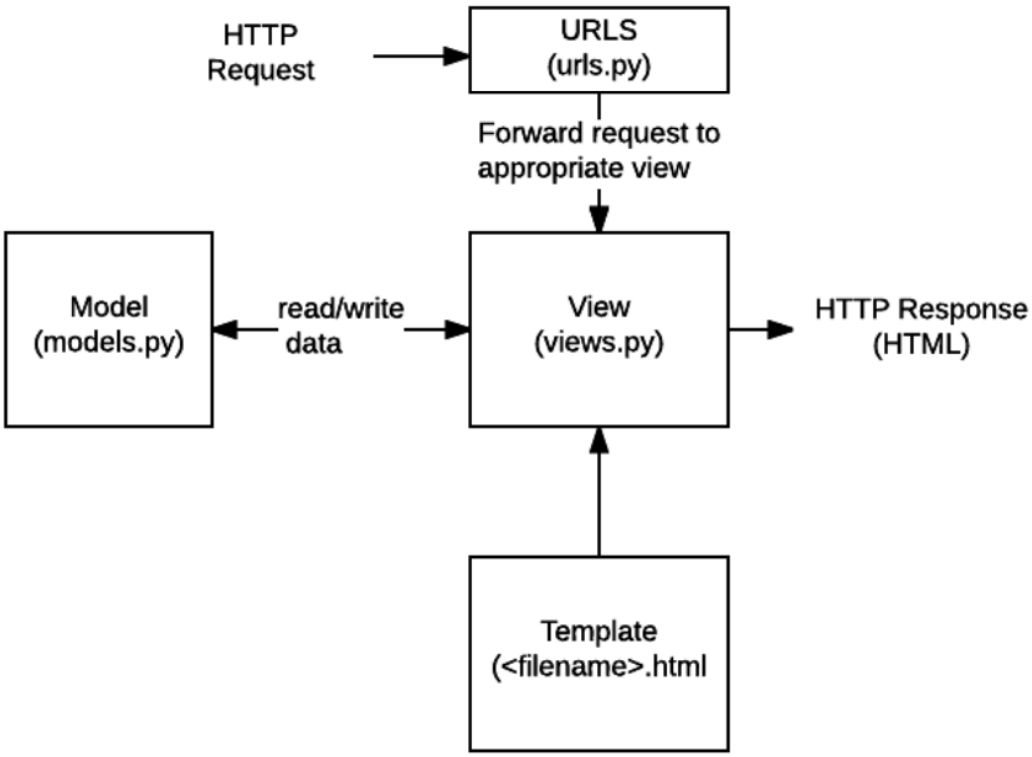


Figure 4.2: Web design

4.2 High Level Design

High level design explains the architecture that will be used to design the system. This architecture will give a general overview of the system architecture and also show, how the data will flow within the multiple components of the system, with the help of the general overview given.

4.2.1 Context Level diagram

A context level diagram shows how the external entities interact with the system. These entities typically are the users of the system. This diagram helps in defining the basic flow of data and establishing the scope of the system.
In the figures below from figure 4.3 - figure 4.5, the overview of the entire system is encompassed.

4.2.1.1 Level 0

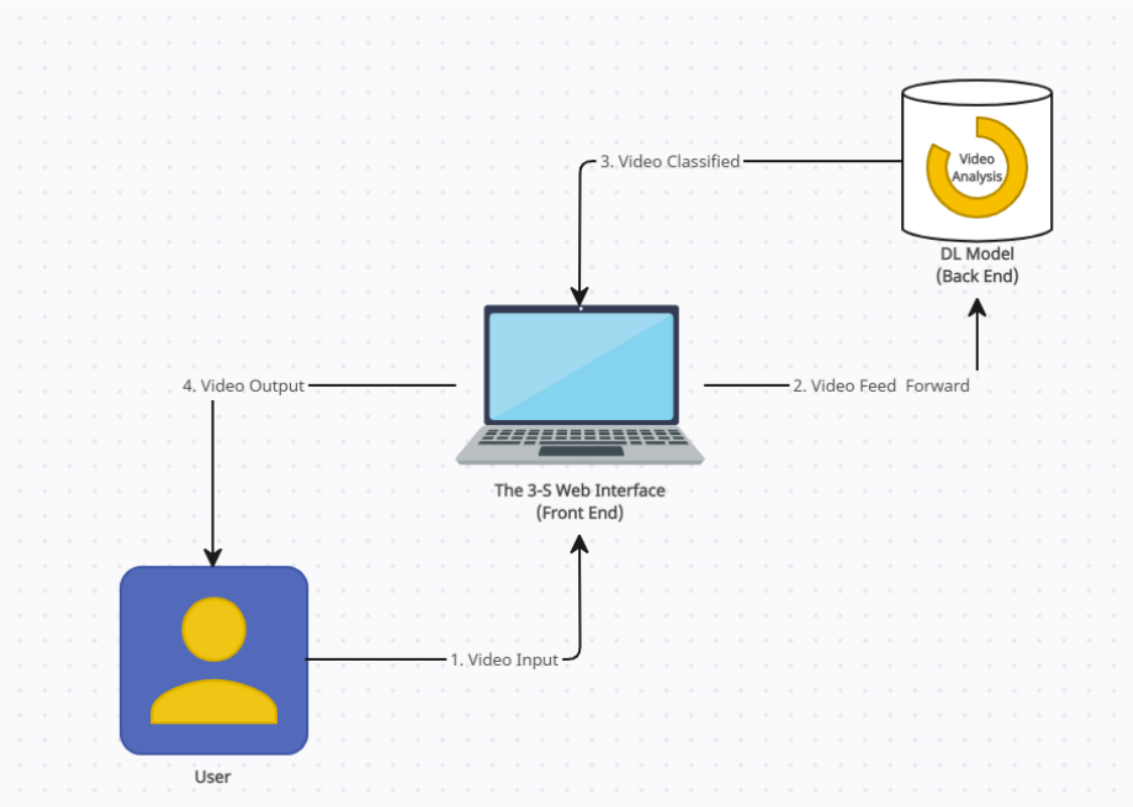


Figure 4.3: Context /(Level 0 data flow) Diagram for video classification process.

In figure 4.3 it is described how the data will flow in the homepage when the user will upload a clip to the front end. Then that clip will be sent to the DL model which is the back end. The DL model will classify the video/clip and then return the label to the front end, which will then display the results to the user. The model used for the project is the 3D-DenseNet(Densely Connected Convolutional Networks), which has dense and transition blocks that help reduce parameters and as a result chances of overfitting are reduced.

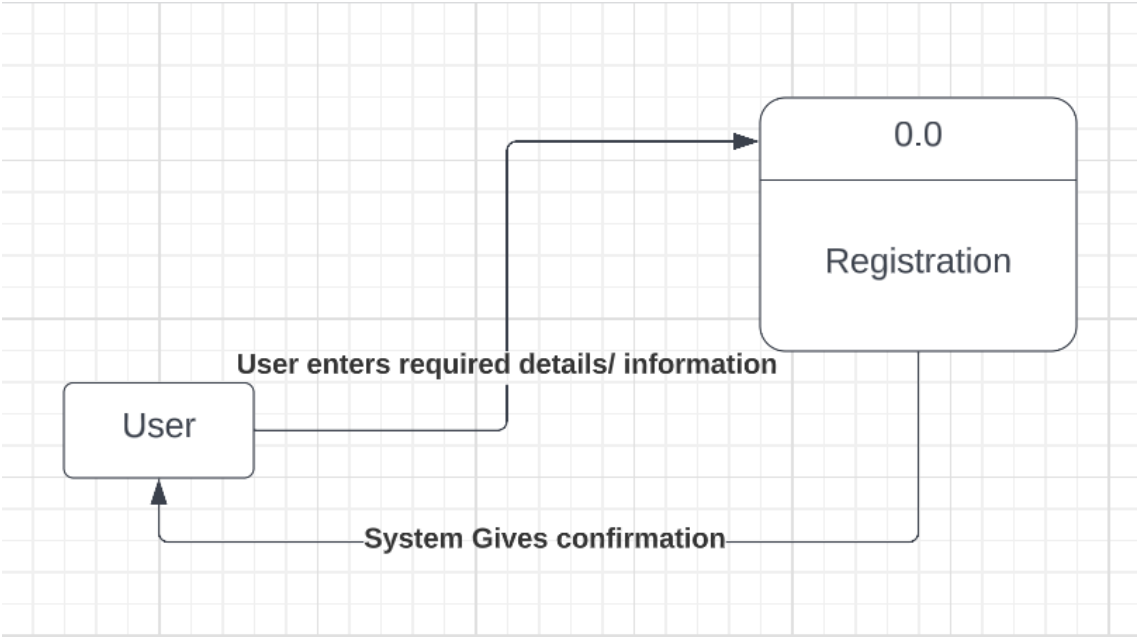


Figure 4.4: Context /(Level 0 data flow) Diagram for user registration

In figure 4.4 it is shown how the user will interact with the system during the process of registration. The user will enter the required details and then the system will give confirmation to the user whether registration is successful or not.

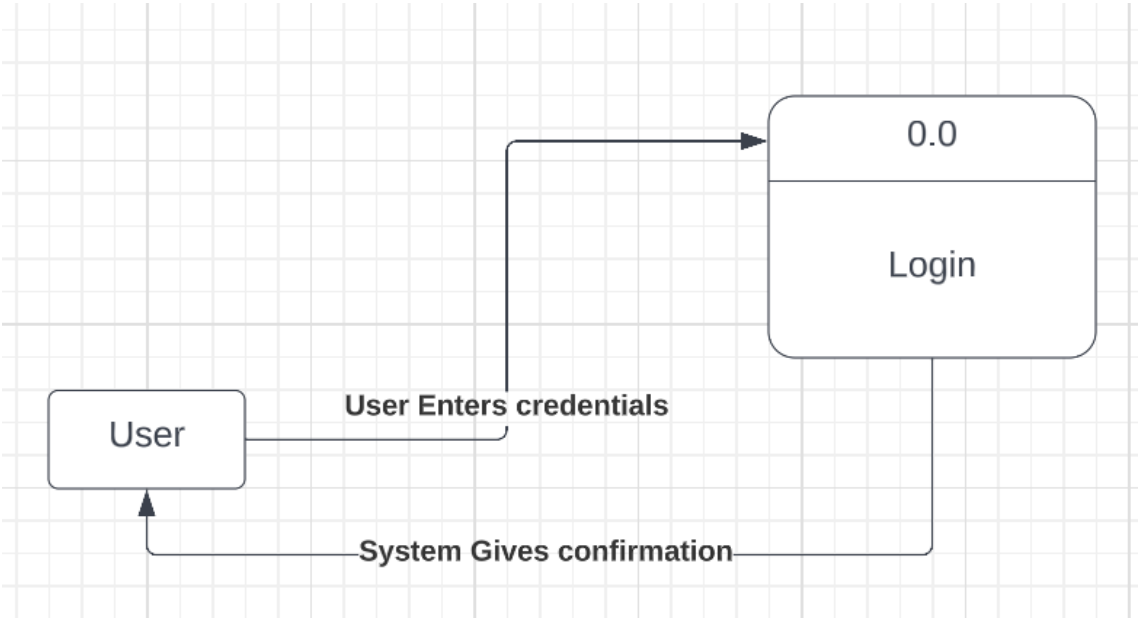


Figure 4.5: Context /(Level 0 data flow) Diagram for user login

In figure 4.5 it is shown how the user will interact with the system during the process of

login. The user will enter the his credentials and then the system will give confirmation to the user whether login is successful or not.

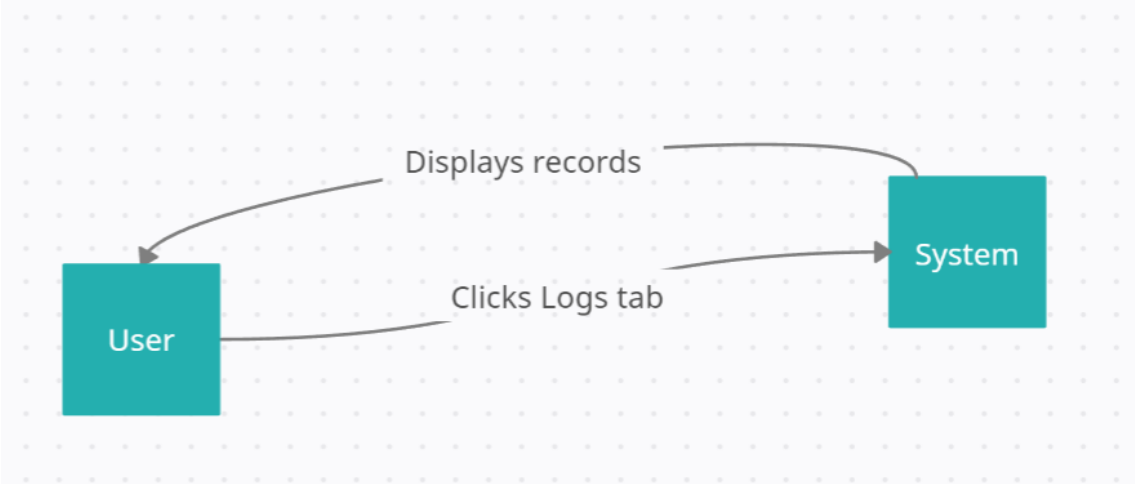


Figure 4.6: Context /(Level 0 data flow) Diagram for display logs

In figure 4.6 it is shown how the user will interact with the system during the process of displaying logs to user. The user will click on the logs tab and then the system will give display logs to the user.

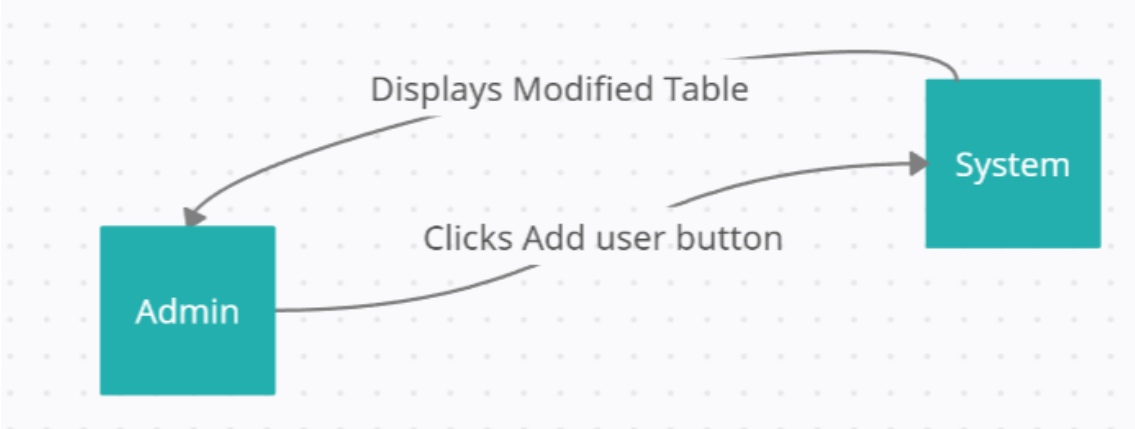


Figure 4.7: Context /(Level 0 data flow) Diagram for admin add user

In figure 4.7 it is shown how the admin user will interact with the system during the process of adding users to the system. When user is finished with interaction the system will confirm whether the user addition is successful or not.

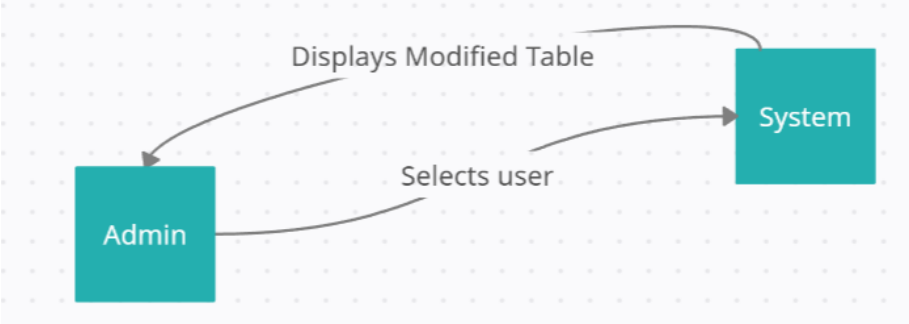


Figure 4.8: Context /(Level 0 data flow) Diagram for admin modify user details

In figure 4.8 it is shown how the admin user will interact with the system during the process of modifying users of the system. When user is finished with interaction the system will confirm whether the user modification is successful or not.

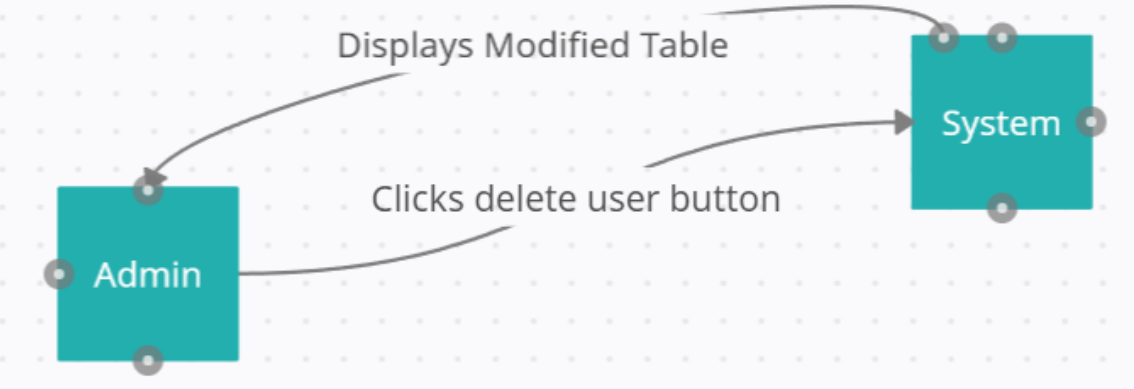


Figure 4.9: Context /(Level 0 data flow) Diagram for admin add user

In figure 4.9 it is shown how the admin user will interact with the system during the process of removing users from the system. When user is finished with interaction the system will confirm whether the user deletion is successful or not.

