Road Pulse

Final Year Project

Session 2017-2021

A project submitted in fulfilment of the

COMSATS University Degree

Of

BS in Computer Science (CUI)

Department of Computer Science

COMSATS University Islamabad, Lahore Campus

26 December 2020

Project Detail

Type (Nature of project)	[ ] **D**evelopment [ ] **R**esearch [✓] **R**&**D**			
Area of specialization	Deep Learning, Crash and Violation detection			
Project Group Members				
Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	SP17-BCS-085	Hamza Latif	hamza.latif773@gmail.com	
(ii)	SP17-BCS-058	Hassan Sharjeel	hsjoiya@gmail.com	

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

Plagiarism Free Certificate

This is to certify that, I am Hamza Latif S/D/o Muhammad Latif, group leader of FYP under registration no CIIT/SP17-BCS-085/LHR at Computer Science Department, COMSATS Institute of Information Technology, Lahore. I declare that my FYP proposal is checked by my supervisor and the similarity index is 8 % that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: 12/28/2020 Name of Group Leader: Hamza Latif Signature: 

Name of Supervisor: DR USAMA IJAZ BAJWA Co-Supervisor (if any): _____

Designation: _ASSISTANT PROFESSOR___ Designation: _____

Signature: _____ Signature: _____

HoD: _____

Signature: _____

Abstract

According to the traffic conditions in Pakistan, traffic violations and anomalies occur daily. Rush drivers do not respect traffic rules and regulations which is a major concern. According to a survey each family in Pakistan owns a vehicle whether it's a car or bike. This results in issues like traffic congestion as the number of vehicles are increasing. Transportation demand exceeds the road capacity. So, we provided an intelligent system to improve traffic management and control Road Pulse. Building new roads, hiring new police officers is not a good option. Fast reaction and prevention of traffic violations play a key role to ensure the safety of vehicles and citizens. For this purpose, in Pakistan surveillance cameras are installed everywhere to prevent traffic violations but they are unable to detect events like a Lane violation, vehicle crash events etc. due to the faulty architecture. In Pakistan, traffic issues are rising day by day which are causing many causalities like accidents, loss of innocent lives, and compromise in the safety of citizens. Our project, Road Pulse, is an application based upon Web as well as Android modules that detects the traffic violations traffic anomalies that are happening daily in Pakistan. Violations and anomalies like lane violations, vehicle crash, etc. which results in many tragic events. The app uses a pre-trained model that's embedded within the Web app. The pre-trained model YOLO v3 is a Fully Convolutional Neural Network that uses machine learning techniques to help us in predicting the violations accurately. The model is trained on a relevant dataset to achieve accuracy. Efforts directed at enforcing traffic laws lack due to no improper surveillance. On the other hand, surveillance systems are limited in duration and intensity, thereby producing weak halo effects on drivers' behaviour. Also, the public attitudes towards compliance are largely conditioned by the visibility of Automated surveillance systems. The effects of the development on driver's behaviour are reviewed and proposals for the improvement of a modified traffic policing structure are urged out. By the presentation of such a framework many innocent lives can be saved and citizens safety can also be restored. The burden of policing departments can also be reduced by introducing such an intelligent and automated surveillance system.

Acknowledgement

It would be unfair and unkind not to acknowledge the guidance, intelligence, and patience Dr. Usama Ijaz Bajwa has shared with us while working on this project. We are incredibly grateful to our supervisor for his vision and his persistence upon the values of hard work, helping others, and not losing morale.

We would also like to thank our institute, Comsats University Islamabad, Lahore Campus, for their providing us a platform to work on as well as helping us in any way possible. This would not have been possible without it.

Our heartiest gratitude to Dr. Usama Ijaz Bajwa, colleagues, and everybody who helped us in any way with their thoughts, skills, or any productive way.

Table of Contents

List of Tables:

List of Figures

Chapter 1

Introduction

1 Introduction

1.1 Introduction

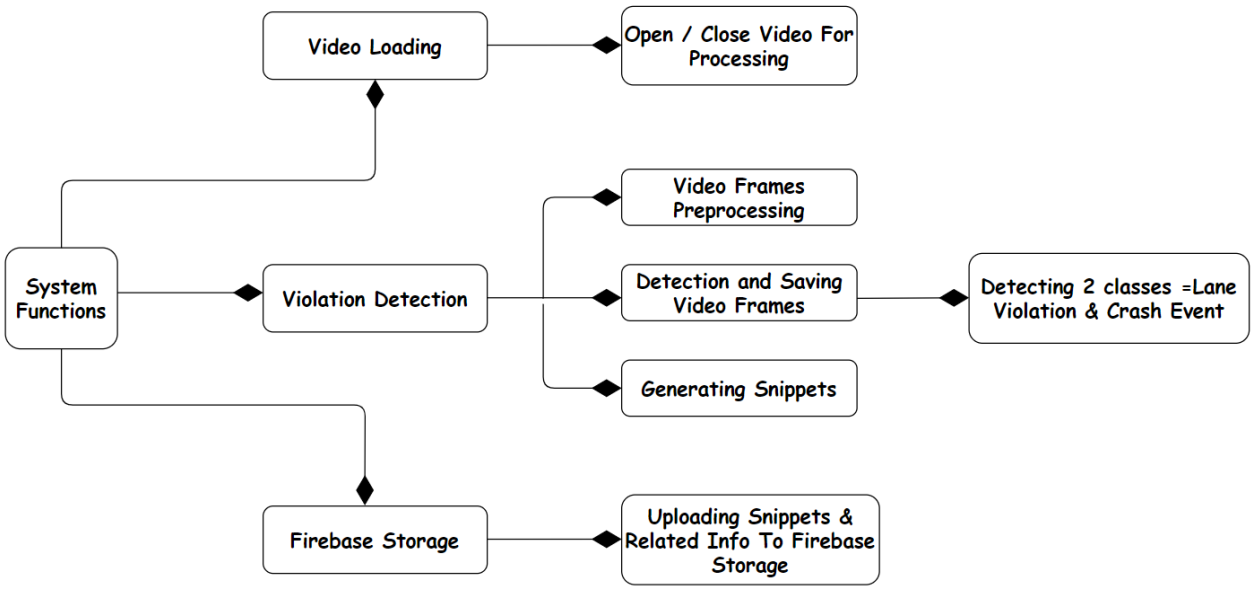
Violation of traffic rules is a critical and rising issue presently. Fast reaction and prevention of traffic violations play a key role to ensure the safety of vehicles and citizens. For this purpose, in Pakistan surveillance cameras are installed everywhere to prevent traffic violations but they are unable to detect events like a red-light violation, wrong way driving, etc. due to the faulty architecture. In Pakistan, traffic issues are rising day by day which are causing many causalities like accidents, loss of innocent lives, and compromise in the safety of citizens. Recently a survey related to the accidents causing deaths in Pakistan increases dramatically due to the traffic violations. [1].So to minimize such alarming factors we should provide an automated solution using the latest technology which will help to reduce these factors.

Rising traffic congestion [2] rules violation and accidents [3] due to a lack of proper traffic management and surveillance need to be addressed with smart solutions. Like automated detection of rules violations, vehicle congestion, etc. integrated with video analytics that can effectively aid traffic administration.

By implementing such system labour costs can be reduced. Reduction in labour will also eliminate many other factors like lack of attention while monitoring many cameras at the same time, pinpointing such events when and where are they happening, etc.

Use of automatic traffic violations in Video analytics can play an important role in getting in-depth insights into traffic conditions [4] and these insights can help by routing this information to traffic administrators which can take effective action on any type of event. Surveillance cameras are cheap and ubiquitous, but the labour required for monitoring them increases the cost now a days. Because generally, it happens that the surveillance cameras are not monitored by proper attention and it is also difficult for a person to pinpoint a violation between many screens. Usually, either no video is monitored at all or infrequent video is observed, alternatively, it is used only for reviewing the incident just for once. But they are also helpful as they are capable of detecting events instead of a passive recording. They make use of the events as they appear to happen and following it, take appropriate actions such as alerting (Notify using android app) the traffic management department. This is the utmost requirement of an automatic violation detection system.

So, our main objective was to derive an optimized and effective solution in traffic violation matter which can detect any type of violation such as red-light violation, wrong-way vehicle detection, over speeding and stop line violation, etc. But later on, after checking the availability of datasets for training custom classes scope was changed to detect two main anomalies Lane violation and vehicle crash. Our proposed system is flexible in nature. Flexibility allows users



to add new modules to the system easily. Some other information can also be inferred from this system in future like vehicle count, provide a prediction of an accident using traffic patterns, etc. This system helps the Traffic management authorities to detect violations accurately so proper action can be taken in real-time and prevent accidents and guarantee citizens safety.

Figure 1: Basic System Module diagram

Figure 1 shows the basic functioning that how the system interacts with other modules. Hence creating a pipeline.

Machine learning algorithms and digital image processing methods are being used for these types of problems still there are many complexities that require greater computation and adaptiveness in algorithms which is possible but are time taking.

They are many solutions proposed to detect the abnormal patterns in traffic, but they require proper guidance on how to interact with the application's UI, costly hardware such as costly sensors to detect any type of violation, and heavy GPU for processing video streams and are cost-effective. So, we proposed a system-based solution by training our model with the latest

available dataset consisting mainly two classes Vehicle crash and Lane violations. The system is cheaper eliminating costly hardware like GPUs, sensors, etc.

1.2 Objectives

The main motive to design and develop such a system is to reduce the number of road accidents by ensuring proper traffic regulation and management. Reduction in the number of road accidents will result in the reduction of human injuries, death rates, and financial loss.

This system does help and assist law enforcement agencies to minimize traffic violations which will ultimately reduce the risk factor for pedestrians etc. Accuracy and quick processing provide help such departments to take regarding the set of actions in real-time. Interactive and user-friendly UI allows law enforcement agencies to use the custom trained model quite effectively. The initial scope of the project involves the detection of traffic rules violation like Lane violation and Vehicle crash, but gradually we will try to extend our domain toward new class problems as per the availability of datasets like wrong way moving vehicle, stop line violation, etc and vehicle count.

1.3 Problem statement

Road Pulse is a modern solution to improve traffic management and restore laws and regulations for the public. As the violation of traffic rules is a critical and rising issue presently. So, such a system assists the law enforcement departments to keep track of such events and response in real-time. The system is based upon Web and Android application where violations and anomalies will be handled.

1.4 Assumptions & constraints

The assumptions and constraints of the system are as follows:

1.4.1 Assumptions

- Our model will be trained using reliable datasets which covers almost all types of violation.
- The model will be deployed on a web application and the android app will be designed for officers on duty. Such a development environment will be selected so that accuracy of the model doesn’t decrease.
- The number of class problems will be limited.

1.4.2 Constraints

- The video stream angle must be accurate enough so that violations can be detected properly.
- The accuracy of detection is dependent upon the quality of the dataset available.
- Live stream quality may affect the detection ratio.
- Compatibility issues aroused during the model deployment.
- AP1 connectivity was complex while the application development.
- As the datasets available are mostly based on foreign roads conditions which affected the detection also. Pakistan's road conditions are different.
- Processing time depends upon the length of the video acting as live stream

1.5 Project scope

We proposed an intelligent Traffic Violation Detection system named "Road Pulse" which makes it easier for the law enforcement department to overcome the violations and irresponsible behaviour of the drivers in Pakistan. Traffic congestion has become a major problem in urban areas because the transportation demand exceeds the road capacity. Traffic congestion is an intense issue in numerous metropolitan zones as a result of its antagonistic impacts on wellbeing, living conditions, and the worldwide economy because of sat around idly. Merely building new roads might not be the best solution for congestion problems due to the enormous investment required and the complex network effects. Instead, there is a huge potential to improve the situation through efficient traffic management and control. The rapid developments in artificial intelligence (AI) and machine learning (ML) have been considered as the primary impetus for the new industrial revolution. So, this project covers major aspects of traffic violations that are happening daily. The trained model is deployed on a web application that tends to generate information about the violations and anomalies. Android applications is used to assist officers to deal with such events in real-time. The information regarding Vehicle crash can be received on mobile app to respond quickly to save lives of citizens.

Chapter 2

Requirements Analysis

2 Requirement Analysis

This section of the report will show all requirements and functionalities of our project Road Pulse.

2.1 Literature review

In Pakistan, there is no such mobile application that receives the information regarding a crash in real-time and generates alarm as soon as the anomaly occurs. The android application is used to assist the traffic department to track the crash events in real time to avoid any tragic events and save precious lives of citizens.

2.1.1 Application Based Related Work

❖ **Waze, GPS, Maps & Traffic Alert**

The most well-known traffic app is free and perhaps its biggest draw is the real-time traffic information provided by users. This allows you to avoid trouble spots on the fly. Many users are also drawn to the app for its social function of sharing route information with friends and family. But it provides only traffic situations. Does not provide anything related to the violations happening in the surroundings. [5]

Screenshots of the application are provided below:

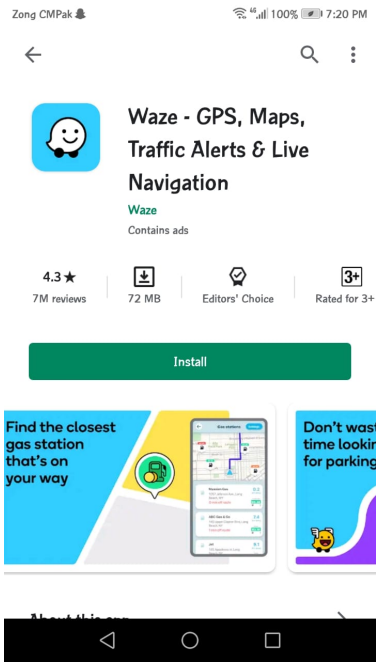


Figure 2: Waze -GPS, Maps, Live Navigation

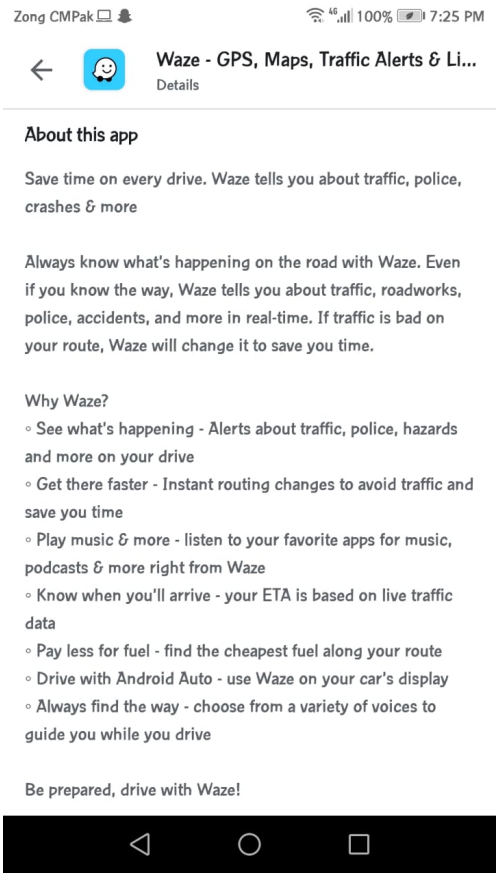


Figure 3: Waze -GPS, Maps, Live Navigation

Figure 2 & 3 show's UI and detailed description of the application. What exactly the Waze application is all about.

❖ Cameras US-Traffic cams, US

This application provides the live stream (doesn't work well) only from the various security cameras installed in the city. This stream is monitored by the security staff to avoid any unpleasant event. But this app is a region bounded only works in the US, poor performance, and has a lot of bugs. Hence there is no proper application introduced yet for the security department. [6]

Screenshot of its main page: below:

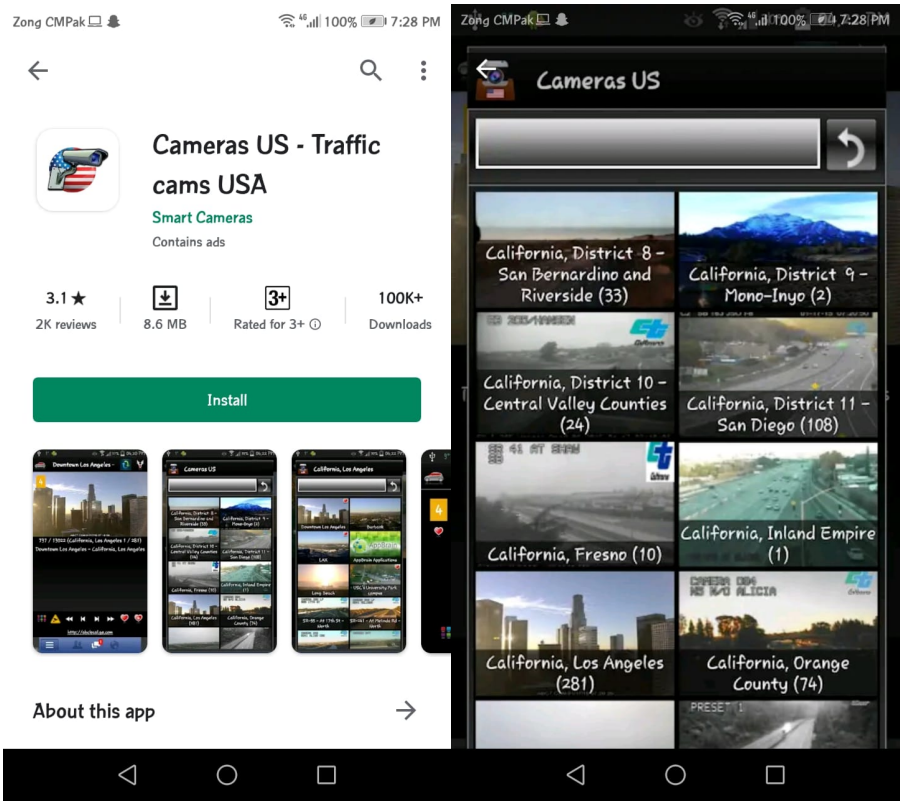


Figure 4: Cameras US

Figure 4 shows' homepage of the application. The application is designed to control different security cameras' live feeds in the US. But there is no scenario of violation detection or anything else. Sometimes the app crashes due to bugs. So, the purpose to include this application is that there is no such application that can detect anomalies and assist the officers to overcome the violations.

