A
Industry Oriented Mini Project Report

On

"AUTOPATROL - AI-POWERED TRAFFIC VIOLATION MONITORING"

Submitted in partial fulfillment of the
Requirements for the award of the degree of

Bachelor of Technology

In

Computer Science & Engineering

By

K. Satish Reddy	**–22R21A05F3**
K. Sai Prakash	**–22R21A05F8**
M. Akshay Kumar	**–22R21A05G7**
M. Manobharath	**–22R21A05G9**

Under the guidance of

Dr. P. Micheal Preetam Raj
Associate Professor

Department of Computer Science & Engineering

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS
MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH, Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

2025

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project entitled **“AutoPatrol - AI-Powered Traffic Violation Monitoring”** has been submitted by **K. Satish Reddy (22R21A05F3), K. Sai Prakash (22R21A05F8), M. Akshay Kumar (22R21A05G7) and M. Manobharath (22R21A05G9)** in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technological University, Hyderabad. The results embodied in this project have not been submitted to any other University or Institution for the award of any degree or diploma.

Internal Guide **Head of the Department**

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

Department of Computer Science & Engineering

DECLARATION

We hereby declare that the project entitled **"AutoPatrol - AI-Powered Traffic Violation Monitoring"** is the work done during the period from **January 2025 to June 2025** and is submitted in partial fulfillment of the requirements for the award of degree of Bachelor of Technology in Computer Science and Engineering from Jawaharlal Nehru Technology University, Hyderabad. The results embodied in this project have not been submitted to any other university or Institution for the award of any degree or diploma.

K. Satish Reddy	**22R21A05F3**
K. Sai Prakash	**22R21A05F8**
M. Akshay Kumar	**22R21A05G7**
M. Manobharath	**22R21A05G9**

MARRI LAXMAN REDDY GROUP OF INSTITUTIONS MLR INSTITUTE OF TECHNOLOGY
(UGC AUTONOMOUS)
Affiliated to JNTUH,Approved by AICTE
Laxman Reddy Avenue, Dundigal, Hyderabad-500 043, Telangana, India

Department of Computer Science & Engineering

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be incomplete without the mention of people who made it possible, whose constant guidance and encouragement crowned our efforts with success. It is a pleasant aspect that we now have the opportunity to express our guidance for all of them.

First of all,we would like to express our deep gratitude towards our internal guide **Dr. P. Micheal Preetam Raj, Associate Professor, Department of CSE** for his support in the completion of our dissertation. We wish to express our sincere thanks to **Dr. A. BALARAM, HOD, Department of CSE** and also **Principal Dr. K. SRINIVAS RAO** for providing the facilities to complete the dissertation.

We would like to thank all our faculty and friends for their help and constructive criticism during the project period. Finally, we are very much indebted to our parents for their moral support and encouragement to achieve goals.

K. Satish Reddy	**22R21A05F3**
K. Sai Prakash	**22R21A05F8**
M. Akshay Kumar	**22R21A05G7**
M. Manobharath	**22R21A05G9**

Department of Computer Science & Engineering

ABSTRACT

AutoPatrol is a traffic violation monitoring system that uses artificial intelligence to detect red signal violations from uploaded video footage. The system allows traffic police to upload traffic surveillance videos through a web interface. During video playback, the police can dynamically toggle the traffic signal state (red or green) using the interface.

The system processes the video continuously, but violation detection is activated only when the signal is toggled to red. While the video continues to play, frames are analyzed using OpenCV for preprocessing and Tesseract OCR for number plate recognition. The extracted number plates of vehicles that violate the red signal are stored in a MySQL database along with the video name and timestamp.

Vehicle owners can search the system by entering their vehicle number to check for any recorded violations. The platform is built using Flask for the backend, HTML, CSS, and JavaScript for the frontend, and MySQL (via XAMPP) for data storage. The system is optimized for project demonstrations using pre-recorded video footage in place of real-time video streams.

AutoPatrol automates the detection of red light violations and provides a preview of detected plates during processing, followed by a message indicating the end of the video. This solution improves the efficiency of monitoring and helps in reducing manual effort in enforcing traffic rules.

LIST OF FIGURES

INDEX

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

This project introduces AutoPatrol: AI-Powered Traffic Violation Monitoring, a web-based traffic surveillance system designed to detect red light violations using artificial intelligence. The system enables traffic police to upload video footage from traffic cameras and dynamically control the traffic signal state (red or green) during video playback.

When the signal is toggled to red, the system activates vehicle detection using OpenCV for frame processing and Tesseract OCR to extract vehicle number plates. Detected violations—specifically vehicles crossing the stop line during the red signal—are automatically logged into a MySQL database along with video details and timestamps.

