Helwan University

Faculty of Computers and Artificial Intelligence

Computer Science Department

Car Accidents Detection

2023 / 2024

Supervised by

Dr. Ahemd Hesham

Implemented By

ID	Name
201900454	Abdallah Ismael Mohammed
201900618	Mohammed Ahmed AbuHassiba
201900453	Abdallah Osama Hasan
201900629	Mohamed Osama Sayed
201900730	Mohammed Mohy Mohammed

Table of Contents

Auto Detection of Car Accidents from Video

Chapter 1: Introduction

Abstract:

According to worldwide statistics, traffic accidents are the cause of a high percentage of violent deaths. The time taken to send the medical response to the accident site is largely affected by the human factor and correlates with survival probability. Due to this and the wide use of video surveillance and intelligent traffic systems, an automated traffic accident detection approach becomes desirable for computer vision researchers. Nowadays, Deep Learning (DL)-based approaches have shown high performance in computer vision tasks that involve a complex features relationship. Identifying accident patterns is one of the most vital research foci of driving analysis. Environmental or safety applications and the growing area of fleet management all benefit from accident detection contributions by minimizing the risk vehicles and drivers are subject to, improving their service and reducing overhead costs. Some solutions have been proposed in the past literature for automated accident detection that are mainly based on traffic data or external sensors. However, traffic data can be difficult to access, while external sensors can end up being difficult to set up and unreliable, depending on how they are used. Additionally, the scarcity of accident detection data has limited the type of approaches used in the past,

leaving. Therefore, this work develops an automated DLR-based method capable of detecting traffic accidents on video. The proposed method assumes that traffic accident events are described by visual features occurring through a temporal way. So after detection that this road has a lot of car accidents. The organization that is responsible of the camera (Device that the program deployed on.) can take some decisions to solve the problems and save human life as fast as possible.

1.1) Motivation:

Daily traffic accidents increase annually, causing a significant number of death and disability cases. Most fatalities occur because of the late response to these emergency cases. The time after the traumatic injury is called the golden hour, when providing essential medical and surgical aid at that time increases the probability of saving human lives by one-third on average. Thus, the focus of this paper was to develop a system based on Deep Learning for accident detection and classification.

Another Motivation We hope to have a database on the roads in Egypt, such as the roads in which many accidents occur We also hope that this database will be used in order to avoid more incidents in the future For example, if there is a road that has accidents, for example, this makes us look at this road so that we avoid more accidents on it by forcing cars to reduce speed, for example, or if there are problems to be solved To save human lives.

1.2 Problem definition :

Nowadays, there is an increase in the number of accidents that happen in the world. As the population is increasing, there is the number of cars increasing on the road that contributes to severe accidents that happen daily. Around 80 per cent of accidents contribute to the loss of many lives. Mostly, the growing countries are being targeted by the day to day road accidents. The major reason is the lack of infrastructure, lack of traffic control and accident management.

Automatic detection of car accidents is that the machine or camera can detect if the viewed car move normally or it has accident and then take the appropriate decision to avoid the problems that can be resulted on any accident.

This can be Detection is determined by the repeated change of a vehicle’s attribute change in position, acceleration, and direction.

1.3 Project Objective (suggested solution):

There are different factors that cause traffic accidents. Among the most common factors that increase the probability of their occurrence are the geometry of the road, the climate of the area, drunk drivers, and speeding. These accidents can cause harm to the people involved and, although most of these present only material damage, each one affects people's quality of life in terms of both traffic mobility and personal safety. Video cameras have become a resource for controlling and regulating traffic in urban areas. They make it possible to analyze and monitor the traffic flowing within the city. However, the number of cameras needed to perform these tasks has been increasing significantly over time.

Several approaches have been proposed to automate tasks within the control and follow-up process. An example of this is a system based on video camera surveillance in traffic. Through these, it is possible to estimate the speeds and trajectories of the objects of interest, with the objective of predicting and controlling the occurrence of traffic accidents in the area.

There are different approaches to detect traffic accidents. These include statistics-based methods, social network data analysis, sensor data, machine learning, and deep learning. These latest techniques have presented improvements in various fields of science, including video based problem solving (video processing).With the advent of

convolutional layers in the domain of neural networks, better performance has been achieved in the solution of problems involving digital image processing. Deep learning techniques have shown high performance in a large number of problems, especially for image understanding and analysis.

These layers exploit the spatial relationship that the input data possess and that, due to the size of the information, it is not possible to achieve with dense neural networks. The use of convolutions on input data with a large number of features makes it possible, among other things, to avoid the problem of the curse of dimensionality. This is a very frequent problem when working with data with high complexity, such as images.

The idea of video processing and detection of car accidents is applicable in many organizations such that Roads and Bridges Authority, traffic department and any organization interested in car accidents reports and analysis.

1.4 Project development methodology :

Method for Automatic Detection of Traffic Accidents :-

The proposed method is based on techniques used in video analytics. In particular, deep learning neural networks architectures trained to detect the occurrence of a traffic accident are used. Before describing the architecture, it was necessary to define the network input. Since a video must be processed, it is separated into segments.

1. **Temporal video segmentation** is a problem that has been studied for many years by the scientific community since it is the first step towards the development of more general solutions, such as scene understanding of videos. A video is a sequence of consecutive images with a particular order. When these images are viewed in the correct order and at a specific speed, it is possible to observe the animated event represented by the recorded video. A video camera can capture, with the help of a mechanism, an event that is happening at the moment, in order to store, observe, and process it in the future. Using the same concept of a digital camera, a video camera makes it possible to capture a number of photographs per second, thus allowing the event that is occurring to be digitally recorded. These images, which represent the video,

are known as frames. Video cameras allow recording at different numbers of frames per second (FPS). This means that the higher the number of FPS, the more fluid the movement of the objects on screen. The most commonly used FPS values are 30, 60, and 120.

Automatic Detection of Traffic Accident :

In order to interpret a video segment to detect whether an event occurs, the data must be exploited in two main ways: visually and temporally. The convolutional-based architectures are the most important techniques for visual analysis of images. These are a significant improvement over traditional artificial neural networks in the performance of image classification solutions. However, convolutional layers do not solve all problems. One of the weaknesses of convolutional layers is that they are not good at extracting temporal features from data. Although convolutional layers are powerful in exploiting the spatial characteristics of the data, recurrent neural networks were designed to exploit the temporal characteristics of the data. Convolutional layers are able to process the data in such a way that the spatial information changes to a more abstract representation saving computational cost. Currently, these architectures are used as automatic extractors of image features due to their performance reducing the dimensionality of the input data. However, spatial data is not everything in a video. Sequential data is

of importance in understanding an event that happens over a time span. Recurrent neural networks perform better when processing a sequence over time compared to feed-forward artificial neural networks.

1.5 The used tools in the project (SW and HW) :

-Android studio

- eclipse

-visual studio

- Anaconda

- Canva Tool

- Star UML

-Kaggle

-Colab

Chapter 2:Related work

Due to the increased need for mobility, driving behavior analysis applications have become an important area of research. The result of driving behavior analysis has significant importance for the automotive and intelligent transportation industry, automobile insurance and the government organizations controlling infrastructure and public transportation. Numerous works address the importance of driving behavior analysis in relation to traffic, safety and ecological concerns, whereas many others concern the driver's behavior analysis. Due to the diversity of research goals, applications, study contributions and data modalities, there is no specific study baseline or research categorization in the domain of driving behavior analysis specifically for accident detection. Therefore, to review the recent state of the art on accident detection, an overview of the relevant works in the field of driving behavior analysis is needed. It can be noted that time-series feature extraction is a topic that is quite well explored .However, the conclusions of such feature extraction studies do not seem to transfer .from one application field to another

1. **Automatic Accident Detection System using Arduino:**

An automatic accident detection system for vehicle. This system can detect accidents in significantly less time in which a vehicle accident had occurred. And this system will send an accident alert message to the proposed mobile number (rescue team, family members, and etc…..), who will help in saving the valuable lives. There is an accelerometer module which detects unexpected angles change that can regarded as an accident. The message is sent through the GSM module and the location of the accident is detected with the help of GPS module. This constructed device provides the optimum solution to poor emergency facilities provided to the roads accidents to the road accidents in the most feasible way.