4.2.1.2 Level 1

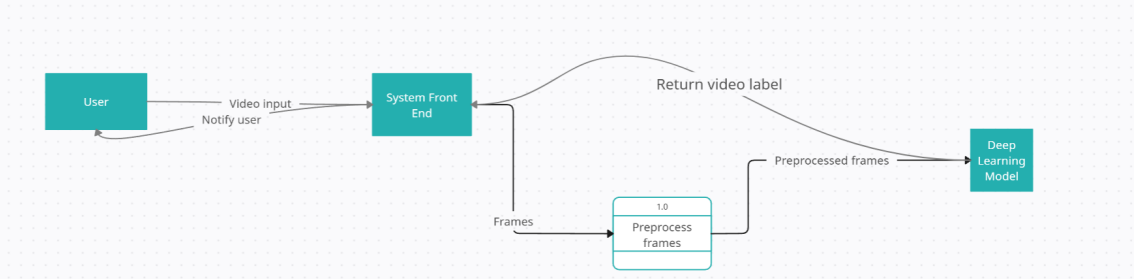


Figure 4.10: Context /(Level 1 data flow) Diagram for Live video classification process.

In figure 4.10 a detailed description is given on how the data will flow when the video input will be taken on a webcam. Initially the system will pre-process the frames and create a small clip. After pre-processing that clip will be sent to the DL model which is in the back end. The DL model will classify the video/clip and then do 2 things. First it will raise an alarm and secondly it will create a red bounding box around the borders of the frames.

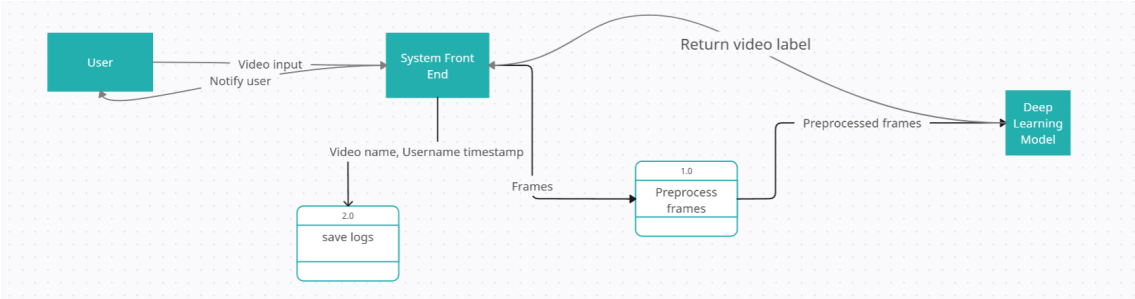


Figure 4.11: Context /(Level 1 data flow) Diagram for recorded video classification process.

In figure 4.11 it is described in further how the data will flow in the homepage when the user will upload a clip to the front end.Initially the system will pre-process the frames and create a small clip. Then that clip will be sent to the DL model which is in the back end. The DL model will classify the video/clip and then return the label to the front end, which will then display the results to the user. After that through the system front-end, a log will be maintained on the timestamp, username and video name in which violence is detected.

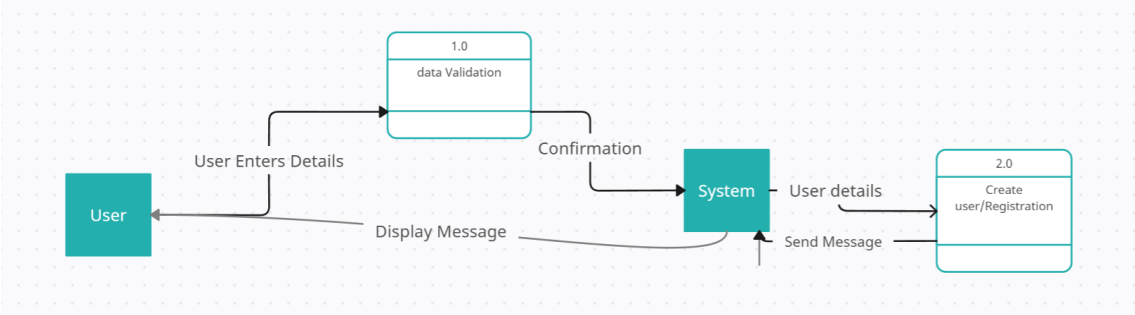


Figure 4.12: Context /(Level 1 data flow) Diagram for user registration

In figure 4.12 it is shown in further detail how the user will interact with the system during the process of registration. The user will enter the his details. After entering the details, the system will validate those details. Then the system will give confirmation to the user whether registration is successful or not.

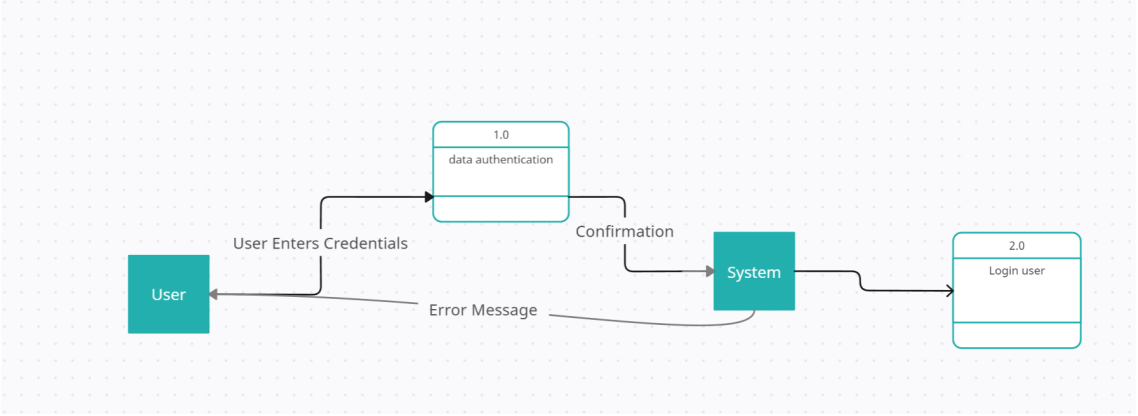


Figure 4.13: Context /(Level 1 data flow) Diagram for user login

In figure 4.13 it is shown in further detail how the user will interact with the system during the process of login. The user will enter the his credentials and then the system will give confirmation to the user whether login is successful or not.

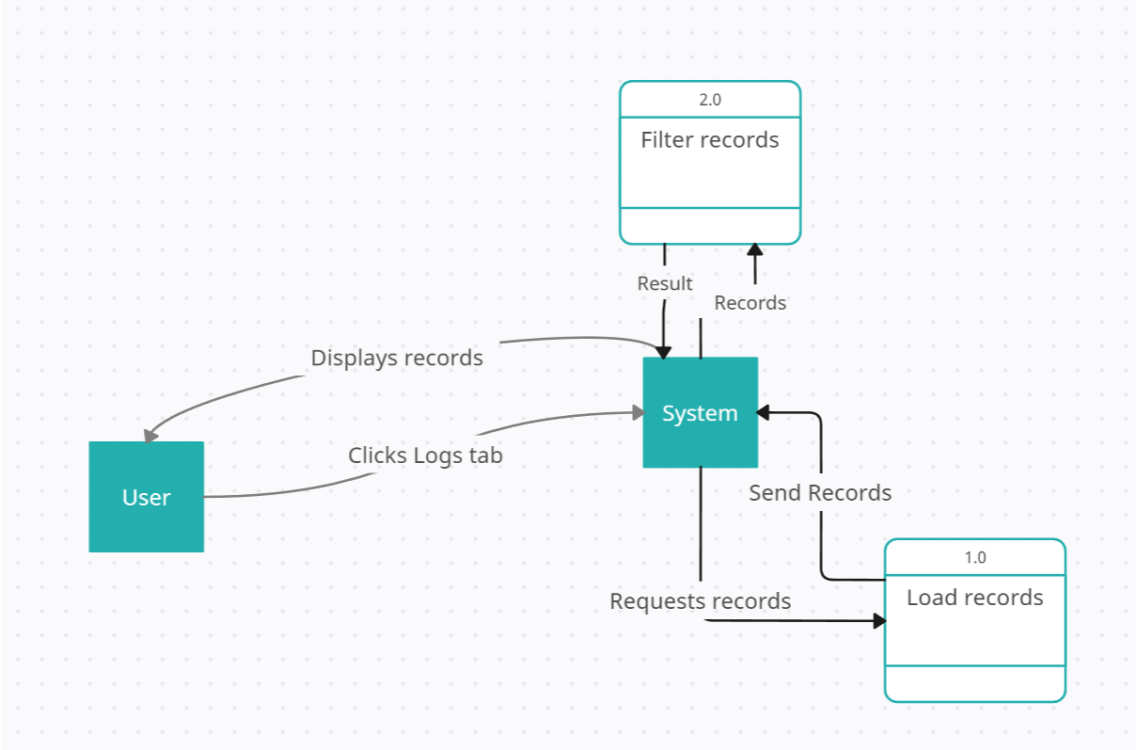


Figure 4.14: Context /(Level 1 data flow) Diagram for displaying logs

In figure 4.14 it is shown in further detail, that what are the series of activities that will take place when displaying logs to user. The user will click on logs tab, after that the system will load the logs file and display only the data which is related to the user by through filtration.

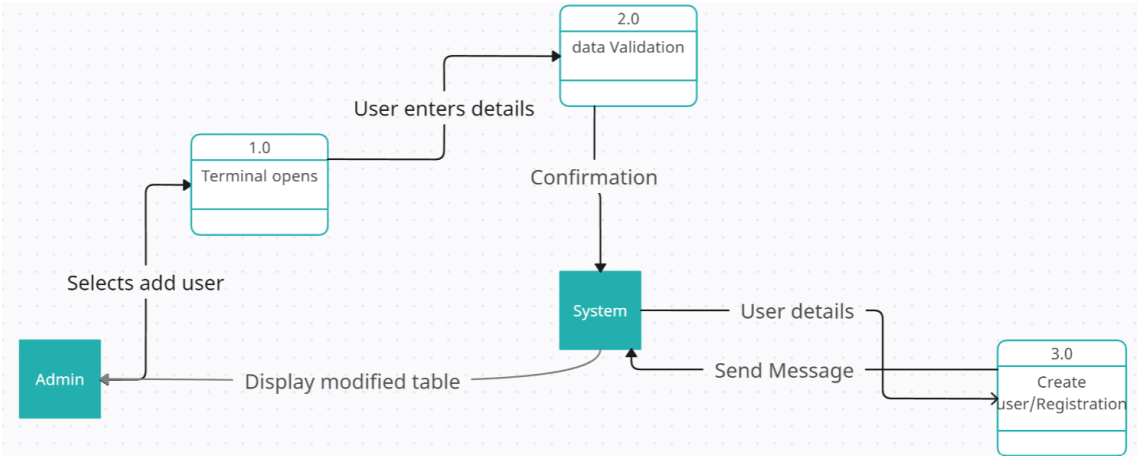


Figure 4.15: Context /(Level 1 data flow) Diagram for admin add user

In figure 4.15 it is shown in greater detail how the admin user will interact with the system during the process of adding users to the system. The admin will click on the add user button and then the system will open terminal in which he will enter details and then submit them. After that the modified user table will be displayed.

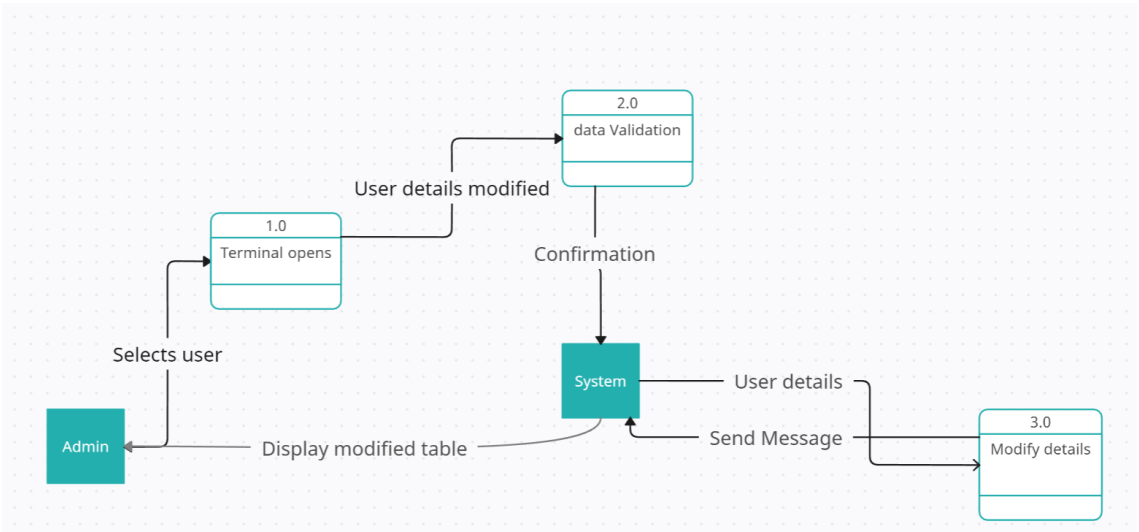


Figure 4.16: Context /(Level 1 data flow) Diagram for admin modify user details

In figure 4.16 it is shown in greater detail how the admin user will interact with the system during the process of modifying users of the system. The admin select user and

then the system will open terminal in which he will modify details and then submit them. After that the modified user table will be displayed.

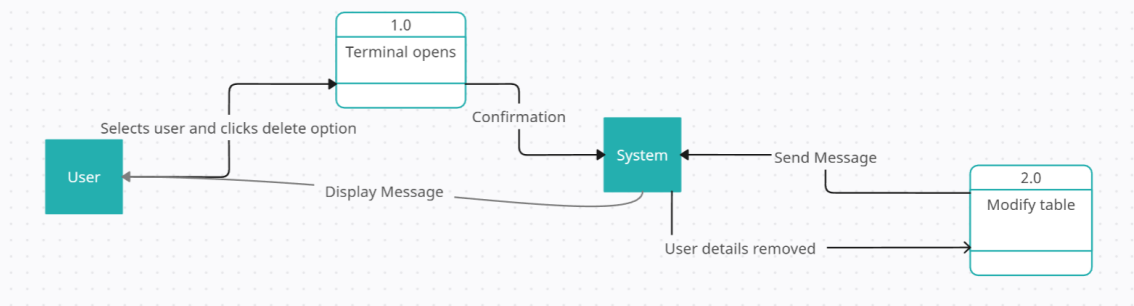


Figure 4.17: Context /(Level 1 data flow) Diagram for admin add user

In figure 4.17 it is shown in greater detail how the admin user will interact with the system during the process of removing users from the system. The admin will select user and, click on the delete user button and then the system will open terminal in which he will give confirmation. After that the modified user table will be displayed.

4.3 Low Level Design

Just like the high level design, the low level design also explains the architecture that will be used to design the system. however, this architecture will give a much more detailed description of the system architecture and also show, how the data will flow within each component and sub-component of the system.