2.1.2 Research papers

There are a few research papers available on the internet which are relevant to our project.

❖ **Embedded Camera for Traffic Surveillance using Video Analysis**

This project is based upon a camera that senses video utilizing a single embedded device using video analytics. A prototype will be developed for traffic surveillance based on a smart camera. So, the surveillance system entirely depends upon a smart camera. [7].

❖ **Accident detection and Traffic monitoring**

The frameworks ought to have the option to distinguish every vehicle and track its conduct and to perceive circumstances or occasions that are probably going to result from a chain of such conduct. The most troublesome issue related to vehicle following is the impediment impact

among vehicles. To tackle this issue, we have built up a calculation, alluded to as Spatio-temporal Markov arbitrary field (MRF), for traffic pictures at crossing points. These calculation models the following issue by deciding the condition of every pixel in a picture and its travel, and how such states travel along both the – image axes as well as time axes [8].

❖ **Traffic Violation Detection System**

The framework gives a continual information discovery and warning instrument to spot traffic violations, additionally to advise the police and therefore the vehicle owner of the submitted infringement so on take the proper procedure at the right time, leading to an increasing rate of saved lives. The framework Utilize RFID innovation, to acknowledge whether the motive force is drunk or not. Then it measures the speed of the car, if the vehicle is in over speed then the shutdown flag is sent to the vehicle then the vehicle gets backtrack and ceased. There are some mechanisms to test the vehicle parameters and archives (Vehicle Reg. no, life belt status, liquor status, protection, charge, and then forth.) status on the off chance that anything finds disgraceful then the auto gets captured and ceased. [9].

2.2 Product Functions:

As a product, Road Pulse consist of the following components:

Web Application: The Flask based web application serves as a web interface for the end users of Road Pulse, providing them with real-time monitoring, database access and generating information functionalities, part of the Road Pulse web application.

Android Interface: The android application serves as an android-based interface for the end users of Road Pulse, providing them with accessing real-time information and, database access functionalities, part of the Road Pulse android application.

The Users of Road Pulse will be able to perform the following top-level functionalities:

- ✓ Log into the system using privileged credentials and process video acting as a stream in real-time. The video is uploaded through the Flask server with firebase database accessibility.
- ✓ Video Processing can be viewed by the user in the browser console. Processing shows model working (YOLO v3) at the backend embedded in the web application.
- ✓ Get information for the anomalous activity detected, through interfaced Web and Android Interfaces of the system in real-time through Google Firebase.

- ✓ Access and View events frames + information of events detected by the web application on android application, in chronological order, residing in backend database.
- ✓ User (Security Officer) can track and navigate the location of event like vehicle crash in real-time

2.3 Stakeholders list (Actors)

The stakeholders' list of our system is as follows:

✓ Traffic Control Officer

The Traffic control officer is the top-level manager who manages the desktop application. The live stream from the security cameras can be monitored. Violation and anomaly can be detected using the model from the video acting as a live stream.

✓ Live stream Manager

The video acting as a live stream is managed by one of the employees of the security department. The stream can be delivered to the desktop application.

✓ Government of Pakistan and relevant authorities

This system helps the Government of Pakistan and the security departments to stop such crimes happening daily which results in many tragic events. Also, to change the irresponsible behaviour of the drivers and restore the law.

✓ Android App Module

The information regarding a certain crime(violation) or anomaly (Vehicle Crash) can be sent to the mobile application of the police officers on duty near to that area. So that certain actions can be taken in real-time.

✓ Any private organization or students

Lastly, any other organizations, students, or researchers who want to use our application and data for further research and studying can benefit from our system

2.4 Requirements elicitation

The requirements of the system from the stakeholders' point of view were as follows:

2.4.1 Functional requirements

The functional requirements of the system are mentioned below:

❖ **FR01 – Handling Video as Live Feed**

Table 1: FR01-Video Processing

FR01	The Desktop application will handle the Video as a live stream from the security cameras. Violation and anomaly will be detected using the trained YOLO v3 model.

Table 1: Describes the functionality of the system which is to handle live stream feed (Video). The video steam from various security cameras will help to monitor each route.

❖ **FR02 – Detecting the violation and generating information**

Table 2: FR02-Showing Detection and Generating Info

FR02	Using the trained model embedded in the web app, the violations and anomaly will be detected from the video stream. Then the violation area and pinned location information will be generated.

Table 2: It is about the detection of violations and anomalies using the trained model. After detection information will be generated against each violation or anomaly.

❖ **FR03 – Uploading the data to the server**

Table 3: FR03-Uploading the data to the server

FR03	After the first two FRs, once the information is generated the info is uploaded to the server which is saved in our real time database Google Firebase. Then it is sent to the relevant police officer who signed in the mobile application.

Table 3: It's about managing the data circulation using the real-time database Google Firebase. Servers will help in sending and receiving the information through web app to the firebase database.

❖ **FR04 – Checking out data and information**

Table 4: FR04-Checking out data and information

FR04	The security office can view and use the information after signing in the android application. The information is uploaded on the firebase database through the webpage server.

Table 4: It's about the android app module where information regarding a certain violation will receive. The officer on duty will take further action by penalizing the responsible one or tracking crash.

❖ **FR05 – Android App Module**

Table 5: FR05-App for traffic officers on duty

FR05	The android app will be designed to assist the traffic officer and other security departments like dolphin to get the information about the violation or anomaly that happened recently.

Table 5: shows that after the data is uploaded on the server, the user can view it any time they want through android application after signing in into the app and act according to the situation.

❖ **FR06 – GPU Requirement?**

Table 6 : GPU Requirement

FR06	The entire system is currently working on CPU environment. No GPU is required for the current system. Although performance can be increased by using GPU for processing. But still on CPU system provides high computational processing.

Table 6: Shows that all the functionalities of the system is working on CPU. No GPU is required to obtain results or to do processing at the blackened. Till now there is no system that is working without a GPU requirement.

2.4.2 Non-functional requirements

- **NFR01 – User Friendly**

Table 7 : User Friendly UI

NFR 01	The GUI of the android application and website will be very user friendly.

Table 7: Shows that our app and other systems will be user friendly and very easy to understand.

- **NFR02 – Simple and minimalistic design**

Table 8 : NFR02-Simple and minimalistic design

NFR02	The design of the application will be simple and minimalistic making it extremely easy to check out all the options and use them.

Table 8: Shows that our app will have an ultra-simple and minimalistic design which will help in transparency and understanding.

- **NFR03 – Portability**

Table 9 : NFR03- Portability

NFR 03	The application will be usable on any device with an Android version of 6.0.1 or higher.

Table 9: Shows that our app will be usable on any android device which has an android version of 6.0.1 Marshmallow or higher. Unfortunately, user's below 6.0.1 android version cannot use our android application.

❖ **NFR04 – Documentations and help**

Table 10 : NFR04-Documentations and help

NFR 04	Complete documentation regarding development process will be provided. A demo regarding system usage and working functionalities will also be provided.

Table 10: Shows that there will be complete documentation about the application and the system which shall be provided. Application working demo will also be provided to the visualize the complete working of both applications Web and Android.

❖ **NFR05 – Responsive and fast**

Table 11 : NFR05-Responsive and fast

NFR 05	The video handling to generating output videos and uploading information to the firebase database will be fast enough. The application will be quick to generate results.

Table 11: Shows that our app will be highly responsive and quick to generate results as well as showing them to the user. There'll be no unnecessary waiting.

❖ **NFR06 – Accuracy**

Table 12 : NFR06-Accuracy

NFR 06	The application will generate results based on the trained model that will be reasonably accurate. Training will be done on quite reliable datasets.

Table 12: Shows that our app will generate results that are accurate up to a very good percentage so that the user can trust the results and the app.

2.5 Use case descriptions and design

2.5.1 Web App

The web application is built using Flask Framework. A custom trained model is embedded at the backend of web app to perform detections and save results in database.

✓ Description

Table 13 : Use Case-App Side

Use case ID: 01
Use case name: Road Pulse Web App
Actors: Traffic control officer, live stream Manager
Use Case Summary: The Desktop application handles the video acting as live stream from the security cameras. Violation and anomaly should be detected using the trained deep neural network model YOLO v3.
Pre-condition: proper internet connection, basic knowledge to handle the site
Course of events: The officer can monitor the video acting as a live stream from the camera and if any violation occurs information must be saved in our real-time database Google Firebase. That information is sent to the traffic officer or relevant department.
Postcondition: Violation is detected using the custom trained YOLO v3 model which is embedded in the site (Working as a backend)
Use Case Cross References:
Includes: Add account, update the account, delete the account, Detection of the anomaly using the trained model, save info, send info
Extends: generate information

Table 13: shows us that the use case “Road Pulse Web App” with the use case ID “01” explains the workflow of the web application side of the project. This application side is the one that the traffic control officer is going to deal with. There are two actors in this use case, the Traffic control officer and live stream Manager. The Traffic control officer performs many of the tasks.

✓ Design

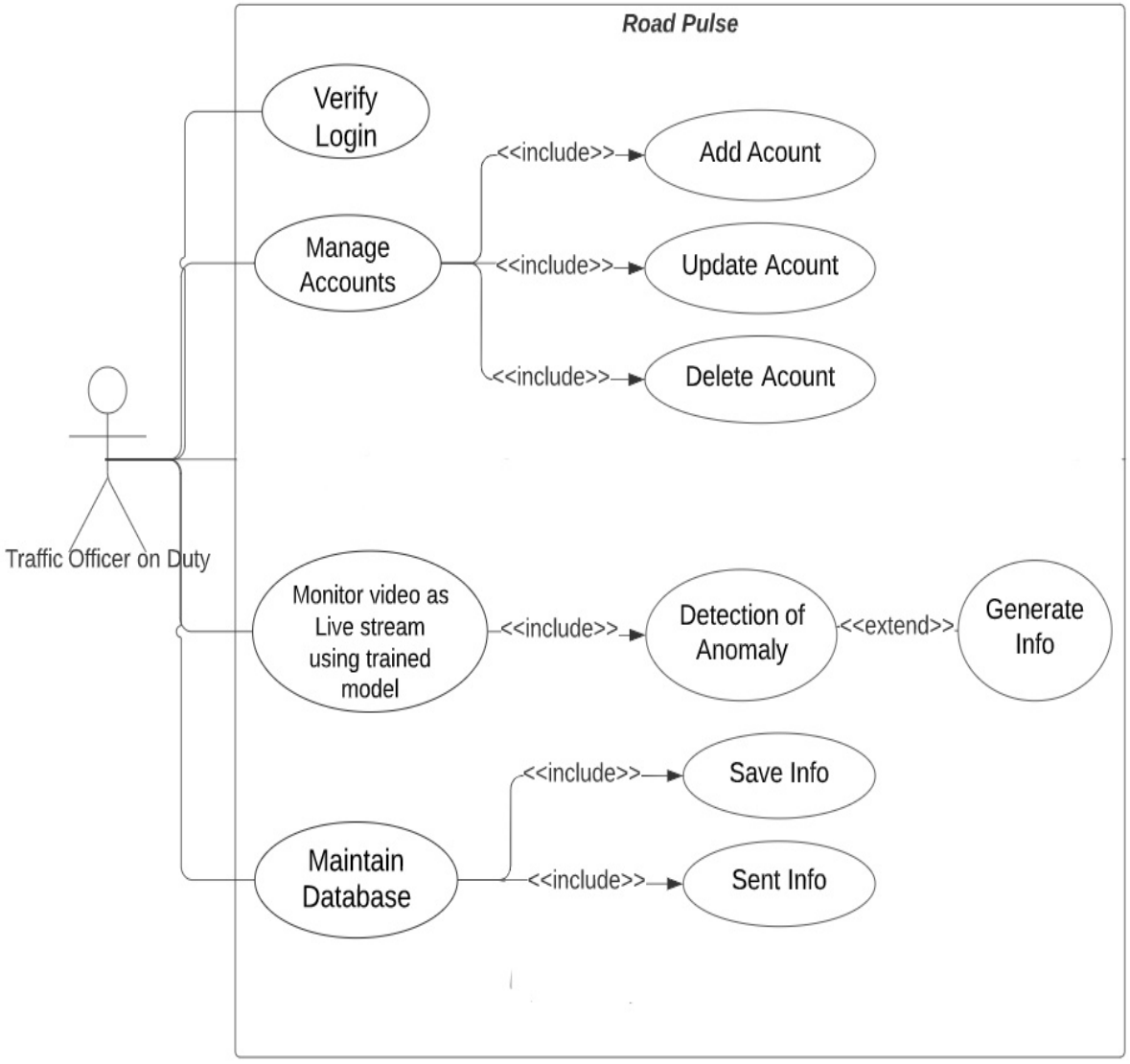


Figure 5: Use case: Web App Side

Figure 5 represents the use case diagram of the Web application. Actors and system modules are shown. The traffic officer on duty is the main actor here. The officer can choose any video for processing and can monitor the video processing to get the results. The complete video and clipped frames are also saved at the backend for later use. The clipped frames are converted into a video clip using OpenCV functions.

2.5.2 Android App side

Android application is synced with web app database. Retrieve anomaly info in real time to track the anomalous events.

- ✓ **Description**

Table 14: Use Case-Android App module

Use case ID: 02
Use case name: Road Pulse Android App
Actors: Officer on Duty
Use Case Summary: The officer acts upon the information received by the web application that include anomaly frame location that can be tracked, time and date will send to the officer. Location can be pinned on the navigation to reach on spot.
Pre-condition: The officer must have access to the internet/server to perform the tasks.
Course of events: The officer opens the app to see the current violation information. After that acts in real-time. After handling the situation updates the status.
Postcondition: None
Use Case Cross References:
Includes: location pinned on the map, update status
Extends: Check information, mark as done, mark as pending

Table 14: shows us that the use case with ID 02 titled “Road Pulse Android App” explains how everything works at the end of the android application. The information regarding a certain crime(violation) or anomaly (Vehicle Crash) can be sent to the mobile application of the police officers on duty near to that area. So that certain actions can be taken in real-time.

✓ Design

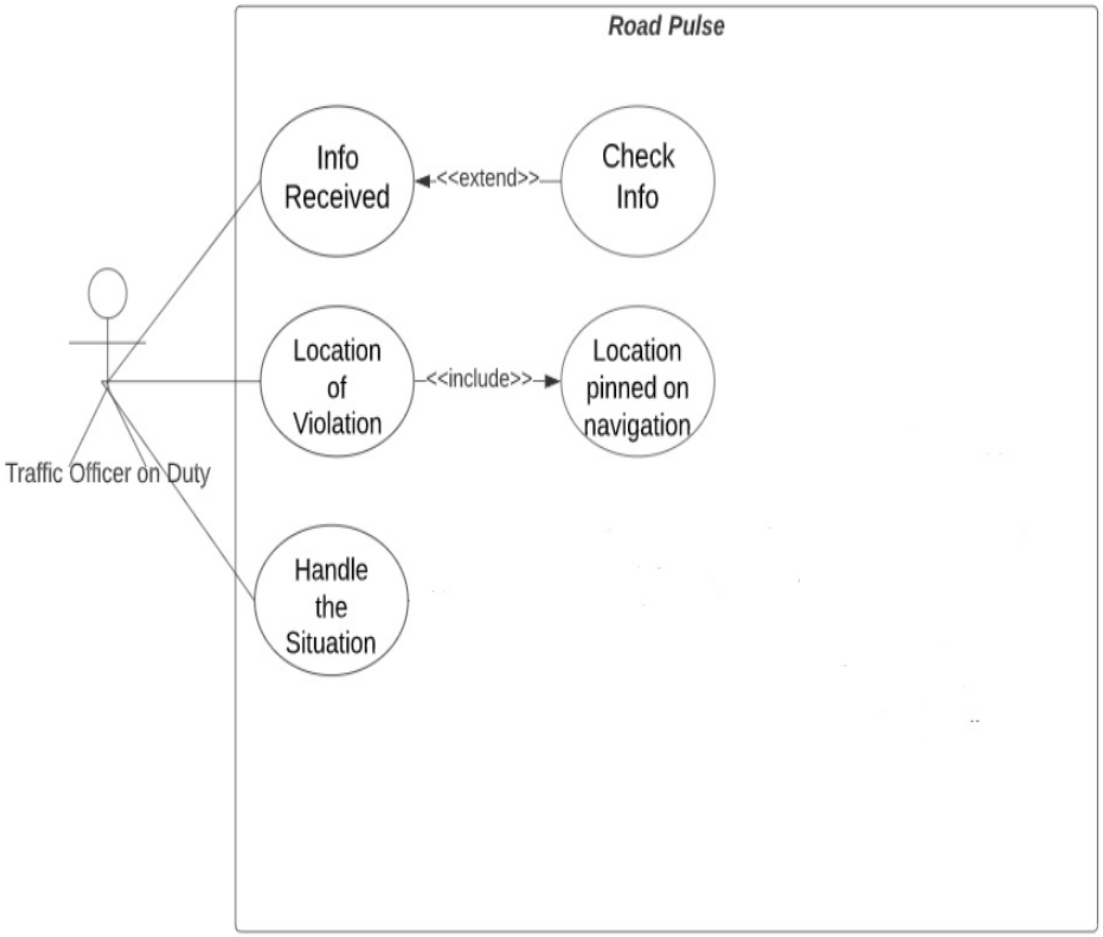


Figure 6: Mobile App Module

Figure 6 shows us the use case of the Android App. There is only one actor in this use case:

- Officer on duty

This figure explains to us that the main actor is the Officer on duty.: The officer acts upon the information received by the web aphtha location, time and date is sent to the officer. Location can be pinned on the navigation to reach on spot. After reaching the spot officer can handling the situation.