2. **IoT based car accident detection and notification algorithm:**

Detecting car accidents and notifying them immediately. This can be achieved by integrating smart sensors with a microcontroller within the car that can trigger at the time of an accident. The other modules like GPS and GSM are integrated with the system to obtain

the location coordinates of the accidents and sending it to registered numbers and nearby ambulance to notify them about the accident to obtain immediate help at the location.

Chapter3 :System Analysis

3.1 Project specification

Functional Requirements

1) Capturing of the accident video correctly.

2) Detect that there are any up normal behavior.

3) Send Warning Notification To Interested Organization.

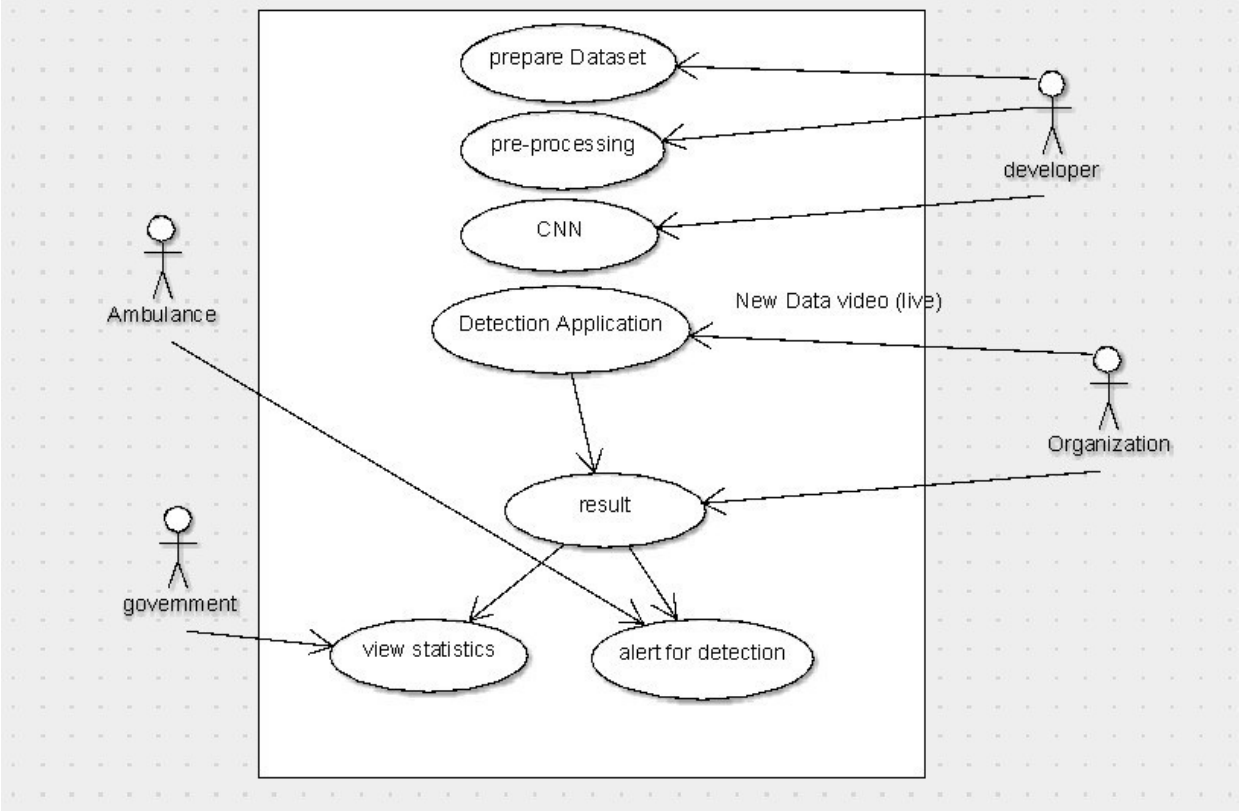
4) Storing Every Thing In Data-Base To Serve Us In Improving Roads Quality.

Non-Functional Requirements

1. **Performance.**
2. **Response of the system.**
3. **Scalability.**
4. **Availability.**
5. **Maintainability.**
6. **Reliability.**

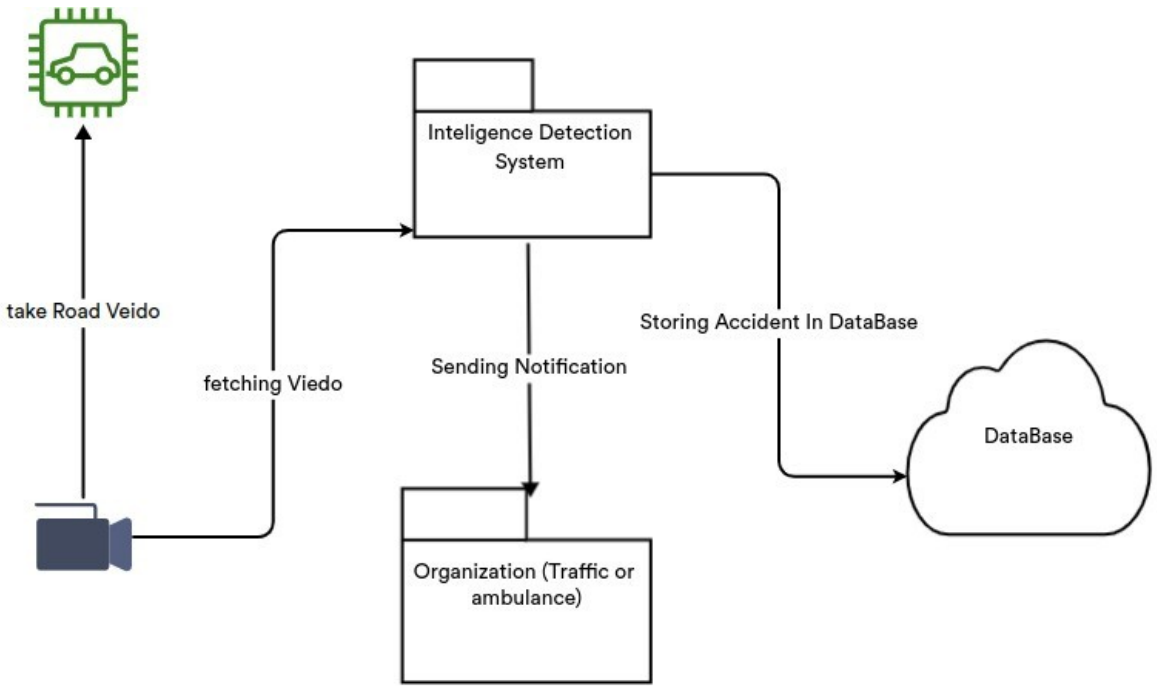
7. Flexibility.

