[Starlight Electronics]

Mobile Application For Traffic Violations

Final Report

[26/05/2024]

Design Studio Section:4

Presented by:
Ismayil Mammadli
Veysel Uysan
İlyas Eğlence
Alparslan Çelik
Bilal Yuksu
Onur Emirhan Çon

Table of Contents

Starlight Electronics

Company Structure

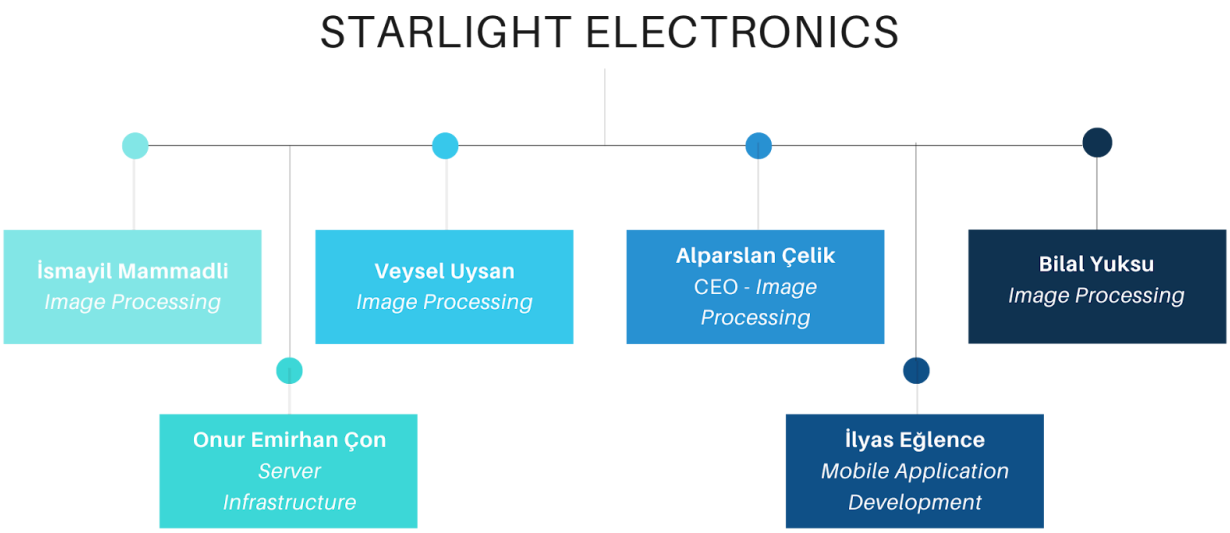


Figure 1: Organizational Chart of Starlight Electronics

Executive Summary

Traffic congestion is a pressing issue in modern cosmopolitan cities, causing significant time wastage for commuters and posing serious safety concerns due to traffic accidents. In 2022, Türkiye witnessed 197,261 traffic accidents resulting in 5,229 fatalities and 288,696 injuries, underscoring the severity of the problem.

While conventional solutions such as city population control, infrastructure development, and increased traffic policing require substantial resources and political backing, Starlight Electronics offers a unique, cost-efficient alternative. Our approach involves empowering individuals with a user-friendly mobile application to monitor and report traffic violations without relying solely on centralized law enforcement.

Our project focuses on developing a mobile app capable of detecting three major traffic violations: speeding, running red lights, and unsafe overtaking. The application records incidents, detects violations, and stores data securely without compromising personal information.

To achieve these objectives, we have structured the project into four subsystems: image processing, mobile application, server, and communication. The image processing subsystem is responsible for detecting traffic violations through video recordings. To do so, it uses cutting edge technologies in the area of image processing and artificial intelligence.

The mobile app subsystem aims for building a user-friendly mobile application which performs video recording and displaying relevant outcomes of the image processing subsystem. The server subsystem securely stores relevant data, while the communication subsystem ensures proper integration of all components, overcoming data transfer limitations through optimization techniques.

In summary, Starlight Electronics offers an innovative, accessible solution to alleviate traffic issues and enhance safety through proactive monitoring and reporting.

Introduction

This project aims to address the challenges posed by traffic violations, which often lead to road accidents and congestion, especially in densely populated areas. The primary objective is to develop a mobile application capable of documenting and identifying various driver behaviors.

The video processing component of the project targets the detection of overspeeding vehicles, those running red lights, and instances of overtaking from the right. Although not within the immediate scope, future enhancements may include additional detection capabilities for various traffic violations such as improper lane changing, illegal parking, turns, wrong-way driving, double lane violation, following distance violations, and driving in a criss-cross manner. Throughout the development process, emphasis is placed on maintaining the mobile app design intuitive and user-friendly.

Background of the project

The challenge of traffic violations persists due to limitations in monitoring by traffic police and the high costs associated with implementing systems like MOBESE universally. Our solution addresses this by proposing a scalable approach through a user-centric mobile application, operating during users' journeys to observe and report traffic events along their routes. By leveraging collective user data, our solution establishes a comprehensive network for traffic monitoring and control, bridging the gap between limited policing resources and the expansive road network.

Key to our solution is the integration of advanced image processing technology, automating incident examination to reduce the manual workload for law enforcement agencies. This streamlined process enables swift and accurate detection of violations, enhancing overall efficiency. Additionally, by providing ubiquitous coverage through the mobile application, we aim to deter opportunistic violations in unsupervised areas, fostering a culture of compliance and accountability among road users.

In essence, our project offers a practical solution to logistical challenges in traffic control while promoting safer driving behaviors and fostering a culture of responsibility on the roads. Through collaborative efforts and technological advancements, we aspire to create a safer and more harmonious urban transportation environment.

Problem statement

The issue with traffic is quite complex. One problem is that the traffic police can't be everywhere at once. It's also not feasible to install advanced systems like Mobese on all roads due to the high cost involved. Putting up Mobese and Traffic Control units everywhere would be too expensive for the government, especially in areas with less traffic and fewer violations. These methods require constant monitoring over large areas for long periods, making them less effective and costly in smaller regions.

To tackle these challenges, we propose creating a mobile app for users during their trips. This app would keep an eye on traffic incidents along the user's route. This way, it forms a network of monitoring points without needing lots of physical infrastructure.

There are several key points to our proposed solution. First, it's cost-effective because it uses smartphones that people already have, avoiding the need for expensive setups. Second, since the app is free and easy to use, we expect many people to use it. This widespread use will make the app more effective in spotting traffic violations across different areas.

The app's backend is equipped with image processing and automatic algorithms to analyze the data users provide. This means it can detect traffic violations without human intervention, making the process efficient and accurate. Additionally, knowing that their actions could be recorded by others using the app may make drivers think twice before breaking traffic rules.

In conclusion, our approach combines technology with user involvement to create a scalable and affordable way to manage traffic issues. This complements traditional methods and promotes safer driving habits, ultimately contributing to improved road safety.

Current status

As of today, we have made huge progress and completed the project, we are only working on minor inconveniences. On the image processing part of the project, our work includes detecting vehicles on the roads, distinguishing between vehicle types (car, bus, truck, etc.), running red traffic lights on the roads, their colors, vehicle speeds, and tracking vehicles and detecting drivers’ attempts to falsely overtake. Adding vehicle tracking at the later parts of our work on the project, allowed us to detect the vehicles overtaking the user’s vehicle on the right. We are also detecting the license plates of the vehicles. We are correctly finding the license plates and working on improving them. Right now, we are improving displaying the violation-type documents on the app and improving speed detection.

We have a mobile app that users can use by opening an account. We also connected the mobile app and server. In the application, we can get data from the map. We can save this data to Excel and send it to the server. We can take photos and videos and send them to the server. We can display the images on the server on the archive page. We can also access the profile information on the profile page. The server can detect when a new image or video arrives and download it to send to image processing. We can also save the data coming from image processing to the desired folder on the server.