Chapter 3

System Design

3 System Design

Chapter 3 shows all the designs that were created for this project:

3.1 Work breakdown structure (WBS)

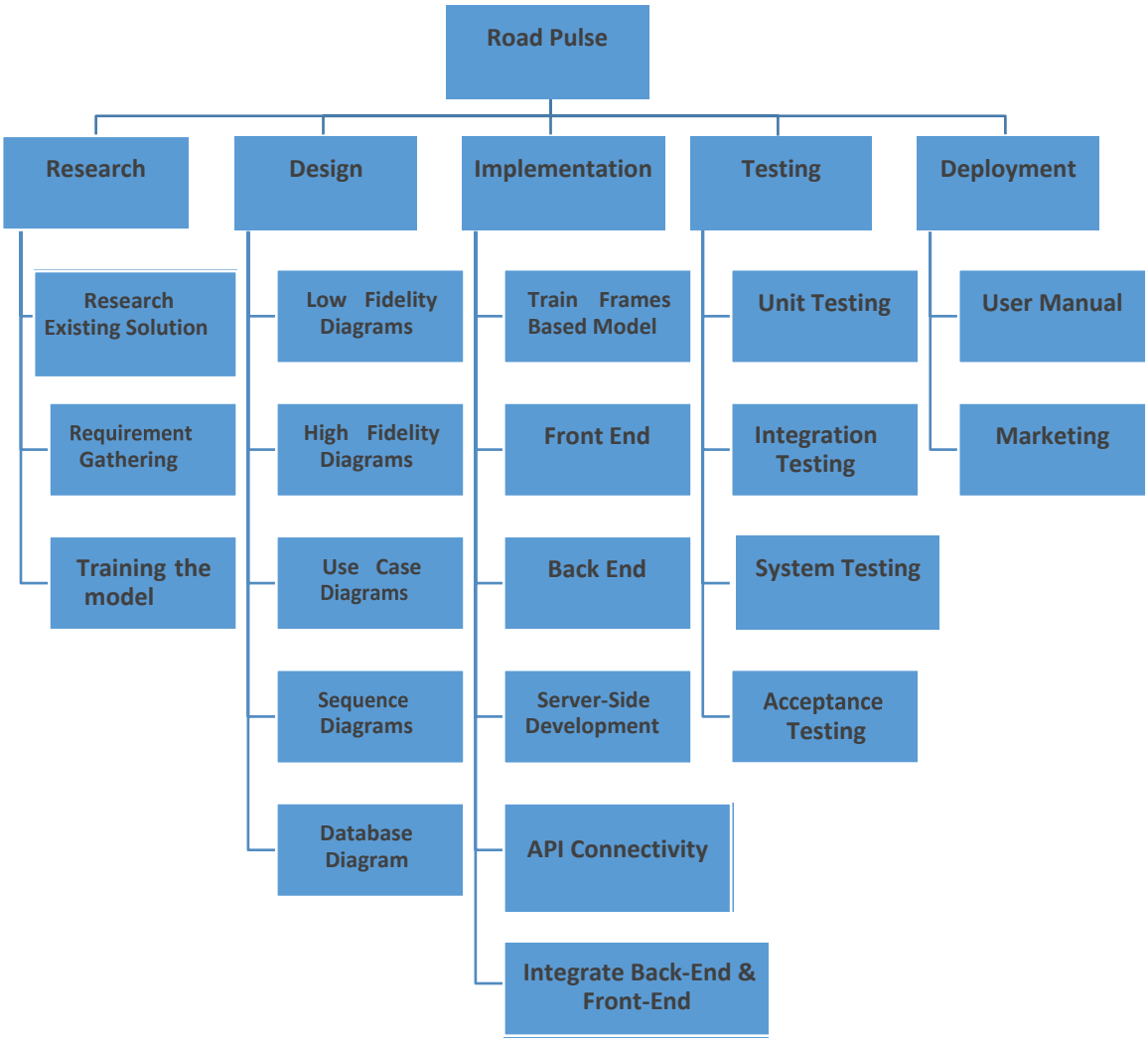


Figure 7: Work Break Down Structure Diagram

Figure 7 shows us the work breakdown structure of the whole project starting from research until deployment. It explains all the workings of the whole project.

3.2 Activity Diagram:

3.2.1 Login:

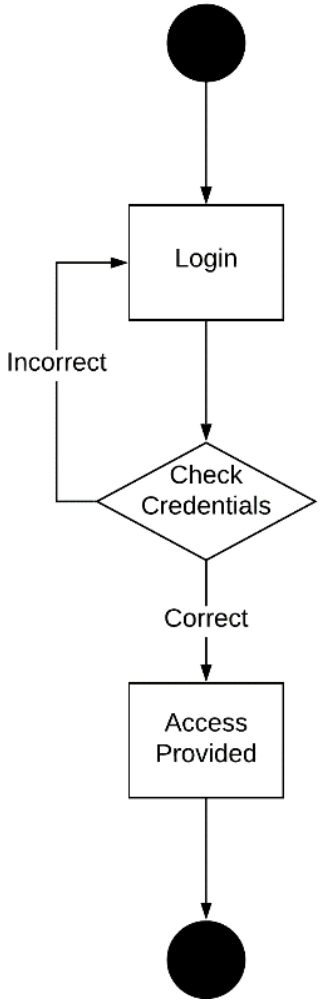


Figure 8 Login

The Figure 8 above presents the activity sequence for system login. The user enters their credentials, which are authenticated with Google Firebase based backend server. If login information is incorrect, the user is asked to enter the correct credentials once again. If correct login information is provided, the used is navigated to the dashboard screen and access is provided to use system functionalities.

3.2.2 Process video through model:

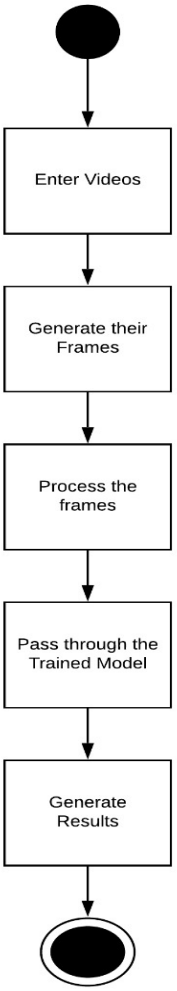


Figure 9 : Video Processing Through YOLO v3

Figure 9 represents the activity diagram of the flow of activities of passing videos through the model YOLO v3

3.3 Sequence diagram

Sequence diagrams explain the whole working of the system.

1) Web-Based to Handle Video Processing and Run Detection

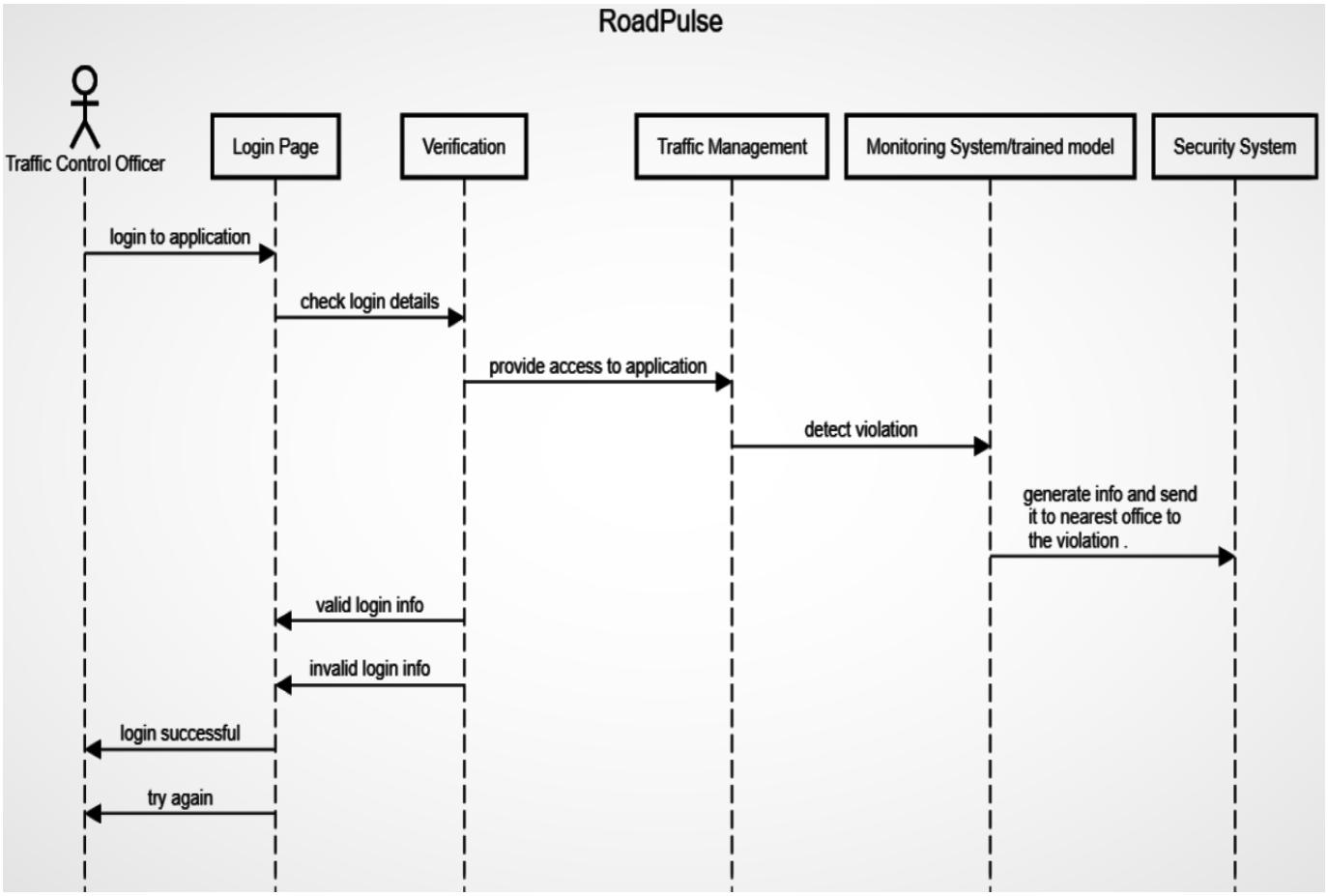


Figure 10: Web-Based App for handling Live stream and detection

Figure 10 explains the workflow of the entire sequence of how a traffic control officer will interact with the web-based application having a series of actions. The live stream and the detection regarding violations and anomalies can be handled here. Information generated can be sent to the android module. So that the officer on duty can respond in real-time.

2) Android App sequence diagram:

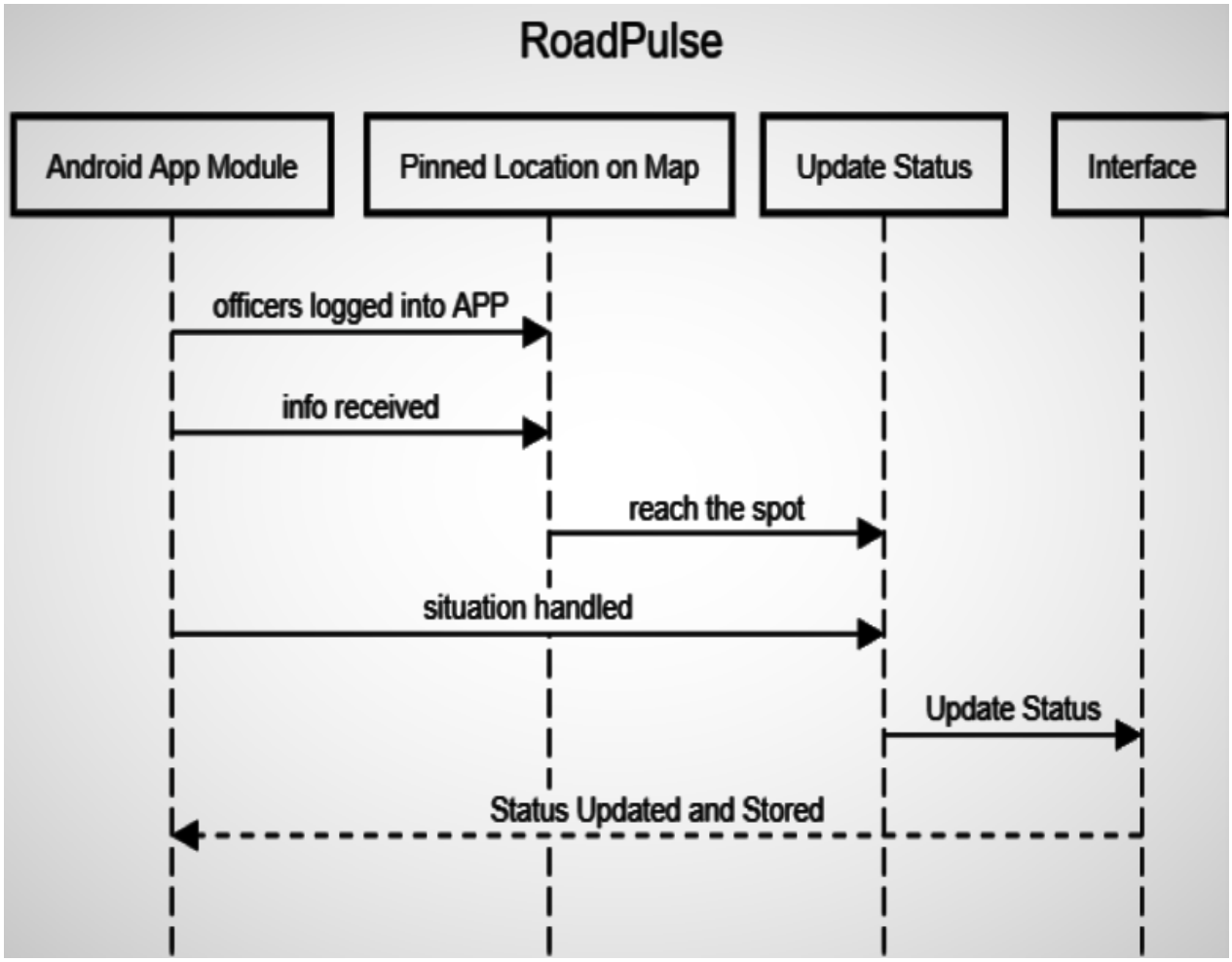


Figure 11: Android App sequence diagram

Figure 11: This module helps the officer on duty to handle the situations like violations happened near to him or any anomaly like the crash in real-time. Access and View events frames + information of events detected by the web application on android application, in chronological order, residing in backend database. The app can assist him to get the information about such events and navigate to the spot in quick time. Location can be pinned on the navigation to reach on spot. After reaching the spot officer can handling the situation.

3.4 System Architecture

The following image represents how system will work:

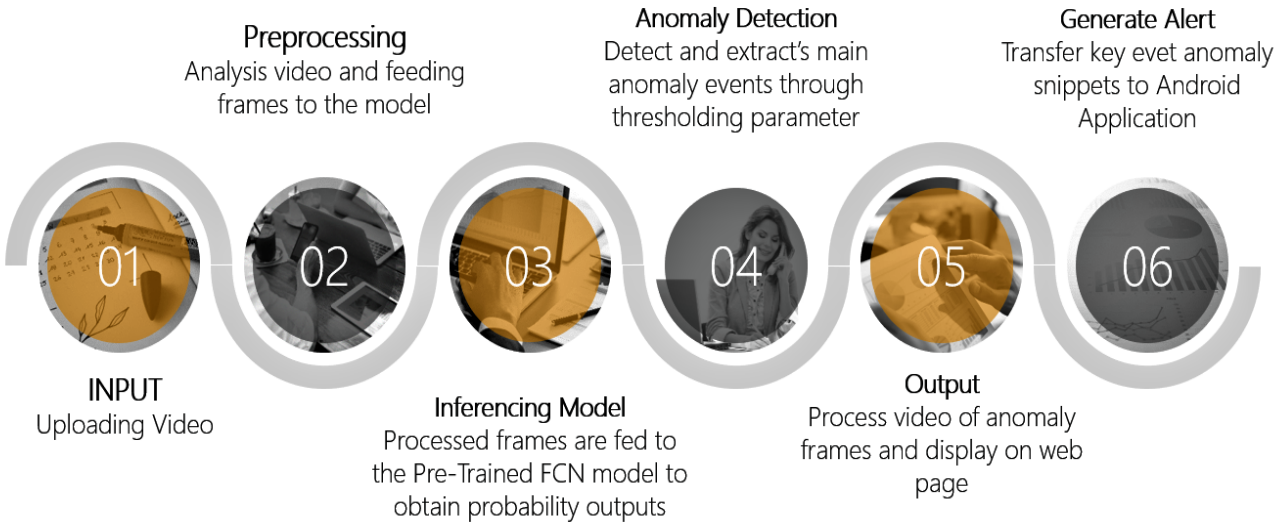


Figure 12: System Architecture & Flow

The Figure 12 above presents the System Architecture for the Road Pulse Application, that takes the input video through user and exploits the AI on the Edge capability of FCN YOLO v3 Embedded Platform, to perform Violation and Anomaly Detection inference.

3.4.1 Violation & Anomaly Detection:

The key steps performed for detection:

- ✓ For each of the frames extracted from the input video. Predict the anomaly scores using our custom trained YOLO V3 model.
- ✓ Run detection after reading video frames and prediction score is then assigned to each frame in that video clip.
- ✓ Get a score for each frame of the video based upon statistical analysis.
- ✓ If there is a deviation in scores, assume that an anomaly must have occurred in that part.