3.2 Use-case Diagram.

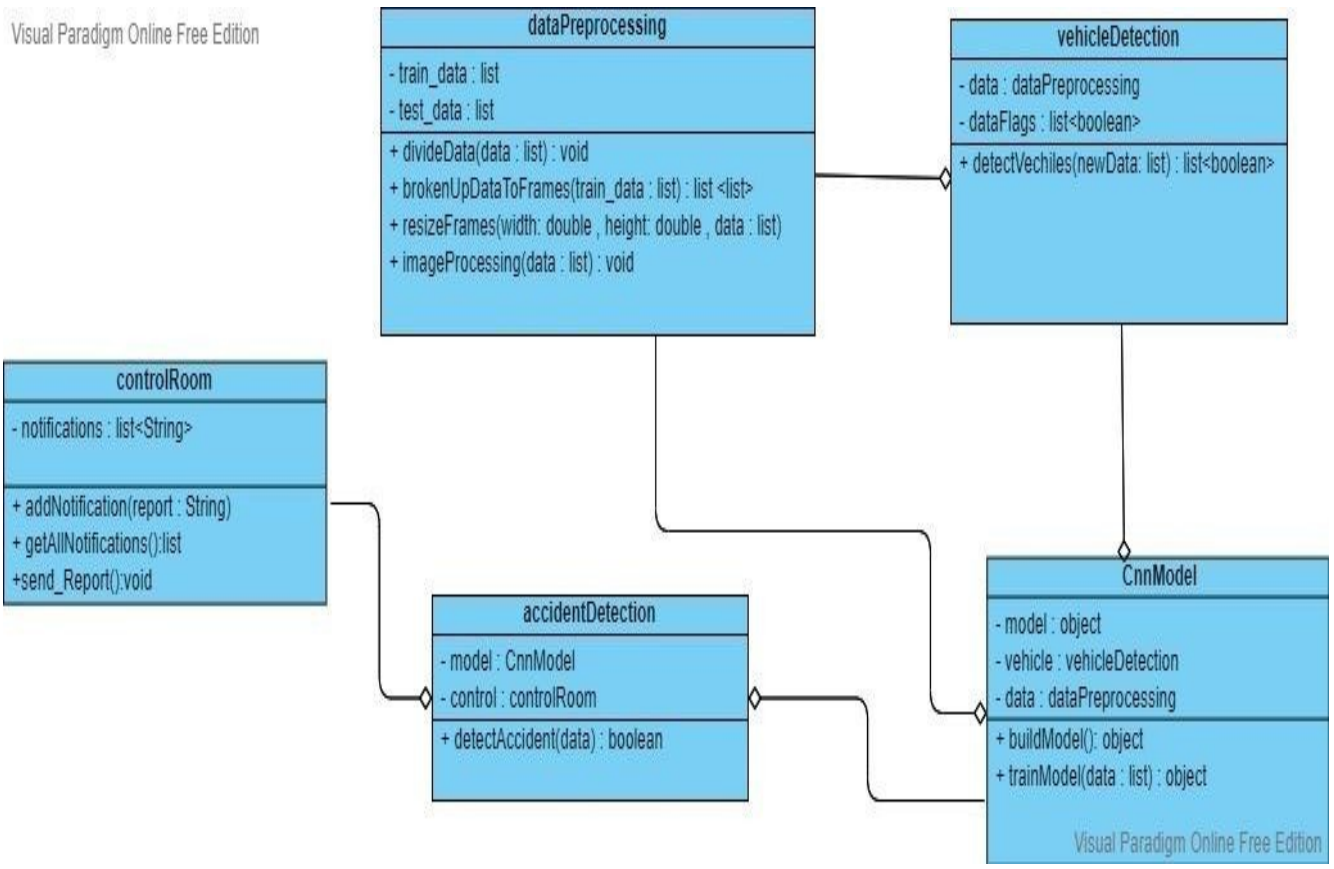


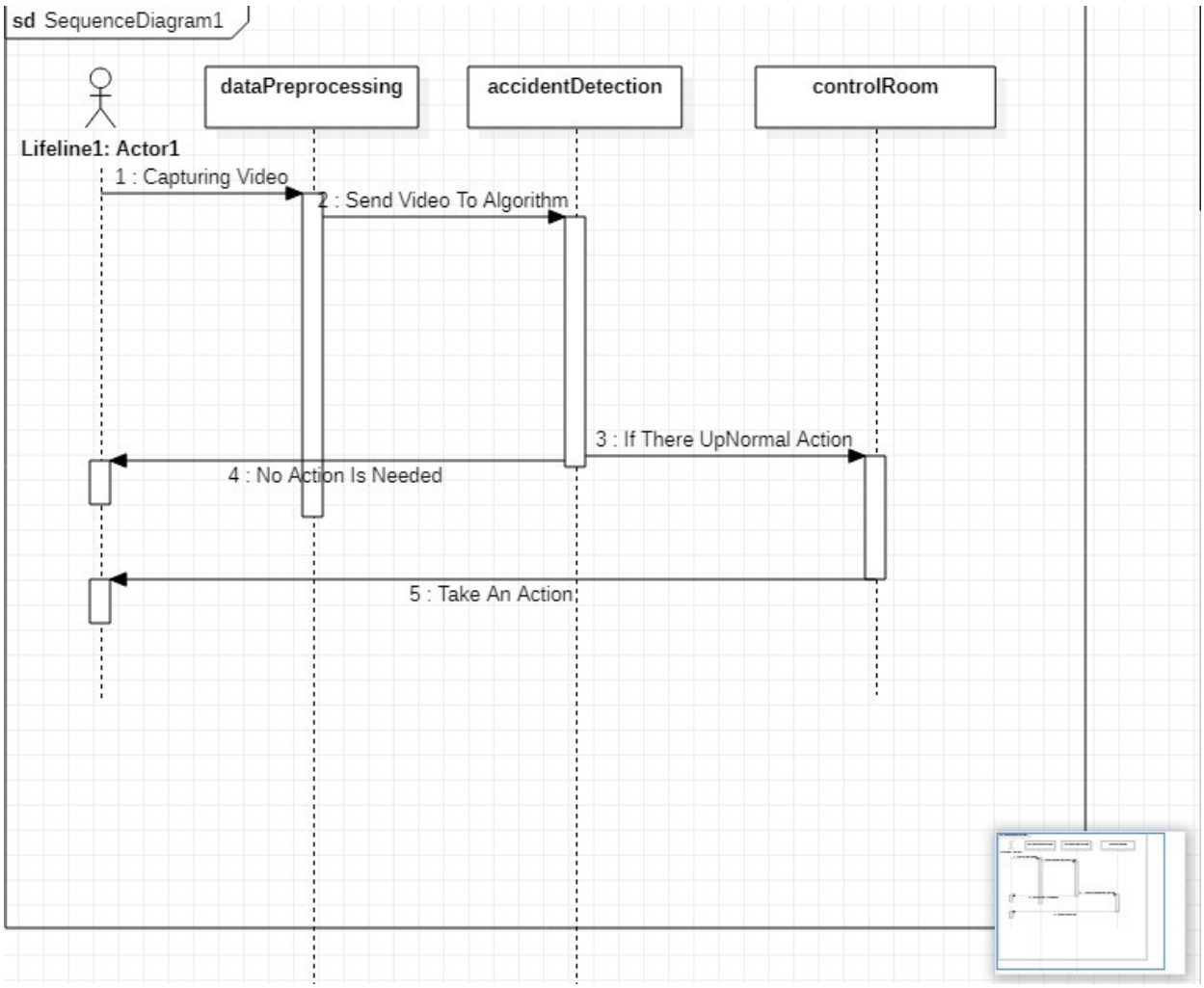
Chapter 4 : System Design

1) System Component Diagram



2) Class Diagram.