Scope and organization

The report covers many parts of the project, including how the whole system is built, diagrams showing how things are connected, how different parts work together, and the basic ideas behind how the project functions. It also talks about research done to solve engineering problems, company strategies, test results for different parts, and overall project results.

System Design

Figure 2. Subsystems Input Output Diagram

The project comprises four key subsystems: image processing, server, communication, and mobile application, as illustrated in Figure 1.

Within the mobile application subsystem, we focus on designing the end product for user interaction. Users can start video recording of the traffic from the application. When they finish the video recording, the app will send it to the server for processing. After the process, the app will show the results which are traffic violations in the recorded video.

The communication subsystem facilitates seamless data exchange between the mobile app and the server, adhering to specified data recording and transmission speeds. This two-way flow of data ensures that inputs from the mobile app are processed and returned as outputs to the user.

The server subsystem serves as the central repository for storing data. It receives data from the image processing code, processes it, and then stores the processed data for access by users through the mobile application.

The image processing subsystem holds significant importance, responsible for analyzing video records and Excel files containing road speed limits. It detects vehicles, license plates, and traffic lights, tracking them for potential violations such as speeding, running red lights, and false overtaking. To manage the workload effectively, this subsystem is divided into several modules, each handling specific detection and violation tasks.

Image Processing

Vehicle and Vehicle Type Detection And Tracking

Vehicle and vehicle-type detection can be realized by using several object detection models. In this project, we firstly choose to use the YOLOv8 object detection model. Instead of training from scratch, we are going to use pre-trained YOLOv8 models which are trained on the MS COCO dataset. During the test procedure, we observed that these models have fulfilled project objectives under the project limitations. However, there is always some room for further development. MS COCO dataset includes 80 different classes but we need only four of them in the project which are car, truck, bus and traffic lights (MS COCO does not include license plate class). Therefore, in order to increase the performance of the pre-trained model, we can apply fine-tuning techniques and train it on a relevant dataset which only includes cars, buses, trucks and traffic lights. The other way of building an object detection model is using different algorithms such as SSD (single shot detector) and RCNN (region-based convolutional neural network). However, we chose the YOLOv8 models over them because it has higher performance at high computational speeds and there are plenty of resources to learn the subject. We chose YOLOv8 over other YOLO versions because of its higher performance as seen in Figure 2.

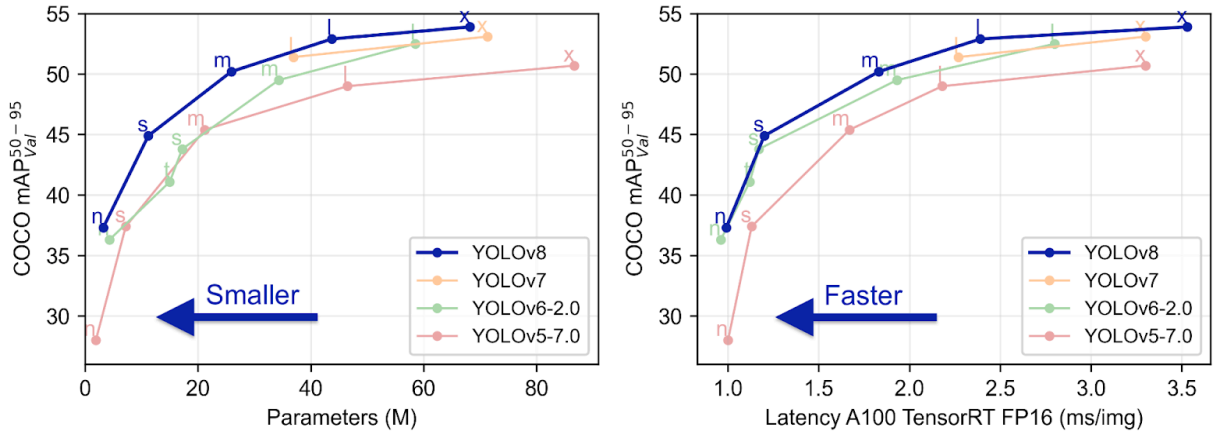


Figure 3: Comparison of different YOLO model versions

Object detection in a video is the same as object detection in a photograph. The object detection code processes each frame of the video one by one. However, detecting vehicles in each frame is not enough. We also need to do object tracking. In object tracking, we create a unique ID for each of the initial detections and track each of the objects as they move around frames in a video. Tracking objects will be important to us in traffic violation detections. To realize object tracking, we are going to implement the built-in object tracking class of Ultralytics YOLO. By doing so, we get a tracking ID for each detection in the video stream.

License Plate Detection

License plate information of the detected vehicles is important for monitoring purposes. So, we are going to detect the license plates of the vehicles. There exists a pre-trained license plate detection model which is trained on YOLOv8 architecture. This model does not read the numbers and the characters on the license plate. Instead, it finds the bounding box locations of the detected license plate.

The license plate detection procedure is closely related to the vehicle detection part. Firstly, the vehicle detection model detects vehicles in the frame of a video. Then, we extract the bounding box of a detected vehicle. It is called ROI (Region of Interest). Later, the license plate detection model takes this ROI as input and finds bounding boxes of the license plate inside the ROI as seen in Figure 3.

Figure 4: Example of Vehicle and License Plate Detection Crop

Detecting the bounding box coordinates of the license plate is not enough. We also have to read the numbers and characters on the license plate. To do so, we extract the license plate bounding box as a second ROI. Then, we implement some fundamental image processing techniques on it in order to prepare it. Later, we use the OCR (Optical Character Recognition) algorithm on processed ROI. By doing so, we are able to read the numbers and characters on the License Plate.

Traffic Lights Detection and Detection of Vehicles Passing Through Red Lights

In this section of our project, we focus on detecting vehicles that violate traffic rules by passing through red lights. To achieve this, we utilize a pre-trained model capable of accurately detecting various traffic light colors, including red, green, and yellow. Our approach involves deploying this model and drawing a horizontal line from the base of the traffic light. Using a custom-developed mechanism, we identify vehicles that violate the traffic rule by crossing the stop line during a red light.

Our mechanism analyzes the bottom line of a vehicle's bounding box to determine whether it intersects with the ground line. Additionally, we have incorporated a tracking code that assigns a unique identifier to each vehicle. This identifier can be used for further analysis and monitoring, allowing us to identify which vehicle violated the rule by marking it with its license plate instead of the assigned ID. Figure 5 illustrates an example of tracked vehicles with their IDs and classes, the baseline (in red), and the traffic light at the top left of the picture.