This system operates on a Flask-based backend, uses HTML, CSS, and JavaScript for the frontend, and is compatible with MySQL via XAMPP. Vehicle owners can access the portal to check for any violations by entering their number plate. AutoPatrol aims to enhance traffic rule enforcement through automation, real-time detection, and a user-friendly interface

1.2 PURPOSE OF THE PROJECT

The primary purpose of this project is to automate the detection and reporting of red signal violations using AI-based video processing techniques. By allowing traffic police to dynamically control the signal state during video playback and enabling automated license plate recognition, the system reduces manual monitoring and enhances traffic law enforcement.

The integration of OCR technology ensures that vehicle number plates are extracted accurately and stored for future reference. The system also supports public transparency by providing vehicle owners with access to their violation records. Overall, the project delivers a practical and scalable solution for improving traffic surveillance in urban environments.

1.3 MOTIVATION

The motivation behind this project arises from the need to improve traditional traffic monitoring systems, which often depend on manual inspection or fixed surveillance infrastructure. Manual enforcement of red light violations is labor-intensive, error-prone, and lacks scalability.

With advancements in AI, computer vision, and OCR, it is now feasible to develop systems that can automatically detect violations from existing video footage. AutoPatrol addresses this gap by providing a robust, automated tool for detecting red signal violations, reducing the burden on traffic authorities, and promoting road safety. This aligns with the broader goal of integrating intelligent systems into urban traffic management to build smarter, more efficient cities.

CHAPTER 2

LITERATURE SURVEY

Recent developments in AI and computer vision have enabled smarter traffic monitoring systems, particularly in detecting and analyzing violations from surveillance footage. Various implementations have explored automatic number plate recognition (ANPR) using OCR tools like Tesseract and EasyOCR, along with object detection models such as YOLO, for identifying vehicles and their movements.

However, most studies focus either on adaptive traffic control or basic number plate recognition, without combining both functionalities in an integrated, actionable system. The use of deep learning has improved accuracy in visual recognition, but real-world applications still struggle with performance under varying conditions and lack real-time adaptability.

Our project addresses these limitations by using Tesseract OCR for plate recognition and integrating it into a Flask-based web interface where police officers can upload videos and control signal states during playback. The system processes video frames in real-time, detects violations during red signals only, and logs data into a centralized MySQL database, thus making AI-based monitoring practical and actionable.

2.1 EXISTING SYSTEM

Existing traffic violation detection systems typically fall into two categories: manual monitoring and automated systems with fixed rule sets. Manual systems rely on traffic police reviewing camera feeds or video footage, which is time-consuming and prone to human error. Some automated systems can detect vehicles and even extract license plates, but they often lack integration with real-time decision-making capabilities.

Additionally, most existing solutions detect objects continuously without context, such as signal color, leading to unnecessary processing or false positives. Earlier systems also struggle with OCR in poor lighting, skewed plates, or blurred footage. There is often no mechanism for user interaction (e.g., registered vehicle owners checking their violations), and these systems are rarely scalable or modular enough to integrate into a broader smart city ecosystem.

In contrast, AutoPatrol combines live video analysis, red signal-controlled detection, and automated database logging into a single, interactive platform, offering a practical alternative to rigid, single-purpose systems.

The Delhi Traffic Police ANPR System, developed by the Delhi Traffic Police in collaboration with private vendors, introduced basic automatic number plate recognition and e-challan generation for traffic violators. While it incorporated helmet and triple-riding detection, the system lacked dynamic control of traffic signals and had no real-time interaction or preview interface for enforcement officers. Its functionality was fixed to pre-installed road cameras, limiting its adaptability and manual signal control during live video playback.

The Kerala Police Speed Violation Detection System, deployed by the Kerala Police in 2021, uses radar-based speed detection along with OCR-based number plate recognition. Although effective in capturing speed violations, it does not detect red light jumping or allow officers to toggle signal states manually. The system works passively and lacks integration with live or recorded video analysis features and adaptive signal management, making it less flexible in multi-scenario traffic enforcement.

The Surat Smart City ANPR Network, implemented by Surat Smart City Development Ltd., focuses on city-wide surveillance and license plate tracking. While it provides good integration with smart cameras across intersections, it is limited to passive monitoring. The system does not support red signal violation detection, adaptive signal toggling, or vehicle preview interfaces, thus restricting its utility in dynamic traffic violation enforcement and live video uploads.

The Chandigarh Red Light Violation Detection Program, rolled out by Chandigarh Police in 2022, utilizes red light violation cameras that issue challans automatically when vehicles cross the stop line. However, this system operates entirely on pre-set traffic signals and cannot respond to officer-initiated changes or dynamically detect violations based on video input. The absence of object detection integration and license plate OCR outside fixed camera feeds limits its flexibility.

The OpenALPR Free Version, developed by OpenALPR and later acquired by Rekor Systems, offers license plate recognition from images and video streams. Though widely used for basic OCR purposes, it does not include red signal logic, object detection, or integration with signal status. Designed for simple recognition tasks, it lacks enforcement logic, signal control features, and an officer review interface, reducing its suitability for real-time smart traffic management.