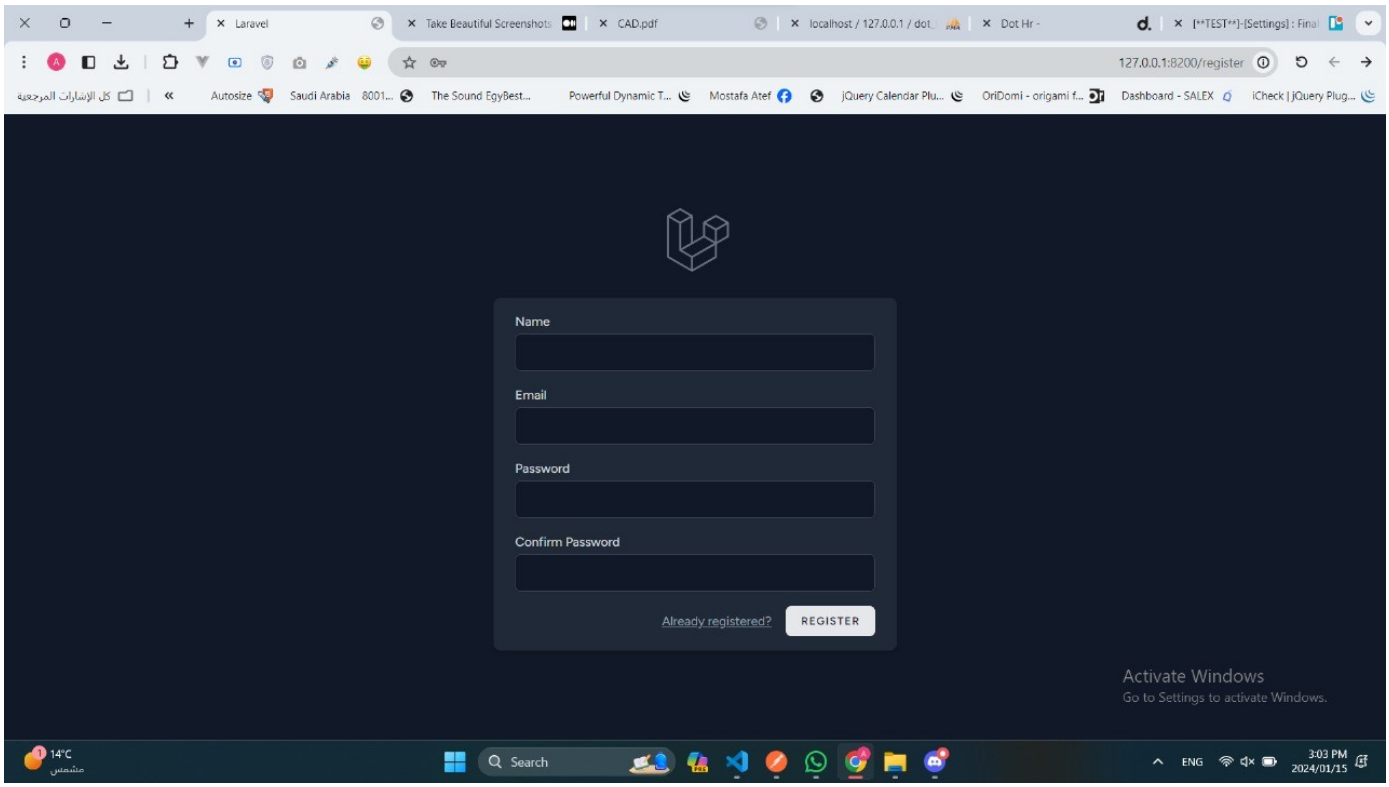




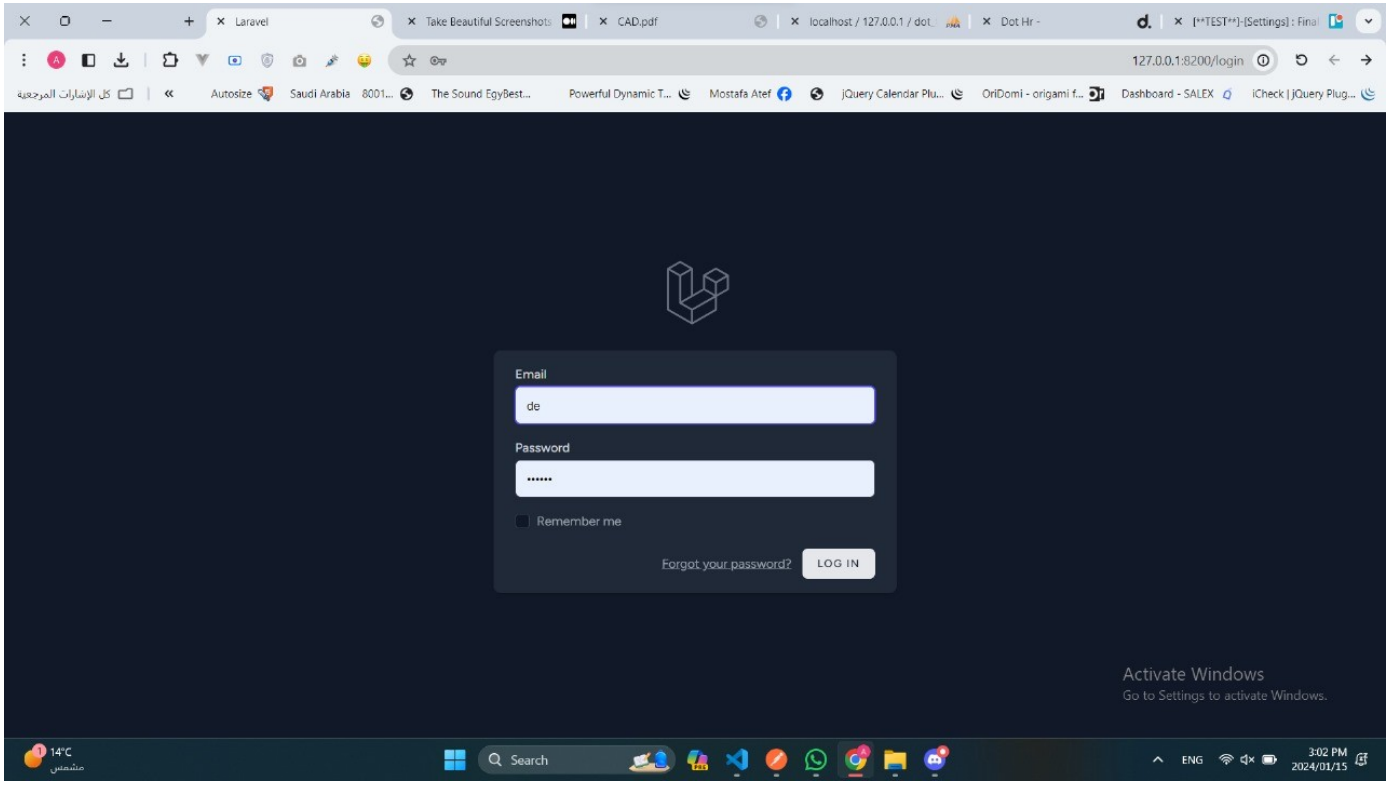
3)Sequence Diagram.