4.3.1 Activity Diagram

4.3.1.1 User Activity

The activity diagram in figure 4.18 models the sequential flow of work that is carried out throughout the system, showing how the user interacts with the system at each stage from registration to login, then to uploading the video clip in recorded tab. It also includes the flow of work for live classification and loading video logs

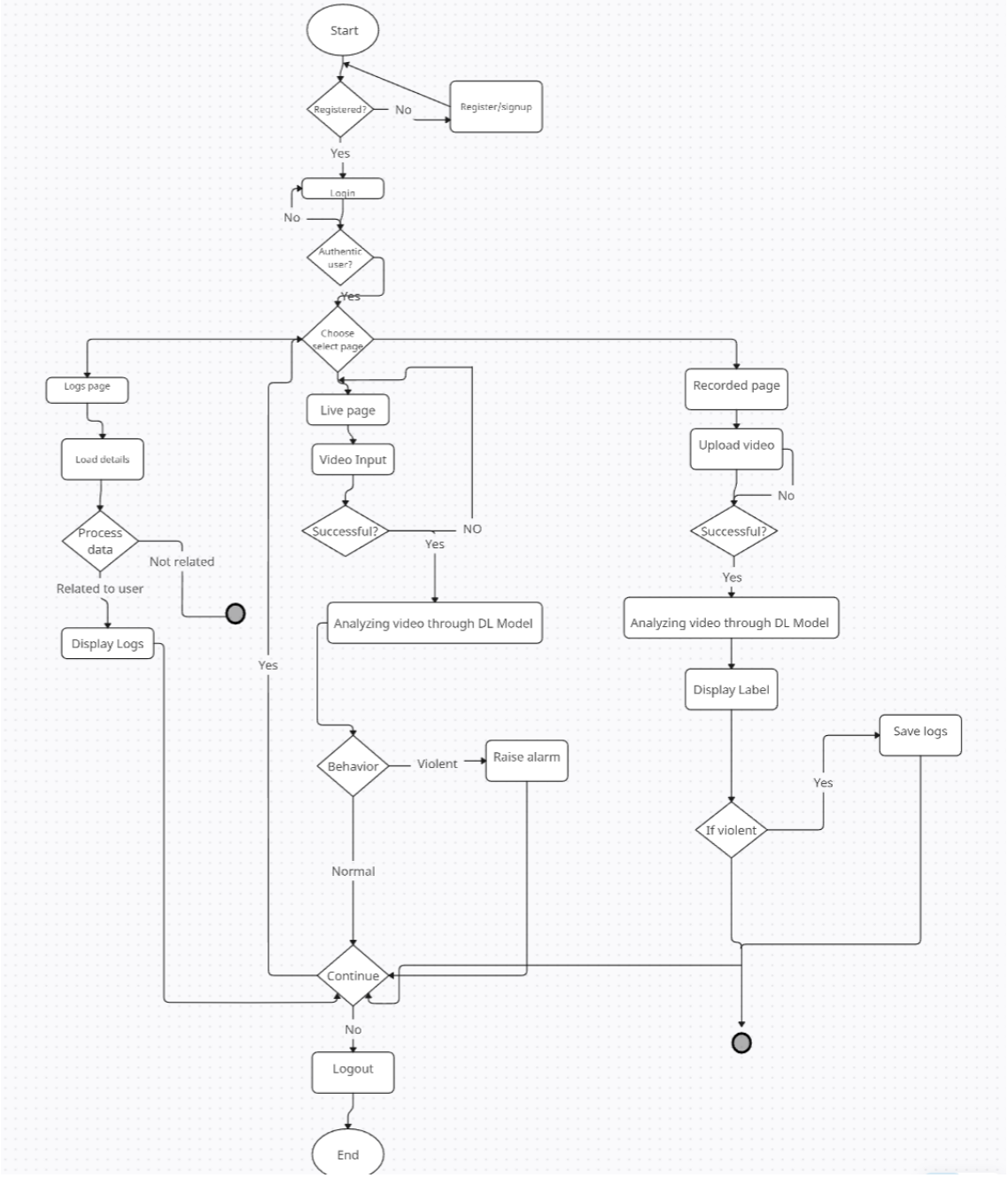


Figure 4.18: Activity Diagram

4.3.1.2 Admin only Activity

The activity diagram in figure 4.19 shows the flow of work that is carried out on the activities only a system admin is permitted to perform. It includes adding, deleting user along with modifying user details. The admin user can also perform normal user activities.

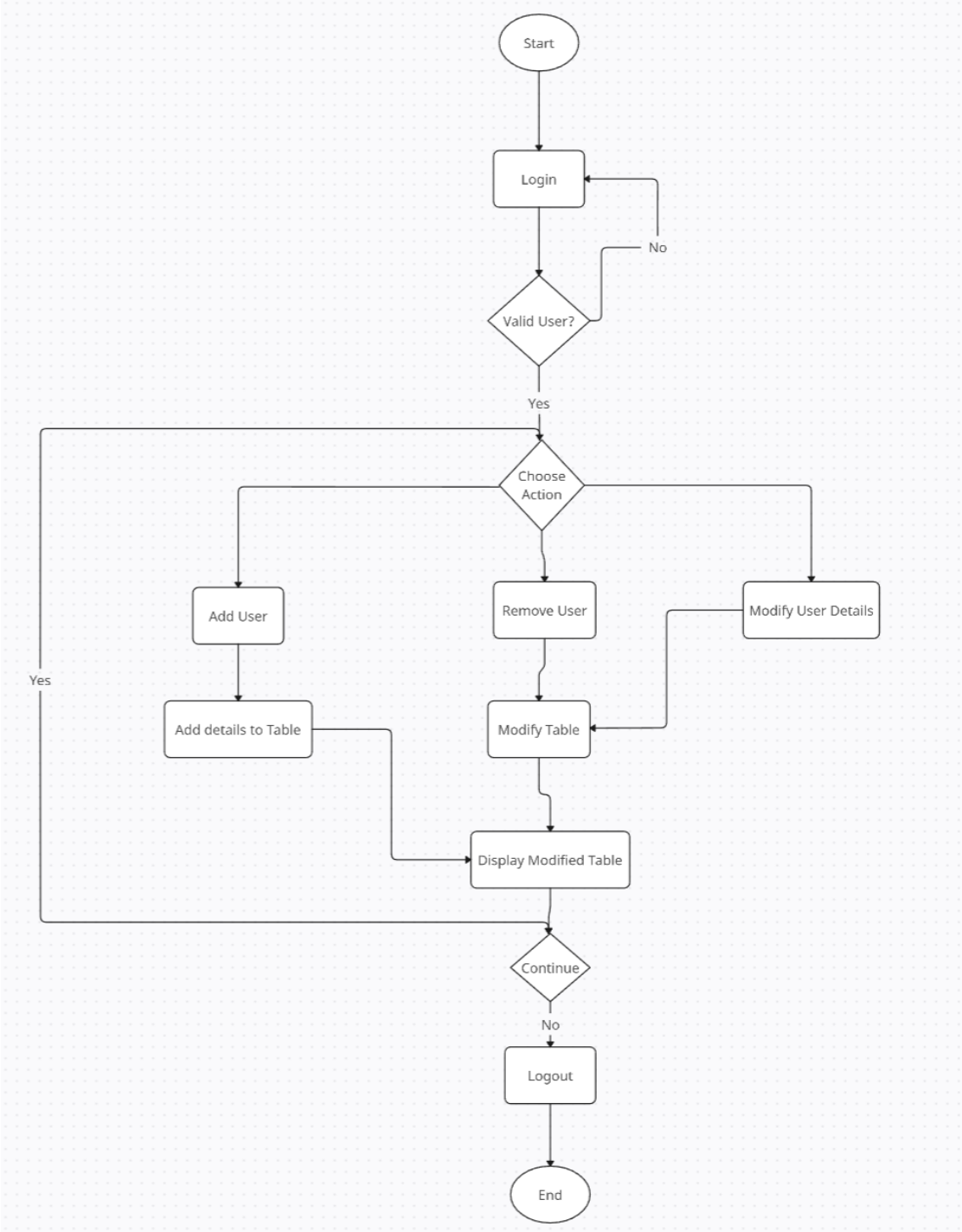


Figure 4.19: Activity Diagram for admin only activities

4.3.2 Sequence diagram

4.3.2.1 User View

The sequence diagram in figure 4.20 shows how different parts of the systems interact with each other when the system is in action. It also shows at what time data is transferred from one part to another. Along with that it shows the life-cycle of each part, when the life-cycle begins and when it ends.

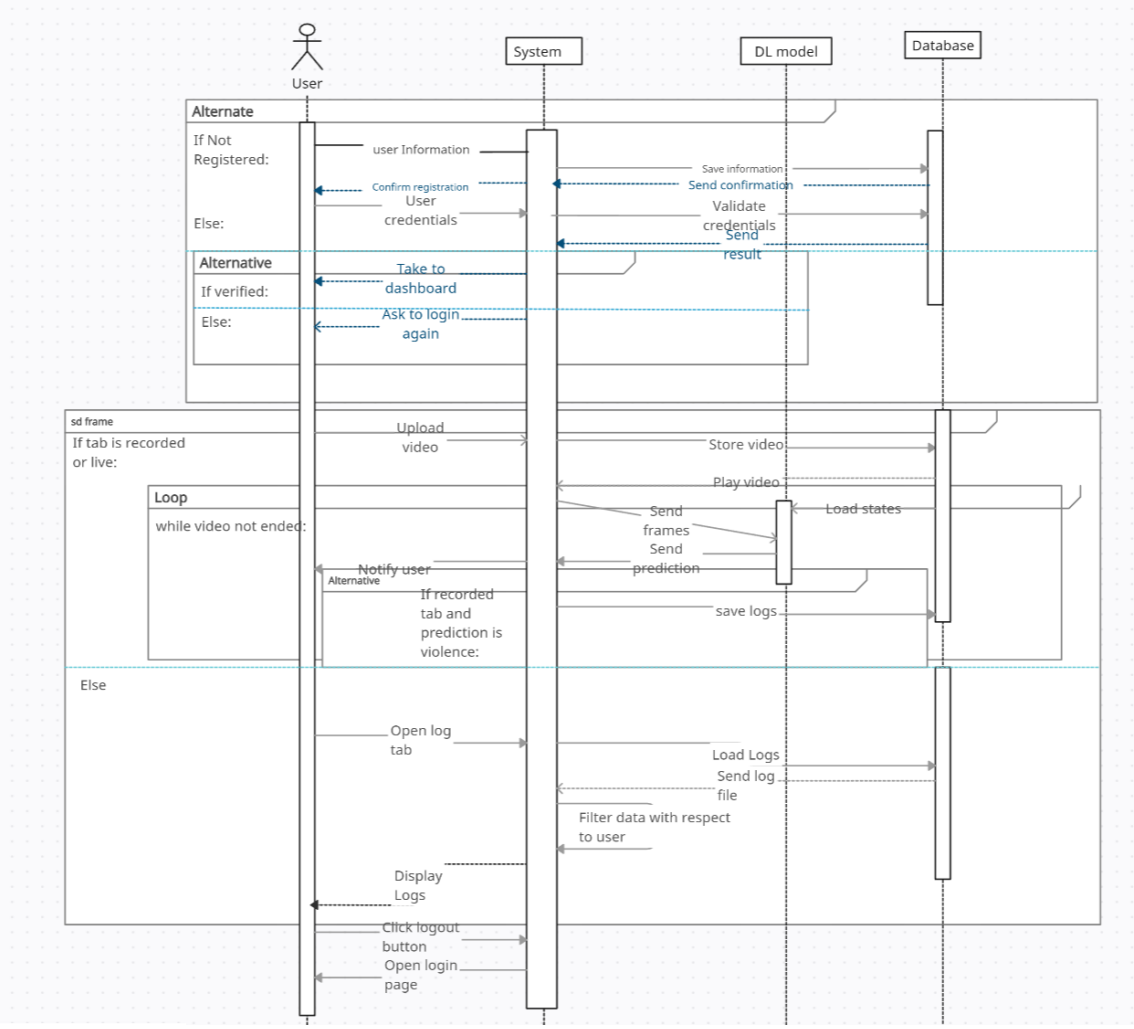


Figure 4.20: Sequence Diagram

4.3.2.2 Admin only View

The sequence diagram in figure 4.21 shows the interactions an admin has with the system with the system for performing the ***admin only*** set of activities related to the system. The admin can also access the user view of the system however the normal user cannot access the admin view.

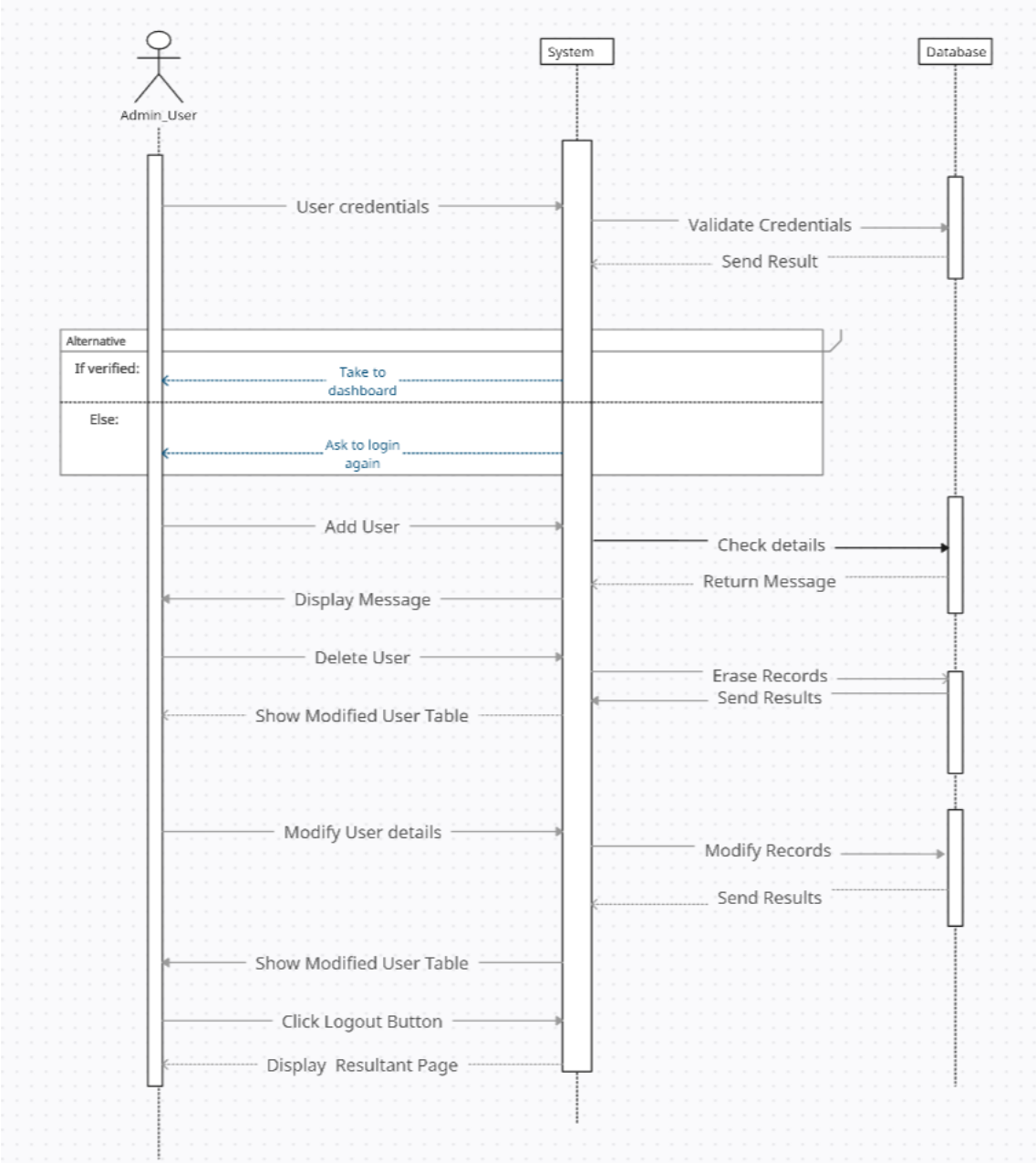


Figure 4.21: Sequence Diagram for Admin only activities

4.4 Database Design

Django's built in user model will be used for the purpose of maintaining a user database.

4.5 GUI Design

The graphical user interface design is done on the Figma application[20]. This application designs prototypes of the web interface. Which help in giving the general idea of how the

users will interact with the system before developing the actual web interface.

4.5.1 Prototype for login page

In figure 4.22 the prototype of the login page is shown. Which help in giving the general idea of how the users will interact with the system during login process.

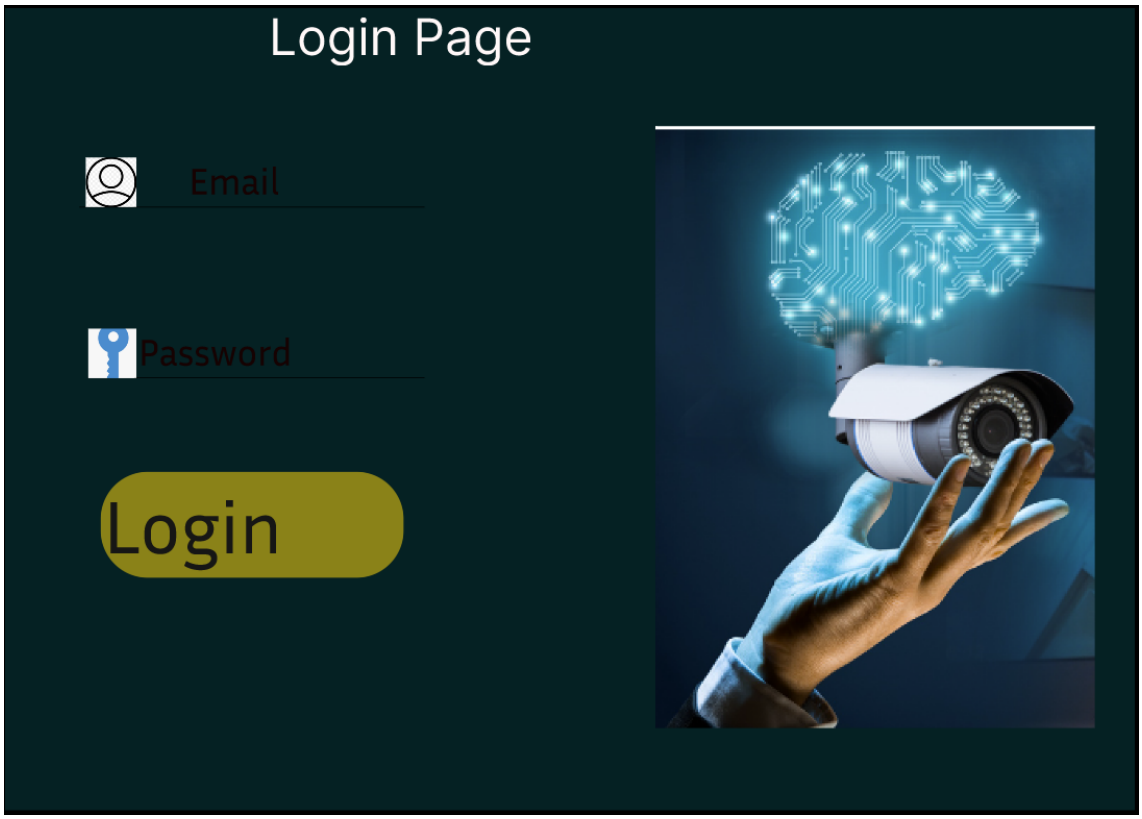


Figure 4.22: Design prototype for login page

4.5.2 Prototype for homepage

In figure 4.23 the prototype of the homepage is shown. Which help in giving the general idea of how the users will interact with the system on the homepage.