3.5 Pipeline created for the Model (YOLO in our case)

3.5.1 Training on datasets:

The training of the model using AI City Challenge Dataset 2019 track 4 dataset to detect the both classes Lane violation and Vehicle Crash accurately:

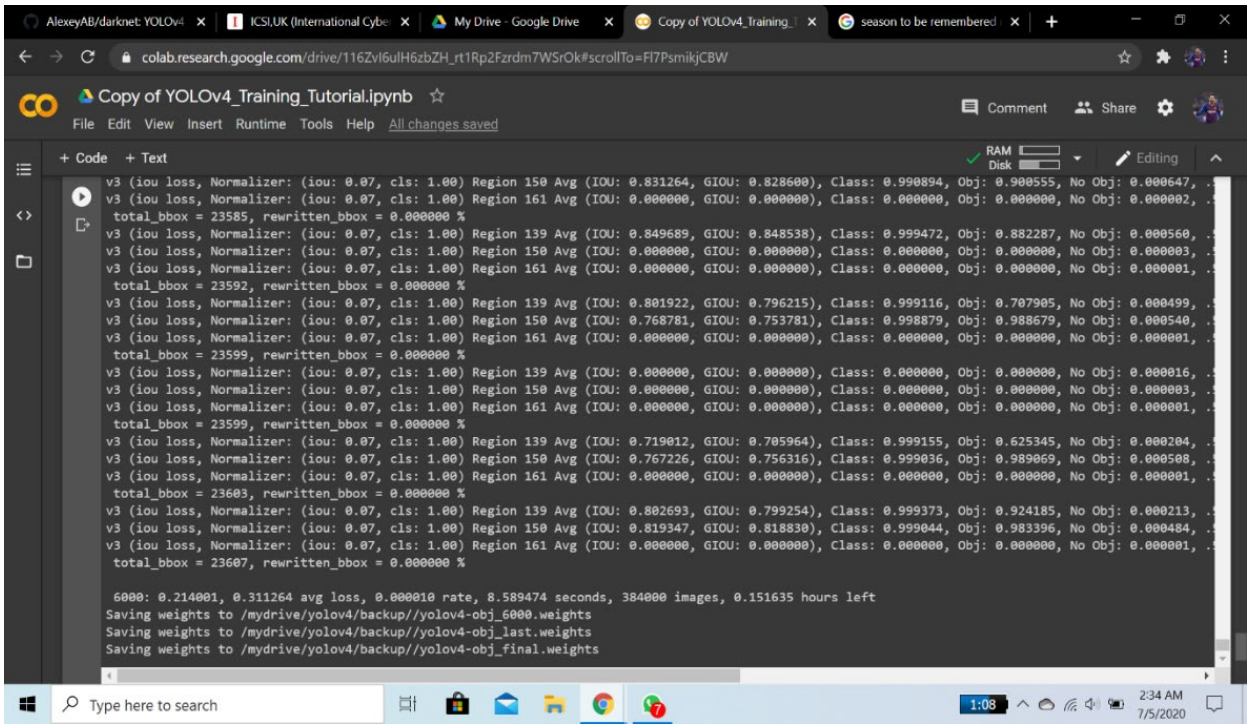


Figure 13: Training process of the model

Figure 13 represents the training process screen of our Model YOLO v3. The images show the number of iterations, average loss, epochs and learning rate, etc. The purpose is to show the training pipeline.

3.5.2 Custom Training:

In training process, we trained our model on custom classes. Dataset used for training process is AI City Challenge 2019. At first faced many issues like dependencies, lack of knowledge etc. but gradually we successfully trained our model using Google Colab GPU environment.

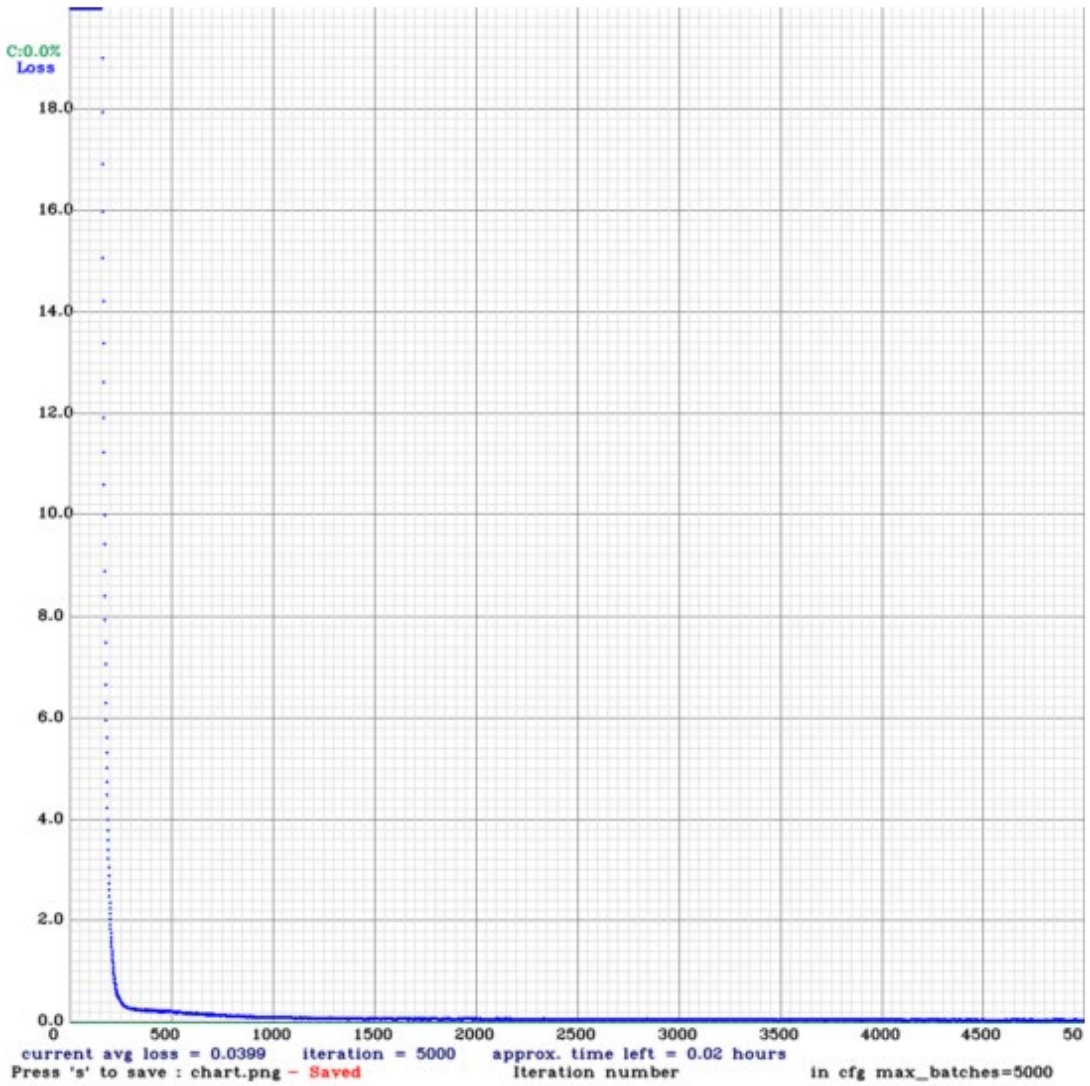


Figure 14 : Training Graph Generated after Training Process

Figure 14 shows a graph generating after training the YOLO v3 on custom classes. The graph shows 'Loss Per Iteration'. We achieved an average loss value of 0.0399 which is the best value for a training process. The max_batch size was set to 5000 according to the class problems requirement. No of iterations performed were 5k almost and custom weight files generated were saved in our mounted G-drive.

Further custom files were made for training process which include CFG file, Text file containing list of all training images using python script, custom classes.name file and obj.names .

The filter size for training was calculated using YOLO v3 formula (No. of classes + 5) * 3. In our case it was 21 for each convolutional layer in the CFG file.

Dataset was uploaded into the G-drive so that we can mount the google drive with colab for training.

3.5.3 Detection on Real world videos:

Currently, the model is trained to provide more accuracy and faster learning to perform detection on unseen data. Both frames (images) and videos can be passed to the model to perform detection. Some testing was performed on the unseen examples and the results are as follow:

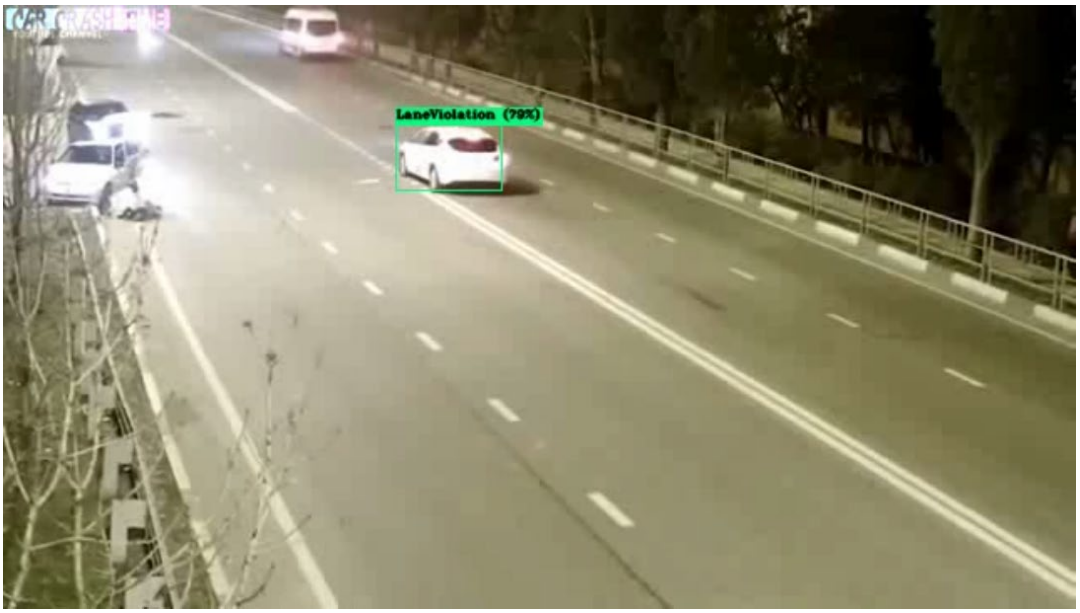


Figure 15: Detection after training on real world videos

Figure 15 shows the detection of vehicles related to the class problem of lane violation. As you can see the results on unseen data are only acceptable because firstly the dataset is based on foreign countries road conditions and secondly the dataset is limited only 100 train videos were trained to the model. No dataset is available according to the Pakistan roads conditions.

3.5.4 Training Pipeline:

- Trained our model on custom classes. Custom classes include Lane Violation and vehicle crash.
- The weight files were saved in form iterations around after every 2000 frames. This was done to be on the safe side that if training stops at any point, we can resume the training.
- The training process was done on Google Collab using GPU environment.
- The accuracy of the model on test videos were around 75 to 85 % as the frames were limited around 10k frames were trained.

3.5.5 Dataset used:

- AI City Challenge 2019 [10]
- No of classes = 2
- Classes Names = Lane Violation & Vehicle Crash
- Plain Dataset which means no annotations were provided with the dataset.
- 100 Train + 100 Test Videos

Submitted online form for the access of AI City Dataset 2019. The dataset received was not annotated. Anomalies in the dataset were clipped by watching every video and were labelled manually

3.6 MPVIR Webpage

Webpage regarding our final year project maintained to present our complete project and its resources.

❖ **Home Page of the website**

This is just a simple website for the project. We are still working on it many options will be available in the future.

All the research we did will working on this project is available on this site [10]. The webpage is update on weekly basis any progress made is update on this webpage. Anyone can access the site and use the information provided for research basis.

Figure 16 : Homepage of MPVIR Group

3.7 Tools Required:

Figure 17: Tools Required

Figure 17 shows the tools and techniques that we used to develop the project. The tools like Collab, Python, TensorFlow are used to train the model. The rest of the tools are used to develop the Web and Android Applications.

3.8 Our Team:

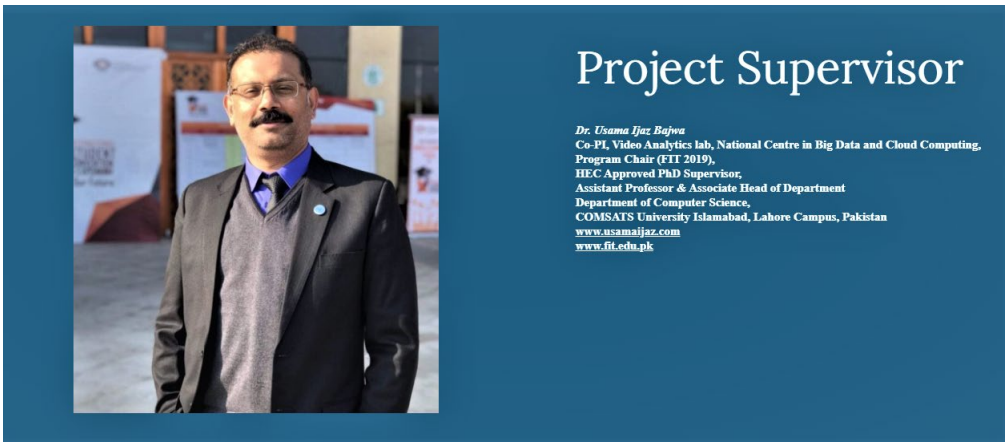


Figure 18 Project Supervisor Dr Usama

Figure 19 : Groupmates for FYP

Figure 19 shows the team of Road Pulse. The whole development was divided between two of us.

3.9 Project Key Milestones

Table 14: Project deliverables and Key Milestones

Elapsed time in (days or weeks or month or quarter) since the start of the project	Milestone	Deliverable
Month 1	Research existing solutions, Requirement gathering, Prioritizing the requirements	SRS Document
Month 2	Documentation (Use cases, Sequence diagrams)	Design Document
Month 3	Gathering the data to train the model, Designing the Interfaces	
Month 4	Development of interfaces	High-fidelity Document
Month 5	Training of the model	
Month 6	Integrating the front-end and back-end	
Month 7	API connectivity	Working prototype
Month 8	Testing of prototype	Working application
Month 9	Formal testing of web and android application	User manual
Month 10	Formal testing of application	Test cases
Month 11	Deployment	Documentation and the final product

Table 14 shows the key milestones of the project and the deliverables of the project.

Chapter 4

System Testing

4 System Testing

4.1 Test Cases

4.1.1 Test Case-01

This test case Table 15 is for the Functional Requirement: Login. It explains what users can do and what can go wrong if not tested. This test case aims to log in with the correct login credentials.

Table 15 : Test Case-01

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

4.1.2 Test Case-02

This test case Table 16 is for the Functional Requirement-FR02: Anomaly and Violation detection.

Table 16 : Test Case-02

Test case Name	TC-02
Application Name	Road Pulse
Use Case	Anomaly and Violation Detection
Input Summary	Video selected by the user through upload screen for processing
Output Summary	SUCCESS: The anomaly is detected, the user is notified, and the violation and anomaly image(s) and video clip are generated.
Pre-Conditions	Video uploaded by the user
Post-Conditions	The video clip and anomaly frames are sent from the firebase database to the user interface.

4.1.3 Test Case-03

This test case table 17 is for the Functional Requirement FR-03: Video Storage

Table 17 : Test Case-03

Test case Name	TC-03
Application Name	Road Pulse
Input Summary	Data storage in Firebase
Output Summary	SUCCESS: The uploaded video and all the anomaly detected clip(s) are stored on firebase database.
Pre-Conditions	Upload video from system
Post-Conditions	Videos and clips are stored on firebase database.

4.2 Unit Testing

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

4.3 Integration testing

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

The following functionalities were checked:

- ✓ Adding a Video Clip from the system to detect if there is any anomaly or not.
- ✓ Save video clip and frames to the Google firebase database
- ✓ Check the if collections in the database were made
- ✓ Sync the web app database with android application
- ✓ Retrieving data through android application

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

4.4 Acceptance testing

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

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

- ✓ Authorization of user to login.
- ✓ Detection of a violation and crash anomaly.
- ✓ Extraction of frames and video clip (of crash or lane event) where the anomaly or violation has occurred.
- ✓ Storing the image and video clip to the firebase database
- ✓ Sync database with android application
- ✓ Retrieve saved anomaly frames on android app to track and navigate

Our application fulfils the listed requirements.

Chapter 5

Application Front-End

5 Application Frontend

This section contains screenshots of the various screens of Road Pulse's Android and Web application Interface.

5.1 Web User Interface:

5.1.1 Login Screen:

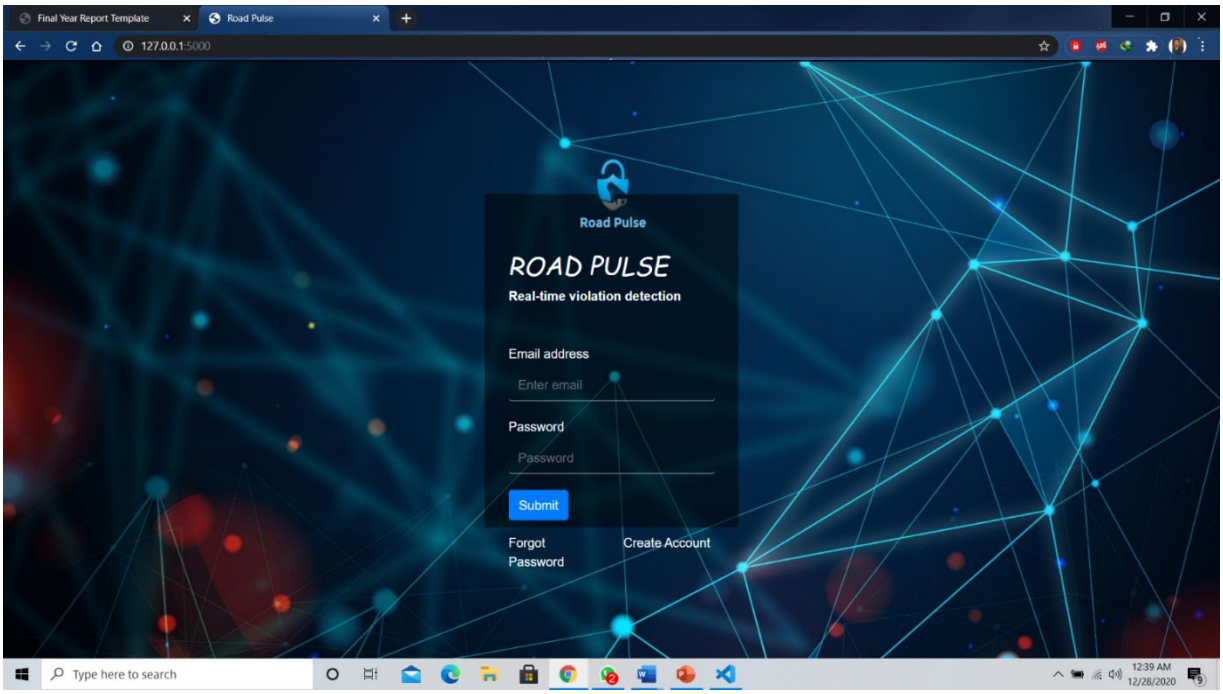


Figure 20 : Login Screen for Web App

The Figure 20 above presents the Login Screen of Flask (python framework) based Web User Interface of Road Pulse web application, where the user can enter authorized credentials to access the application features. The user will have to provide validated Username and Password to gain access to the system services. The authentication is done using Google Firebase where all the records of authorized system users are saved. Only authorized users can access the system.