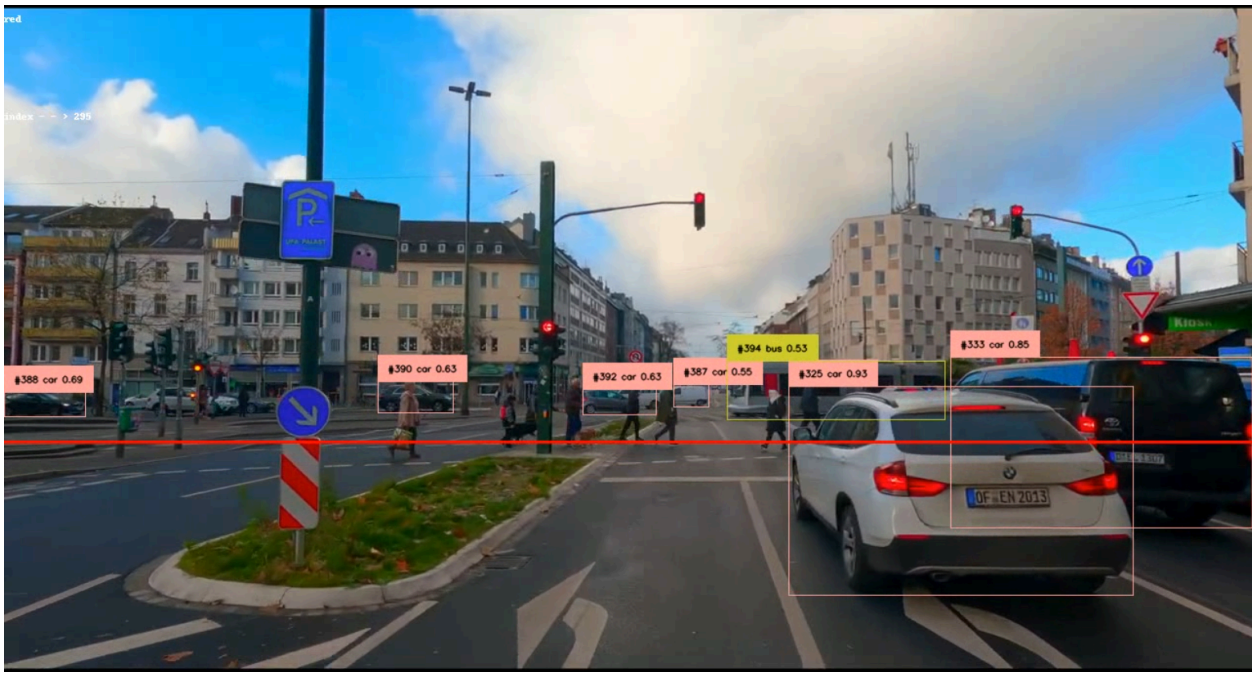


Figure 5: detection of the red light violation example

The flowchart in Appendix 1 outlines the process of detecting red light violations using video footage. It begins with the initialization step, where the traffic light detection code is run, setting the number of frames to zero and initializing a counter 'i' to zero. As the video progresses, the number of frames increments, and for each frame, the traffic light status is saved in a buffer (TrafficLightBuffer) along with the line coordinates where the vehicles are detected (LineCoordinates).

As each frame of the video is processed, the YOLO model detects traffic lights. For each detected traffic light, a line is drawn downward from the center of the bounding box, extending twice the height of the box. The line with the lowest y-coordinate is selected, and this y-coordinate is used to draw a red line across the width of the screen. The left and right coordinates of the red line are calculated to be equal to the screen's width. If no line coordinates are found for a frame, no values are assigned. This process is repeated for each frame, and the results are appended to the line_coordinates list.

Next, the vehicle tracking code is executed, which monitors the positions and movements of vehicles. The red light violation function then proceeds to extract the current traffic light line and its status from the respective buffers. The core logic involves checking for intersections between vehicle positions and the line coordinates. If a vehicle crosses the line, it is recorded. For each detected intersection, the flowchart checks whether the intersection value is 0 or 1. If an intersection is found, the algorithm further checks the previous value of the same VehicleID to determine if the vehicle was already in violation in the preceding frame. To enhance accuracy, the system can examine up to seven consecutive frames instead of just one, ensuring more reliable detection of red light violations.

If the traffic light is red at the moment of intersection and the vehicle is detected crossing the line, it is flagged as a red light violation. This is noted for the particular VehicleID. The counter 'i' is then incremented. The process repeats for all frames in the video until index 'i' reaches the total number of frames. Once all frames have been processed, the flowchart reaches the end, completing the detection process for red light violations, as shown in Appendix 1.

Speed Detection

In the speed estimation part of our image processing subsystem, we have approached the design to make it as robust and reliable as possible. In the speed detection algorithm, we first detect cars that are moving by our sides or in front of us. Then, according to frames of video, the algorithm detects how many pixels of movement are between time frames, by looking at the car's bounding box frame. We have taken every 4 pixels of the road to be equal to 1 meter, for a video of 640 vertical pixels (downscaled from 1280 pixels of a vertical video). Then the algorithm divides the road taken by the time (or rather, the frame passed in this case). And this results in our speed being displayed. This system estimates the car's speed in one portion of the video. From the design standpoint, this is a big advantage for us since we do not want our system to update the speed as frequently as possible. When the cars move beyond 20 meters, the algorithm diverges from ground truth, making it harder for us to get a speed estimation reliably. Cases for the speed detection algorithm’s detection distances when it is reliable and diverging from reliable cases are shown in Appendix 2.

Detection of False Overtaking

This part is responsible for detecting vehicles passing other vehicles on the right. It consists of two detections: Vehicles passing the vehicle camera is mounted and vehicles passing other vehicles on the right. For both of these, we utilize our object-tracking algorithm. To detect the first case scenario we check the centroids of the vehicle the moment they are detected. Then if any of these coordinates are in the box that we define (bottom right corner of the screen), python code writes "True" to the cell corresponding to the ID of that vehicle. The logic of the code can be seen from figures 5 and 6 (Any new vehicle appearing on the bottom right corner of the video violates the rule).