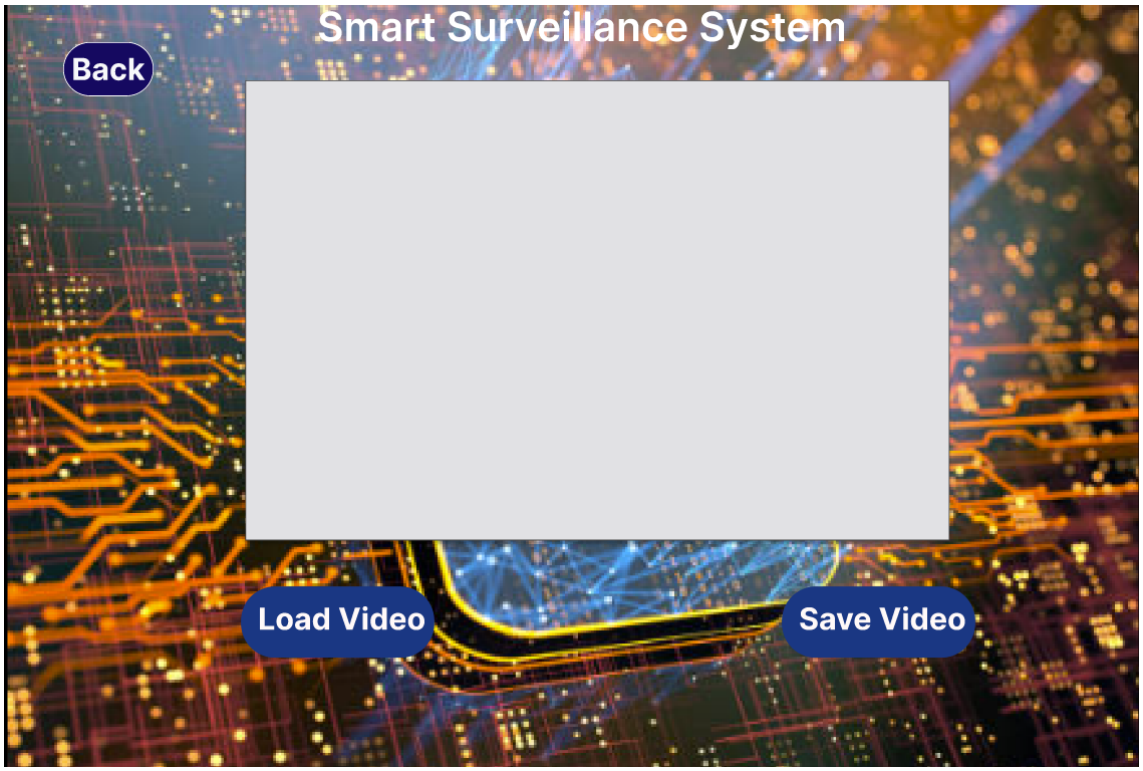


Figure 4.23: Design prototype for homepage

Chapter 5

System Implementation

5.1 System Architecture

5.1.1 System Back-end:

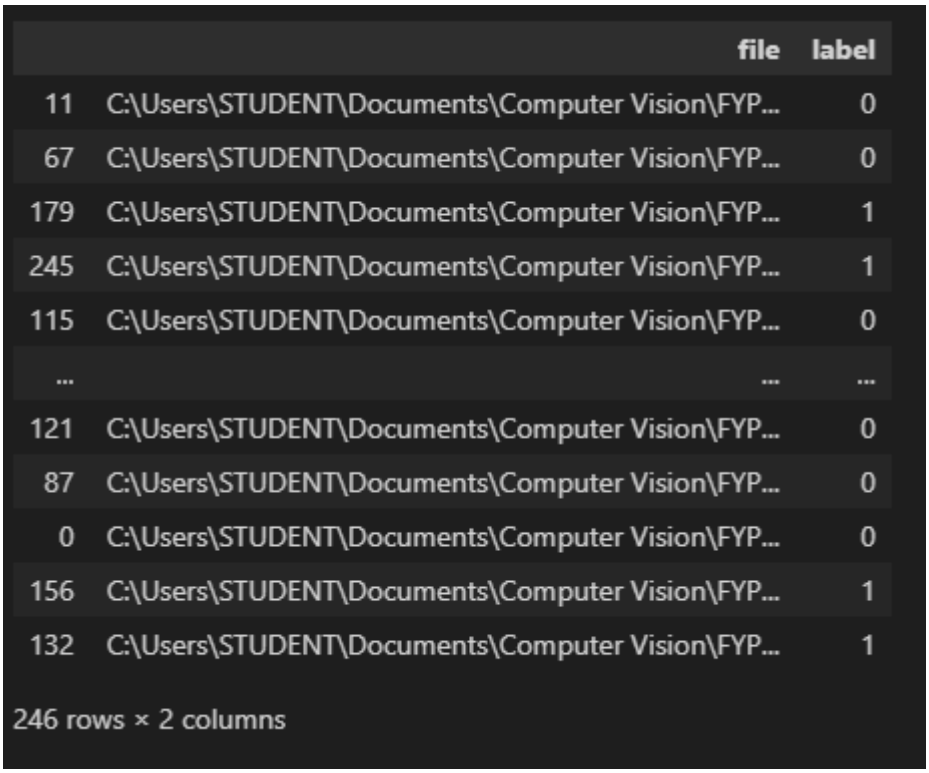
For the back-end of the system, we first train our deep learning model on the data-set and then deploy the model on the Django framework.

5.1.1.1 Training Phase

The dats-set used for this project is violent flows. It contains 246 videos ranging from 1-6 seconds. 123 videos are containing violent and 123 videos are containing non violent actions. The data-set is divided in 5 folders, and those 5 folders are further sub-divided into folders containing violent and non-violent videos.

- Firstly, we take the path of each video and make a list of those paths.
- Secondly, we create a list of labels (0 for non-violence and 1 for violence) of the same length of the video-path list.
- Then we make a data-frame(Df) by combining the list of paths in one column and the label list in the other.

After creation of the data-frame we split it into training and testing data-frames and then concatenate both the data-frames back together. Then we proceed towards the creation of a labeled video data-set which will be done using PyTorch[21].
The figure 5.1 shows the data frame created using pd(pandas) library containing a path to the file and label of the video (O for Non-violent/Normal and 1 for Violent videos).



	file	label
11	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
67	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
179	C:\Users\STUDENT\Documents\Computer Vision\FYP...	1
245	C:\Users\STUDENT\Documents\Computer Vision\FYP...	1
115	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
...	...	...
121	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
87	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
0	C:\Users\STUDENT\Documents\Computer Vision\FYP...	0
156	C:\Users\STUDENT\Documents\Computer Vision\FYP...	1
132	C:\Users\STUDENT\Documents\Computer Vision\FYP...	1

246 rows × 2 columns

Figure 5.1: Df created by combining path and label of video.

Before creating the labelled video data-set, first we must define the augmentations that are to be used in the pre-processing. The augmentations used are:

- *ApplyTransformToKey:* This specifies on which element of the video the transformations/ augmentations will be applied. In our case the key is set to video, meaning the transformations will be performed on the video only.
- *UniformTemporalSubsample(N):* This enables us to extract N number of frames from 1 second in a video. For instance, if there is a 3 second video and N = 16, then a total of 48 frames will be extracted.
- *Lambda:* Applies user defined transformation to the video input. In our case it normalizes the range of the pixels from 0-255 to 0-1.
- *Normalize:* Defines the mean and standard deviation which is to be applied in an input clip. Mean = (0.485, 0.456, 0.406), standard deviation = (0.229, 0.224, 0.225).
- *Resize:* changes the size of the frames. In our case width = 128 and height = 171.
- *RandomResizedCrop:* Resizes the input frames by cropping them through scale=(0.8,1) and ratio=(3/4,4/3) at the desired size, which is 112x112 pixels.
- *RandomHorizontalFlip:* Flips the input frame in horizontal direction with respect to a set probability. In our case the probability is 0.5.

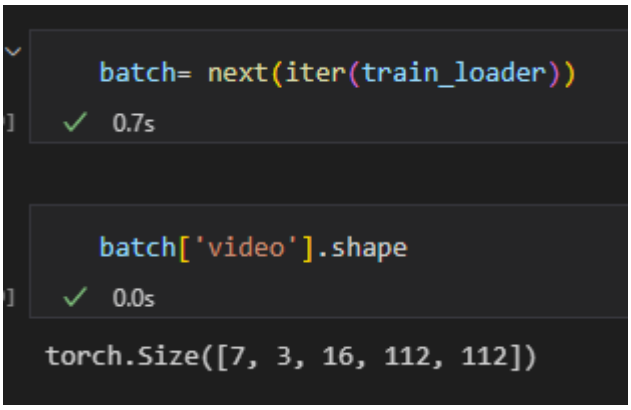


Figure 5.2: Shape of tensor after preprocessing the video frames

All the above mentioned augmentations/ transformations are brought together by the *compose* library. The compose library not only brings the augmentations together, but also sets the order in which they are to occur. After composing our transformations/augmentations, we proceed towards defining our labeled video dataset. We use the Labeled-video-dataset and make-clip-sampler libraries from pytorchvideo.data. As inputs we give the data-frame, make-clip-sampler in which the length and type of the clip is defined, and the transformations to be used. Then we defined the PyTorch dataloader, which provides an iterable on the labeled video dataset. The parameters of the dataloader are, the data-set, batch size which is the number of clips to be passed to the model at a time, numworkers which is the number of sub-processes to be working on the training and validation. Lastly we have pin memory through which it is decided whether the dataloader is passed to the gpu or not.

In figure 5.2 the final shape of the video tensor is shown. In this shape the tensor will be fed to the model for training and validation. 7 is the batch size of tensors, 3 is the number of dimensions, 16 is number of frames per tensor and 112X112 is the size of each frame. The training will be done using the 5-folds cross validation technique. In this technique the data-set is divided into 5 parts, 4 of which are used as training data-set and 1 is used as testing/validation data-set. Then the training loop is run until the defined epochs. Then in the next fold the testing part will be merged into training data-set and another portion will be used for testing. This will continue until all 5 parts have been shuffled into the testing data-set. Hence the labeled video data-sets and the dataloaders will be redefined when the fold changes. This is done by passing the indices of the data-frame that to the labeled-video-dataset library in the training loop.

In the training loop, we call the train-epoch and val-epoch functions which return the training and validation accuracy, loss per each epoch. The optimizer used is Adam with learning rate set to 1e-4 and weight decay set to 1e-3. the loss function used is BCE-WithLogitsLoss(). The train-epoch function takes the model, device(gup/cpu), dataloader, loss function and optimizer as input. To calculate the accuracy, BinaryAccuracy from

torchmetrics.classification is imported. Then the dataloader is enumerated in a for loop. From the dataloader the video input is taken and passed through the model and the output is compared with the assigned labels. Through the comparison the training accuracy and loss is computed. The same procedure is repeated in the valid-epoch function.It takes model, device(gup/cpu), dataloader, loss function as input parameters.

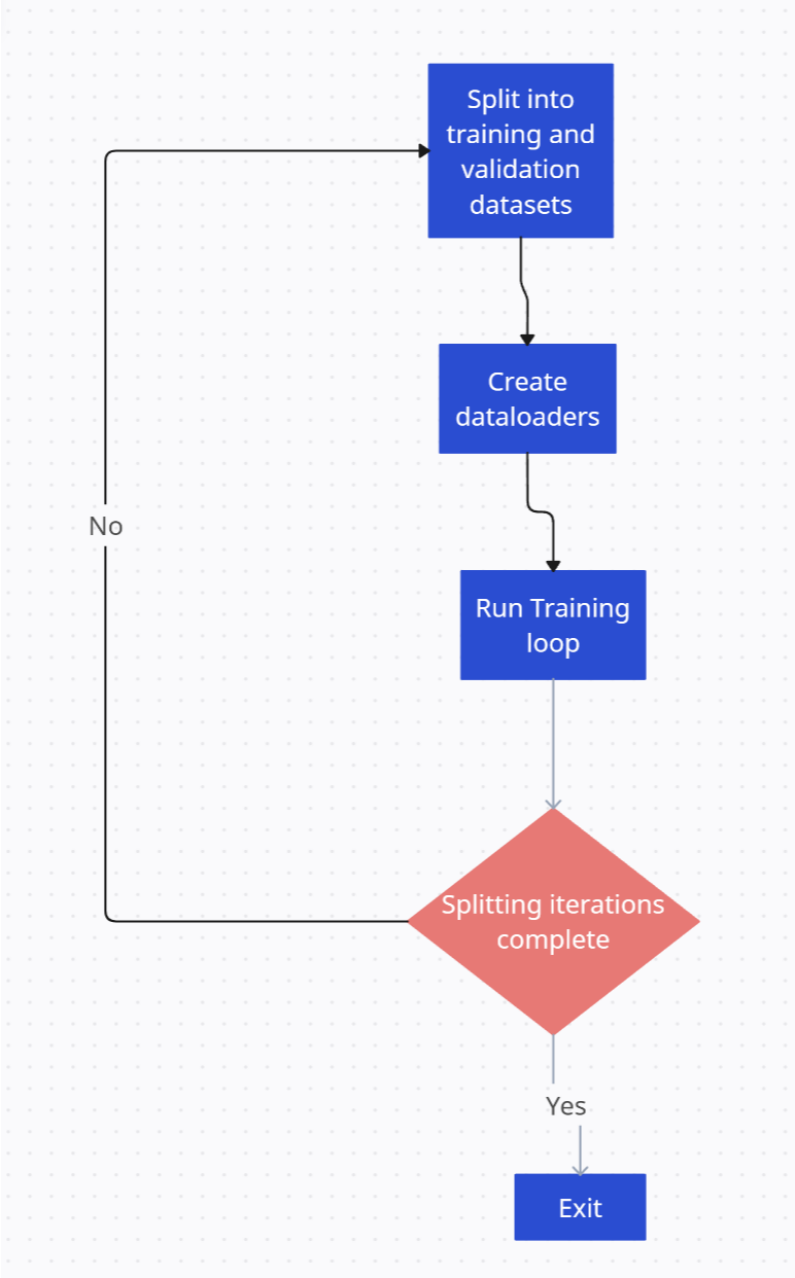


Figure 5.3: Training Overview

The deep learning model used for training and classification is 3D-DenseNet. The DenseNet architecture consists of a sequence of dense blocks, where each dense block is made up of of several densely connected layers, followed by a transition layer that reduces

the number of feature maps before entering the next dense block. The code defines three classes which are, DenseLayer, DenseBlock, and Transition that are used to construct the overall architecture of the DenseNet model. The DenseNet class is the main class that defines the complete architecture of the model.

DenseLayer class: This class defines a single layer of the dense block. It takes as input the number of input features, growth rate, batch normalization size, and dropout rate. It initializes the layer by adding batch normalization, ReLU activation, 1x1 convolution, batch normalization, ReLU activation, and 3x3 convolution to the sequential module. It also defines a forward function that concatenates the input with the output of the dense layer and applies dropout if the drop rate is greater than zero.

DenseBlock class: This class defines a dense block consisting of several DenseLayer layers. It takes as input the number of layers, number of input features, batch normalization size, growth rate, and dropout rate. It initializes the dense block by adding DenseLayer layers to the sequential module.

Transition class: This class defines a transition layer that reduces the number of feature maps by half. It takes as input the number of input features and the number of output features. It initializes the transition layer by adding batch normalization, ReLU activation, 1x1 convolution, and 3D average pooling to the sequential module.

DenseNet class: This class defines the complete architecture of the DenseNet model. It takes several input parameters such as number of input channels, size and stride of the first convolutional layer, number of blocks in the dense layers, number of initial features, batch normalization size, growth rate, dropout rate, and number of output classes. It initializes the model by adding the first convolutional layer, batch normalization, and ReLU activation to the sequential module. It then adds dense blocks and transition layers to the sequential module. Finally, it adds batch normalization and linear layer to the sequential module. It also initializes the weights of the model using the Kaiming initialization method and defines a forward function that passes the input through the layers of the model and returns the output.