5.1.2 Reset Password:

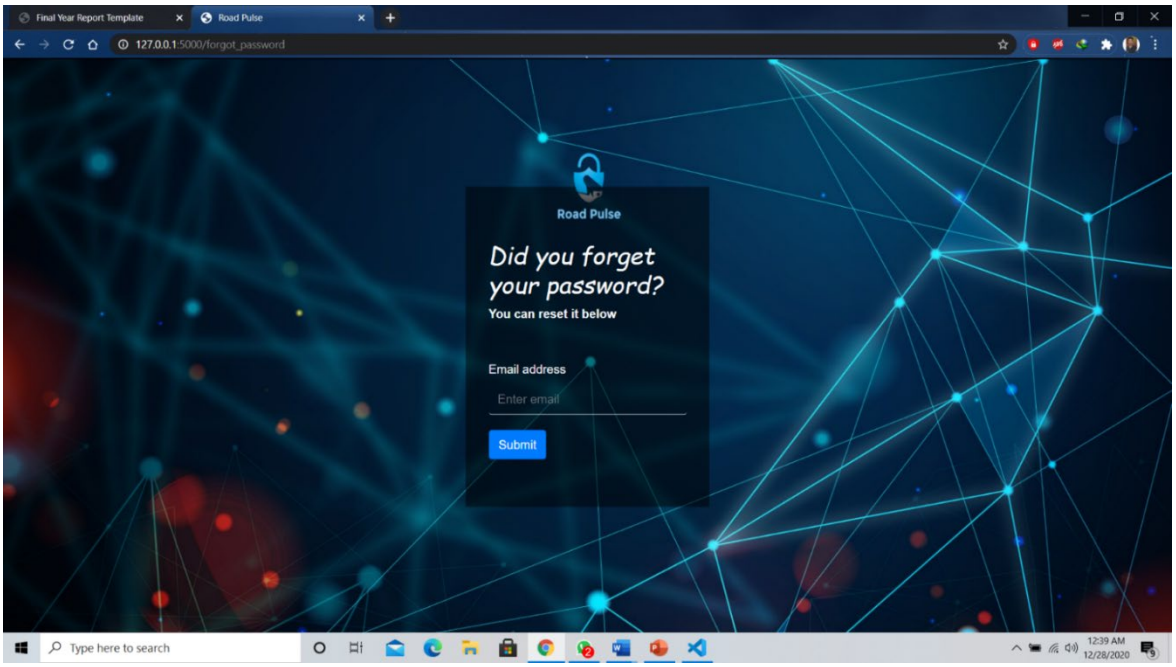


Figure 21 : Forgot password Screen to reset the password

Figure 21 represents a Password Recovery Screen, well its obvious many of forgot there password somehow. But if you forgot your password you can reset your password using your registered or granted email. An online password recovery mail is set to the user email through which password can be reset. Once the password is reset it is automatically updated in our Firebase database.

5.1.3 Dashboard Screen:

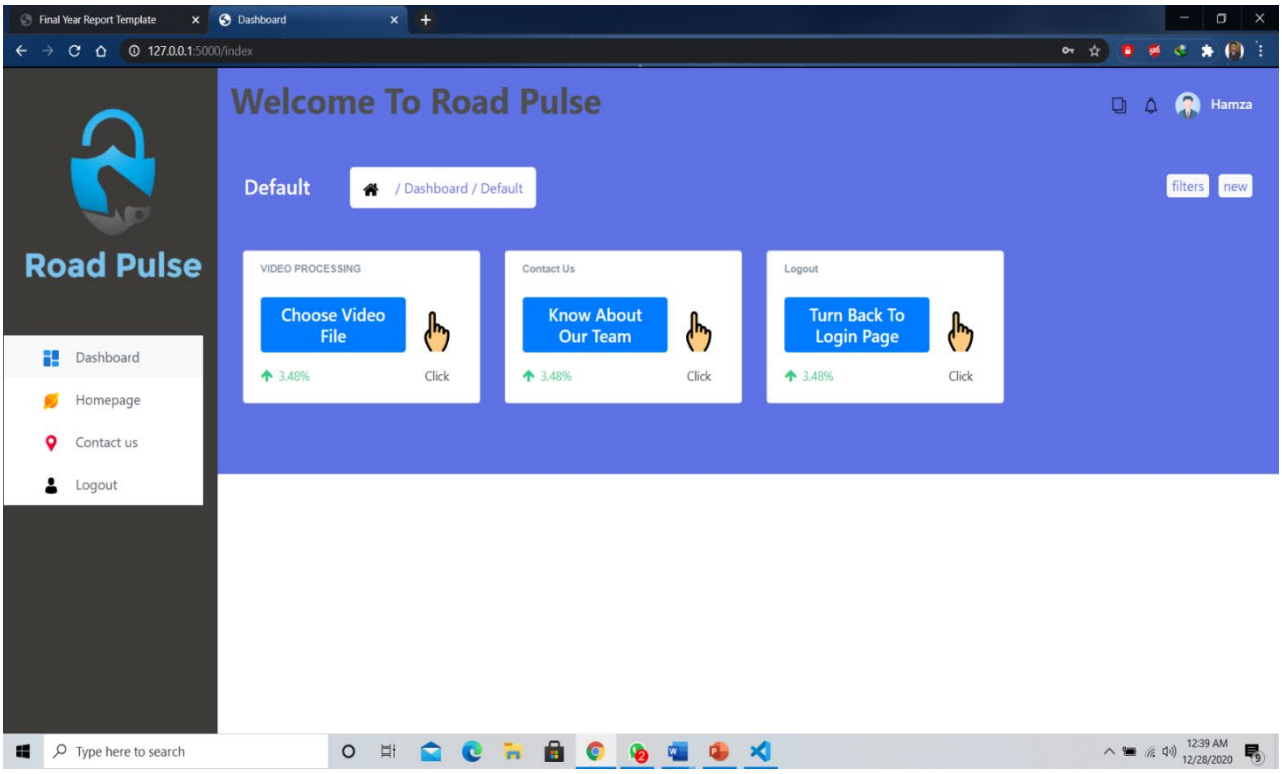


Figure 22 : Dashboard for User

Figure 22 shows the dashboard screen where can choose desired option. User can directly proceed to video processing page to use our system's main functionalities. This screen represents the information well organized and its user friendly so that user can easily navigate to the processing page or know information about our team.

5.1.4 Video Uploading Through Server API:

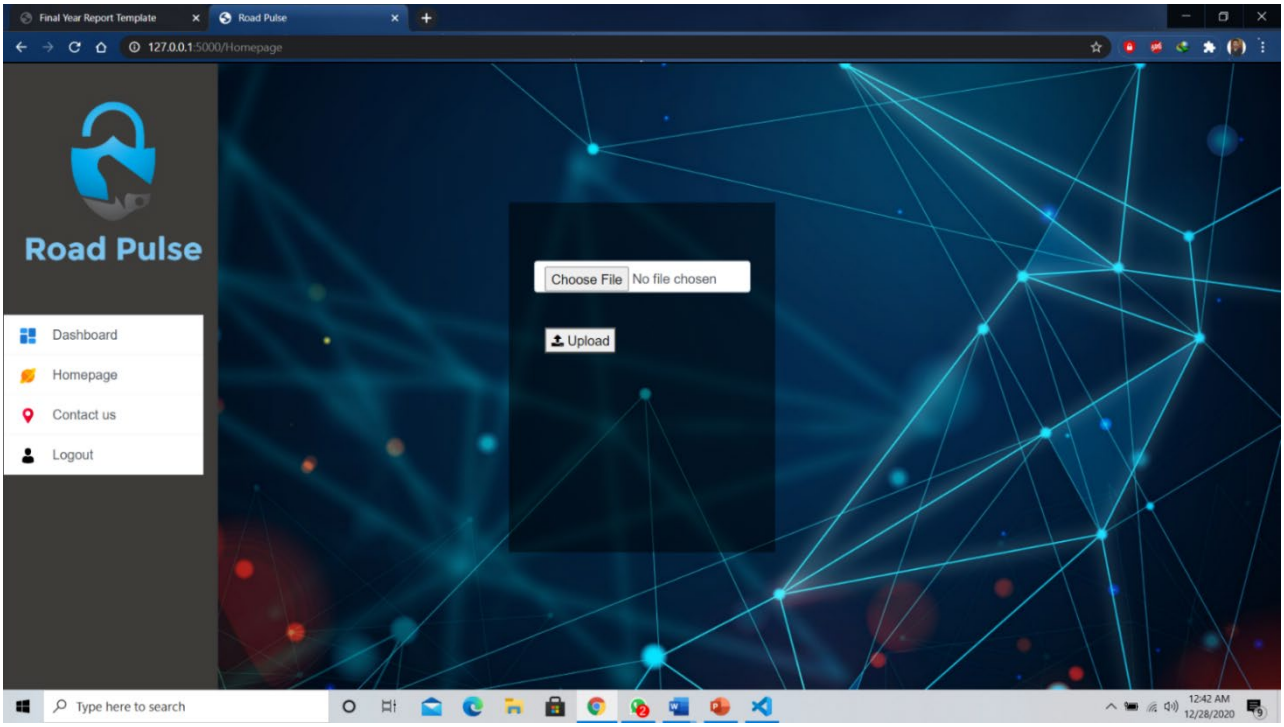


Figure 23 : Video Selection to start processing

Figure 23 shows a screen where can upload video through Flask server API. As soon the video is upload to the server processing is started at the backend. How does the user will know what is happening? well we have solution the backend processing is displayed to the user using browser console. Below you can see frames and background processing:

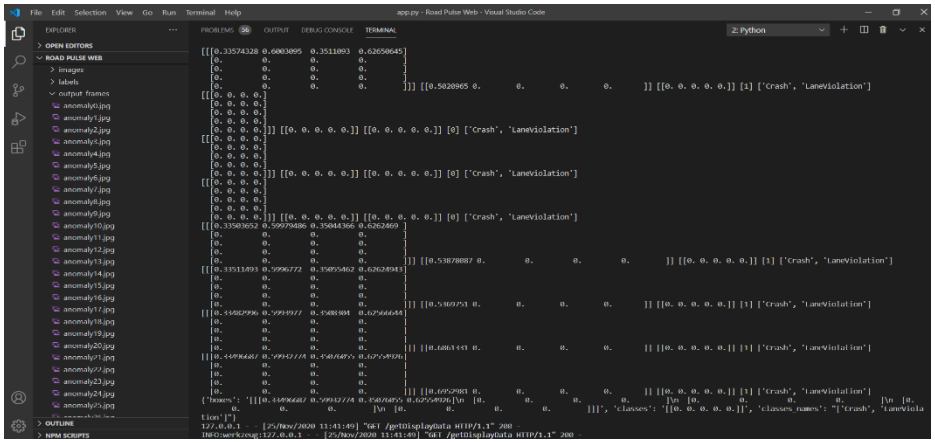


Figure 24 : Backend Processing

Here in Figure 24 is the screen shot where you can see the browser console where all the backend processing is displayed to the user. We used this approach so that user can stay update about the processing of the video. As user cannot see what's happening in the backend

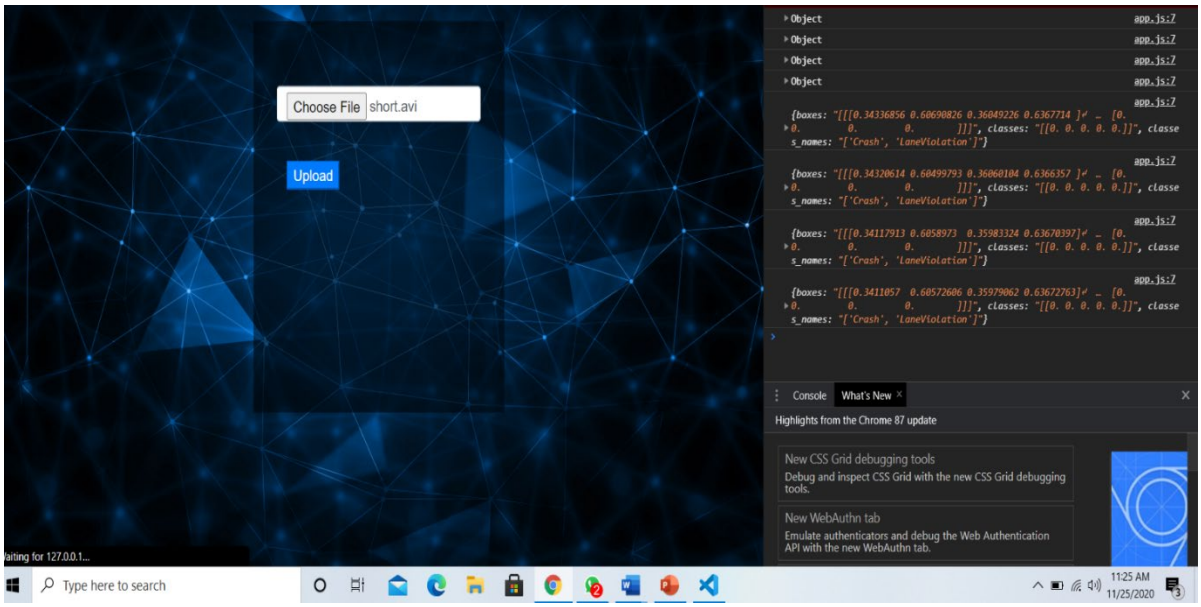


Figure 25 : Backend Processing in User Browser Console

In Figure 25 the processing terms include computational results using our embedded model YOLO v3 trained on custom classes to produce final results. Probabilities, bounding boxes, classes and their probabilities are displayed in the browser console.

5.1.5 Processed Output Screen:

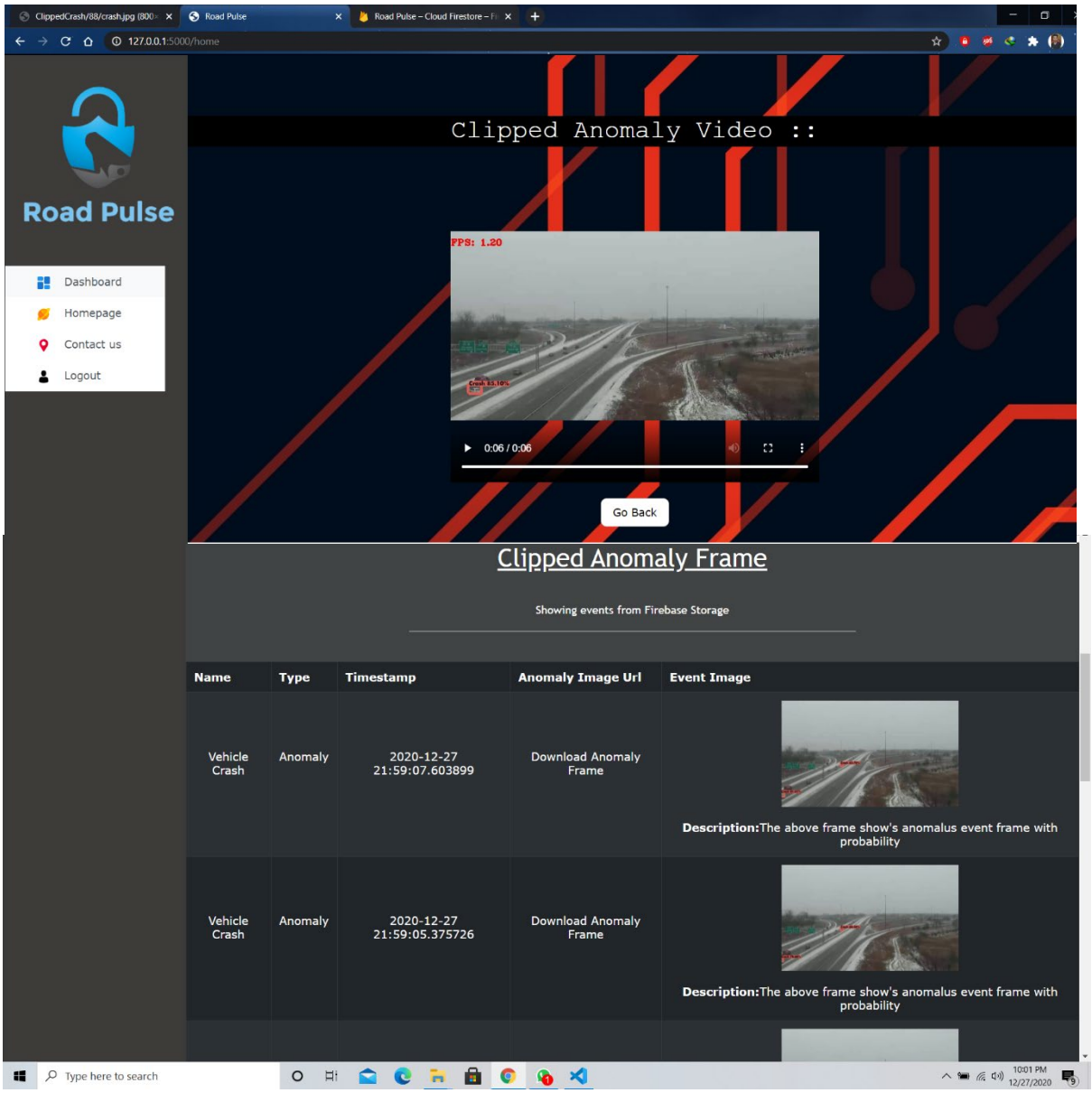


Figure 26 : Processed Output Screen

Figure 26 shows the output screen after video processing. The output video here is generated by our embedded model YOLO v3 at the backend. The model is trained on custom dataset to detect two main type of anomaly Lane violation and Vehicle crash. User can view the detection video on our web page which shows detection on the processed video. All the processing is done on CPU environment. The table below presents the information regarding the anomalous

events detected in the video. High probability frames are clipped while processing the video frames at the backend.