2.2 DISADVANTAGES OF EXISTING SYSTEM

Concisely summarizing the disadvantages of the above implementations:

- **Lack of Real-Time Signal Control**

Existing traffic violation systems often rely on static input parameters and do not allow law enforcement to toggle traffic signal states (e.g., Red or Green) while analyzing video feeds. This limits the accuracy of violation detection, as the system cannot distinguish whether a vehicle crossed during an active red signal or not.

- **Always-On Detection**

Most current solutions continuously detect and log vehicles without considering the current state of the traffic signal. This results in excessive or irrelevant data capture, including vehicles that are not violating any rules. It leads to false positives and reduces the reliability of the violation reports.

- **Low OCR Robustness**

Existing OCR modules fail to perform accurately under poor lighting, low-resolution video, non-standard fonts, or angled number plates. This severely affects number plate recognition and the reliability of vehicle identification, especially in practical urban scenarios with inconsistent video quality.

- **No User Access Portal**

Many systems do not offer a dedicated portal for vehicle owners to log in and check for violations against their vehicle. This creates a transparency gap and increases dependency on physical challan delivery or manual checking with traffic authorities.

- **Not Action-Oriented**

In many research-based or semi-automated systems, detection is limited to drawing bounding boxes or logging data. There is no integrated enforcement mechanism such as challan generation, alert dispatch via SMS/email, or official logging of offenses, which reduces the real-world utility of such systems.

- **Non-Scalable Architectures**

Most implementations are local and lack scalability. They are typically designed for a single-machine setup without proper consideration for multi-user environments, concurrent usage, or expansion across multiple traffic junctions or cities.

- **No Web-Based Management Interface**

Many systems are CLI-based or desktop-only applications, which do not provide a centralized dashboard or web interface for traffic police. As a result, officers cannot conveniently upload videos, review violations, or issue challans from a remote browser-based portal.

- **Poor Modularity**

Existing implementations often use tightly coupled code where components like vehicle detection, OCR, and database logging are hard-coded and interdependent. This design makes future updates (e.g., switching to a different OCR engine or integrating a new UI) cumbersome and time-consuming.

CHAPTER 3

PROPOSED SYSTEM

3.1 PROPOSED SYSTEM

The proposed **AutoPatrol** system is an intelligent, AI-driven traffic violation monitoring and management solution that leverages advanced computer vision and deep learning techniques for real-time vehicle detection and automatic number plate recognition. Utilizing the YOLOv8 object detection model, the system accurately detects vehicles in video frames from live traffic cameras or uploaded footage. Integrated with EasyOCR, it extracts license plate information from the detected vehicles to identify and log traffic violations efficiently.

The system dynamically controls traffic signals (Red, Yellow, Green) based on real-time vehicle density, enabling adaptive and smarter traffic flow management. Unlike traditional sensor-based or manual traffic systems, AutoPatrol operates as a vision-based, data-driven platform that processes video feeds without requiring costly hardware installations.

AutoPatrol's modular architecture allows easy scalability and future upgrades, making it suitable for integration with broader smart city traffic networks. This system enhances traffic regulation by combining fast detection, automated violation tracking, and dynamic signal control, ultimately improving road safety and easing congestion.