Parameter	**Value**
No. of input channels	3
conv1 size	7
conv1 stride	1
Block configuration	(6, 12, 24)
No. of initial features	64
Batch normalization size	4
Growth rate	32
Dropout rate	0
No. of output classes	1

Table 5.1: Parameters of DenseNet class with values

```
The model will be running on cuda device
Fold 1
Epoch:1/10 AVG Training Loss:0.720 AVG Test Loss:0.611 AVG Training Acc 0.74  AVG Test Acc 0.84
Epoch:2/10 AVG Training Loss:0.585 AVG Test Loss:0.731 AVG Training Acc 0.80  AVG Test Acc 0.80
Epoch:3/10 AVG Training Loss:0.562 AVG Test Loss:0.987 AVG Training Acc 0.82  AVG Test Acc 0.75
Epoch:4/10 AVG Training Loss:0.531 AVG Test Loss:0.675 AVG Training Acc 0.84  AVG Test Acc 0.81
Epoch:5/10 AVG Training Loss:0.540 AVG Test Loss:0.477 AVG Training Acc 0.84  AVG Test Acc 0.84
Epoch:6/10 AVG Training Loss:0.514 AVG Test Loss:0.644 AVG Training Acc 0.81  AVG Test Acc 0.83
Epoch:7/10 AVG Training Loss:0.458 AVG Test Loss:0.848 AVG Training Acc 0.85  AVG Test Acc 0.83
Epoch:8/10 AVG Training Loss:0.487 AVG Test Loss:0.446 AVG Training Acc 0.81  AVG Test Acc 0.86
Epoch:9/10 AVG Training Loss:0.460 AVG Test Loss:0.662 AVG Training Acc 0.85  AVG Test Acc 0.75
Epoch:10/10 AVG Training Loss:0.480 AVG Test Loss:0.768 AVG Training Acc 0.83  AVG Test Acc 0.78
```

Figure 5.4: Training history

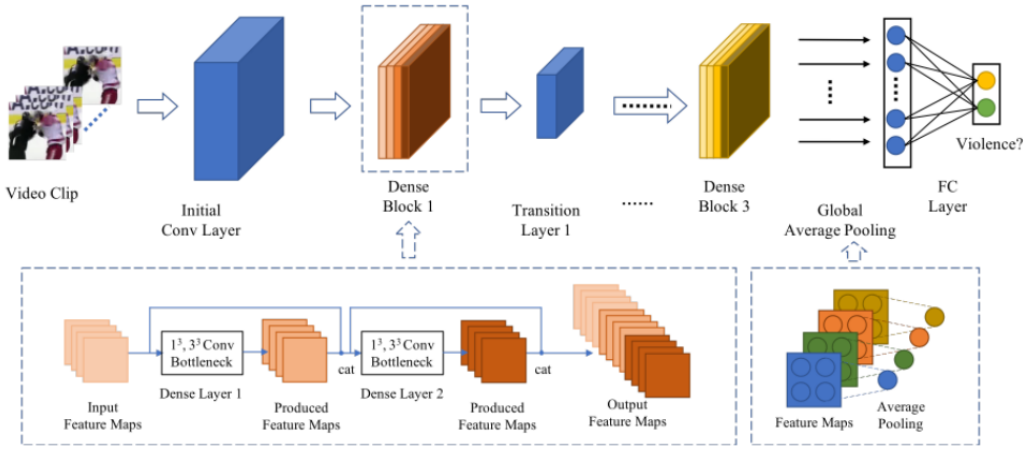


Figure 5.5: DenseNet Architecture

5.1.2 Single video inference

After training the model, we save its states and weight in a .pt file. Once the saving process is done we proceed towards single video inference. For single video inference we first insert the source code of the model and initialize it. Then we load the states into the model from the .pt file, and after loading the states we push the model to the cuda device. Then we proceed towards playing the video, which is done by using the cv2 library. when the video is loaded and displayed, a batch of 16 frames is extracted and pre-processed in accordance to the model. Firstly, the frames are converted into PIL image format and then to PyTorch tensors. After turning them into tensors they are appended into a list. If the length of the list is equal to the batch size, the list is converted into a PyTorch stack. After conversion to stack the batch of frames is un-squeezed to add an extra dimension and then the shape of the tensor permuted in order to fit the model. The tensor is then moved to the cuda device and the passed to the model. The output of the model is detached from the gpu and the prediction is displayed (0 for non violent and 1 for violent).

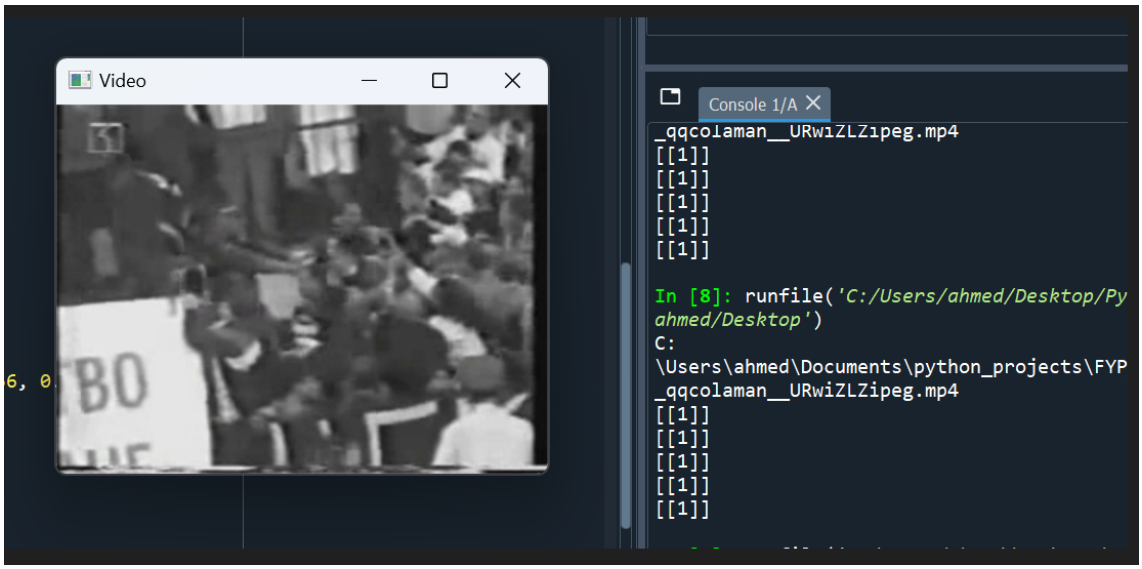


Figure 5.6: Single Video Inference

5.1.3 System Front-end:

The front end of the system is designed using the Django framework[22]. It consists of a user registration/sign-up page, a user login/sign-in page. A main dashboard, which is used for classification of recorded videos, a logs page which keeps record of the videos uploaded by the user and the timestamp in which the violence is detected. Lastly it contains a page for live classification in which the webcam of the PC is accessed and classification is done. In order to perform the necessary functionalities effectively, the Django framework uses the MVT(Model View Template) structure.

5.1.3.1 Model

In Django, the model consists of databases created by the user or/and present in the framework by default. These databases store the data and retrieve it when the user request is given. The user can also modify the data present in the tables. The database used in this project is the user model provided by Django, to register and then login users into the system.

5.1.3.2 Template

The template consists of the files that are used to create the interface that is displayed to the user. These files are the HTML(Hyper Text Markup Language), css(cascading style sheets) and js(java-script) files, and these files are combined in order to create a responsive front-end that the user interacts with. The templates can consist of tags and/or components that the user creates by himself or the use of framework can be done. For this system the

templates were created using the Bootstrap framework[23] and its components. within the all templates, the CSRF (Cross Site Request Forgery) token is added so that the responses sent to the user are secured and only the valid party can send and access data of the user.

5.1.3.3 Views

The view is the most important section of the Django framework. It performs the functionality of handling the url requests, which connects the front-end and back-end with each other enabling the system to perform its functionalities. In order to connect front-end and back-end successfully, many functions are created in the views.py folder. These functions have the request as their parameter and once a request comes a function in the views.py folder will validate it through either the CSRF token or/and the type of request that has been sent and then perform its function. Some of these functions enable video classification, while others enable the connection between the models and users or storing of data in different files. In order to access functions in the views.py folder urls for each of them have been created through the url function of Django in the urls.py folder.

5.1.3.4 User registration

If the user wishes to register/ create id on the system, he is rendered to the Register.html page via the register function in views.py. If the user is already registered and is on the registration page he can be rendered to the login page by clicking the text *already a user? click here to login* at the bottom of the page. If the user is not registered, he/she provides details to the system which are:

- username.
- email.
- First name.
- Last name.
- password.
- password again.

After providing these details the system checks on the front-end whether any of the fields are missing or not. If any field is missing the user is unable to submit the details until the fields are filled. If all the required information is entered, the Register function function in views.py is called. In the Register function, firstly the length of the username is checked. Secondly, it is verified whether the username is alphanumeric or not and lastly if the two passwords entered match or not. If any of the conditions are not met, an error message is

displayed and the user is redirected to the registration page again. Otherwise, the user is created bu using the create-user object in Django User model and the user is directed to the login page where he can login to his created account and access the dashboard.

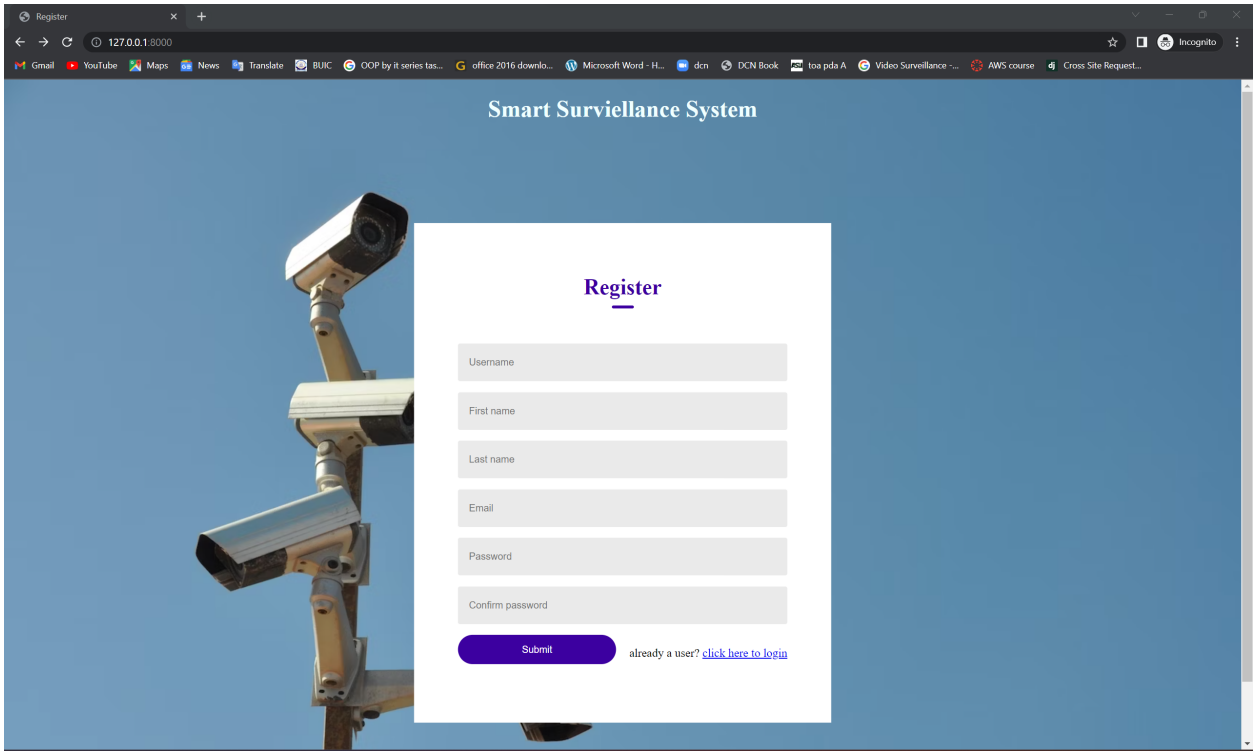


Figure 5.7: User register page

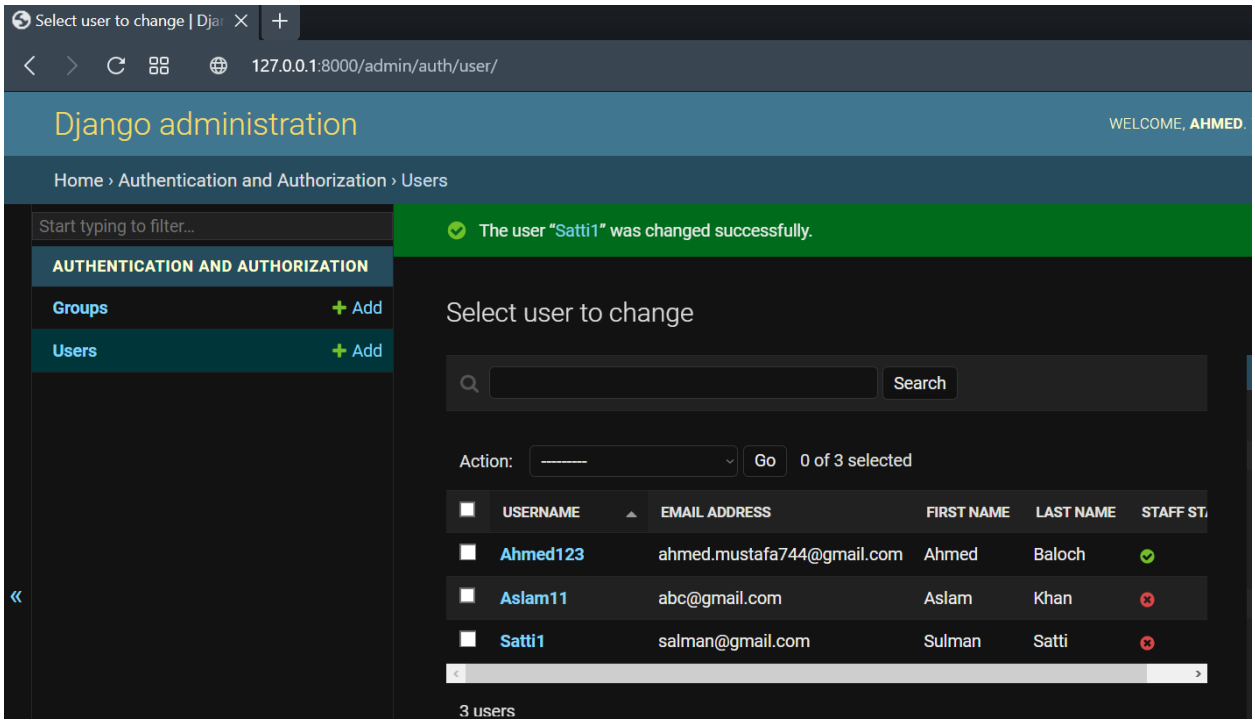


Figure 5.8: User Database after registering user

We can access user table through Django administration. Through this if we login as admin we can add and delete users, and also modify user details such as name or/and emails.

5.1.3.5 User login

If the user wishes to login to his account, he is rendered to the login.html page via the login function in views.py. If the user is not registered then he can be rendered to the registration page by clicking the text *not a user? click here to register*. If the user is already registered, he/she provides details to the system which are:

- username.
- password.

After providing these details the system checks on the front-end whether any of the fields are missing or not. If any field is missing the user is unable to login until the fields are filled. If all required information is entered, the Login function in views.py is called. In the Login function, the user is authenticated via the authenticate library provided by Django authentication which takes the provided user details as parameters. If the user is not authentic,the authenticate function returns none, which stops the user from accessing the main dashboard and an error message is displayed stating *Invalid username or password* .

Then user is redirected to the login page again. If user is authentic he is logged into the system using built in auth-log function and rendered to the dashboard.

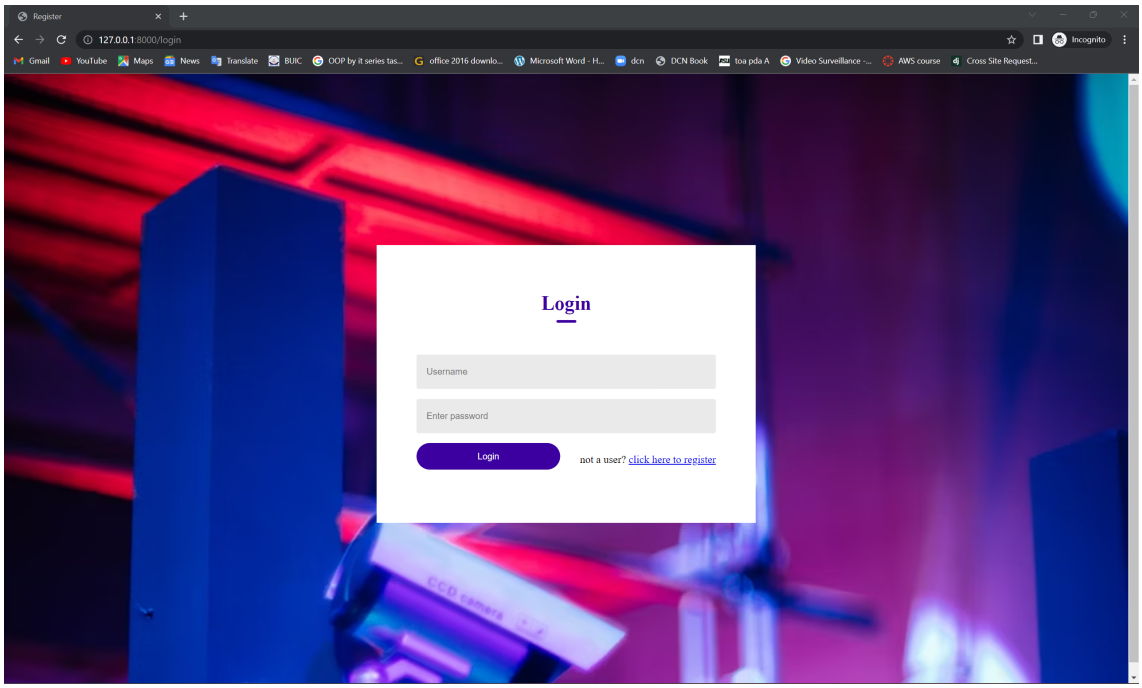


Figure 5.9: User login page

5.1.3.6 Classification of recorded videos

After successfully logging in the main dashboard is displayed to the user. If the user wishes to check recorded videos on his PC or device for violence he can upload it on the web-page. If not, then the user can be redirected to the live classification page to detect violence in real-time or records page to see the records of the video in which the video name and the timestamp of violence detected is shown. The user can also log out. When the user uploads the video by clicking the submit button the Classify function in views.py is called. This function does two tasks, firstly it takes the path of the video and assigns it to the thread calling the Pred function as a variable, this starts the classification of the video. Secondly it uploads the video to the media directory of the project and then assigns its path to the *src* of the video tag. In the Pred function the entire process of the single video inference is repeated and then the classification is done. After the classification is done, the label is then saved in the file var.js. This saved variable is accessed by the dashboard via java script and displayed on the page and the background of the the page is changed to red. The accessing of the variable is done every 300 milliseconds until the end of the video so that modification of the label is displayed. This is achieved by sending a GET request to the assign-value function in views.py which opens the var.js file, extracts the variable and then sends it back to the web page as a json response. on the front-end the json object

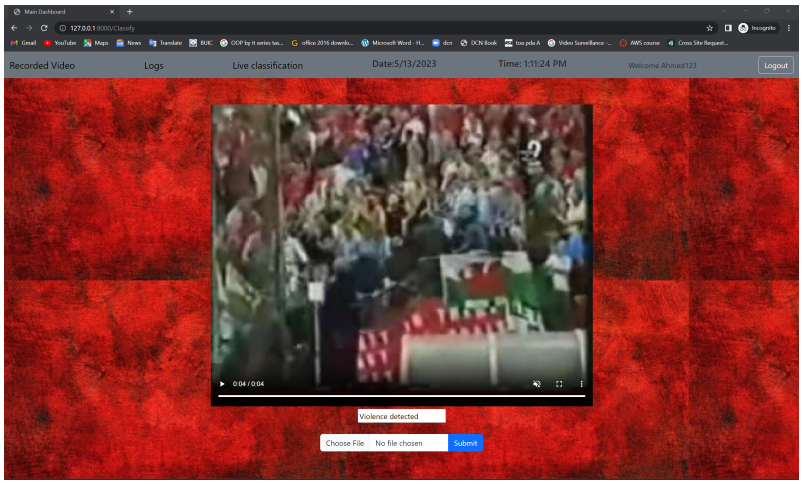


Figure 5.10: Recorded video classification page

is received and the value is extracted before displaying it to the user. Furthermore, if the label contains classification of violence detection then the video name and username along with the current time of the video is saved in logs.json file. This is achieved by calling the write-to-json function in which the logs.json file is opened. If the file is empty, the a json array is created and then the data is stored, or else the data is appended in the array. This file is then accessed to display logs.