After displaying this data to the user all the information including video is saved in our database Google firebase. The data is stored uniquely so the android device user which is a security department representative can access this information to act accordingly. User can also download the anomaly frame using the image URL in the table.

5.1.6 Firebase Firestore:

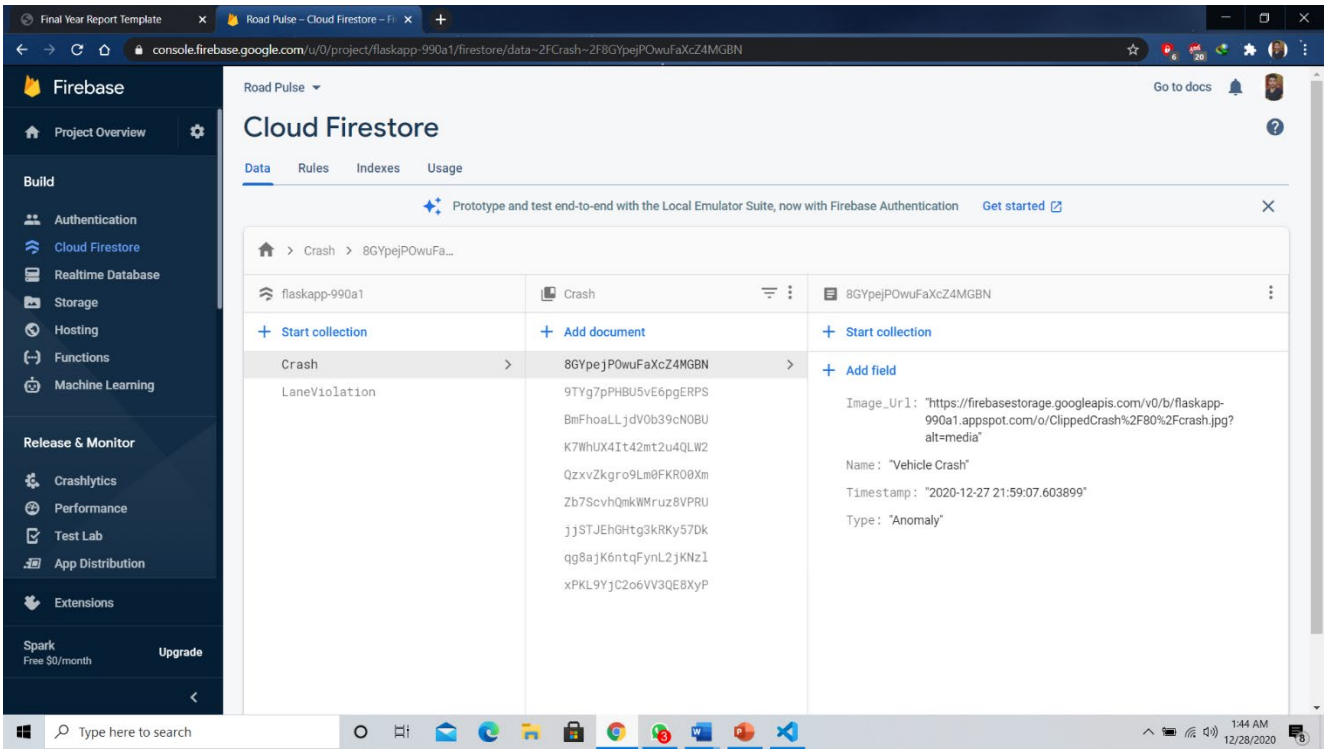


Figure 27 : Firebase Firestore Data Representation

Figure 27 represents data presentation and collections created through web application. This information is further synced to our android application.

5.2 Android Application Frontend

5.2.1 Login Screen

Figure 28 : Login Screen for Android App

Figure 28 reflects the Road Pulse Login Screen for the Android application. To use the functionality of the application, the user must have the valid login credentials (saved in the Google Firebase database). The UI is designed and functionality is developed using Android Studio. We have also provided an option for a user to sign up. The information is automatically stored in the database when the user signs up.

5.2.2 Home Screen:

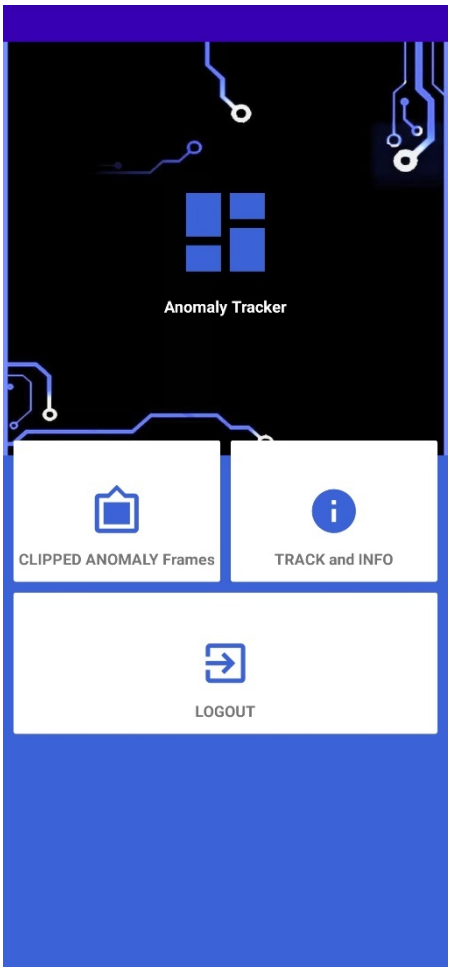


Figure 29 : Homepage of Android App

Figure 29 After logging in the user is directed to this screen, it is the home screen for our application. We have designed it as a simple, user-friendly, easier to understand, and well-organized UI.

The user gets two options i.e., Clipped Anomaly Frames, and Track and Info. The clipped anomaly frames display an anomaly list of current events in the form of multiple frames, which when tapped, submits a request to the database. Resultantly, the information is displayed on the screen in real-time. Similarly, when the Track and info button is pressed, the keyframes with higher probability can be viewed (coming from the firebase database). In real-time, users may even trace the position of an anomalous incident.

5.2.3 Clipped Anomaly Frames:

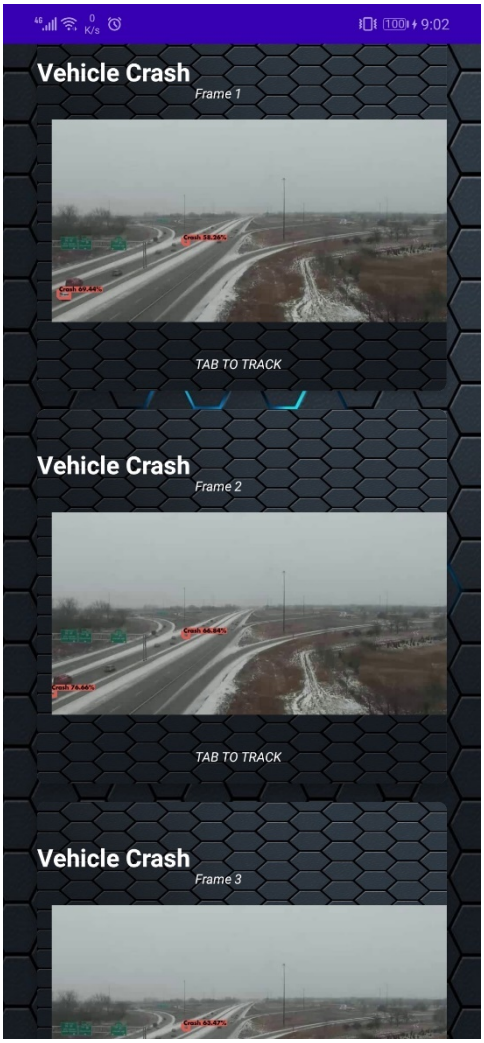


Figure 30 : Clipped Anomaly Screen

Figure 30 The user is directed to this screen after he/she clicks the Clipped Anomaly Frames Button. The multiple frames representing detected anomalous events which are stored in database through Web app processing. Recent anomaly frame is retrieved from firebase database which is synced with web app database. Each displayed frame has different probability value as per detection. To view information regarding a particular event, the user may tap any frame card.

5.2.4 Track and INFO & Navigation Screen:

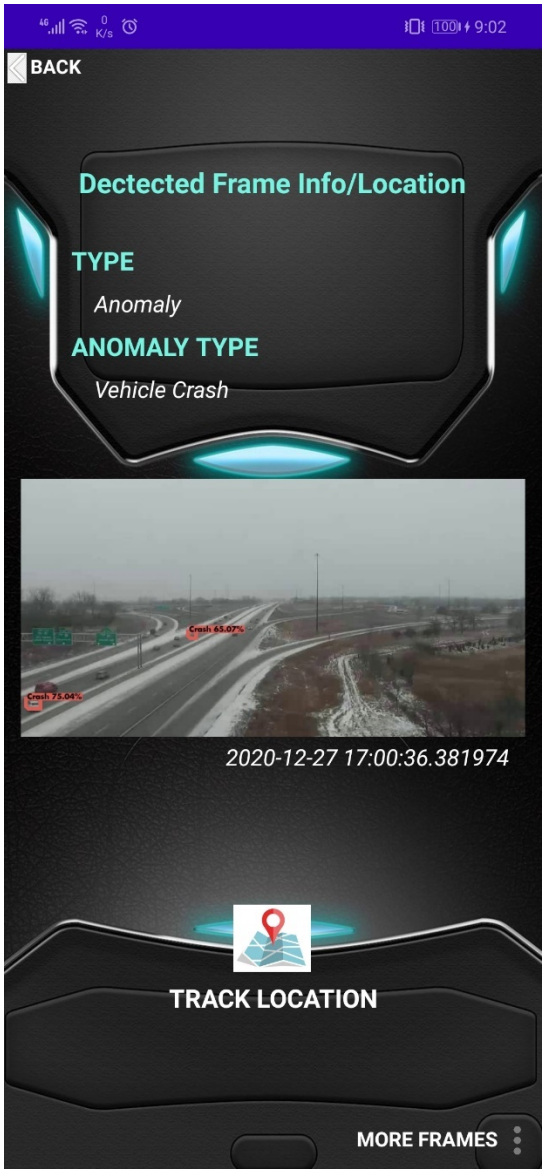


Figure 31 : Track & View INFO Screen

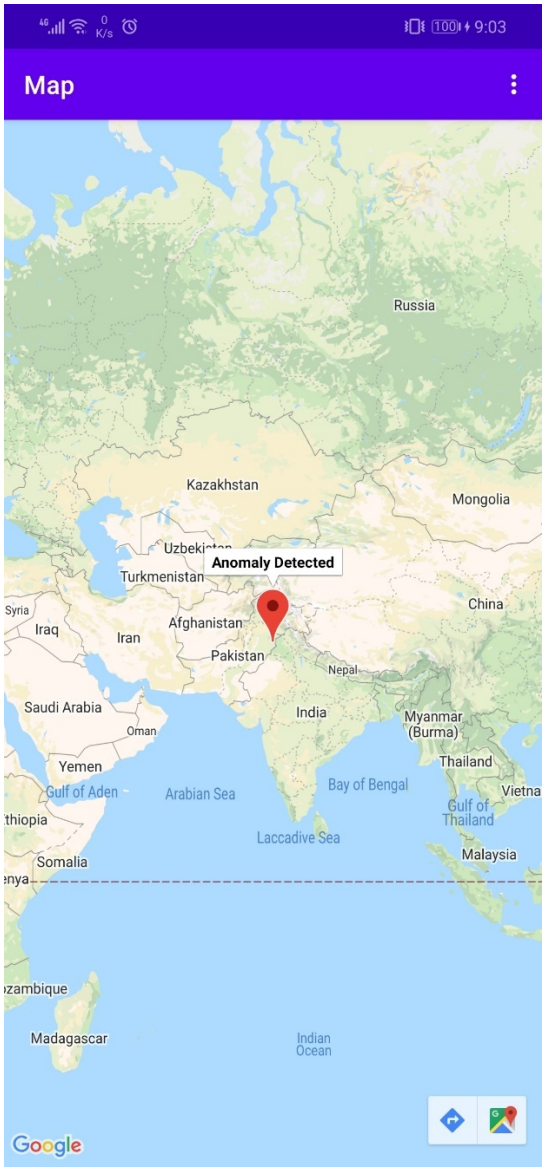


Figure 32: Pinned Location to Navigate

Figure 31 depicts the view of detected frame info/location. The keyframe and its time of occurrence are displayed. The overall information is acquired from the Google Firebase servers, which runs synchronized with the web application on the backend. As soon as the web application generates information, it is saved into the firebase database and can be viewed from the android application. We have also offered users the option of monitoring the event's location in real-time.

In figure 32 the location of the detected anomaly is pinned on the map along with the occurrence time. This entire process is in fact, in real-time. As soon as the anomaly is detected it is updated on the map. The user can navigate to the pinned position by clicking on the google maps option in the bottom right corner and can get directions from his current location to the anomaly location.

The primary goal behind the development of this android application is to target our traffic department and Dolphin Police to track and handle vehicle crash accidents on time to save citizens precious lives.

Chapter 6

Project Code

6 Project Code

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

- YOLO v3 Model
- Save Frames of Violation & Anomalous Events to the folder
- Loading Pre-Trained Weights and Custom Classes File

And many other sections that are vital for the project.

The code is built with the following libraries:

- Flask Framework 1.1.2
- HTML Jinja Templating
- PyTorch version 1.6
- OpenCV version 3.4.2.16
- Tensorflow
- NumPy version 1.18.1
- Pandas version 1.0.1
- Firebase Python SDK