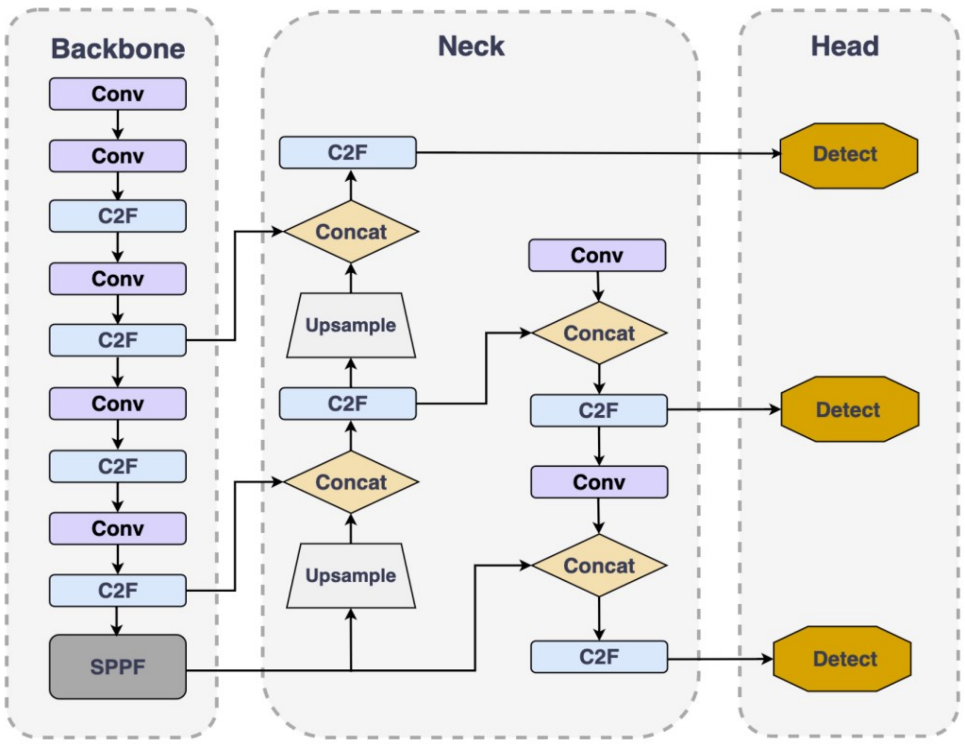


Figure 3.1.1 YOLOv8 Process Flow

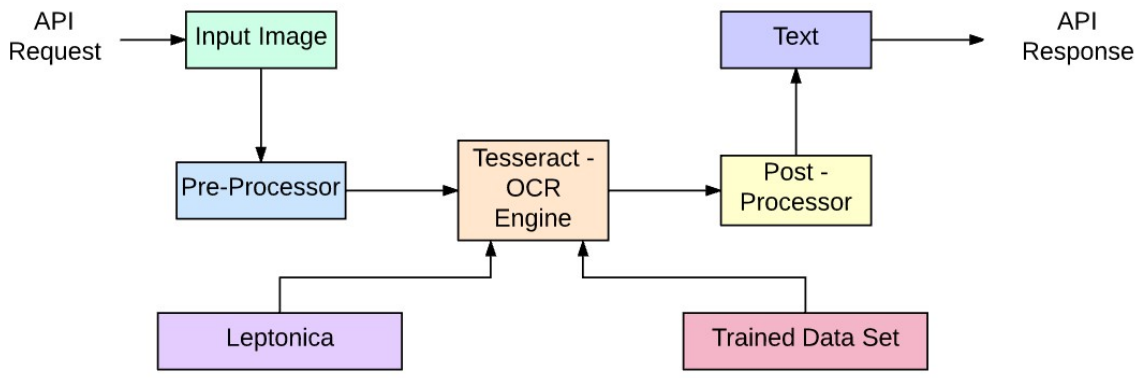


Figure 3.1.2 EasyOCR Process Flow

3.2 ADVANTAGES OF PROPOSED SYSTEM

The proposed AutoPatrol system offers the following advantages:

- **Real-Time Vehicle and Violation Detection:** Utilizes YOLOv8 for fast and accurate detection of vehicles and traffic violations directly from live or recorded video feeds.
- **Dynamic Traffic Signal Control:** Enables traffic police to toggle traffic signals and automates signal timing based on vehicle density, helping reduce congestion and improve traffic flow.
- **Integrated Automatic Number Plate Recognition:** Employs EasyOCR to accurately read and log vehicle number plates, facilitating automated violation recording and enforcement.
- **Cost-Effective and Easy to Deploy:** Operates on standard hardware like webcams or existing CCTV cameras, eliminating the need for expensive sensors or infrastructure upgrades.
- **Enhanced Monitoring and Enforcement:** Improves accuracy and consistency over manual methods by automatically tracking violations only during red signals, reducing human error and improving law enforcement efficiency.

3.3 SYSTEM ARCHITECTURE

The AutoPatrol system architecture is designed to provide a comprehensive, intelligent solution for real-time traffic violation monitoring and adaptive traffic signal control. The system integrates video input processing, vehicle detection, number plate recognition, and dynamic signal management within a modular framework to ensure scalability and ease of maintenance.

1. Input Source

- **Live Camera Feed:** Real-time video captured from traffic surveillance cameras or webcams installed at intersections. This live feed is the primary data source for vehicle detection and traffic signal monitoring.
- **Pre-recorded Video:** Stored traffic footage used for system testing, model training, or offline analysis of traffic violations and flow patterns.

2. Processing Modules

- **Vehicle Detection (YOLOv8):** The system uses the YOLOv8 deep learning model to detect and classify vehicles (cars, bikes, trucks, buses, etc.) within each video frame. This module outputs bounding boxes and vehicle counts for further processing.
- **Violation Detection Logic:** This logic monitors vehicle behavior relative to the traffic signal state (red/green). It flags and tracks vehicles that violate traffic rules such as crossing on red lights.
- **Number Plate Recognition (EasyOCR):** Once a violating vehicle is detected, EasyOCR extracts the license plate number from the vehicle's bounding box in the video frame, enabling automated logging of violations.

3. Data Processing and Decision Making

- **Vehicle Counting and Signal Control:** Based on vehicle density and traffic conditions, the system either automatically suggests or allows traffic police to toggle traffic signals dynamically, optimizing flow and reducing congestion.
- **Violation Logging:** Violations detected during red signal phases are logged with vehicle number plate details, timestamp, and video evidence for enforcement purposes.
- **Data Storage:** All detection results, including vehicle counts, violation records, and license plate data, are stored securely in a MySQL database for retrieval and reporting.