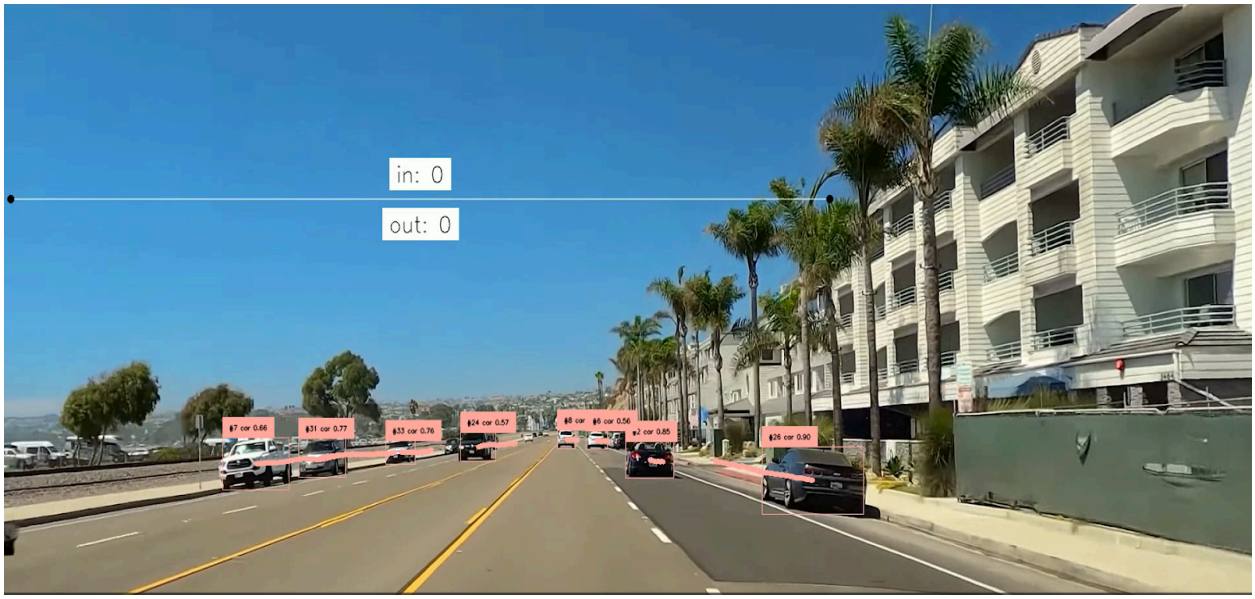


Figure 6: Initial frame of a video

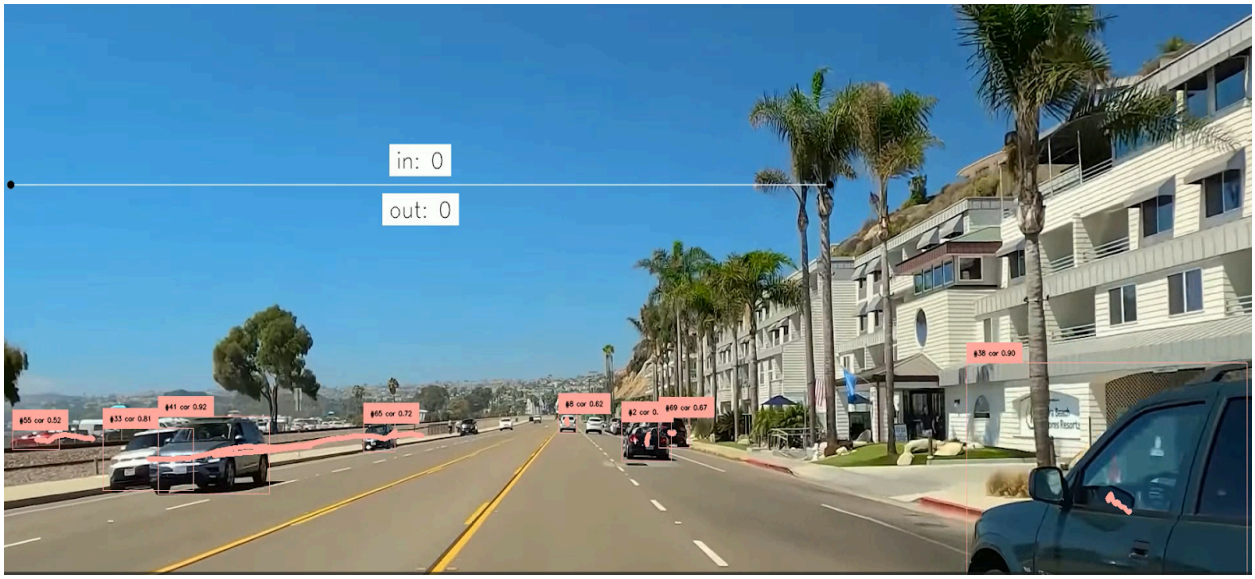


Figure 7: Second frame of a video

Detection of the second scenario needed a more complex approach. Firstly, the vehicles that are very close to each other are saved. Then the positions of these vehicles are checked in

the next frames. If the position of the automobile on the right goes above the position of the automobile on the left then “True” is written in the corresponding cell of the Excel file. In Figure 7 the vehicles numbered 1 and 2 are saved for checking in the following frames and in Figure 8 the detection is made.

Figure 8: Initial frame of the video

Figure 9: Last frame of the video

These codes were tested on multiple videos and improved based on the results. The main improvements include: Not checking the vehicles that are very far away as they decrease our accuracy and checking for more than one frame after the detection is done since relying on only one frame in a video decreases the confidence rate of our detections.

Server

To store and manage our data effectively, we needed a reliable server. We considered different options and decided to go with Google Firebase. We chose Firebase because it's free and easy to use. Its interface is user-friendly and straightforward. We also used other Google services like Google Colab for image processing and Flutter for mobile app development. Firebase works well with these services, which makes our work easier.

The mobile application uploads videos along with their related data, which is stored in an Excel file, to Firebase. Because both Firebase and Flutter are Google services, Firebase provides an API specifically designed for Flutter. By granting the necessary permissions from Firebase to the mobile application and utilizing the relevant libraries such as cloud_firestore, firebase_storage and firebase_auth in Flutter, the upload process from the mobile app to Firebase is successfully completed. This seamless integration between Firebase and Flutter streamlines the data upload functionality, ensuring efficient and effective data management within the application.

After the raw videos are uploaded to the Firebase, these videos must be sent to the image processing subsystem. For that purpose, we wrote a Python script in Google Colab by using Pyrebase and FirebaseAdminSDK libraries. The working principle of the code is first it reads the all file paths in the firebase and stores it in an array. After creating the array, it reads the file paths again and compares it with the array at certain periods. If there is a new file in the Firebase, it updates the array and downloads the new files to the Colab. Similarly, it tracks the processed video folder in the Colab and uploads the recently processed videos to Firebase.

Mobile Application

We developed a mobile application. Since our application is connected to the server, users must have an account. At the beginning of the application, there is a "**Register Screen''** where users can open an account. On this page, we request the user information required for the account. There is a "**Login Screen''** for users who create an account to enter the application. Users can log in to the account by entering their email and password. This only needs to be done once when opening the application. Unless you log out, the application will be opened with the same user account. Additionally, there is a "**Forgot Password Screen**" for users who forget their password. It helps to set a new password by sending a password reset email to your e-mail address.

When the application is opened, the first of the five home pages welcomes the user with the "**Home Screen**". Other pages are "Map Screen"," Camera Screen"," Archive Screen" and" Profile Screen". The "Home Screen" will contain information explaining to users how to use the application. In this way, the user will learn how to create a complaint to detect traffic violations.

In "**Map Screen**", users can access map data. There are features such as being able to navigate the map, detect their location, and see the traffic density. Thanks to this page, we can access the user's traffic speed and the speed limit of the road he is on. This

information is important for detecting traffic violations. We save this information together with the complaint in an Excel table and send it to the server.

There are two different pages in "**Camera Screen**": "**Photo Screen**" and "**Video Screen**". First, "Photo Screen" opens. Here, a complaint will be created by taking a photo. The captured photo is sent to the server and entered into image processing. The other camera page is ”Video Screen”. Here, users can create a complaint by recording a video. We give the name of the time when the record was created in Excel, where the data from the captured video and map are saved. In this way, we will know which video is linked to which Excel. When the submitted complaint is examined with image processing, the results will be written in the Excel table. In this way, violations will be shown to the user as output.

"**Archive Screen**" is the page where we can see the user data on the server in the application. The complaint image we sent to the server, the output video from image processing and the Excel filled in after image processing will be displayed on this page. Users will only be able to access their own data. However, the "**admin**" account will be able to access all user data.

In the "**Profile Screen**", users will be able to access user data on the server. Users can change their profile information from this page. This page will also include application settings. Users can change the theme of the application to light and dark themes. The "**Logout Screen**" where users will log out of their accounts is also here.

Communication

There is a rule "Similarly, while transmitting the information (or recording) to a central database, the data usage of your mobile app should not exceed 10 MB per 1 minute of recording (approximate WhatsApp or FaceTime video call data usage)". For this reason, we are considering restricting the application's internet usage. Normally we had the idea of reducing the data size. But because of this, image processing quality may decrease.
Instead, we think that we will comply with the rule if we save the data in good quality and restrict the internet usage speed of the application.

"While recording the relevant data, the storage usage of your mobile app should not exceed 10 MB per 1 minute of recording." is another rule. We do our registration on the server. We don't need to save anything on the phone. That's why we provide this rule.

Design Requirements

In this project, our main objective is to build a traffic monitoring app that detects some of the traffic violations. These violations are:

- Speeding
- Running red lights
- Passing other vehicles from the right lane (False Overtaking)

In addition to these, there are some optional objectives such as wrong-way driving, illegal parking, illegal turns, improper lane change, double lane violation and so on.

While realizing these objectives, we have to follow several requirements:

1. Storage Requirement: While recording the storage of the mobile application should not be filled more than 10 MB/min
2. Transmission Requirement: While transmitting the information to the server (our central database) data usage should not be higher than 10 MB/sec

Another requirement is an interface for viewing the information on the server. Here the analysis of the violations will be shown. The interface must provide:

1. Admin and user accounts where users are able to see their records but not the others
2. A system where users can start using the application after admin approval.
3. Support for at least four different devices connected at the same time
4. Each user with their reported violations
5. The admin with an opportunity to filter the records based on the type of violation
6. Encrypted communication between database (server) and mobile application (optional)

Design Modifications

We have made a modification on the vehicle detection and tracking module. In the conceptual design report, we have stated that a centroid tracking algorithm is used to track vehicles. However, during the development process, we agreed on that using built-in multi-object tracking classes of Ultralytics YOLO has better results than centroid tracking algorithm in performance and easy-to-implementation aspects. Therefore, we made a modification about that.

We have completely changed our lane detection method since the conceptual design report. We are following our B plan stated in the previous report. A model trained on a dataset named CUlane is now used for detection. The Python code finds the estimated coordinates of the lane lines using this model. Previously, we were using edge detection and Hough transform but it was not that efficient.