5.1.3.7 Live classification

If the user wishes to perform live classification, he clicks on the live tab on the dashboard which renders the live.html page. When this page is loaded the live-classification function in views.py is called. This function returns the frames as a StreamingHttpResponse to the front-end. the frames are returned via calling the gen function. In this function, the web-cam is opened using open-cv and then the process of single video inference is initiated again. The video frames are pre-processed within the while loop and the pre-processed batch is sent to the pred-live function which then returns the label. If the label classifies violence, then a beep sound/alarm is generated and the frame border is painted red. Along with that the system also sends an auto generated email to the user in the instance of violence detection in which it sends an alert message along with a snapshot as attachment. It does that by creating an ssl connection and connecting to a secure smtp gmail server through python. In the front-end the live-classification function's url is called in *src* of the image, which displays the frames on the front-end.

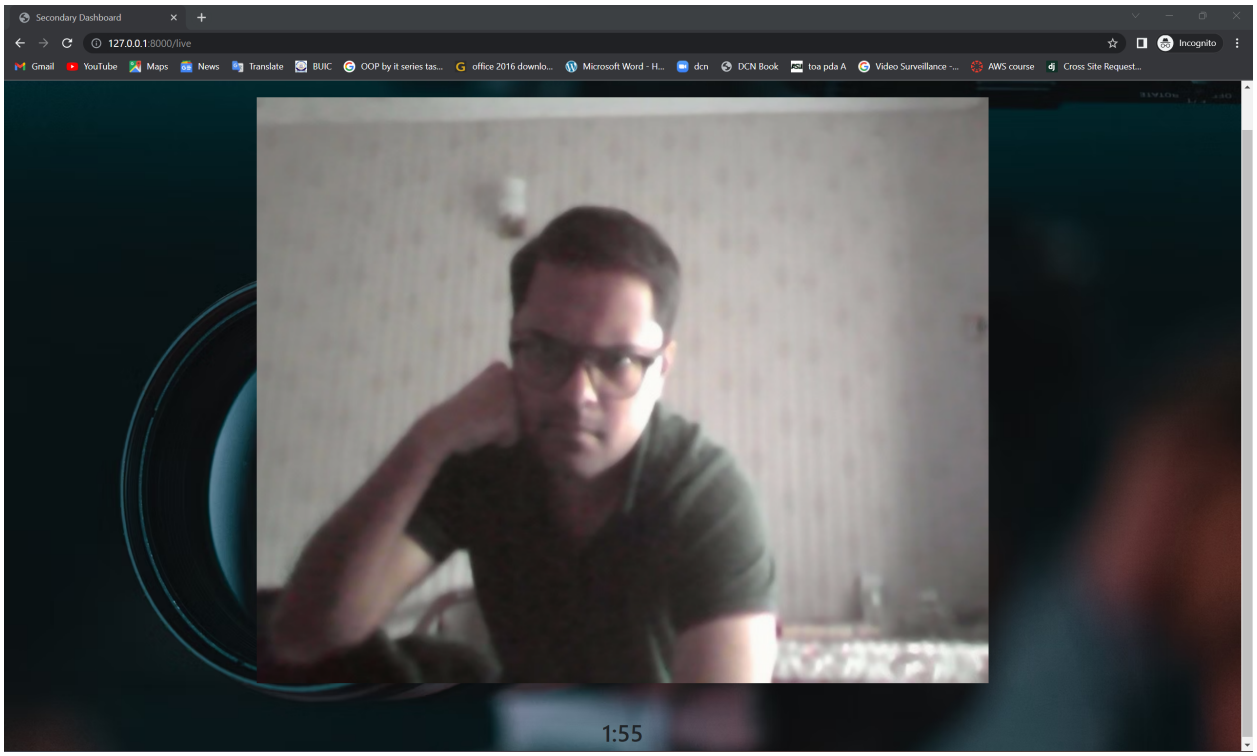


Figure 5.11: Live video classification page

5.1.3.8 Viewing logs

When the user clicks on Logs tab, it renders the logs.html page via the logs function in the views.py file. In the logs page the records of the video uploaded by the user along with the time in the video in which the violent activity occurred. This is achieved by calling the send-json function in the views.py file. The send-json function opens the logs.json file and loads the json array containing the details. After loading, it sends the array as a json response to the front-end. Then the front-end filters out the the data of videos that were not uploaded by the logged in user and creates a table of the remaining data.

5.1.3.9 Admin Tasks

In order to perform tasks of a system administrator the user must log into the Django admin site. Django admin site is the built in admin interface of the Django framework. The user firstly creates a super user which is the admin of our system. This task is done before beginning the development of the front-end. For the purpose of our system, the admin site is used to do the following tasks:

- Add User, as shown in figure 5.16.
- Delete User, as shown in figure 5.18.
- Modify User details, as shown in figure5.17.