4. Output Components

- **Traffic Signal Controller Interface:** Allows traffic authorities to manually or automatically control the traffic signals in real time, based on system insights.
- **Violation Preview Display:** Shows detected violations and vehicle information in a user-friendly interface, including a preview of vehicle images and license plates.
- **End-of-Video Notification:** Displays a message when the video feed ends, indicating the completion of the monitoring session.

This modular architecture ensures AutoPatrol can be integrated with existing traffic infrastructure, scaled to cover multiple intersections, and enhanced with additional features like or data analytics.

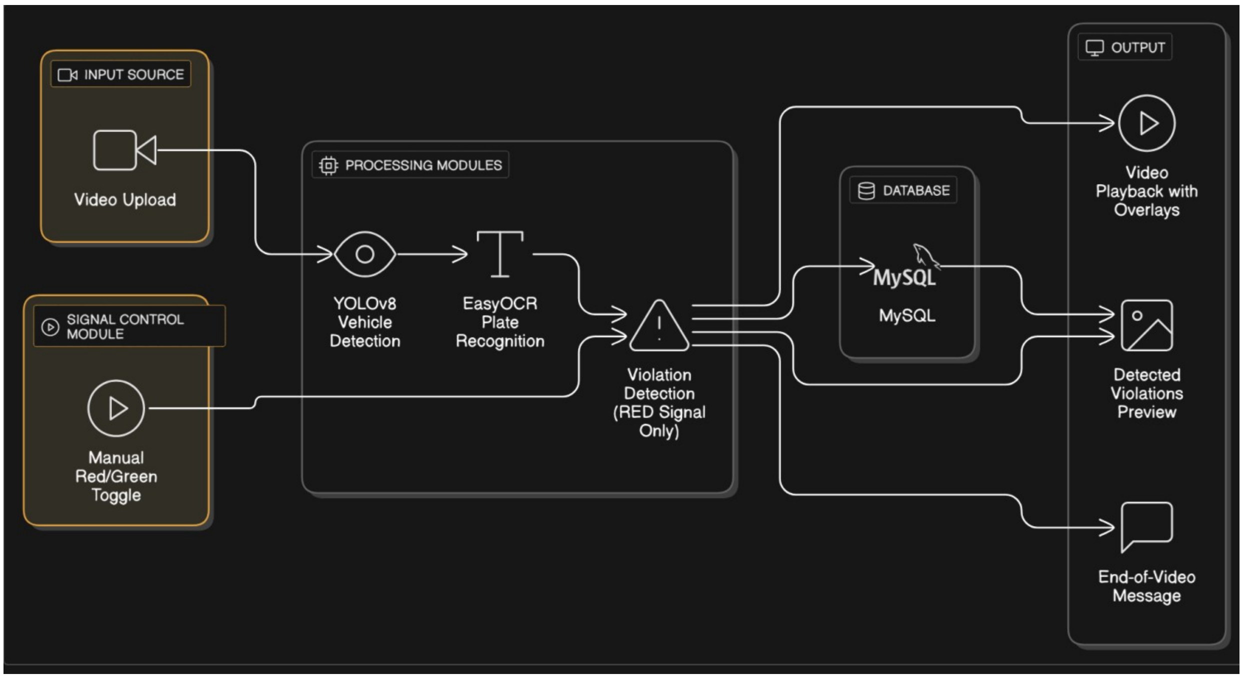


Figure 3.3.1 System Architecture

CHAPTER 4

SYSTEM REQUIREMENTS

This section specifies the system requirements necessary for the development, deployment, and smooth operation of the AutoPatrol traffic violation monitoring application. These requirements focus on the development environment and backend infrastructure. Since the system is designed to be versatile and cross-platform, there are no strict end-user device constraints.

4.1 SOFTWARE REQUIREMENTS

Below are the software requirements for application development:

Operating System: Windows 10/11, Ubuntu 20.04 or later (64-bit recommended)

IDE: Visual Studio Code, PyCharm, or Jupyter Notebook

Programming Language: Python 3.8 or higher

Libraries & Frameworks:

- OpenCV (for video processing and image manipulation)
- Ultralytics YOLOv8 (for vehicle detection)
- EasyOCR (for automatic number plate recognition)
- NumPy, Matplotlib (for data handling and visualization)
- PyTorch (deep learning framework supporting YOLOv8)

Database: MySQL (for storing violation logs, vehicle data, and system records)

Deep Learning Model: Pre-trained YOLOv8n model (yolov8n.pt) for fast and accurate detection.