4) System GUI Design

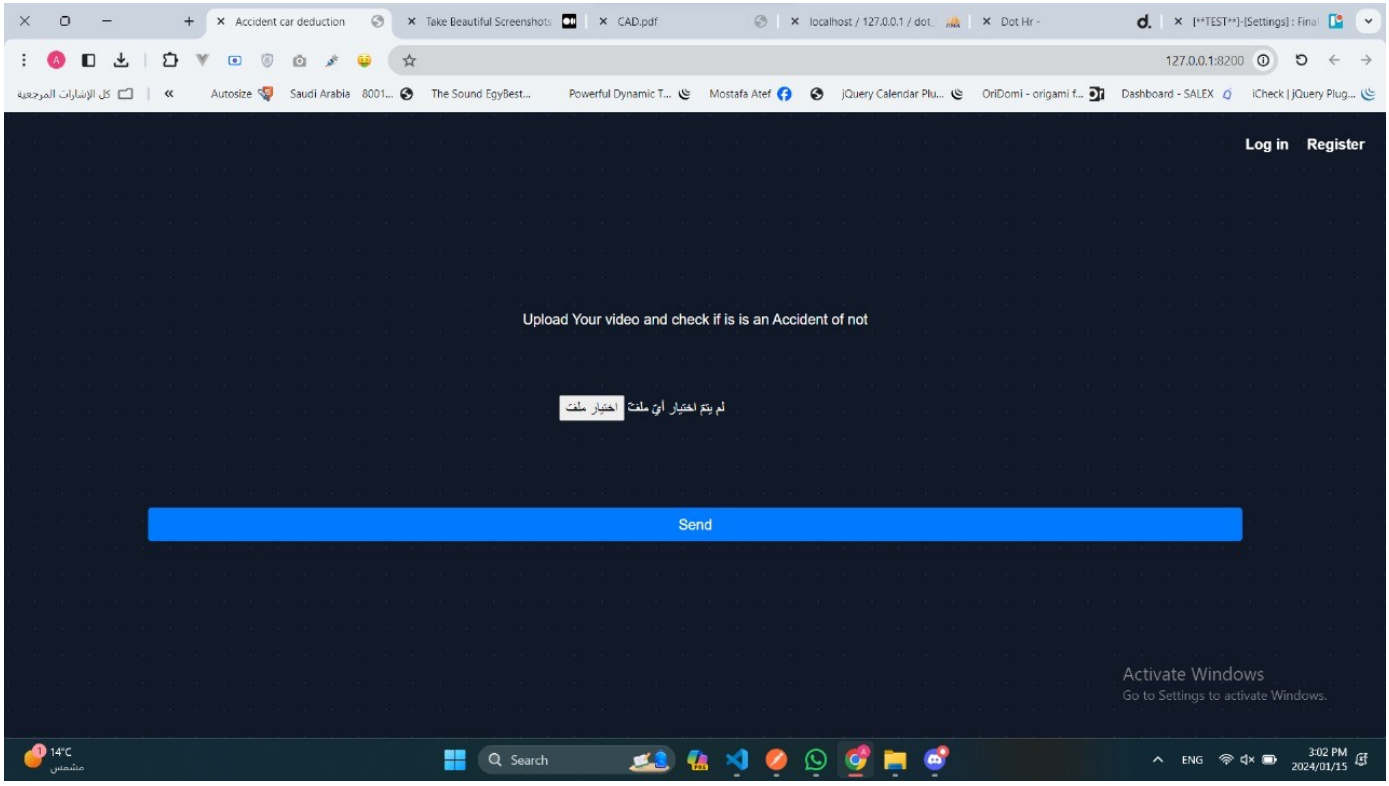
1) Creating a new user



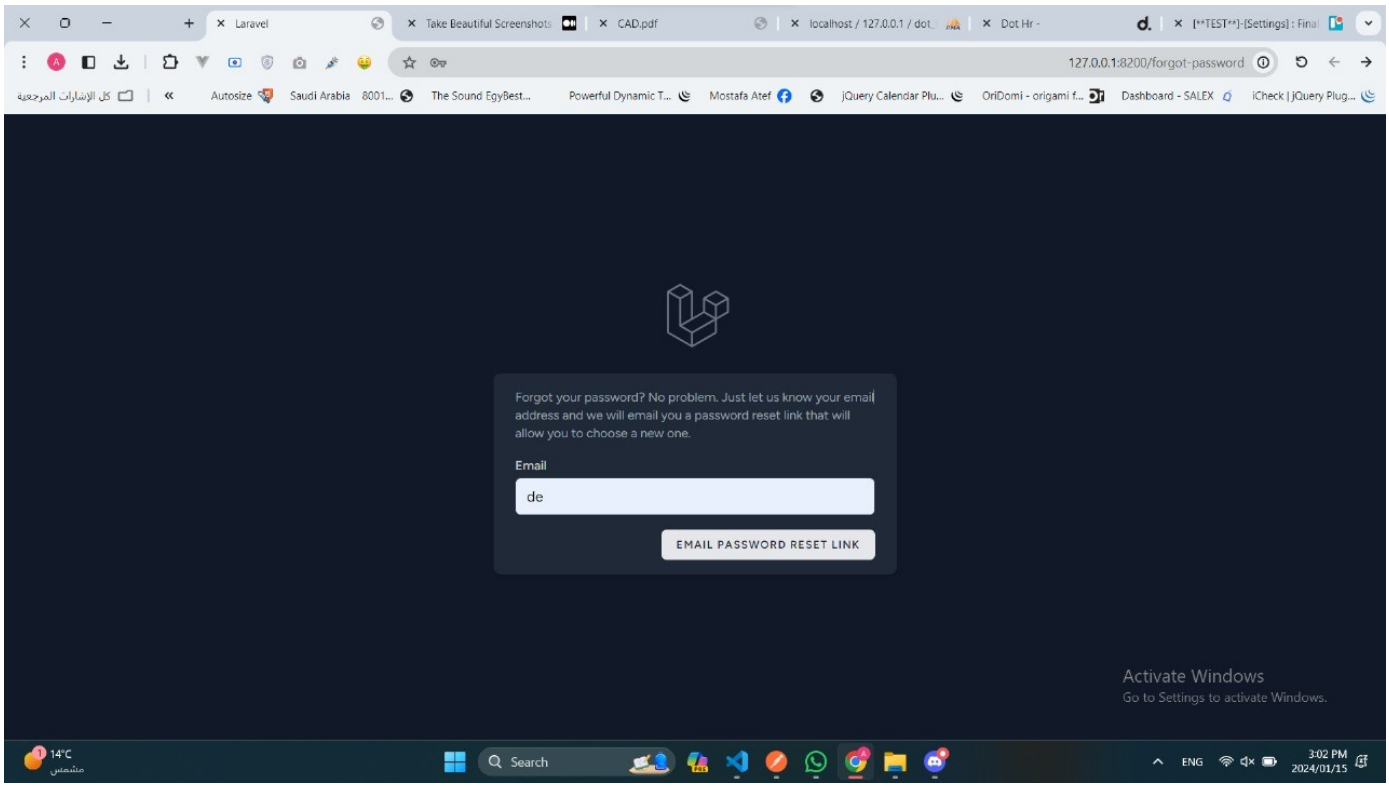
2) Log-in page


Email
de
Password
Remember me
Forgot your password?
LOG IN
Activate Windows
Go to Settings to activate Windows.
Search