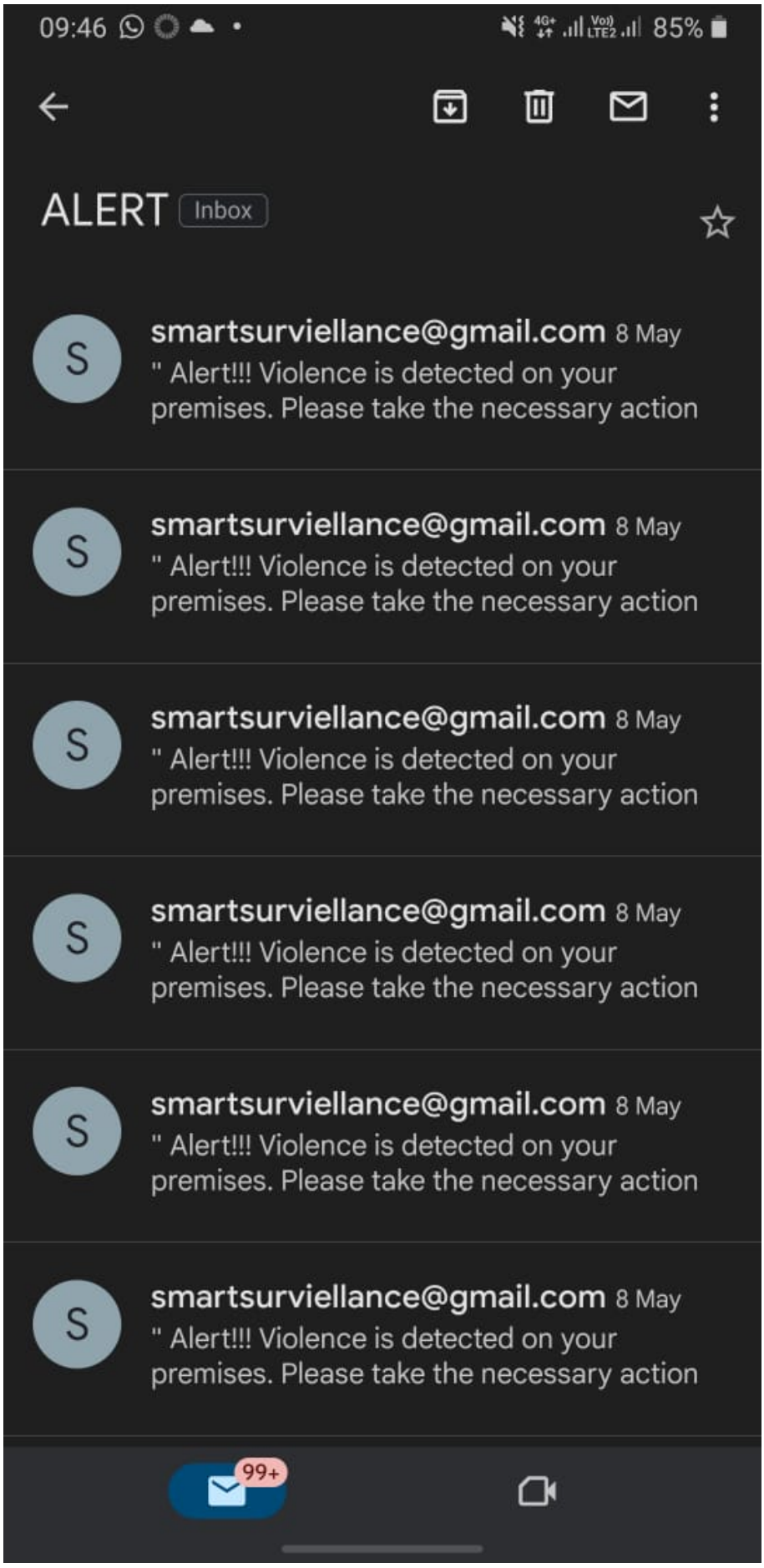


Figure 5.12: Email alert received by user

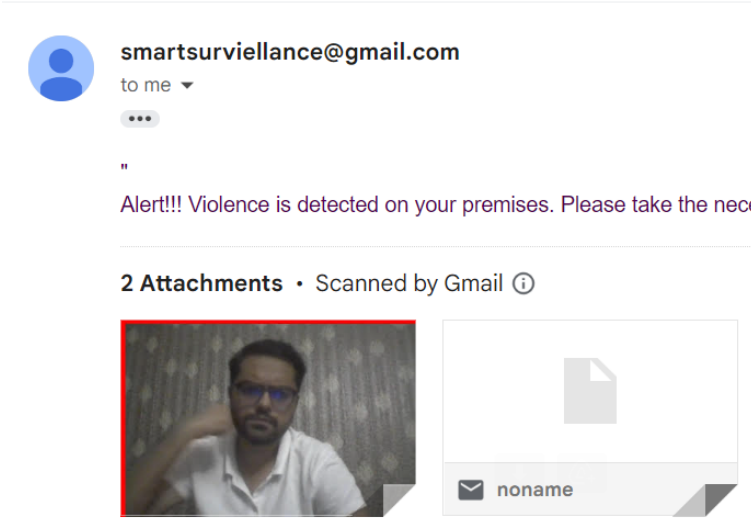


Figure 5.13: image sent as attachment in email

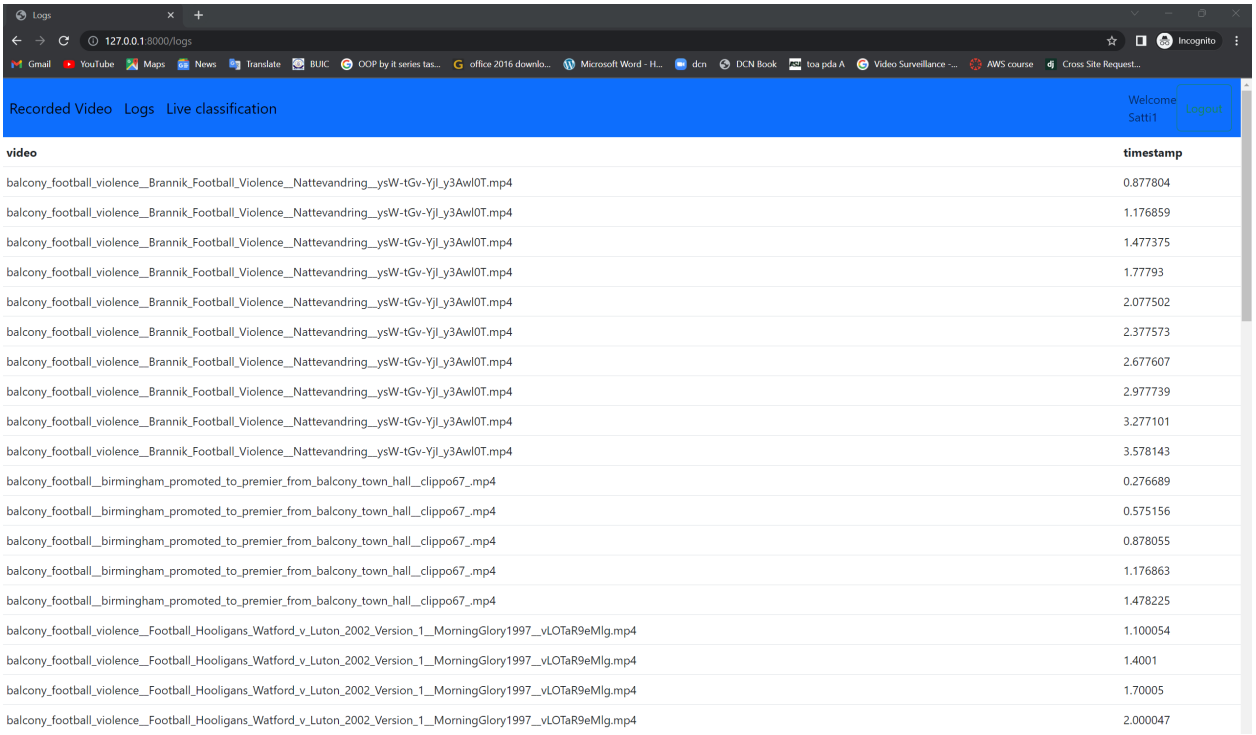


Figure 5.14: Logs page

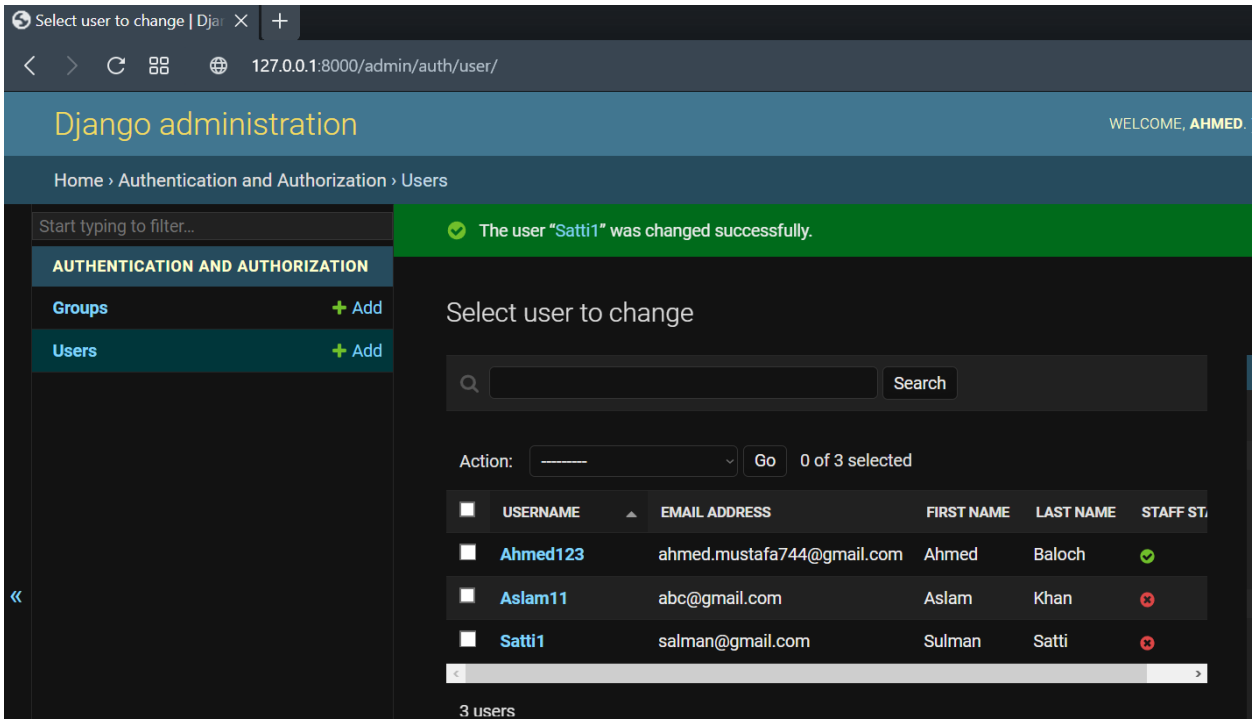


Figure 5.15: User Database

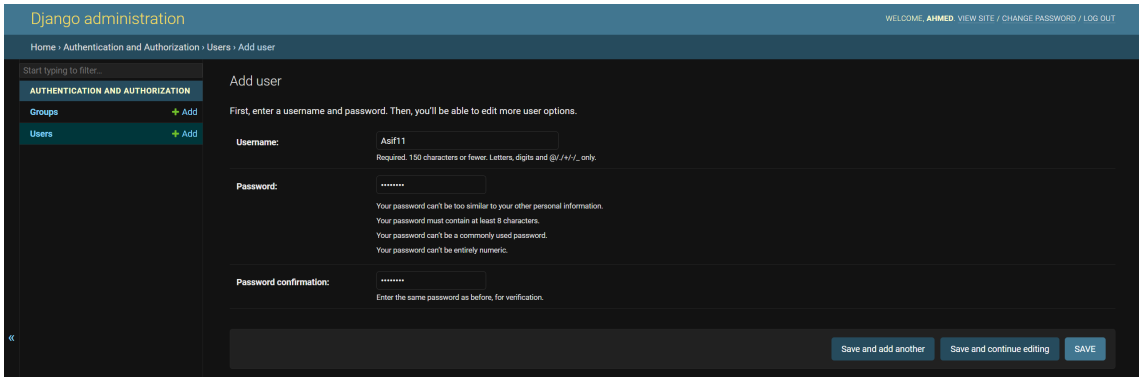


Figure 5.16: Add User by admin

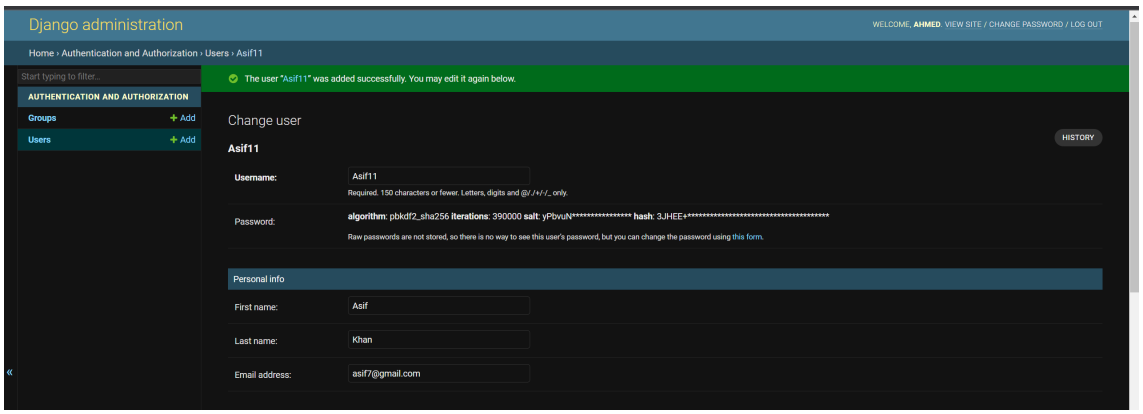


Figure 5.17: Modify User details by admin

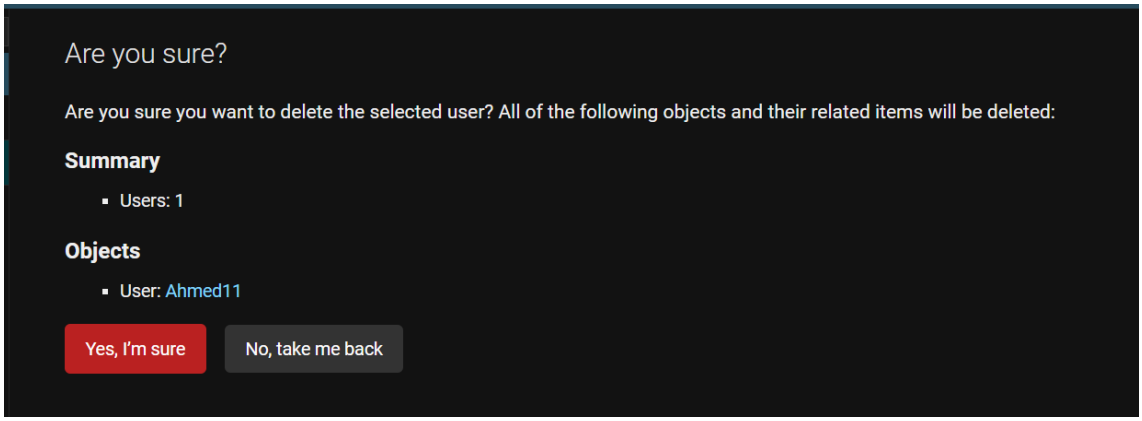


Figure 5.18: Remove User by admin

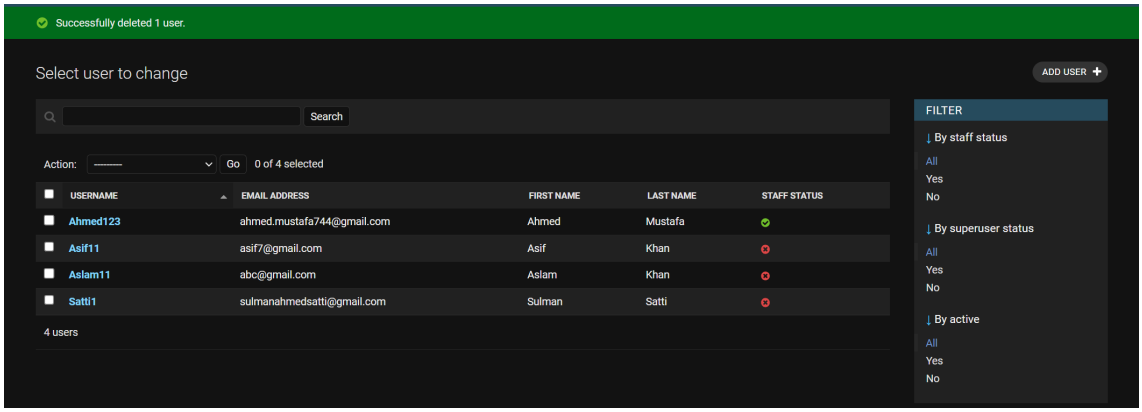


Figure 5.19: User Database after modifications

Chapter 6

System Testing and Evaluation

6.1 Tests Conducted

6.1.1 Graphical user interface testing

The graphical user interface (GUI) consists of the components that the user will be able to interact with in order to perform tasks on the system. Hence in this testing we will ensure that these components are performing the required tasks in a way that satisfies the user requirements. In the figure below the user proceeds to enter his/her details for registration to see whether the password he enters is replaced by **dot** symbol.

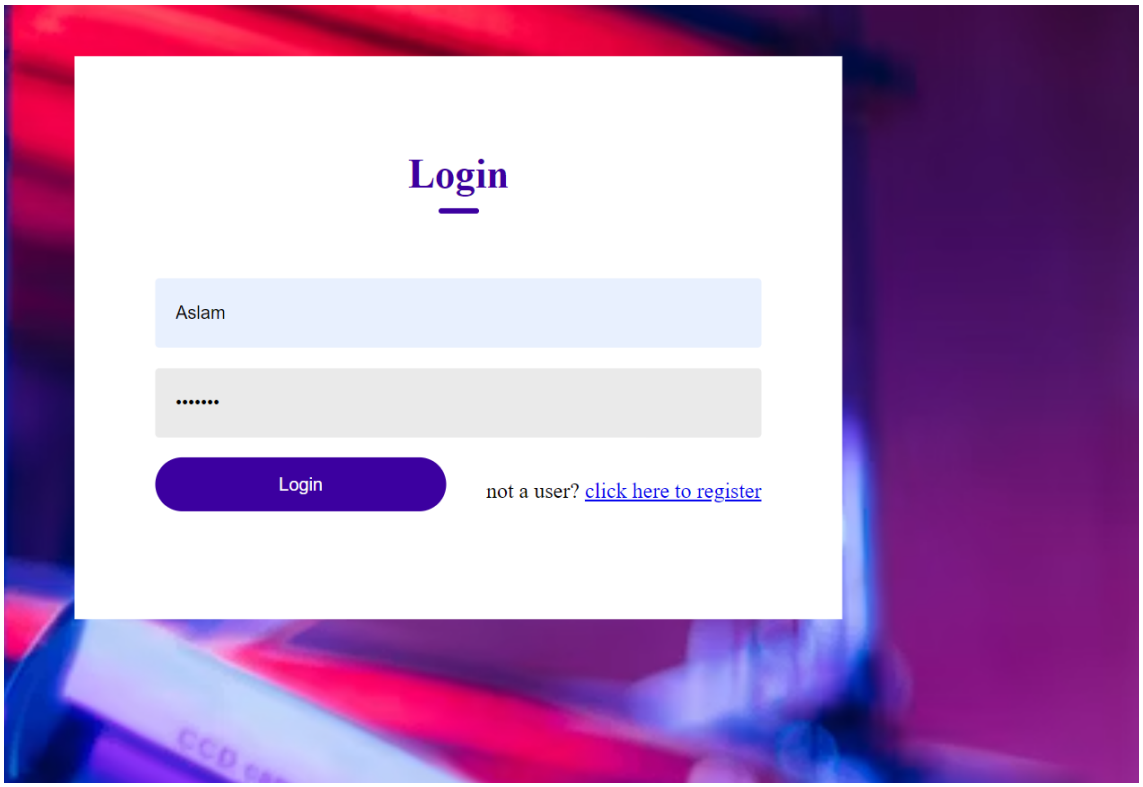


Figure 6.1: GUI Testing

6.1.2 Usability testing

The application is a combination of multiple pages which are registration, login, main dashboard, live classification page and logs. In this testing we see whether the user can easily navigate through these pages and for classification of recorded videos, is able to successfully upload a video file.

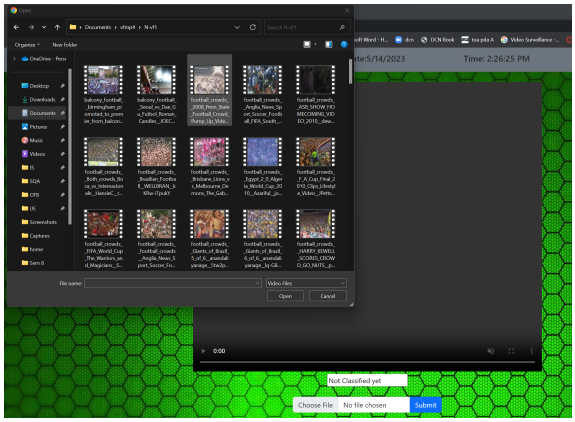


Figure 6.2: Usability testing 1

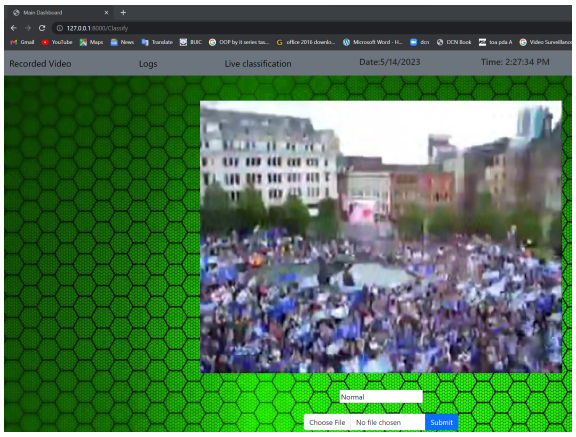


Figure 6.3: Usability testing 2

6.1.3 Performance testing

In this testing we check whether the application is performing its functionalities. In the figure below we check whether the system can successfully classify violence in recorded videos. First we check whether the label is changed in var.js file during classification and then whether the changed value is shown on screen or not. Also we also monitor whether the background gets changed when violence is detected.

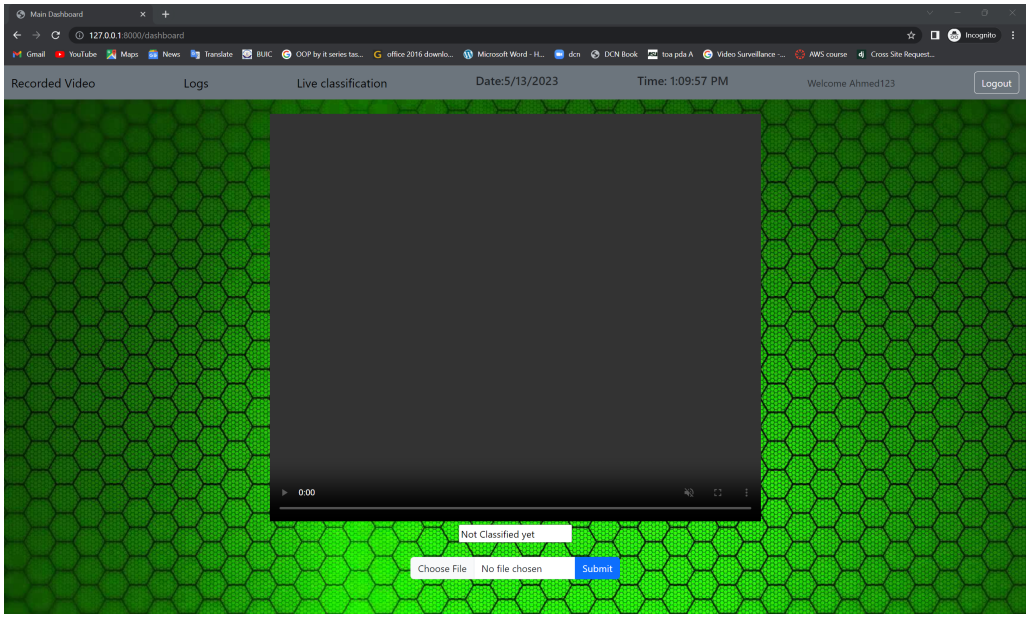


Figure 6.4: before label and background change

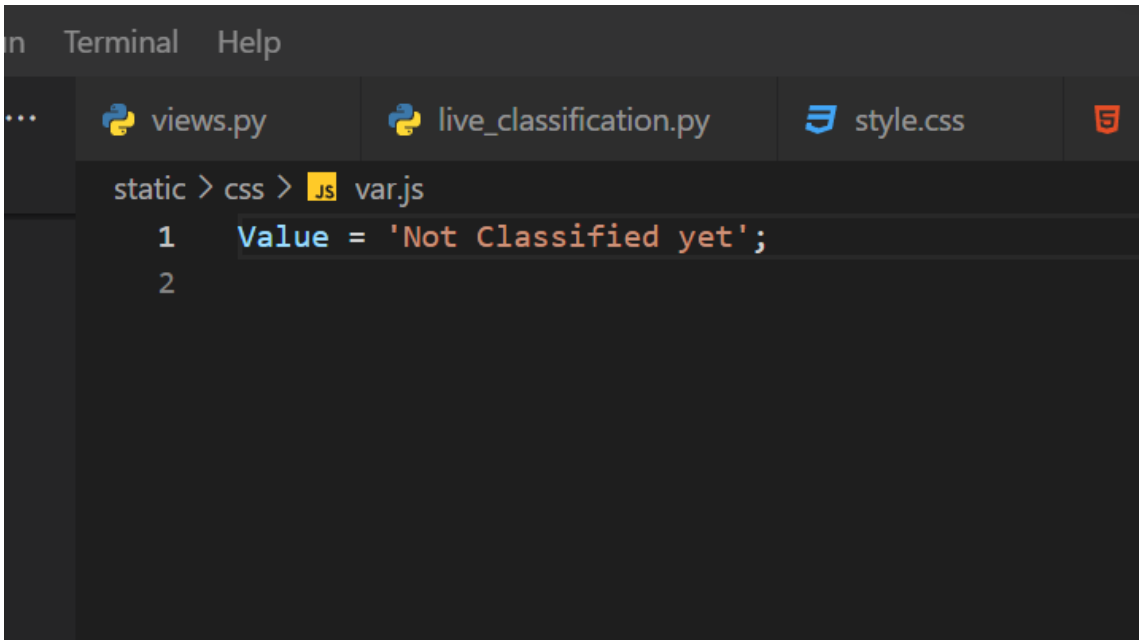


Figure 6.5: Var.js file before

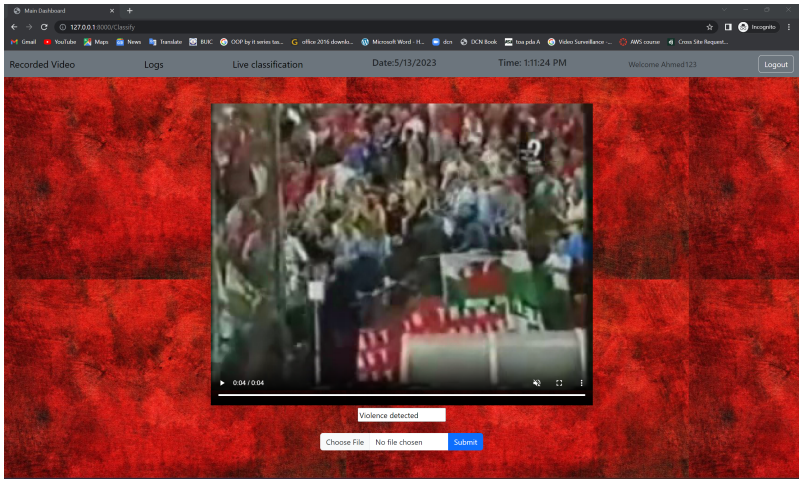


Figure 6.6: label and background change displayed on screen

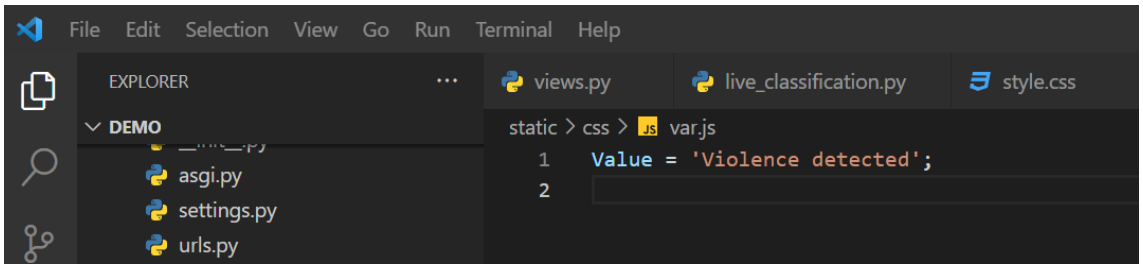


Figure 6.7: Var.js file after

6.1.4 Compatibility testing

In compatibility testing we check whether the application can run on different browser. In this case the Opera browser is used.