Compatibility Analysis

As outlined in the "System Design" section of the report, our entire system comprises four subsystems. These subsystems must function both compatibly and independently, allowing for easy integration without complications.

The subsystems do not have to work simultaneously which makes our job easier. For example, while the video is recorded through the mobile app, other subsystems do not work. Or, while the video processing subsystem is working others don't. Therefore, the individual success of our subsystems and reliable data transfer between them is enough for the project.

The process starts with the mobile application. The mobile application only interacts with the server subsystem. First, it sends the video and Excel table to the server. Then, it takes the processed video and the detected violation information from the server.

After the video and Excel table are uploaded to the server, the video is sent to the image processing subsystem. This process is done by a Python script that is explained in the System Design section. The server subsystem takes the processed video and the Excel table from the image processing subsystems. The format of the Excel tables coming from mobile applications and the image processing subsystem must be compatible with each other in order to compare the tables properly and decide whether a violation occurred.

Compliance with Requirements

There are seven Interface Requirements as shown in Table 1. We can say that our system can achieve design requirements of the project with a high success rate. We can detect speeding violations, running red light violations and false overtaking violations. We can read the license plates of the vehicles. Moreover, the mobile app can record videos and show the processed video and outcomes of violations successfully.

However, there is one design requirement that is not achieved consciously which is storage requirement. To fulfill this requirement, we need to decrease the quality of the video significantly. This can be achieved easily by mobile application subsystem. However, decreasing the quality of a video has a dramatic effect on the success rate of reading license plates. It can be seen on table 4. Therefore, we are not able to compromise these two design requirements under given limitations.

Requirements	**Image Processing**	**Server**	**Mobile Application**	**Communication**
Storage R.	-	x	-	-
Transmission R.	-	-	-	✓
Interface R.	-	-	✓	-
Speeding Violation	✓	-	-	-
Running Red Light Violation	✓	-	-	-
False Overtaking	✓	-	-	-

Violation				
Reading License Plates	✓	-	-	-

Table 1: Compliance with Design Requirements table

Test Results

Mobile App

Mobile application tests were conducted to see whether the features worked or not.

The server connection was tested by opening and logging in from six different devices and ten different email addresses. It was tested whether the application's interface (texts, images, buttons...) works correctly on six different devices. It was tested whether all five home pages opened as desired. The application has successfully completed all of these tests.

The registration process was attempted more than a hundred times on different devices. Video shooting and data saving to Excel were successful. Except for a few exceptional cases due to user error, sending these records to the server was also successful.

Viewing the records on the "Archive Page" was mostly successful. In fact, displaying the recording video and data on the archive page was completely successful. The test did not yield successful results in cases where the video and Excel output from the image processing code were faulty. The application gave successful results in all tests where video and Excel were problem-free.

Server

We successfully tested the downloading and uploading the required files to the Firebase. When a new file is added to the Firebase it is automatically downloaded to the Colab and when a video is processed it is automatically uploaded to the Firebase. Test procedure and results of our server are discussed above, in the Mobile App part.

Running Red Lights

During the traffic light detection testing phase, night, day, near (up to 20 meters) and far (between 20 meters and 50 meters) parameters were determined. Approximately 10 videos were shot during the day, night, far and near, with lengths ranging from 10 to 40 seconds, and photographs were created at 10 frames per second. The test results are shown below in normalized form in Appendix-1. Analysis of video feeds from mobile phone cameras 10 videos were utilized for testing, comprising a combination of self-captured footage and red light violation videos sourced from the internet. The chart and relation formed between running red lights and false overtaking systems can be seen in Appendix.

Quantitative Results:

- Correct detections: 86%
- Missed violations: 14%
- False positives: 39%

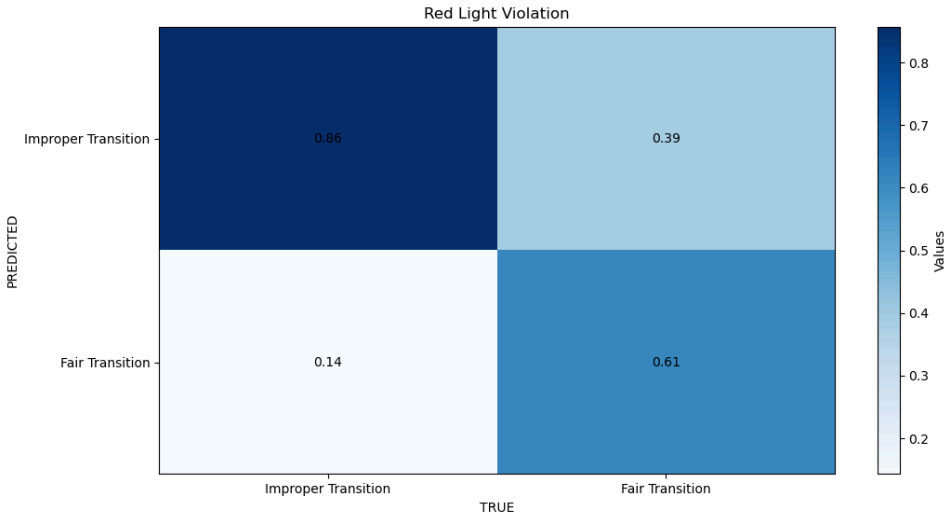


Figure 10: *Test results for red light violation in normalized form.*

Test Requirement	**Test Conducted/Solution Implemented**
Accurate detection of violations	Correct detections: 86%
Movement of the detection line	• Adjusted detection thresholds based on traffic light distance • Another threshold was set for multi-frame control
Addressing re-identification challenges	Proposed vehicle re-identification using license plates

Table 2: Running Red Lights Test Results

Speed Detection

For the speed detection test 3 videos are tested.Results are recorded in the below table.

Video Name	Relative Speed (km/h)	Detected Speed (km/h)	Difference
hız30_sol.mp4	30	30.53	1.77%
hız50_sol.mp4	50	50.777	1.55%
hız70_sol.mp4	70	67.562	3.48%
hız30_sağ.mp4	30	32.61	8.70%
hız50_sağ.mp4	47	49.62	6.21%

Table 3. Speed Detection Tests Results

As can be seen from Table 4, the speed detection test gives a 60% truth rate. The data tested for the speed detection test is limited, therefore more tests will be done. The error in the second test is due to the algorithm. The speed detection is done according to the vehicle location. If the location is near the bottom, detection is close to the real values. On the other hand, if the location is far away from the bottom, test results get worse.

License Plate Detection

During the license plate detection test procedure, we examined several videos to check whether the license plate detection module works properly or not. We set the test environment such that we consider cars on the 5-meter radius approximately (one car distance) and on a sunny and clear day. We note that test results of the license plate readings can significantly change under different test environments.

The outcomes of our test results show us that the quality of the video has a significant effect on the correctness of the readings.

* Almost correct title stands for one or two wrong character detections in the readings.

Video Name	Video Quality	Correct	Almost Correct	Wrong/Not Read
v5.mp4	1920 x 1080	3/7	2/7	2/7
v20.mp4	1280 x 720	0/5	3/5	2/5
v21.mp4	1280 x 720	0/3	0/3	3/3
v32.mp4	1280 x 720	1/1	0/1	0/1
Total		4/16	5/16	7/16
		25%	31%	44%

Table 4: License Plate Detection Test Results

Detection of overtaking on the right

Test Method: Analysis of video feeds from mobile phone cameras

Test Procedure:

- 10 videos were utilized for testing, comprising a combination of self-captured footage. They included scenarios where the violators are very far away, the roads are crowded, there are parked cars on the side of the roads and vehicles in the videos are of different types.

Encountered Problems and Solutions:

Problem 1: False positives caused by the movement of bounding boxes of vehicles very far away.

- **Solution Implemented:** Do not take into consideration the vehicles that are observed to be very small.

Problem 2: Vehicles passing to the right of the user's vehicle cause parked cars on the right to be seen as violators.

- **Solution Implemented:** By checking if the violators appear one second after detection.