3) Front-end Upload Video That Simulate Camera



4) Resetting password



Chapter 5
Implementation and Testing

1) Model Implementation

2) Flask Application

3) Notification System

4) Front-end System

1) Model Implement

Model Methods

1) Load Model

```
5
6   def load_model():
7   |
8   |   model_path = 'model'
9   |   loaded_model = tf.keras.models.load_model(model_path)
10  |   return loaded_model
11
```

Loading The pretrained Model Weights

2) Process frames for training

```
11
12  def process_frame(frame):
13  |
14  |   resized_frame = tf.keras.preprocessing.image.smart_resize(frame, (48, 48), interpolation='bilinear')
15  |   resized_gray_frame = np.dot(resized_frame[..., :3], [0.299, 0.587, 0.114])
16  |   image_array = tf.keras.utils.img_to_array(resized_gray_frame)
17  |   image_batch = np.expand_dims(image_array, axis=0)
18  |   return image_batch
```

Processing frames method

3) Predict frame

```
def predict_frame(model, frame):

    processed_frame = process_frame(frame)
    full_prediction = (model.predict(processed_frame) > 0.5).astype('int32')
    return full_prediction

def is_frame_accident(model, frame):

    prediction = predict_frame(model, frame)
    if prediction[0][0] == 1:
        return "Accident detected"
    else:
        return "Not accident"
```

Two methods one to predict if the frame is an accident and one to return a string with the result

4) Predict the Top Classes

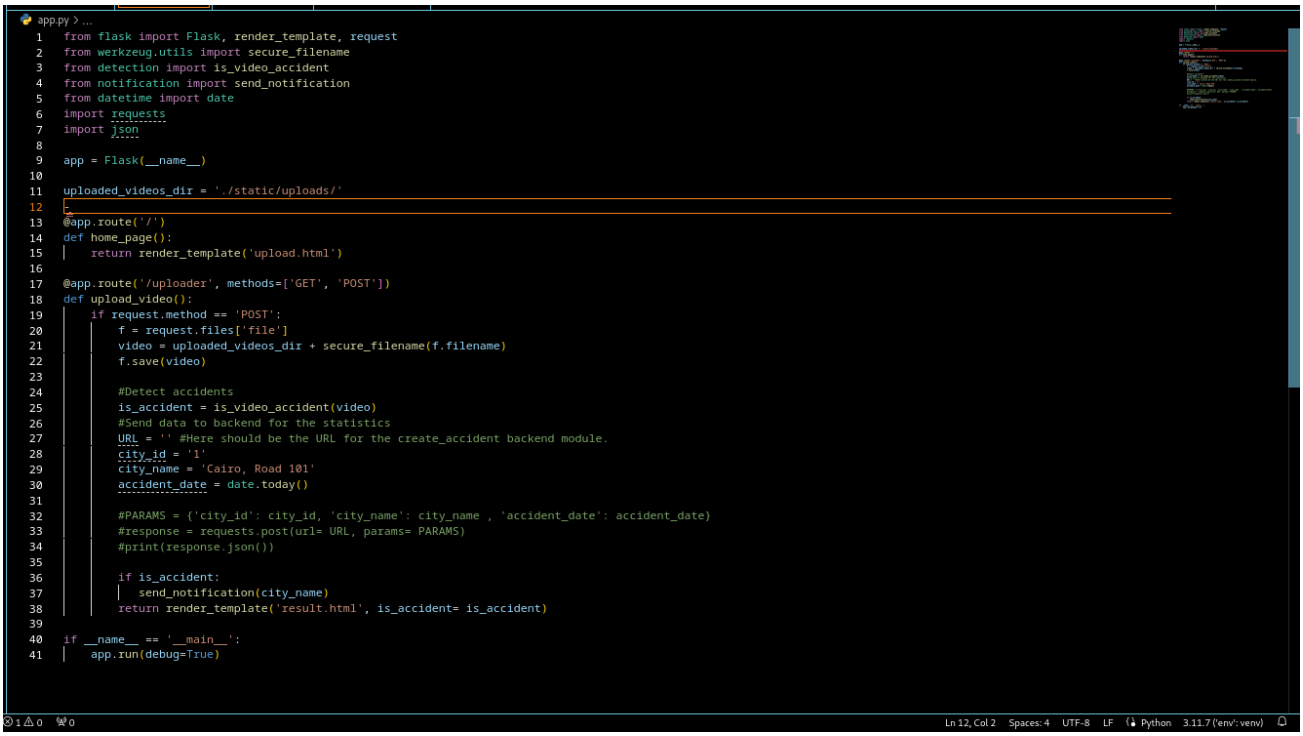
```
def predict_every_video_frame(model, video_path):

    frames=[]
    labels=[]
    video = cv2.VideoCapture(video_path)
    video_frames_count = int(video.get(cv2.CAP_PROP_FRAME_COUNT))
    for x in range(video_frames_count):
        if x%30 == 0:
            _, frame = video.read()
            frames.append(frame)
            labels.append(is_frame_accident(model, frame))

    video.release()
    video_preds = [labels, frames]

    return video_preds
```

This method takes every frame in the video and predict it individually

5) the Main Function To predict Class (accident or Not)

```
def is_video_accident(video_path):

    model = load_model()
    labels = predict_every_video_frame(model, video_path)[0]
    for label in labels:
        if label == 'Accident detected':
            return True
    return False
```

3) Flask APP



```python
from flask import Flask, render_template, request
from werkzeug.utils import secure_filename
from detection import is_video_accident
from notification import send_notification
from datetime import date
import requests
import json

app = Flask(__name__)

uploaded_videos_dir = './static/uploads/'

@app.route('/')
def home_page():
    return render_template('upload.html')

@app.route('/uploader', methods=['GET', 'POST'])
def upload_video():
    if request.method == 'POST':
        f = request.files['file']
        video = uploaded_videos_dir + secure_filename(f.filename)
        f.save(video)

        #Detect accidents
        is_accident = is_video_accident(video)
        #Send data to backend for the statistics
        URL = '' #Here should be the URL for the create_accident backend module.
        city_id = '1'
        city_name = 'Cairo, Road 101'
        accident_date = date.today()

        #PARAMS = {'city_id': city_id, 'city_name': city_name , 'accident_date': accident_date}
        #response = requests.post(url= URL, params= PARAMS)
        #print(response.json())

        if is_accident:
            send_notification(city_name)
        return render_template('result.html', is_accident= is_accident)

if __name__ == '__main__':
    app.run(debug=True)
```

app.py explanation:

A. import the libraries

B. we have two routes ‘/’ and ‘/uploader’

C. ‘/’ has a function called home_page that will return html page.

D.‘/uploader’ has a function called upload_file that will do the following:

a. get the uploaded file and save it in a folder
b. call the (is_video_accident) function from the detection.py file to check if the model detects an accident or not and save a new video that contains one of the values (accident, no accident) on each frame.
c. (is_video_accident) function returns a boolean value and the path of the new video.

d. send the result with the city_id, date, city_name to API that will help us in the statistics

e. **check if there is an accident, call the (sendNotification) function from the notification.py file that will send a notification to the firebase cloud messaging (FCM)**