6.1 YOLO v3:

YOLO perform detection using its convolutional layers present in the weight files. Filter size for each convolutional are defined using CFG file that model reads.In our case we trained custom classes using 79 convolutional layered weights to generate our own custom weights.

```
yolo_max_boxes = 5
yolo_iou_threshold = 0.5
yolo_score_threshold = 0.5
# customize your model through the following parameters
flags.DEFINE_integer('yolo_max_boxes', 10, 'maximum number of detectio
ns at one time')
flags.DEFINE_float('yolo_iou_threshold', 0.5, 'iou threshold')
flags.DEFINE_float('yolo_score_threshold', 0.5, 'score threshold')

yolo_anchors = np.array([(10, 13), (16, 30), (33, 23), (30, 61), (62,
45),
(59, 119), (116, 90), (156, 198), (373, 326)],
np.float32) / 416
yolo_anchor_masks = np.array([[6, 7, 8], [3, 4, 5], [0, 1, 2]])

yolo_tiny_anchors = np.array([(10, 14), (23, 27), (37, 58),
(81, 82), (135, 169),  (344, 319)],
np.float32) / 416
yolo_tiny_anchor_masks = np.array([[3, 4, 5], [0, 1, 2]])

def DarknetConv(x, filters, size, strides=1, batch_norm=True):
```

```
if strides == 1:
padding = 'same'
else:
x = ZeroPadding2D(((1, 0), (1, 0)))(x)  # top left half-padding
padding = 'valid'
x = Conv2D(filters=filters, kernel_size=size,
strides=strides, padding=padding,
use_bias=not batch_norm, kernel_regularizer=l2(0.0005))(x)
if batch_norm:
x = BatchNormalization()(x)
x = LeakyReLU(alpha=0.1)(x)
return x

def DarknetResidual(x, filters):
prev = x
x = DarknetConv(x, filters // 2, 1)
x = DarknetConv(x, filters, 3)
x = Add()([prev, x])
return x

def DarknetBlock(x, filters, blocks):
x = DarknetConv(x, filters, 3, strides=2)
for _ in range(blocks):
x = DarknetResidual(x, filters)
return x

def Darknet(name=None):
x = inputs = Input([None, None, 3])
x = DarknetConv(x, 32, 3)
x = DarknetBlock(x, 64, 1)
x = DarknetBlock(x, 128, 2)  # skip connection
x = x_36 = DarknetBlock(x, 256, 8)  # skip connection
x = x_61 = DarknetBlock(x, 512, 8)
x = DarknetBlock(x, 1024, 4)
return tf.keras.Model(inputs, (x_36, x_61, x), name=name)

def DarknetTiny(name=None):
x = inputs = Input([None, None, 3])
x = DarknetConv(x, 16, 3)
x = MaxPool2D(2, 2, 'same')(x)
x = DarknetConv(x, 32, 3)
x = MaxPool2D(2, 2, 'same')(x)
x = DarknetConv(x, 64, 3)
x = MaxPool2D(2, 2, 'same')(x)
x = DarknetConv(x, 128, 3)
x = MaxPool2D(2, 2, 'same')(x)
x = x_8 = DarknetConv(x, 256, 3)  # skip connection
x = MaxPool2D(2, 2, 'same')(x)
x = DarknetConv(x, 512, 3)
x = MaxPool2D(2, 1, 'same')(x)
x = DarknetConv(x, 1024, 3)
return tf.keras.Model(inputs, (x_8, x), name=name)

def YoloConv(filters, name=None):
def yolo_conv(x_in):
if isinstance(x_in, tuple):
inputs = Input(x_in[0].shape[1:]), Input(x_in[1].shape[1:])
x, x_skip = inputs

# concat with skip connection
x = DarknetConv(x, filters, 1)
x = UpSampling2D(2)(x)
x = Concatenate()([x, x_skip])
else:
```

```
x = inputs = Input(x_in.shape[1:])

x = DarknetConv(x, filters, 1)
x = DarknetConv(x, filters * 2, 3)
x = DarknetConv(x, filters, 1)
x = DarknetConv(x, filters * 2, 3)
x = DarknetConv(x, filters, 1)
return Model(inputs, x, name=name)(x_in)
return yolo_conv

def YoloConvTiny(filters, name=None):
def yolo_conv(x_in):
if isinstance(x_in, tuple):
inputs = Input(x_in[0].shape[1:]), Input(x_in[1].shape[1:])
x, x_skip = inputs

# concat with skip connection
x = DarknetConv(x, filters, 1)
x = UpSampling2D(2)(x)
x = Concatenate()([x, x_skip])
else:
x = inputs = Input(x_in.shape[1:])
x = DarknetConv(x, filters, 1)

return Model(inputs, x, name=name)(x_in)
return yolo_conv

def YoloOutput(filters, anchors, classes, name=None):
def yolo_output(x_in):
x = inputs = Input(x_in.shape[1:])
x = DarknetConv(x, filters * 2, 3)
x = DarknetConv(x, anchors * (classes + 5), 1, batch_norm=False)
x = Lambda(lambda x: tf.reshape(x, (-
1, tf.shape(x)[1], tf.shape(x)[2],
anchors, classes + 5)))(x)
return tf.keras.Model(inputs, x, name=name)(x_in)
return yolo_output

def yolo_boxes(pred, anchors, classes):
# pred: (batch_size, grid, grid, anchors, (x, y, w, h, obj, ...classes
))
grid_size = tf.shape(pred)[1]
box_xy, box_wh, objectness, class_probs = tf.split(
pred, (2, 2, 1, classes), axis=-1)

box_xy = tf.sigmoid(box_xy)
objectness = tf.sigmoid(objectness)
class_probs = tf.sigmoid(class_probs)
pred_box = tf.concat((box_xy, box_wh), axis=-
1)  # original xywh for loss

# !!! grid[x][y] == (y, x)
grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size))
grid = tf.expand_dims(tf.stack(grid, axis=-
1), axis=2)  # [gx, gy, 1, 2]

box_xy = (box_xy + tf.cast(grid, tf.float32)) / \
tf.cast(grid_size, tf.float32)
box_wh = tf.exp(box_wh) * anchors

box_x1y1 = box_xy - box_wh / 2
box_x2y2 = box_xy + box_wh / 2
bbox = tf.concat([box_x1y1, box_x2y2], axis=-1)
```

```
return bbox, objectness, class_probs, pred_box

def yolo_nms(outputs, anchors, masks, classes):
# boxes, conf, type
b, c, t = [], [], []

for o in outputs:
b.append(tf.reshape(o[0], (tf.shape(o[0])[0], -1, tf.shape(o[0])[-
1])))
c.append(tf.reshape(o[1], (tf.shape(o[1])[0], -1, tf.shape(o[1])[-
1])))
t.append(tf.reshape(o[2], (tf.shape(o[2])[0], -1, tf.shape(o[2])[-
1])))

bbox = tf.concat(b, axis=1)
confidence = tf.concat(c, axis=1)
class_probs = tf.concat(t, axis=1)

scores = confidence * class_probs
boxes, scores, classes, valid_detections = tf.image.combined_non_max_s
uppression(
boxes=tf.reshape(bbox, (tf.shape(bbox)[0], -1, 1, 4)),
scores=tf.reshape(
scores, (tf.shape(scores)[0], -1, tf.shape(scores)[-1])),
max_output_size_per_class=yolo_max_boxes,
max_total_size=yolo_max_boxes,
iou_threshold=yolo_iou_threshold,
score_threshold=yolo_score_threshold
)

return boxes, scores, classes, valid_detections

def YoloV3(size=None, channels=3, anchors=yolo_anchors,
masks=yolo_anchor_masks, classes=80, training=False):
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
tf.config.experimental.set_memory_growth(physical_devices[0], True)
x = inputs = Input([size, size, channels], name='input')

x_36, x_61, x = Darknet(name='yolo_darknet')(x)

x = YoloConv(512, name='yolo_conv_0')(x)
output_0 = YoloOutput(512, len(masks[0]), classes, name='yolo_output_0
')(x)

x = YoloConv(256, name='yolo_conv_1')((x, x_61))
output_1 = YoloOutput(256, len(masks[1]), classes, name='yolo_output_1
')(x)

x = YoloConv(128, name='yolo_conv_2')((x, x_36))
output_2 = YoloOutput(128, len(masks[2]), classes, name='yolo_output_2
')(x)

if training:
return Model(inputs, (output_0, output_1, output_2), name='yolov3')

boxes_0 = Lambda(lambda x: yolo_boxes(x, anchors[masks[0]], classes),
name='yolo_boxes_0')(output_0)
boxes_1 = Lambda(lambda x: yolo_boxes(x, anchors[masks[1]], classes),
name='yolo_boxes_1')(output_1)
boxes_2 = Lambda(lambda x: yolo_boxes(x, anchors[masks[2]], classes),
name='yolo_boxes_2')(output_2)

outputs = Lambda(lambda x: yolo_nms(x, anchors, masks, classes),
```

```
name='yolo_nms')((boxes_0[:3], boxes_1[:3], boxes_2[:3]))

return Model(inputs, outputs, name='yolov3')

def YoloV3Tiny(size=None, channels=3, anchors=yolo_tiny_anchors,
masks=yolo_tiny_anchor_masks, classes=80, training=False):
physical_devices = tf.config.experimental.list_physical_devices('GPU')
if len(physical_devices) > 0:
tf.config.experimental.set_memory_growth(physical_devices[0], True)
x = inputs = Input([size, size, channels], name='input')

x_8, x = DarknetTiny(name='yolo_darknet')(x)

x = YoloConvTiny(256, name='yolo_conv_0')(x)
output_0 = YoloOutput(256, len(masks[0]), classes, name='yolo_output_0
')(x)

x = YoloConvTiny(128, name='yolo_conv_1')((x, x_8))
output_1 = YoloOutput(128, len(masks[1]), classes, name='yolo_output_1
')(x)

if training:
return Model(inputs, (output_0, output_1), name='yolov3')

boxes_0 = Lambda(lambda x: yolo_boxes(x, anchors[masks[0]], classes),
name='yolo_boxes_0')(output_0)
boxes_1 = Lambda(lambda x: yolo_boxes(x, anchors[masks[1]], classes),
name='yolo_boxes_1')(output_1)
outputs = Lambda(lambda x: yolo_nms(x, anchors, masks, classes),
name='yolo_nms')((boxes_0[:3], boxes_1[:3]))
return Model(inputs, outputs, name='yolov3_tiny')

def YoloLoss(anchors, classes=80, ignore_thresh=0.5):
def yolo_loss(y_true, y_pred):
# 1. transform all pred outputs
# y_pred: (batch_size, grid, grid, anchors, (x, y, w, h, obj, ...cls))
pred_box, pred_obj, pred_class, pred_xywh = yolo_boxes(
y_pred, anchors, classes)
pred_xy = pred_xywh[..., 0:2]
pred_wh = pred_xywh[..., 2:4]

# 2. transform all true outputs
# y_true: (batch_size, grid, grid, anchors, (x1, y1, x2, y2, obj, cls)
)
true_box, true_obj, true_class_idx = tf.split(
y_true, (4, 1, 1), axis=-1)
true_xy = (true_box[..., 0:2] + true_box[..., 2:4]) / 2
true_wh = true_box[..., 2:4] - true_box[..., 0:2]

# give higher weights to small boxes
box_loss_scale = 2 - true_wh[..., 0] * true_wh[..., 1]

# 3. inverting the pred box equations
grid_size = tf.shape(y_true)[1]
grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size))
grid = tf.expand_dims(tf.stack(grid, axis=-1), axis=2)
true_xy = true_xy * tf.cast(grid_size, tf.float32) - \
tf.cast(grid, tf.float32)
true_wh = tf.math.log(true_wh / anchors)
true_wh = tf.where(tf.math.is_inf(true_wh),
tf.zeros_like(true_wh), true_wh)

# 4. calculate all masks
obj_mask = tf.squeeze(true_obj, -1)
```

```
# ignore false positive when iou is over threshold
best_iou = tf.map_fn(
lambda x: tf.reduce_max(broadcast_iou(x[0], tf.boolean_mask(
x[1], tf.cast(x[2], tf.bool))), axis=-1),
(pred_box, true_box, obj_mask),
tf.float32)
ignore_mask = tf.cast(best_iou < ignore_thresh, tf.float32)

# 5. calculate all losses
xy_loss = obj_mask * box_loss_scale * \
tf.reduce_sum(tf.square(true_xy - pred_xy), axis=-1)
wh_loss = obj_mask * box_loss_scale * \
tf.reduce_sum(tf.square(true_wh - pred_wh), axis=-1)
obj_loss = binary_crossentropy(true_obj, pred_obj)
obj_loss = obj_mask * obj_loss + \
(1 - obj_mask) * ignore_mask * obj_loss
# TODO: use binary_crossentropy instead
class_loss = obj_mask * sparse_categorical_crossentropy(
true_class_idx, pred_class)

# 6. sum over (batch, gridx, gridy, anchors) => (batch, 1)
xy_loss = tf.reduce_sum(xy_loss, axis=(1, 2, 3))
wh_loss = tf.reduce_sum(wh_loss, axis=(1, 2, 3))
obj_loss = tf.reduce_sum(obj_loss, axis=(1, 2, 3))
class_loss = tf.reduce_sum(class_loss, axis=(1, 2, 3))

return xy_loss + wh_loss + obj_loss + class_loss
return yolo_loss
```

6.2 Updating Layer Using Weight Files:

The layers are to updated using custom trained weight files to detect our custom class problems. In our case for Lane violation and Vehicle crash events. After updating the layers detection is performed and 4 values are returned bounding_boxes value, detection score , class ID and custom classes name .

```
yolo_max_boxes = 5
yolo_iou_threshold = 0.5
yolo_score_threshold = 0.5
# customize your model through the following parameters
flags.DEFINE_integer('yolo_max_boxes', 10, 'maximum number of detectio
ns at one time')
flags.DEFINE_float('yolo_iou_threshold', 0.5, 'iou threshold')
flags.DEFINE_float('yolo_score_threshold', 0.5, 'score threshold')

yolo_anchors = np.array([(10, 13), (16, 30), (33, 23), (30, 61), (62,
45),
                         (59, 119), (116, 90), (156, 198), (373, 326)]
,
                        np.float32) / 416
yolo_anchor_masks = np.array([[6, 7, 8], [3, 4, 5], [0, 1, 2]])

yolo_tiny_anchors = np.array([(10, 14), (23, 27), (37, 58),
                              (81, 82), (135, 169),  (344, 319)],
                             np.float32) / 416
yolo_tiny_anchor_masks = np.array([[3, 4, 5], [0, 1, 2]])
```

```
def DarknetConv(x, filters, size, strides=1, batch_norm=True):
    if strides == 1:
        padding = 'same'
    else:
        x = ZeroPadding2D(((1, 0), (1, 0)))(x)  # top left half-
padding
        padding = 'valid'
    x = Conv2D(filters=filters, kernel_size=size,
               strides=strides, padding=padding,
               use_bias=not batch_norm, kernel_regularizer=l2(0.0005))
(x)
    if batch_norm:
        x = BatchNormalization()(x)
        x = LeakyReLU(alpha=0.1)(x)
    return x

def DarknetResidual(x, filters):
    prev = x
    x = DarknetConv(x, filters // 2, 1)
    x = DarknetConv(x, filters, 3)
    x = Add()([prev, x])
    return x

def DarknetBlock(x, filters, blocks):
    x = DarknetConv(x, filters, 3, strides=2)
    for _ in range(blocks):
        x = DarknetResidual(x, filters)
    return x

def Darknet(name=None):
    x = inputs = Input([None, None, 3])
    x = DarknetConv(x, 32, 3)
    x = DarknetBlock(x, 64, 1)
    x = DarknetBlock(x, 128, 2)  # skip connection
    x = x_36 = DarknetBlock(x, 256, 8)  # skip connection
    x = x_61 = DarknetBlock(x, 512, 8)
    x = DarknetBlock(x, 1024, 4)
    return tf.keras.Model(inputs, (x_36, x_61, x), name=name)

def DarknetTiny(name=None):
    x = inputs = Input([None, None, 3])
    x = DarknetConv(x, 16, 3)
    x = MaxPool2D(2, 2, 'same')(x)
    x = DarknetConv(x, 32, 3)
    x = MaxPool2D(2, 2, 'same')(x)
    x = DarknetConv(x, 64, 3)
    x = MaxPool2D(2, 2, 'same')(x)
    x = DarknetConv(x, 128, 3)
    x = MaxPool2D(2, 2, 'same')(x)
    x = x_8 = DarknetConv(x, 256, 3)  # skip connection
    x = MaxPool2D(2, 2, 'same')(x)
    x = DarknetConv(x, 512, 3)
    x = MaxPool2D(2, 1, 'same')(x)
    x = DarknetConv(x, 1024, 3)
    return tf.keras.Model(inputs, (x_8, x), name=name)

def YoloConv(filters, name=None):
    def yolo_conv(x_in):
        if isinstance(x_in, tuple):
            inputs = Input(x_in[0].shape[1:]), Input(x_in[1].shape[1:]
)
            x, x_skip = inputs

            # concat with skip connection
```

```
            x = DarknetConv(x, filters, 1)
            x = UpSampling2D(2)(x)
            x = Concatenate()([x, x_skip])
        else:
            x = inputs = Input(x_in.shape[1:])

        x = DarknetConv(x, filters, 1)
        x = DarknetConv(x, filters * 2, 3)
        x = DarknetConv(x, filters, 1)
        x = DarknetConv(x, filters * 2, 3)
        x = DarknetConv(x, filters, 1)
        return Model(inputs, x, name=name)(x_in)
    return yolo_conv

def YoloConvTiny(filters, name=None):
    def yolo_conv(x_in):
        if isinstance(x_in, tuple):
            inputs = Input(x_in[0].shape[1:]), Input(x_in[1].shape[1:]
)
            x, x_skip = inputs

            # concat with skip connection
            x = DarknetConv(x, filters, 1)
            x = UpSampling2D(2)(x)
            x = Concatenate()([x, x_skip])
        else:
            x = inputs = Input(x_in.shape[1:])
            x = DarknetConv(x, filters, 1)

        return Model(inputs, x, name=name)(x_in)
    return yolo_conv

def YoloOutput(filters, anchors, classes, name=None):
    def yolo_output(x_in):
        x = inputs = Input(x_in.shape[1:])
        x = DarknetConv(x, filters * 2, 3)
        x = DarknetConv(x, anchors * (classes + 5), 1, batch_norm=Fals
e)
        x = Lambda(lambda x: tf.reshape(x, (-
1, tf.shape(x)[1], tf.shape(x)[2],
                                            anchors, classes + 5)))(x)
        return tf.keras.Model(inputs, x, name=name)(x_in)
    return yolo_output

def yolo_boxes(pred, anchors, classes):
    # pred: (batch_size, grid, grid, anchors, (x, y, w, h, obj, ...cla
sses))
    grid_size = tf.shape(pred)[1]
    box_xy, box_wh, objectness, class_probs = tf.split(
        pred, (2, 2, 1, classes), axis=-1)

    box_xy = tf.sigmoid(box_xy)
    objectness = tf.sigmoid(objectness)
    class_probs = tf.sigmoid(class_probs)
    pred_box = tf.concat((box_xy, box_wh), axis=-
1)  # original xywh for loss

    # !!! grid[x][y] == (y, x)
    grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size))
    grid = tf.expand_dims(tf.stack(grid, axis=-
1), axis=2)  # [gx, gy, 1, 2]

    box_xy = (box_xy + tf.cast(grid, tf.float32)) / \
        tf.cast(grid_size, tf.float32)
```

```
    box_wh = tf.exp(box_wh) * anchors

    box_x1y1 = box_xy - box_wh / 2
    box_x2y2 = box_xy + box_wh / 2
    bbox = tf.concat([box_x1y1, box_x2y2], axis=-1)

    return bbox, objectness, class_probs, pred_box

def yolo_nms(outputs, anchors, masks, classes):
    # boxes, conf, type
    b, c, t = [], [], []

    for o in outputs:
        b.append(tf.reshape(o[0], (tf.shape(o[0])[0], -
1, tf.shape(o[0])[-1])))
        c.append(tf.reshape(o[1], (tf.shape(o[1])[0], -
1, tf.shape(o[1])[-1])))
        t.append(tf.reshape(o[2], (tf.shape(o[2])[0], -
1, tf.shape(o[2])[-1])))

    bbox = tf.concat(b, axis=1)
    confidence = tf.concat(c, axis=1)
    class_probs = tf.concat(t, axis=1)

    scores = confidence * class_probs
    boxes, scores, classes, valid_detections = tf.image.combined_non_m
ax_suppression(
        boxes=tf.reshape(bbox, (tf.shape(bbox)[0], -1, 1, 4)),
        scores=tf.reshape(
            scores, (tf.shape(scores)[0], -1, tf.shape(scores)[-1])),
        max_output_size_per_class=yolo_max_boxes,
        max_total_size=yolo_max_boxes,
        iou_threshold=yolo_iou_threshold,
        score_threshold=yolo_score_threshold
    )

    return boxes, scores, classes, valid_detections

def YoloV3(size=None, channels=3, anchors=yolo_anchors,
           masks=yolo_anchor_masks, classes=80, training=False):
    physical_devices = tf.config.experimental.list_physical_devices('G
PU')
    if len(physical_devices) > 0:
        tf.config.experimental.set_memory_growth(physical_devices[0],
True)
    x = inputs = Input([size, size, channels], name='input')

    x_36, x_61, x = Darknet(name='yolo_darknet')(x)

    x = YoloConv(512, name='yolo_conv_0')(x)
    output_0 = YoloOutput(512, len(masks[0]), classes, name='yolo_outp
ut_0')(x)

    x = YoloConv(256, name='yolo_conv_1')((x, x_61))
    output_1 = YoloOutput(256, len(masks[1]), classes, name='yolo_outp
ut_1')(x)

    x = YoloConv(128, name='yolo_conv_2')((x, x_36))
    output_2 = YoloOutput(128, len(masks[2]), classes, name='yolo_outp
ut_2')(x)

    if training:
        return Model(inputs, (output_0, output_1, output_2), name='yol
ov3')
```

```
    boxes_0 = Lambda(lambda x: yolo_boxes(x, anchors[masks[0]], classe
s),
                     name='yolo_boxes_0')(output_0)
    boxes_1 = Lambda(lambda x: yolo_boxes(x, anchors[masks[1]], classe
s),
                     name='yolo_boxes_1')(output_1)
    boxes_2 = Lambda(lambda x: yolo_boxes(x, anchors[masks[2]], classe
s),
                     name='yolo_boxes_2')(output_2)

    outputs = Lambda(lambda x: yolo_nms(x, anchors, masks, classes),
                     name='yolo_nms')((boxes_0[:3], boxes_1[:3], boxes
_2[:3]))

    return Model(inputs, outputs, name='yolov3')

def YoloV3Tiny(size=None, channels=3, anchors=yolo_tiny_anchors,
               masks=yolo_tiny_anchor_masks, classes=80, training=Fals
e):
    physical_devices = tf.config.experimental.list_physical_devices('G
PU')
    if len(physical_devices) > 0:
        tf.config.experimental.set_memory_growth(physical_devices[0],
True)
    x = inputs = Input([size, size, channels], name='input')

    x_8, x = DarknetTiny(name='yolo_darknet')(x)

    x = YoloConvTiny(256, name='yolo_conv_0')(x)
    output_0 = YoloOutput(256, len(masks[0]), classes, name='yolo_outp
ut_0')(x)

    x = YoloConvTiny(128, name='yolo_conv_1')((x, x_8))
    output_1 = YoloOutput(128, len(masks[1]), classes, name='yolo_outp
ut_1')(x)

    if training:
        return Model(inputs, (output_0, output_1), name='yolov3')

    boxes_0 = Lambda(lambda x: yolo_boxes(x, anchors[masks[0]], classe
s),
                     name='yolo_boxes_0')(output_0)
    boxes_1 = Lambda(lambda x: yolo_boxes(x, anchors[masks[1]], classe
s),
                     name='yolo_boxes_1')(output_1)
    outputs = Lambda(lambda x: yolo_nms(x, anchors, masks, classes),
                     name='yolo_nms')((boxes_0[:3], boxes_1[:3]))
    return Model(inputs, outputs, name='yolov3_tiny')

def YoloLoss(anchors, classes=80, ignore_thresh=0.5):
    def yolo_loss(y_true, y_pred):
        # 1. transform all pred outputs
        # y_pred: (batch_size, grid, grid, anchors, (x, y, w, h, obj,
...cls))
        pred_box, pred_obj, pred_class, pred_xywh = yolo_boxes(
            y_pred, anchors, classes)
        pred_xy = pred_xywh[..., 0:2]
        pred_wh = pred_xywh[..., 2:4]

        # 2. transform all true outputs
        # y_true: (batch_size, grid, grid, anchors, (x1, y1, x2, y2, o
bj, cls))
        true_box, true_obj, true_class_idx = tf.split(
```

```
            y_true, (4, 1, 1), axis=-1)
        true_xy = (true_box[..., 0:2] + true_box[..., 2:4]) / 2
        true_wh = true_box[..., 2:4] - true_box[..., 0:2]

        # give higher weights to small boxes
        box_loss_scale = 2 - true_wh[..., 0] * true_wh[..., 1]

        # 3. inverting the pred box equations
        grid_size = tf.shape(y_true)[1]
        grid = tf.meshgrid(tf.range(grid_size), tf.range(grid_size))
        grid = tf.expand_dims(tf.stack(grid, axis=-1), axis=2)
        true_xy = true_xy * tf.cast(grid_size, tf.float32) - \
            tf.cast(grid, tf.float32)
        true_wh = tf.math.log(true_wh / anchors)
        true_wh = tf.where(tf.math.is_inf(true_wh),
                           tf.zeros_like(true_wh), true_wh)

        # 4. calculate all masks
        obj_mask = tf.squeeze(true_obj, -1)
        # ignore false positive when iou is over threshold
        best_iou = tf.map_fn(
            lambda x: tf.reduce_max(broadcast_iou(x[0], tf.boolean_mas
k(
                x[1], tf.cast(x[2], tf.bool))), axis=-1),
            (pred_box, true_box, obj_mask),
            tf.float32)
        ignore_mask = tf.cast(best_iou < ignore_thresh, tf.float32)

        # 5. calculate all losses
        xy_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_xy - pred_xy), axis=-1)
        wh_loss = obj_mask * box_loss_scale * \
            tf.reduce_sum(tf.square(true_wh - pred_wh), axis=-1)
        obj_loss = binary_crossentropy(true_obj, pred_obj)
        obj_loss = obj_mask * obj_loss + \
            (1 - obj_mask) * ignore_mask * obj_loss
        # TODO: use binary_crossentropy instead
        class_loss = obj_mask * sparse_categorical_crossentropy(
            true_class_idx, pred_class)

        # 6. sum over (batch, gridx, gridy, anchors) => (batch, 1)
        xy_loss = tf.reduce_sum(xy_loss, axis=(1, 2, 3))
        wh_loss = tf.reduce_sum(wh_loss, axis=(1, 2, 3))
        obj_loss = tf.reduce_sum(obj_loss, axis=(1, 2, 3))
        class_loss = tf.reduce_sum(class_loss, axis=(1, 2, 3))

        return xy_loss + wh_loss + obj_loss + class_loss
    return yolo_loss
```