Version Control & Package Management: Git, pip

Optional Tools: Anaconda (for environment and package management)

4.2 HARDWARE REQUIREMENTS

- **Processor:** Intel Core i5/i7 or AMD Ryzen 5/7 (quad-core or better recommended)
- **RAM:** Minimum 8 GB (16 GB recommended for smoother real-time processing)
- **Storage:** Minimum 512 GB SSD (to enable fast read/write operations for video data)
- **Camera:** Webcam or IP camera for live video feed (if using live monitoring)
- **Graphics:** Dedicated GPU (NVIDIA recommended) for improved YOLOv8 inference speed. (optional but beneficial)

4.3 FUNCTIONAL REQUIREMENTS

- **Real-Time Vehicle Detection:** Utilize YOLOv8 to detect vehicles in real-time from live camera feed or recorded video.
- **Number Plate Recognition:** Extract vehicle number plates using EasyOCR on detected vehicle images.
- **Traffic Signal Management:** Dynamically control traffic signals (Red, Yellow, Green) based on real-time vehicle density and traffic rules.
- **Violation Detection:** Detects and logs traffic violations (e.g., vehicles crossing during red light) automatically.
- **Multi-Input Video Support:** Accept and process both live video streams and pre-recorded traffic footage.
- **User Interface:** Display live video with overlaid bounding boxes, vehicle counts, recognized plate numbers, and current traffic signal status.
- **Data Logging:** Store violation details and traffic data in the MySQL database for reporting and analysis.

4.4 NON-FUNCTIONAL REQUIREMENTS

- **Accuracy:** High precision in vehicle detection and OCR to minimize false positives and negatives in violation detection.

- **Performance:** System must operate in near real-time with minimal latency to support live traffic control.

- **Reliability:** Ensure continuous operation without crashes, memory leaks, or data loss during extended monitoring periods.

- **Usability:** Provide an intuitive interface for traffic authorities to monitor live feeds, receive alerts, and manage signals.

- **Compatibility:** Seamlessly integrate with existing traffic camera hardware and software infrastructure; support various video formats and streaming protocols.

- **Scalability:** Designed to scale from single intersections to city-wide deployments with multiple video feeds and centralized management.

- **Security:** Protect stored data and system access with appropriate authentication and encryption mechanisms.

CHAPTER 5

MODULE DESIGN

5.1 MODULE DESIGN

The module design for the AutoPatrol system involves decomposing the overall functionality into discrete, manageable components that collectively contribute to intelligent, real-time traffic monitoring and violation detection. Each module is developed to handle a specific task, ensuring modularity, scalability, and ease of maintenance.

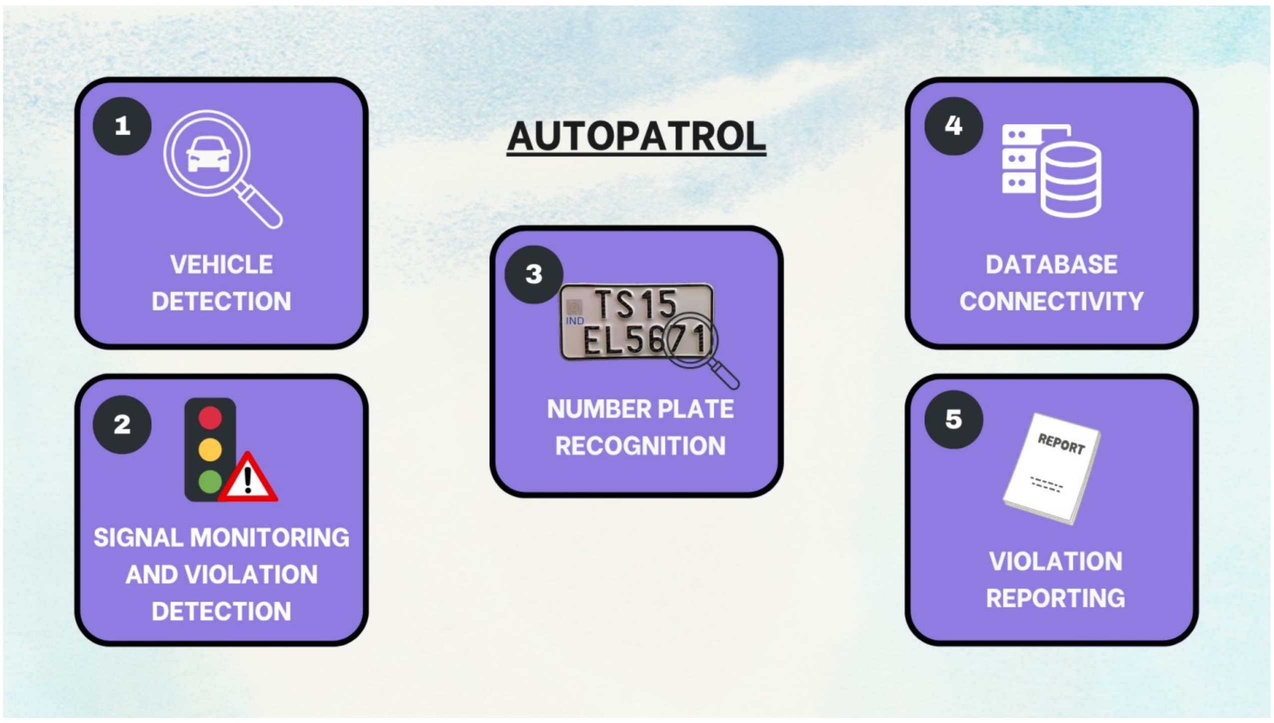


Figure 5.1.1 Module Diagram

Vehicle Detection Module

This is the core module that initiates the traffic monitoring process. It uses the YOLOv8 (You Only Look Once) object detection model to identify and locate vehicles such as cars, bikes, buses, and trucks from live video feeds or uploaded traffic footage. Bounding boxes are drawn around detected vehicles to visually represent the detection in the video feed.