4) Notification System

notification.py > send_notification

```python
import requests
import json

def send_notification(city_name):
    server_token = 'AAAArz9M-tE:APA91bGXRq_CjCMmRnBU7PJpOmxTop1dswRiiCWJy3RbK3QLwhAs6Mw4MYx6VM9ndrhhSEeMdz8d3CNx924xMFCQV3IsFvhtaDcCiW_kzaTXmWwfxjMiGG048_2YQ3xHYW
    device_token = 'fGqE1MsbSgq5VjKgZSa24N:APA91bHXFCbLK3JN8evyHXcYkScG8zLcVxRkaEMjmsY1ApkpbUxwHckRZJYwWf_VkdUD2TvURcMsntJ1N8WqevNUyftShK3UBTqdVgGCBPyInJ_f4pmD4gWl

    headers = {
        'Content-Type': 'application/json',
        'Authorization': 'key=' + server_token,
    }

    body = {
        'notification': {
            'title': 'ALERT',
            'body': 'Accident at ' + city_name
        },

        'to': device_token,
        'priority': 'high',
    }

    #Sending a post request to firebase
    response = requests.post("https://fcm.googleapis.com/fcm/send", headers= headers, data= json.dumps(body))
    if response.status_code == 200:
        print(response.json())
        return True
    return False
```

notification.py explanation:

1- import the libraries
2- we have a function called send-Notification() that will send a notification to the firebase cloud messaging using API and return a boolean value.
1- We initialized the server token (we get it from the firebase cloud messaging) and the device token (we get it from the mobile application)
2- We initialized the headers and body of the API to send the request
3- Send the request to the firebase cloud messaging API and check if the status_code of the result = 200 then return True else return False

5) Front-end System

1) Front Home Page

```blade
<body class="antialiased" style="width: 100% !important">
    <div style="width: 100% !important"
        class="relative sm:flex sm:justify-center sm:items-center min-h-screen bg-dots-darker bg-center bg-gray-100 dark:bg-dots-lighter dark:bg-gray-900 selection:bg-red-500 selection:text-white">
        @if (Route::has('login'))
            <div class="sm:fixed sm:top-0 sm:right-0 p-6 text-right z-10">
                @auth
                    <a href="{{ url('/dashboard') }}"
                        class="font-semibold text-gray-600 hover:text-gray-900 dark:text-gray-400 dark:hover:text-white focus:outline focus:outline-2 focus:rounded-sm focus:outline-red-500">Dashboard</a>
                @else
                    <a href="{{ route('login') }}"
                        class="font-semibold text-gray-600 hover:text-gray-900 dark:text-gray-400 dark:hover:text-white focus:outline focus:outline-2 focus:rounded-sm focus:outline-red-500">Log
                        in</a>

                    @if (Route::has('register'))
                        <a href="{{ route('register') }}"
                            class="ml-4 font-semibold text-gray-600 hover:text-gray-900 dark:text-gray-400 dark:hover:text-white focus:outline focus:outline-2 focus:rounded-sm focus:outline-red-500">Register</a>
                    @endif
                @endauth
            </div>
        @endif

        <div class="max-w-7xl mx-auto p-6 lg:p-8" style="width: 100% !important">
            <center>
                <h1>Upload Your video and check if is is an Accident of not</h1>
            </center>
            <br>

            <form id="uploadForm" enctype="multipart/form-data">
                @csrf
                <div id="fileUpload"></div><br><br>
                <center><input type="file" name="file" accept="video/*"></center><br><br><br><br>
                <button type="button" class="btn btn-primary" id="uploadButton">Send</button>
            </form>
        </div>
    </div>
    <script src="https://code.jquery.com/jquery-3.7.1.min.js"></script>
    <script>
        $(document).ready(function() {
            $('#uploadButton').on('click', function() {
                $.ajax({
                    url: '{{ route('user.upload') }}', // Replace with your server-side upload endpoint
                    type: 'POST',
                    data: new FormData($('#uploadForm')[0]),
                    processData: false,
                    contentType: false,
                    success: function(response) {
                        if (response == "is_accident") {
                            alert("oh no.! , this is an accident");
                        } else {
                            alert("don't worry , this is not an accident");
                        }
                    },
                    error: function(error) {
                        alert("oh no.! , this is an accident");
                    }
                });
            });
        });
    </script>
</body>

</html>

```

6)Mobile Application (Accidents Alert)

The Application consists of two parts

3- UI (activity_main.xml and MainActivity.Java)

4- Get Notification using firebase cloud messaging (FCM) (MyFirebaseMessagingService.java).

First Part: UI (User Interface)

It is a very simple application that contains a simple appbar with the application name and a text at the center of the application (‘Hello From CAD’)

Code:

We have two files and we will explain each one of them in the following pages.

4- MainActivity.Java
5- front-end xml file

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Device registration token" />

    <EditText
        android:id="@+id/etToken"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:ems="10"
        android:inputType="textMultiLine|textPersonName"
        android:text="Name" />

</LinearLayout>
```

we just added a TextView with an id ‘textview’ and add the following attributes:

1- change the text color
2- change the text size
3- make the text style bold
4- make the text the center of the screen

1- **MainActivity.java** (everything explained in comments)

```java
package com.example.myapplication;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.EditText;
import android.widget.Toast;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.messaging.FirebaseMessaging;

public class MainActivity extends AppCompatActivity {

    EditText etToken;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        etToken = findViewById(R.id.etToken);


        FirebaseMessaging.getInstance().getToken()
                .addOnCompleteListener(new OnCompleteListener<String>() {
                    @Override
                    public void onComplete(@NonNull Task<String> task) {
                        if (!task.isSuccessful()) {
                            System.out.println("Fetching FCM registration token failed");
                            return;
                        }

                        // Get new FCM registration token
                        String token = task.getResult();

                        // Log and toast
                        System.out.println(token);
                        Toast.makeText(MainActivity.this, "Your device registration token is" + token
                                , Toast.LENGTH_SHORT).show();