6.3 Main API to Run Detection and Save Frames:

The following API was built for our web application to perform detection at the backend .As soon the detection start's marked and clipped frames are saved in the project direct which are uploaded to the database as soon the processing stops.

```
try:
                vid = cv2.VideoCapture(f.filename)
            except:
                vid = cv2.VideoCapture(f.filename)
            out = None
```

```
            fps = 0.0
            count = 0
            while True:
                _, img = vid.read()
                if img is None:
                    logging.warning("Empty Frame")
                    time.sleep(0.1)
                    count+=1
                    if count < 3:
                        continue
                    else:
                        break
                img_in = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
                img_in = tf.expand_dims(img_in, 0)
                img_in = transform_images(img_in,416)

                t1 = time.time()
                boxes, scores, classes, nums = yolo.predict(img_in)
                fps  = ( fps + (1./(time.time()-t1)) ) / 2

                img = draw_outputs(img, (boxes, scores, classes, nums)
, class_names)
                # Checking threshold i first row of 2D "scores array"
because score array has only one row
                if(scores[0][1].any() >0.50):
                    cv2.imwrite('data/Clipped/clipp'+str(i)+'.jpg',img
)
                    storage.child('ClippedCrash/' +str(i)+ '/crash.jpg
').put('data/Clipped/clipp'+str(i)+'.jpg')
                    link_image = storage.child('ClippedCrash/' +str(i)
+ '/crash.jpg').get_url(None)
                    doc_ref = db.collection(u'Crash')
                    doc_ref.add({
                        u'Name': u'Vehicle Crash',
                        u'Type': u'Anomaly',
                        u'Timestamp': str(datetime.now()),
                        u'Image_Url': link_image
                    })
                elif(scores[0][0].any() >0.50):
                    cv2.imwrite('data/Clipped/clipp'+str(i)+'.jpg',img
)
                     storage.child('LaneClipped/' +str(i)+ '/Lane.jpg'
).put('data/Clipped/clipp'+str(i)+'.jpg')
                     link_image = storage.child('ClippedLane/' +str(i)
+ '/Lane.jpg').get_url(None)
                     doc_ref = db.collection(u'LaneVoilation')
                     doc_ref.add({
                         u'Name': u'Lane Voilation',
                         u'Type': u'Anomaly',
                         u'Timestamp': datetime.now(),
                         u'Image Url': link_image
                     })

                print(boxes, scores, classes, nums, class_names)
                global displayData
                displayData = {

                    "scores":str(scores),
                    "classes":str(classes),
                    "classes_names":str(class_names)
                }
                # print(displayData)
                # data['boxes'] = i
```

```
                img = cv2.putText(img, "FPS: {:.2f}".format(fps), (0,
30),
                cv2.FONT_HERSHEY_COMPLEX_SMALL, 1, (0, 0, 255), 2)
                cv2.imwrite('data/output_frames/anomaly'+str(i)+'.jpg'
,img)
                i=i+1

                cv2.destroyAllWindows()

            # os.remove('data\output-vid\short.mp4')
            vid_array = []
            for img_video in glob.glob('data/output_frames/*.jpg'):
                vid_img = cv2.imread(img_video)
                height, width, layers = vid_img.shape
                SIZE_vid = (width,height)
                vid_array.append(vid_img)

            out = cv2.VideoWriter('data/output-
vid/short.mp4',cv2.VideoWriter_fourcc(*'mp4a'), 15,SIZE_vid)

            for n in range(len(vid_array)):
                out.write(vid_array[n])
            out.release()

            cv2.destroyAllWindows()

            users_ref = db.collection(u'Crash')
            Crashdata = users_ref.stream()
            sasta = []
            print(Crashdata)
            for doc in Crashdata:
                print(f'{doc.id} => {doc.to_dict()}')

                my_dict = doc.to_dict()
                sasta.append(my_dict)
                # print(my_dict)
            print(sasta)
            storage.child("videos/new.mp4").put("data\output-
vid\short.mp4")
            links = storage.child('videos/new.mp4').get_url(None)
            return render_template('upload.html', l=(links,sasta))
```

The complete code is available at the GitHub Repository:

- **Application Complete Code(UI + Custom Files + Custom Trained Model)**

  https://github.com/Hamza-773/Road-Pulse

- **Custom Trained Weight Files After Training(Custom Classes Trained Weights)**

  https://drive.google.com/drive/folders/1hqnlCoep39CK6YH9VAKVJm8STIdhKs3a?usp=sharing

Chapter 7

Conclusion

7 Conclusion

7.1 Project summary

In conclusion, our project, Road Pulse, consists of an android and a web application. The functionality that detects the Traffic violations events, as well as anomalies and the model embedment, can be shared across the app and the website. The app and website will use a machine learning model YOLO v3 that helps in accurately detecting the violations + anomalies to assist the law enforcement departments. Both platforms can be synchronized in terms of data through a mutual database Google Firebase that relates to both websites and android applications. All the results and data are stored in the database.

The live stream video is handled through the web application with an embedded model to perform detection. Once the detection occurs the information including clipped key anomaly event and its description are saved in the real-time database. This info is synced using database to the android app module. The system looks for signed in officers and information regarding the violation is sent to the officer. Officer will take further action in real-time to punish the drivers showing irresponsible behaviour.

Further after handling the situation, the officer can update the department by sending a response using their communication channel. This helps to restore law and change the behaviour of the people regarding traffic to obey the law and respond in real time to save citizens precious lives from tragic event like crash.

7.2 Problems Faced and Lessons Learned

7.2.1 Problem-1

Dataset availability and labelling of dataset was a major issue. Datasets available online were based on foreign roads condition and no dataset was available according to the Pakistan's road condition. Dataset quality can affect the training process and the detection rate of the model.

Partial Solution:

We minimized this problem using the best quality dataset from reliable source which is AI City Challenge 2019 Dataset. Hope in future more dataset will be available to improve the model accuracy.

Dataset labelling solution:

- ✓ Manually labelled the dataset by watching every video (200 Videos).
- ✓ The labels of dataset were in XML format but later on we discovered that YOLO V3 use labels of .txt format
- ✓ So, all the labels were converted then to YOLO V3 format using a script written in python
- ✓ Average video length was around 12-13 minutes

7.2.2 Problem-2:

First time interaction with Neural Network Model YOLO v3 which was issue for us at the start-up:

Solution: Extensive research which led to these results:

- ✓ YOLO makes use of only convolutional layers, making it a fully convolutional network (FCN) which uses deeper architecture of a feature extractor called Darknet-53.
- ✓ Each layer is followed by a batch normalization layer. No form of pooling is used, and a convolutional layer with stride 2 is used to down sample the feature maps.
- ✓ This network applies single Neural network to the Full Image which divides the image into regions and predicts bounding boxes and probabilities for each region.

7.2.3 Problem-3

Problem: Creating custom training files for our custom classes and extensive training time

Solution:

- ✓ Used the already available CFG files. Update the main parameters like batch size, classes, filter size in the convolutional layers.
- ✓ After first training which was done in around 11 hours reduced the convolutional layers to reduce training time.
- ✓ Used colab GPU Environment for the training
- ✓ Implemented the model on our CPU.

7.2.4 Problem-4

Problem: Defining the threshold for what can be considered an anomalous snippet

Solution:

- ✓ Static Thresholding:
- ✓ Use a static value i.e., 0.5 to classify anomalous segment.
- ✓ Dynamic Thresholding:
- ✓ Frames with scores exceeding the mean of smoothened scores are considered as anomalous.

7.3 Future work

We worked to perform detection on the video acting as a live stream to point out the violations. A simple android app is developed for officers. But in future we will try to introduce more modules to our system as follow:

- ❖ **Accident predictor:**

This module will provide a prediction regarding accident (crash) using road conditions. Also, the traffic patterns will help to make an accurate prediction. The system will be a subpart of the current system.

- ❖ **Vehicle Count:**

This module will help to count the vehicles on certain roads to avoid traffic congestion. The traffic will be managed to overcome such an issue on busy roads. This will help to keep the traffic going.

In the future, it can be further enhanced by providing the model with a much larger dataset that includes various countries to train it further so it can work for those places too. The model can also be enhanced in a way that the accuracy can be increased and can work at night-time as well. This will increase the efficiency and effectiveness of this project in the future.

- ❖ **Extend Class Problems Domain:**

In future we want to extend our problems classes by using more datasets and train our FCN YOLO model for more class problems which can extend the surveillance domain to the next level. As for now we have only 2 class problems due to the unavailability of datasets.

Chapter 8

References

8 References

[1] S. K. H. Kazmi, “Pakistan And Gulf Economist,” news,research, 11 2017. [Online]. Available: http://www.pakistaneconomist.com/2017/11/13/alarming-road-accidents-rate-pakistan-rules-laws-need-overhaul/. [Accessed 14 2 2020].

[2] C. Wang, S. Ison and M. Quddus, “Impact of traffic congestion on road accidents: A spatial analysis of the M25 motorway in England,” *Accident; analysis and prevention,* p. 12, 2009.

[3] D. Deme and M. Bari, “Traffic Accident Causes and Its Countermeasures on Addis Ababa-Adama Expressway,” *Journal of Equity in Science and Sustainable Development,* p. 12, 2016.

[4] V. .B*, and M. Babu, “Dynamic Traffic- Rule- Violation Monitoring and Detection System,” *IJESRT,* p. 14, 2014.

[5] W. Corporation, “Waze ,GPS ,Maps & Traffic Alert,” 30 July 2020. [Online]. Available: https://play.google.com/store/apps/details?id=com.waze&hl=en.

[6] C. U. Corporation, “Cameras US-Traffic cams US,” Android Application, 30 04 2018. [Online]. Available: https://play.google.com/store/apps/details?id=com.vision.cameras.us&hl=en. [Accessed 30 07 2020].

[7] M. Bramberger, J. Brunner and B. Rinner, “Real-Time Video Analysis on an Embedded Smart Camera,” *10th IEEE Real-Time and Embedded Technology and Applications Symposium,* p. 11, 2004.

[8] S. Kamijo, Y. Matsushita, K. Ikeuchi and M. Sakauchi, “Traffic Monitoring and Accident Detection at,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 1, p. 20, 2000.

[9] A. mariya T.P, A. M.J, F. Aishwarya and L. George, “TRAFFIC VIOLATION DETECTION SYSTEM,” *Traffic Enforcement System,* vol. 4, no. 03, p. 4, 2017.

[10] D. U. I. Bajwa, “MPVIR Group,” Research, 2015. [Online]. Available: https://sites.google.com/view/mpvir/projects/road-pulse?authuser=0. [Accessed 28 12 2020].

Turnitin Originality Report

CS29_RoadPulse.pdf by Anonymous

From Research Papers 2 (Research Papers 2)

- Processed on 28-Dec-2020 7:45 AM PST
- ID: 1481699955
- Word Count: 8836

Similarity Index
8%
Similarity by Source

Internet Sources:
5%
Publications:
3%
Student Papers:
5%

sources:

1 1% match (Internet from 27-Sep-2018)
https://www.diva-portal.org/smash/get/diva2:1205236/FULLTEXT01.pdf

2 1% match (student papers from 05-Aug-2020)
Submitted to Higher Education Commission Pakistan on 2020-08-05

3 1% match (Internet from 12-Dec-2020)
https://www.sixt.com/magazine/tips/top-traffic-apps/

4 < 1% match (Internet from 18-Oct-2011)
http://www.ncbi.nlm.nih.gov/m/pubmed/15676815/

5 < 1% match (publications)
S. Kamijo, Y. Matsushita, K. Ikeuchi, M. Sakauchi. "Traffic monitoring and accident detection at intersections", IEEE Transactions on Intelligent Transportation Systems, 2000

6 < 1% match (student papers from 15-Dec-2019)
Submitted to Higher Education Commission Pakistan on 2019-12-15

7 < 1% match (student papers from 11-Jan-2018)
Submitted to Higher Education Commission Pakistan on 2018-01-11