- **Function:** Detects and classifies the vehicles in real-time.
- **Input:** Live feed or pre-recorded video.
- **Output:** Detected vehicle coordinates and class types.
- **Role:** Serves as the trigger for signal monitoring and number plate detection modules.

Signal Monitoring & Violation Detection Module

This module enables traffic police to dynamically toggle the signal status (Red/Green) while the video continues playing. The violation detection logic is activated only when the red signal is ON, identifying vehicles that cross the stop line unlawfully.

- **Function:** Detects red light violations.
- **Input:** Vehicle coordinates and real-time signal state.
- **Output:** Violation flag and time/frame of occurrence.
- **Role:** Determines whether a detected vehicle has violated traffic rules.

Number Plate Recognition Module

Once a vehicle is flagged during a red signal, this module performs Optical Character Recognition (OCR) using EasyOCR to extract the vehicle's number plate from the frame.

- **Function:** Extract and read alphanumeric characters from license plates.
- **Input:** Cropped vehicle image containing the number plate.
- **Output:** Recognized plate text.
- **Role:** Identifies violators for legal and administrative purposes.

Database Connectivity Module

This module is responsible for securely storing and retrieving data from the MySQL backend. It logs all violations along with the timestamp, number plate, video frame image, and signal state. It also manages information related to registered users and police authorities.

- **Function:** Insert and fetch violation data, user details, and vehicle info.
- **Database:** MySQL running locally via XAMPP.
- **Role:** Provides persistent storage for reporting and future reference.

Violation Reporting

This module generates a detailed violation report upon detection. The system displays a preview of detected vehicles with their plate number and signal status. Upon video completion, a notification message is shown to inform the user that the analysis is complete.

- **Function:** Display previews and notify end of detection.
- **Optional Features:** Extendable to send emails or SMS to vehicle owners (future scope).
- **Role:** Bridges detection and user feedback to support action-taking.