                        etToken.setText(token);
                    }
                });
    }
}
```

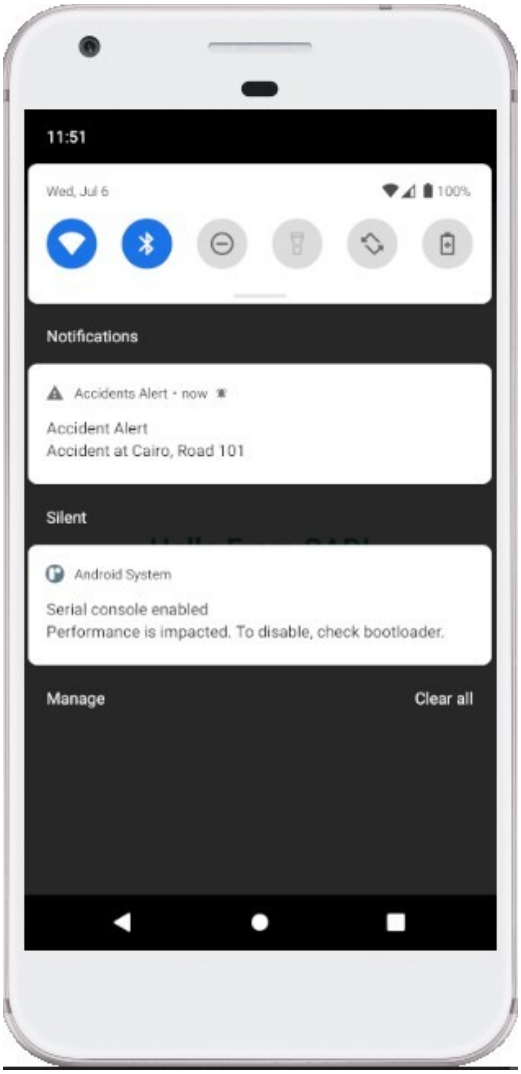
Second Part: Notifications

we used Firebase Cloud Messaging (FCM) to implement this part in the application

The application will display a notification when the accident is occurred whether the application on the background or not (we handled the two cases)

This is a screenshot of the application when getting the notification

- Here we got an Accident Alert at Cairo, Road 101



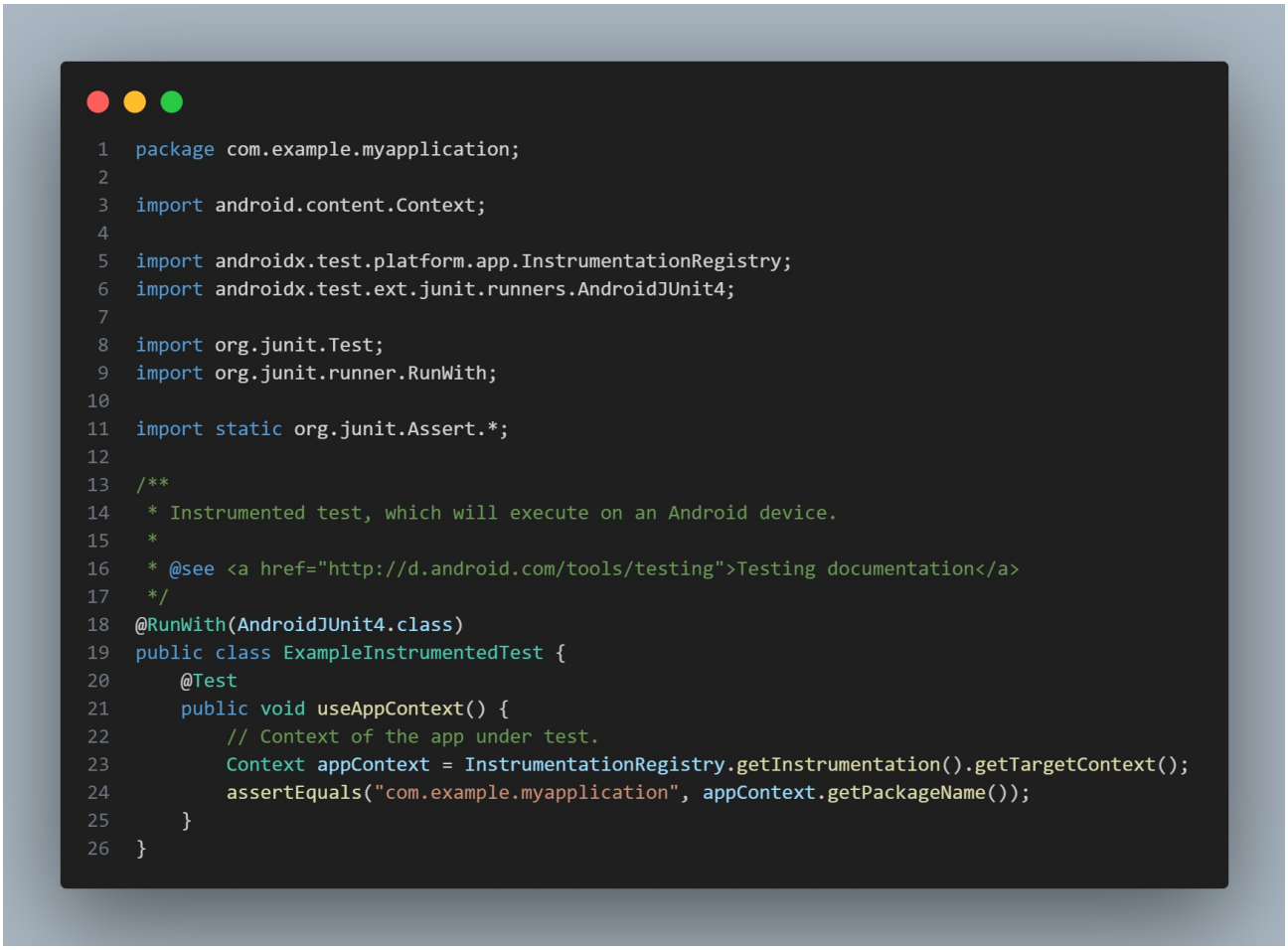
Coding:

We have one file called

MyFirebaseMessagingService.java and we explained everything in comments

MyFirebaseMessagingService.Java

```java
package com.example.myapplication;

import android.app.NotificationChannel;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.media.RingtoneManager;
import android.net.Uri;
import android.os.Build;
import android.os.Handler;
import android.os.Looper;
import android.widget.Toast;

import androidx.core.app.NotificationCompat;

import com.google.firebase.messaging.FirebaseMessagingService;
import com.google.firebase.messaging.RemoteMessage;

public class MyFirebaseMessagingService extends FirebaseMessagingService {
    @Override
    public void onMessageReceived(RemoteMessage remoteMessage) {
        // ...

        // TODO(developer): Handle FCM messages here.
        // Not getting messages here? See why this may be: https://goo.gl/39bRNJ
        System.out.println("From: " + remoteMessage.getFrom());

        // Check if message contains a notification payload.
        if (remoteMessage.getNotification() != null) {
            System.out.println("Message Notification Body: " + remoteMessage.getNotification().getBody());
        }

        // Also if you intend on generating your own notifications as a result of a received FCM
        // message, here is where that should be initiated. See sendNotification method below.
        sendNotification(remoteMessage.getFrom(), remoteMessage.getNotification().getBody());
        sendNotification(remoteMessage.getNotification().getBody());
    }

    private void sendNotification(String from, String body) {
        new Handler(Looper.getMainLooper()).post(new Runnable() {

            @Override
            public void run() {
                Toast.makeText(MyFirebaseMessagingService.this.getApplicationContext(), from + " -> " + body,Toast.LENGTH_SHORT).show();
            }
        });
    }

    private void sendNotification(String messageBody) {
        Intent intent = new Intent(this, MainActivity.class);
        intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0 /* Request code */, intent,
                PendingIntent.FLAG_ONE_SHOT | PendingIntent.FLAG_IMMUTABLE);


        String channelId = "My channel ID";
        Uri defaultSoundUri = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
        NotificationCompat.Builder notificationBuilder =
                new NotificationCompat.Builder(this, channelId)
                        .setSmallIcon(R.drawable.ic_launcher_background)
                        .setContentTitle("My new notification")
                        .setContentText(messageBody)
                        .setAutoCancel(true)
                        .setSound(defaultSoundUri)
                        .setContentIntent(pendingIntent);

        NotificationManager notificationManager =
                (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);

        // Since android Oreo notification channel is needed.
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            NotificationChannel channel = new NotificationChannel(channelId,
                    "Channel human readable title",
                    NotificationManager.IMPORTANCE_DEFAULT);
            notificationManager.createNotificationChannel(channel);
        }

        notificationManager.notify(0 /* ID of notification */, notificationBuilder.build());
    }
}

```



```java
package com.example.myapplication;

import android.content.Context;

import androidx.test.platform.app.InstrumentationRegistry;
import androidx.test.ext.junit.runners.AndroidJUnit4;

import org.junit.Test;
import org.junit.runner.RunWith;

import static org.junit.Assert.*;

/**
 * Instrumented test, which will execute on an Android device.
 *
 * @see <a href="http://d.android.com/tools/testing">Testing documentation</a>
 */
@RunWith(AndroidJUnit4.class)
public class ExampleInstrumentedTest {
    @Test
    public void useAppContext() {
        // Context of the app under test.
        Context appContext = InstrumentationRegistry.getInstrumentation().getTargetContext();
        assertEquals("com.example.myapplication", appContext.getPackageName());
    }
}
```