Total cars in videos	Total violations in videos	Correctly detected violations	Falsely detected violations	Accuracy of correctly defining a vehicle	Accuracy of finding a violator if there is one	Accuracy of detected violators being the real violators
142	9	9	11	92.54%	100%	45%

Table 5

According to Table 2, detection accuracy showing if a vehicle is violator or not is 92.54%. If there are violators in the recorded video it will be detected with 100% accuracy. However, when our code labels a car as a violator there is a 45% chance that it is actually a violator. We will use the mentioned solutions to increase this stat and we believe we can make it above 75%.

Resource Management

Cost

We use Flutter for mobile application, Google Firebase for database and server. These are free software programs, so they have zero cost. We use Google Colab for image processing and all other coding stuff. Although Google Colab has charge free version, we bought Google Colab Pro for non-interrupting operations. This cost $5 with no additional expenditures.

Expenses	Cost
Budget	$300
Project Expenses	$5
Additional Costs	$0
Total Costs	**$5**

Table 6: Costs of products and services

Power Management

The only equipment we use for the project are mobile phones and a computer. Therefore, the project does not need an extra power management process and the power consumption of the project is equal to the power consumption of the mobile application while running and the power consumption of the computer while connecting to Google Colab.

Final Product

List of Deliverables

1.1. Install the traffic.apk in Teams.
1.2. After the installation process is completed, open the "Settings" application of the phone. Open the “Apps” menu. Open the settings of the application named

“traffic_app_try1”. Open the “Permissions” menu. Grant the necessary permissions:

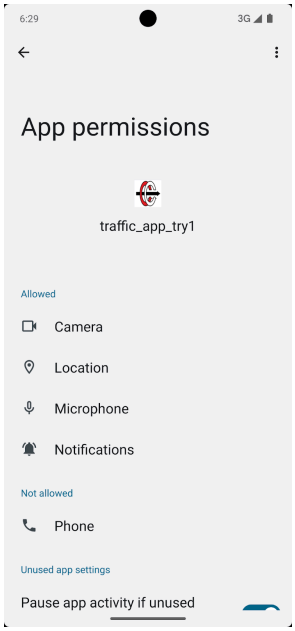
1.2.1. Camera
1.2.2. Location
1.2.3. Microphone
1.2.4. Notifications

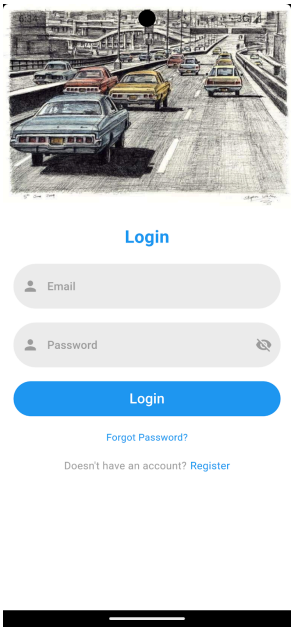
1.3. After giving the necessary permissions, ensure the phone's internet connection. You can use your Wi-Fi or Mobile internet. Then open the "Location" setting of your phone.

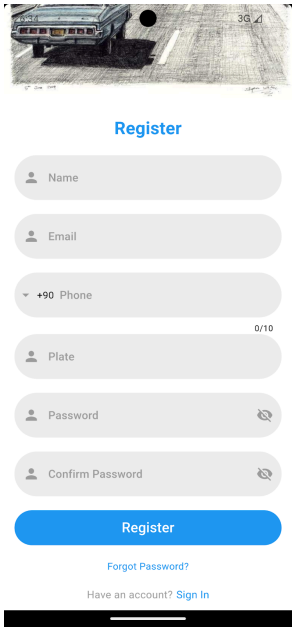
1.4. The application is now ready for use. Find and open the application from the phone's menu. The first “Login Screen” will appear from the application. You must log in with a user account registered on the server.

1.4.1. If you do not have an account, open the “Register Screen” by pressing the “Register” button. Fill in the required information: Name, email, phone, plate and password. Create an account by pressing the “Register” button. The application will log in to the account you opened.

1.4.2. If you have an account, fill in the required information in the “Login Screen”: Email, password. You can open the application with your account by pressing the “Login” button.







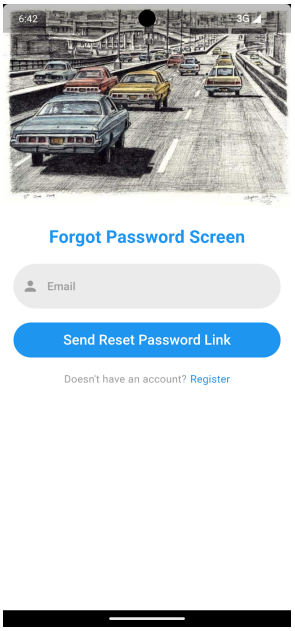


Figure 11 ***Figure 12*** ***Figure 13*** ***Figure 14***

- **Figure 11**: Permission screen for app
- **Figure 12**: "Login Screen" of the application
- **Figure 13**: "Register Screen" of the application
- **Figure 14**: "Forgot Password Screen" of the application

1.5. When you enter the application with your user account, you will see the "Home Screen" from five main pages. This page contains a general introduction of the application. Read this information.

1.6. To create a recording, open the third page, “Camera Screen”. You must use the rear camera. If the front camera is on, switch between the front camera and rear camera using the camera switch button at the bottom right of the screen. You can adjust the video resolution from the settings button at the top right of the screen.

1.7. Adjust the position of your phone. Turn on the "Auto-rotate" feature of the phone and use the phone in a horizontal position for a better viewing angle. Start recording by pressing the “play” button at the bottom of the screen. Video recording will begin. Excel will open in the background. It will take your current speed and the speed limit of the road you are on from GPS and save it to Excel second by second. You can track the elapsed time during recording from the timer at the top of the application.

1.8. When you want to end the recording, press the "stop" button, which replaces the "play" button. Video and Excel recording will end. First, the video will be saved to the gallery. When recording starts, a folder will be opened on the server. The name of the folder will be the date and time when the recording started. Once the recording is completed, Excel and video will be sent to the server. Make sure your internet connection is not interrupted during this process. After pressing the “stop” button, the files will be saved and sent to the server automatically.

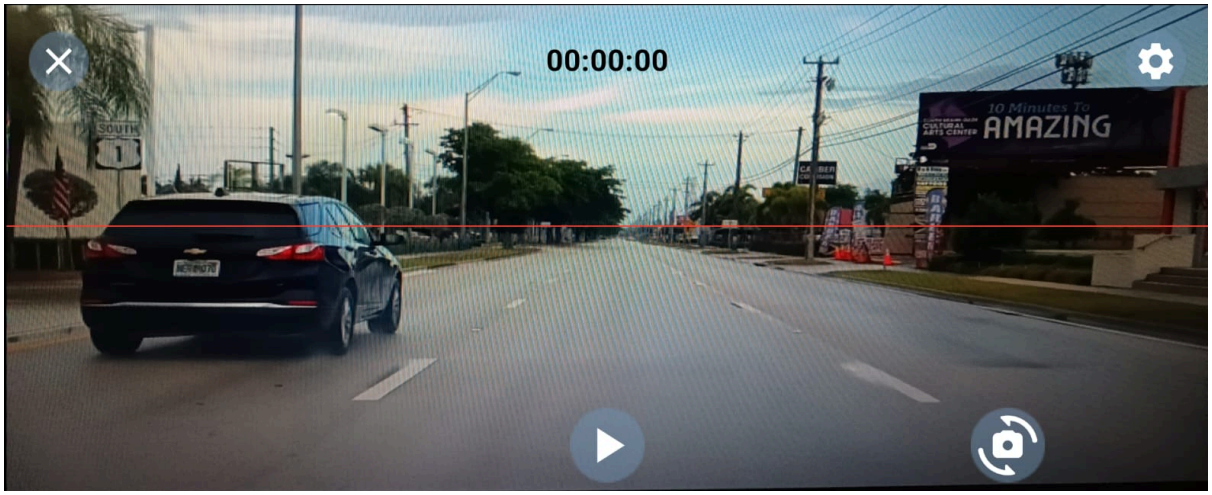
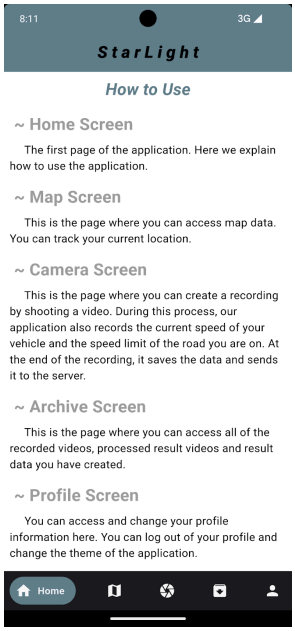