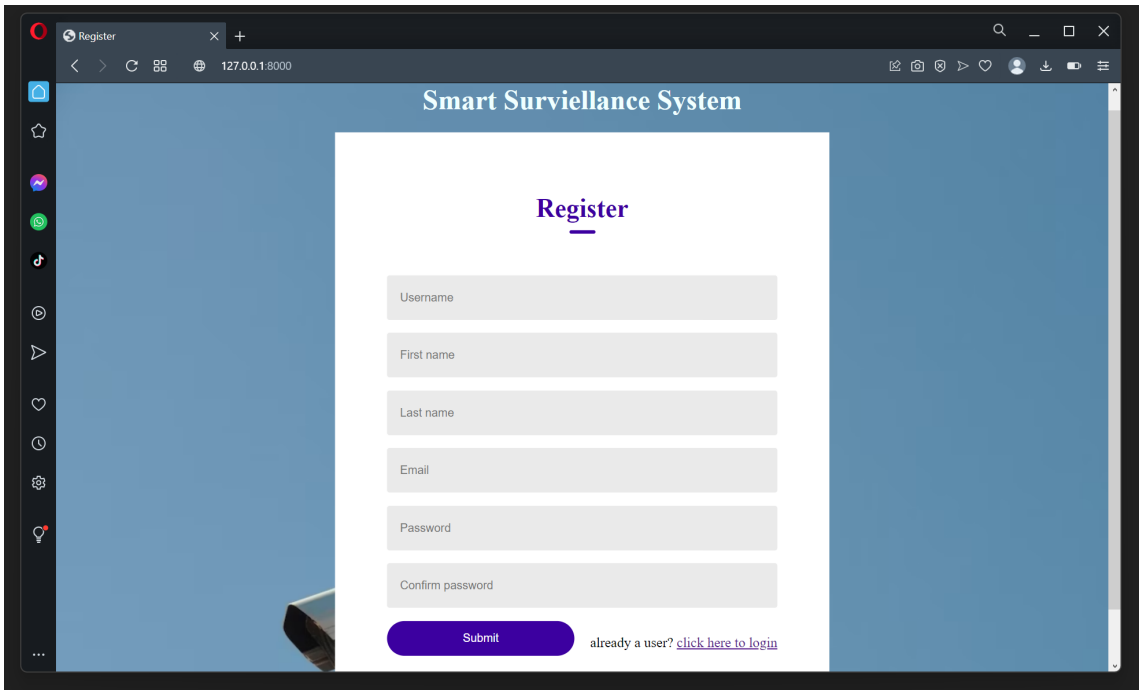


Figure 6.8: Compatibility

6.1.5 Security testing

In this testing we check the security aspect of the application. We check this by trying to access the pages without logging in.

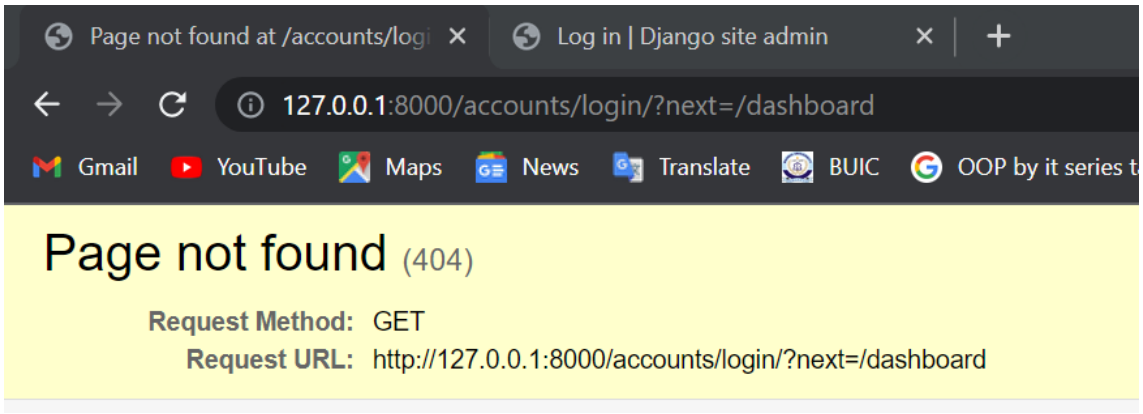


Figure 6.9: Security Testing

6.1.5.1 Training and Testing accuracies:

The graphs below show the training and validation accuracy and loss for each epoch,

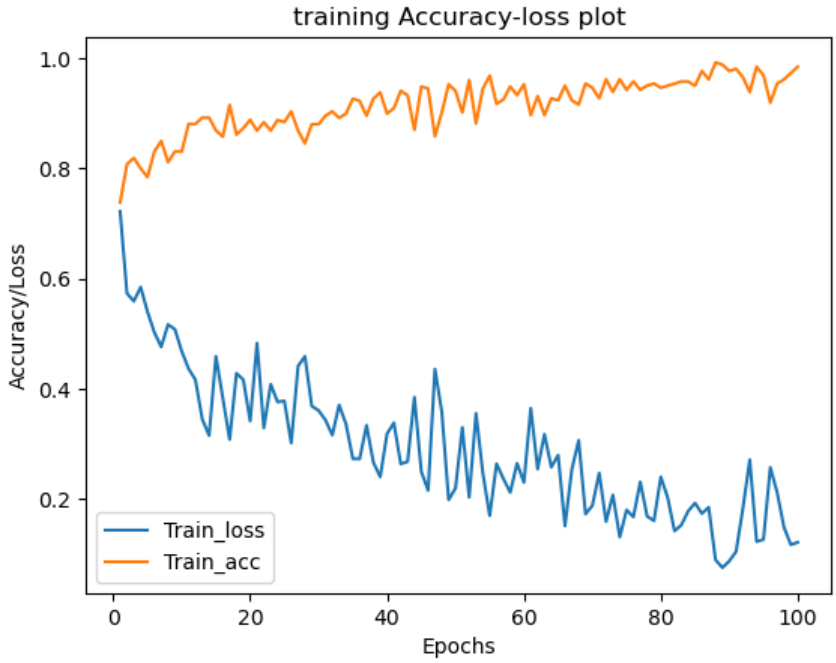


Figure 6.10: Training Accuracy/Loss plot

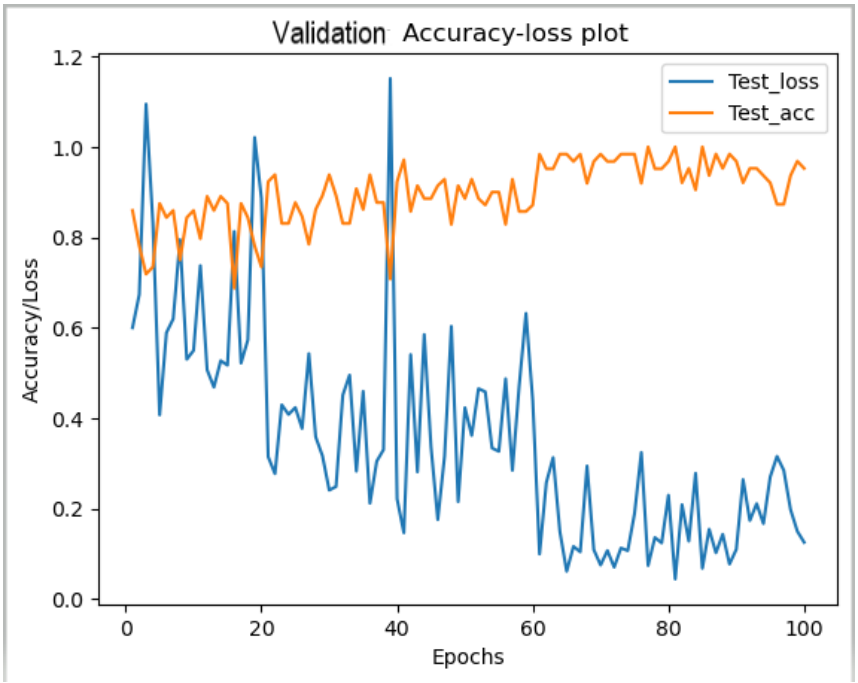


Figure 6.11: Validation Accuracy/Loss plot

6.2 Test Cases

6.2.1 Test Case 1: User registration

ID	UR-01
Pre-condition	Website must be loaded
Input	Username, email, first name, last name, password, repeat password.
Expected result	User registers successfully .
Actual Result	User registers successfully .
Post condition	login page is loaded
Status	success

Table 6.1: User registration success

ID	Ur-02
Pre-condition	Website must be loaded.
Input	One or multiple fields kept empty.
Expected result	User is unable to submit details.
Actual Result	User is unable to submit details.
Post condition	Login page doesnot load.
Status	fail

Table 6.2: User registration failure1

ID	UR-03
Pre-condition	Website must be loaded.
Input	Passwords donot match
Expected result	User gets error message.
Actual Result	User gets error message and.
Post Condition	User is redirected to registration page.
Status	fail

Table 6.3: User registration failure2

6.2.2 Test Case 2: User Login

ID	UL-01
Pre-condition	User is registered.
Input	User enters username and password
Expected result	User logs in successfully
Actual Result	User logs in successfully
Post Condition	Main dashboard is loaded.
Status	success

Table 6.4: User login success

ID	UL-02
Pre-condition	User is registered
Input	User enters username and password. The username and/or password donot match
Expected result	error message is displayed and user does not login
Actual Result	error message is displayed and user does not login
Post Condition	User is redirected to login page.
Status	fail

Table 6.5: User login failure

6.2.3 Test Case 3: Classification on recorded video

ID	URec-01
Pre-condition	User must be logged in
Input	User uploads video
Expected result	Violence is successfully classified in video
Actual Result	Violence is successfully classified in video
Post Condition	Label is displayed on web page.
Status	success

Table 6.6: Classification on recorded video success

ID	URec-02
Pre-condition	User must be logged in
Input	User uploads video
Expected result	Violence is not successfully classified in video
Actual Result	Violence is not successfully classified in video
Post Condition	Wrong label is displayed in video.
Status	fail

Table 6.7: Classification on recorded video failures

6.2.4 Test Case 4: Live classification

ID	Ulive-01
Pre-condition	User must be logged in
Input	user opens live tab and frames are extracted from webcam
Expected result	Alarm is generated and red box is made on border
Actual Result	Alarm is generated and red box is made on border
Post Condition	None
Status	success

Table 6.8: Live classification success

ID	Ulive-02
Pre-condition	User must be logged in
Input	user opens live tab and webcam doesnot open
Expected result	Live tab displays black screen instead of video
Actual Result	Live tab displays black screen instead of video
Post Condition	None
Status	fail

Table 6.9: Live classification failure

6.2.5 Test Case 5: View Logs

ID	ULog-01
Pre-condition	User is logged in
Input	User clicks the logs tab
Expected result	The names and time in the video where violence was detected is displayed
Actual Result	The names and time in the video where violence was detected is displayed
Post Condition	None
Status	success

Table 6.10: Logs

ID	ULog-02
Pre-condition	User is logged in
Input	User clicks the logs tab
Expected result	Page is empty
Actual Result	page is empty
Post Condition	None
Status	fail

Table 6.11: Logs failure

6.2.6 Test Case 6: User logout

ID	LogOut
Pre-condition	User must be logged in
Input	User clicks logout button
Expected result	User successfully logs out
Actual Result	User successfully logs out
Post Condition	User is redirected to login page.
Status	success

Table 6.12: Logout

Chapter 7

Conclusions

7.1 Summary

In recent times many violent incidents have shown the importance of increase in security mainly through improving technology. This then created a need for a system that assists the security personnel. The aim of this project was to fulfill this need by creating a web based application.In this project we used the Violent Flows dataset to train our DL model. We initially trained a DenseNet model which gave an accuracy of 70 percent. Then this was improved by increasing the number of epochs that led to the accuracy rates going up to 97 percent. After the training of the DL model a front end was created through the assistance of the Django and Bootstrap frameworks.This system assists the security personnel in classification of violence in both live stream and recorded videos. Also, this system stores the logs of recorded video which contains the name of the video and the time(s) in the video in which violence was detected and sends an email to the user when violence is detected in live streaming.

7.2 Future Enhancements

At this stage the project is only deployed in local host server and can only handle one user at a time. In the future, we aim to increase the scale of the project and deploy it on a hosting which is accessible to the general public. Also we aim to increase the capacity of the system to handle multiple users at the same time.Another improvement we would like to incorporate in our system would be to use a grid structure in which we show output of multiple cameras and play multiple recordings at the same time and classify the in parallel manner. We also intend on continuing the research on effective tools and techniques to

increase the accuracy of the of our back-end system and constantly improve the deep learning model.

References

[1] Hindustan Times. Watch how an islamist mob in pakistan tortured and burnt a sri lankan man over blasphemy charges, Dec. 2021. Cited on p. 2.

[2] Bench Mark. Junaid jamshed beaten up at islamabad airport, 2020. Cited on p. 2.

[3] Violence detection for smart surveillance systems. https://www.abtosoftware.com/blog/violence-detection. Cited on p. 7.

[4] Save lives by detecting warning signs of suicide, cyberbullying, and school violence. https://www.lightspeedsystems.com/solutions/lightspeed-alert/. Cited on p. 8.

[5] Meet oddity.ai the future of safety. https://oddity.ai/. Cited on p. 9.

[6] Swathikiran Sudhakaran and Oswald Lanz. Learning to detect violent videos using convolutional long short-term memory. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2017. Cited on pp. 9 and 13.

[7] Ling Zhu, Zhenbo Li, Chen Li, Jing Wu, and Jun Yue. High performance vegetable classification from images based on alexnet deep learning model. *International Journal of Agricultural and Biological Engineering*, 11(4):217–223, 2018. Cited on p. 10.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. Cited on p. 10.

[9] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202, 2011. Cited on p. 10.

[10] Alex Hanson, Koutilya PNVR, Sanjukta Krishnagopal, and Larry Davis. Bidirectional convolutional lstm for the detection of violence in videos. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018. Cited on pp. 11 and 13.

[11] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Cited on p. 11.

[12] Yuan Gao, Hong Liu, Xiaohu Sun, Can Wang, and Yi Liu. Violence detection using oriented violent flows. *Image and Vision Computing*, 48-49:37–41, 2016. Cited on pp. 12 and 13.

[13] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000. Cited on p. 12.

[14] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3), may 2011. Cited on p. 12.

[15] Tal Hassner, Yossi Itcher, and Orit Kliper-Gross. Violent flows: Real-time detection of violent crowd behavior. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–6, 2012. Cited on p. 12.

[16] Long Xu, Chen Gong, Jie Yang, Qiang Wu, and Lixiu Yao. Violent video detection based on mosift feature and sparse coding. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3538–3542. IEEE, 2014. Cited on pp. 12 and 13.

[17] Ming-yu Chen and Alexander Hauptmann. Mosift: Recognizing human actions in surveillance videos. 2009. Cited on p. 12.

[18] Xinling Geng and Guangshu Hu. Unsupervised feature selection by kernel density estimation in wavelet-based spike sorting. *Biomedical Signal Processing and Control*, 7(2):112–117, 2012. Cited on pp. 12 and 13.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. Cited on p. 12.

[20] Figma (software). https://en.wikipedia.org/wiki/Figma_(software). Cited on p. 52.

[21] Pytorch. https://pytorch.org/get-started/locally/. Cited on p. 55.

[22] Let's understand the pros and cons of using django. https://www.benchmarkit.solutions/lets-understand-the-pros-and-cons-of-using-django/. Cited on p. 61.

[23] Bootstrap framework. https://getbootstrap.com/docs/5.0/getting-started/introduction/. Cited on p. 62.