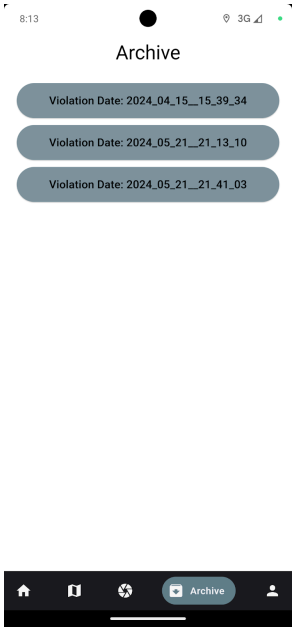
Figure 15: "Camera Screen" of the application

1.9. File sizes will vary depending on the length of the recording and image quality. Since Excel size is very small, it will be sent to the server immediately. But sending the video will take a little longer.

1.10. When the recording reaches the server, you will receive a notification on your phone.

1.11. Our code on the server will automatically detect the new incoming record. The image processing process will begin. It will download the video from storage and enter it into the image processing code. After the image is processed, the necessary outputs will be written to the video and Excel. Output video and Excel server will be re-saved to the relevant folder. This entire process will occur automatically on the server. But it will take time to download the video, process it and upload the outputs back to the server storage.





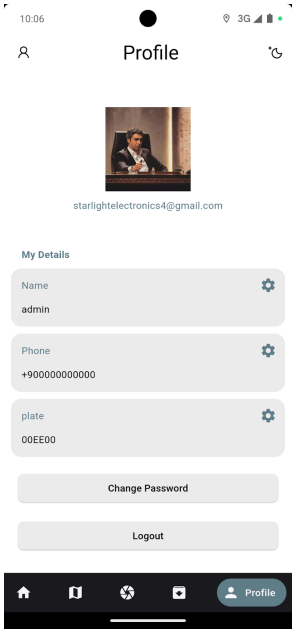


Figure 16 ***Figure 17*** ***Figure 18*** ***Figure 19***

- **Figure 16**: "Home Screen" of the application
- **Figure 17**: "Map Screen" of the application
- **Figure 18**: "Archive Screen" for the user of the application
- **Figure 19**: "Profile Screen" of the application

1.12. When the recording process is completed in the "Camera Screen", turn off the camera with the "x" button on the top left. To view your recording from the application, open the fourth page, “Archive Screen”. Press the button with the date and time you started recording. When the recording files reach the server, you will see the video you shot first. When the processed output video is uploaded to the server, you will see this video secondly. Then the number of vehicles detected in the video will be written. Afterward, you will see lines with information about each vehicle. You will see which vehicle was detected, its license plate and whether it violated three rules. If "false" is written next to the violation, it means that the violation has not been committed; if it says "true", it means that the violation has been committed. Recording video and output video can be played, paused and rewound.

2. Admin Usage

2.1. If you log in with an admin account, buttons with the "IDs" of registered users will appear on the archive page. Press the button with the “ID” of any user. On the page that opens, you will see the records of that user. Press the button with the name of the record you want. You will see the videos and vehicle information of the record you opened.

2.2. Normal users can only access their own records on the archive page. All users' records can be accessed in the Admin account.

Figure 20

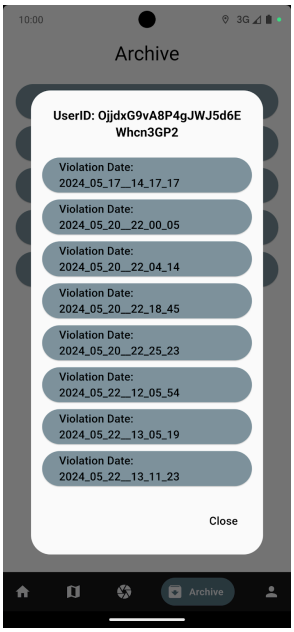


Figure 21

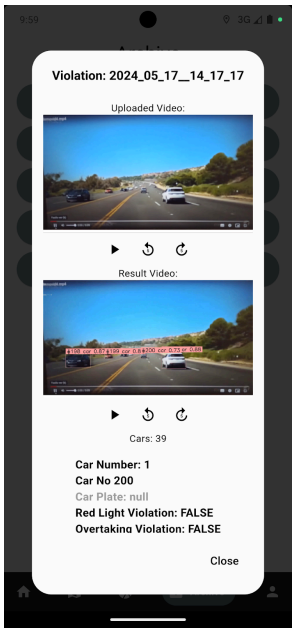


Figure 22

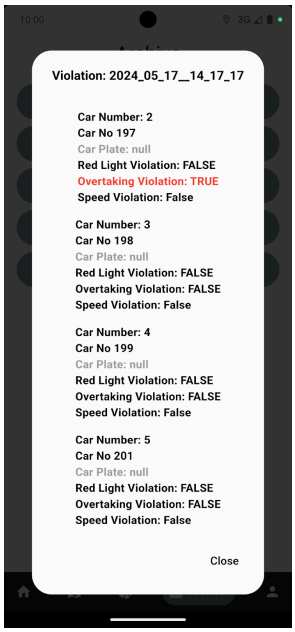


Figure 23

- **Figure 20**: "Archive Screen" for the admin of the application (user IDs)
- **Figure 21**: "Archive Screen" for the admin of the application (users' records)
- **Figure 22**: View video recordings in the app's "Archive Screen"
- **Figure 23**: View recording results in the app's "Archive Screen"

3. Additional instructions

3.1. “Map Screen”

3.1.1. You can see its current location on the map on the second page, "Map Screen". If you can't see it, press the "location" button at the top right of the screen. Your current location will open.

3.1.2. To turn the traffic density feature on or off, press the first button at the top left of the screen.

3.1.3. You can switch between map types: normal, satellite, hybrid and terrain. To do this, press the second button at the top left of the screen.

3.1.4. Use the “+” and “-” buttons at the bottom right to zoom in and out on the map.

3.2. “Profile Screen”

3.2.1. You will see your user information on the fourth page, “Profile Screen”. Press the “settings” button to the right of the information you want to change. Enter the new information on the page that opens and press the "save" button. Your relevant information will change on the server and in the application.

3.2.2. Press the “Change Password” button to change your password. A password change email will be sent to your email address. If you can't find the email, check your spam folder. Click the link in the email and set your new password.

3.2.3. To switch between the night and day themes of the application, press the “theme” button at the top right of the screen.

3.2.4. Press the “Logout” button to log out of your account.” Your account will be closed and you can log in again from the “Login Screen”.

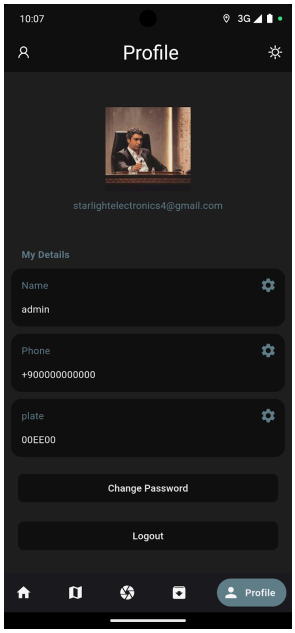


Figure 24

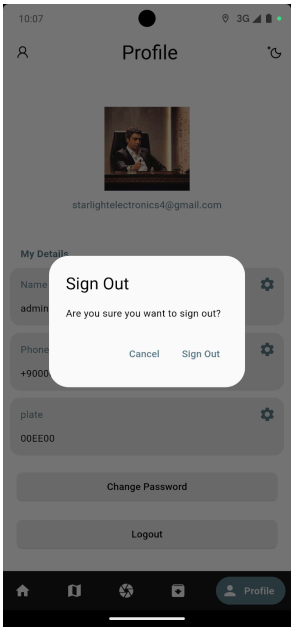


Figure 25

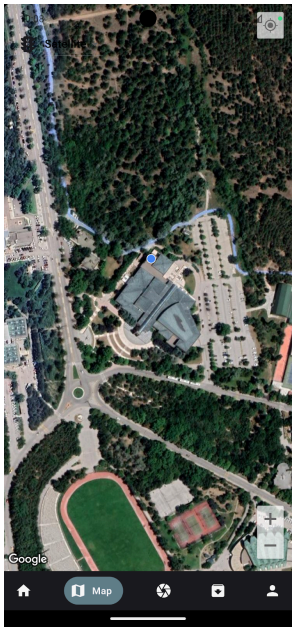


Figure 26

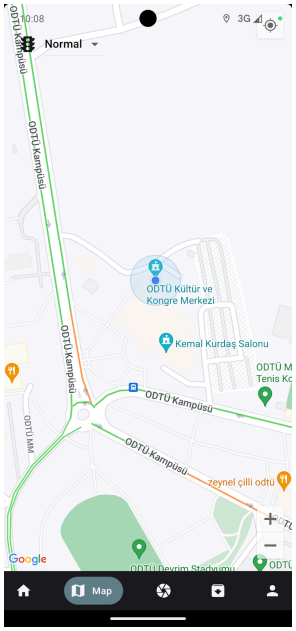


Figure 27

- **Figure 24**: Dark theme of the application
- **Figure 25**: “Sign Out Screen” of the application
- **Figure 26**: Map type feature in the application
- **Figure 27**: Traffic feature on the map in the application

Conclusion

The Starlight Electronics team has developed a novel solution to address the pressing issue of traffic violations in crowded cities. Our mobile application is designed to detect three major traffic violations: speeding, not stopping at red lights and false overtaking. The application utilizes advanced image processing techniques to monitor traffic in real time, providing a cost-effective and scalable alternative to traditional traffic control methods.

At the beginning of the project, we have faced so many challenges. There were so many new things to learn. We started the project by first analyzing it. Then, we divided the project into different subsystems and we further divided the subsystems into necessary submodules. After that, we started to work on them one by one.

The project is divided into four subsystems: image processing, mobile application, server, and communication. The image processing subsystem is the core of the project which is responsible for vehicle detection, license plate recognition, and traffic violation detection. The mobile application serves as the user interface, allowing users to record traffic incidents and submit them for analysis. The server subsystem manages data storage and processing, while the communication subsystem ensures seamless data transfer between the mobile app and the server.

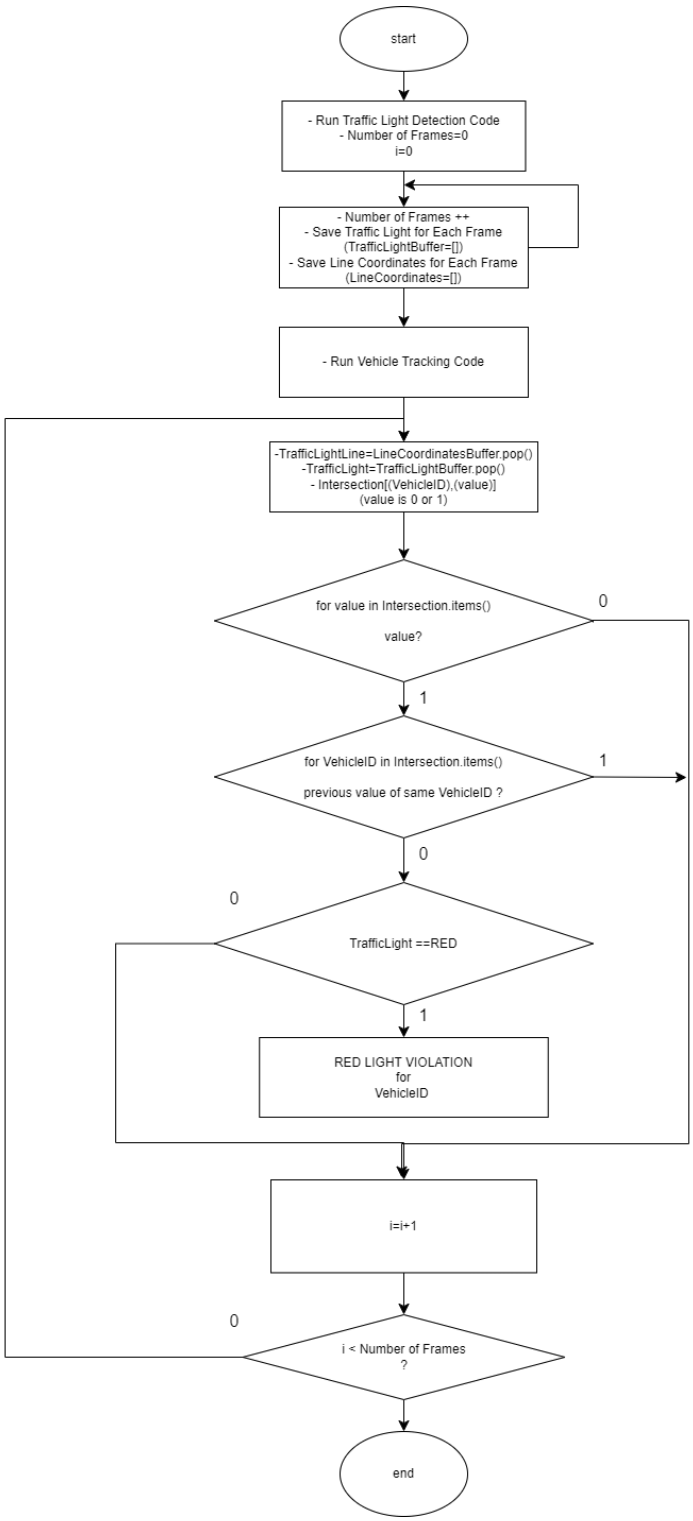
At the final point of the project, we can say that our project can successfully fulfill the design requirements. We have a mobile app working on Android OS that can record videos and show the processed video and the outcomes of the violations. We have a successful image processing subsystem which can detect traffic lights, false overtaking and speeding violations and can read the license plates of the vehicles. We have a successful server and communication subsystems which perform proper operations of different subsystems under required data transmission and storage limitations.

In terms of resource management, the project has been cost-efficient, utilizing free platforms and tools such as Flutter, Firebase, and Google Colab. Power management has been straightforward, with the only power consumption being that of the mobile phones and computers used for the application and image processing.

Overall, our solution offers a promising approach to improving traffic safety and compliance with traffic regulations. By leveraging the widespread use of smartphones, we can create a network of monitoring points that can cover large areas without the need for extensive physical infrastructure. As we continue to refine our system and expand its capabilities, we aim to contribute to safer and more efficient urban transportation.

In closing, the Starlight Electronics team is proud to present this innovative solution to the challenge of traffic violations. We believe that our mobile application has the potential to revolutionize traffic monitoring and enforcement, making our roads safer for everyone. We are committed to further developing and improving our system, and we look forward to seeing its positive impact on communities around the world.

Appendix: Flowchart



Flowchart of the merging traffic light detection and vehicle tracking for red light violations