CHAPTER 6

IMPLEMENTATION

6.1 SOURCE CODE

```
#app.py
from flask import Flask, render_template, request, redirect, session, jsonify, send_from_directory
from flask_mysqldb import MySQL
import MySQLdb.cursors
import os
import cv2
import pytesseract
import re
import threading
import time

# Set Tesseract path
pytesseract.pytesseract.tesseract_cmd = r'D:\tesseract\tesseract.exe'

app = Flask(__name__)
app.secret_key = 'your_secret_key_here'

# MySQL configuration for XAMPP
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'traffic_db'

# Upload folder config
app.config['UPLOAD_FOLDER'] = 'uploads'
os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)

mysql = MySQL(app)

# Shared state
detection_state = {
    'signal': 'green',
    'video_processing': False,
    'video_finished': False,
    'current_video': '',
    'latest_plates': set()
}
```

```
@app.route('/')
def home():
    return redirect('/login')

@app.route('/login', methods=['GET', 'POST'])
def login():
    msg = ''
    if request.method == 'POST' and 'username' in request.form and 'password' in request.form:
        username = request.form['username']
        password = request.form['password']
        cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
        cursor.execute('SELECT * FROM users WHERE username = %s AND password = %s AND
role = %s',
                       (username, password, 'police'))
        account = cursor.fetchone()
        if account:
            session['loggedin'] = True
            session['id'] = account['id']
            session['username'] = account['username']
            session['role'] = account['role']
            return redirect('/police_dashboard')
        else:
            msg = 'Incorrect username/password for police!'
    return render_template('login.html', msg=msg)
@app.route('/logout')
def logout():
    session.clear()
    return redirect('/login')
@app.route('/police_dashboard')
def police_dashboard():
    if 'loggedin' in session and session.get('role') == 'police':
        return render_template('police_dashboard.html', username=session['username'])
    return redirect('/login')
@app.route('/upload_video', methods=['POST'])
def upload_video():
    if 'loggedin' not in session or session.get('role') != 'police':
        return jsonify({'success': False, 'error': 'Unauthorized'}), 401
    if 'video' not in request.files or request.files['video'].filename == '':
        return jsonify({'success': False, 'error': 'No video file provided'})
    video = request.files['video']
    filename = video.filename
    save_path = os.path.join(app.config['UPLOAD_FOLDER'], filename)
    video.save(save_path)
    detection_state.update({
        'current_video': filename,
        'video_processing': True,
        'video_finished': False,
        'latest_plates': set()
    })
    threading.Thread(target=process_video, args=(save_path,)).start()
    return jsonify({'success': True, 'filename': filename})
@app.route('/toggle_signal', methods=['POST'])
def toggle_signal():
    data = request.get_json()
    signal = data.get('signal')
    if signal in ['red', 'green']:
        detection_state['signal'] = signal
        return jsonify({'success': True, 'signal': signal})
    return jsonify({'success': False, 'error': 'Invalid signal'})
@app.route('/detection_status')
def detection_status():
    return jsonify({
        'video_processing': detection_state['video_processing'],
        'video_finished': detection_state['video_finished'],
        'latest_plates': list(detection_state['latest_plates']),
        'signal': detection_state['signal']
    })
def process_video(video_path):
    cap = cv2.VideoCapture(video_path)
    print("[INFO] Video processing started...")
    while True:
        ret, frame = cap.read()
        if not ret:
            break
        if detection_state['signal'] == 'red':
            gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
            gray = cv2.bilateralFilter(gray, 11, 17, 17)
            edged = cv2.Canny(gray, 30, 200)
            contours, _ = cv2.findContours(edged.copy(), cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
            contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10]
            for cnt in contours:
                approx = cv2.approxPolyDP(cnt, 0.018 * cv2.arcLength(cnt, True), True)
                if len(approx) == 4:
                    x, y, w, h = cv2.boundingRect(cnt)
                    plate_img = gray[y:y + h, x:x + w]
                    text = pytesseract.image_to_string(plate_img, config='--oem 3 --psm 6')
                    print(f"[DEBUG] OCR Plate Region Text: {text}")
                    plates = re.findall(r'[A-Z]{2}\d{1,2}[A-Z]{1,2}\d{4}', text.replace(" ", "").upper())
                    if plates:
                        try:
                            db = MySQLdb.connect(host="localhost", user="root", passwd="",
db="traffic_db")
                            cursor = db.cursor()
                            for plate in plates:
                                if plate not in detection_state['latest_plates']:
                                    cursor.execute("INSERT INTO violations (plate_number, video_name)
VALUES (%s, %s)",
                                                   (plate, detection_state['current_video']))
                                    db.commit()
                                    detection_state['latest_plates'].add(plate)
                                    print(f"[INFO] Detected and saved: {plate}")
                        except Exception as e:
                            print(f"[ERROR] DB error: {e}")
                        finally:
                            db.close()
                    break
        time.sleep(0.05)
    cap.release()
    detection_state['video_processing'] = False
    detection_state['video_finished'] = True
    print("[INFO] Video processing finished.")
@app.route('/search_violation')
def search_violation():
    plate_number = request.args.get('plate_number', '').strip().upper()
    if not plate_number:
        return redirect('/login')

    cursor = mysql.connection.cursor(MySQLdb.cursors.DictCursor)
    cursor.execute("SELECT plate_number, violation_time FROM violations WHERE
plate_number = %s ORDER BY violation_time DESC", (plate_number,))
    violations = cursor.fetchall()
    return render_template('violations_result.html', plate_number=plate_number,
violations=violations)
@app.route('/video/<filename>')
def serve_video(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)
if __name__ == '__main__':
    app.run(debug=True)
```

#login.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Traffic Management Login</title>
  <style>
    body, html {
      margin: 0; padding: 0; height: 100%; font-family: Arial, sans-serif;
    }
    .container {
      display: flex;
      height: 100vh;
    }
    .half {
      flex: 1;
      padding: 40px;
      box-sizing: border-box;
    }
    .left {
      background-color: #f0f0f0;
      border-right: 2px solid #ccc;
    }
    .right {
      background-color: #fff;
    }
    h2 {
      margin-bottom: 20px;
    }
    input[type=text], input[type=password], input[type=search] {
      width: 100%;
      padding: 10px;
      margin: 10px 0 20px 0;
      box-sizing: border-box;
      font-size: 16px;
    }
    button {
      padding: 10px 20px;
      font-size: 16px;
      cursor: pointer;
    }
    .msg {
      color: red;
      margin-top: 10px;
    }
  </style>
</head>
<body>
  <div class="container">
    <!-- Police Login -->
    <div class="half left">
      <h2>Police Login</h2>
      <form method="POST" action="/login">
        <input type="text" name="username" placeholder="Username" required>
        <input type="password" name="password" placeholder="Password" required>
        <button type="submit">Login</button>
      </form>
      {% if msg %}
        <div class="msg">{{ msg }}</div>
      {% endif %}
    </div>
    <!-- Citizen Search -->
    <div class="half right">
      <h2>Citizen Vehicle Violation Search</h2>
      <form method="GET" action="/search_violation">
        <input type="search" name="plate_number" placeholder="Enter Vehicle Number Plate"
required>
        <button type="submit">Search</button>
      </form>
    </div>
  </div>
</body>
</html>
```

#police_dashboard.html

```
<!DOCTYPE html>

<html>

<head>

  <title>Police Dashboard</title>

  <script src="https://code.jquery.com/jquery-
3.6.0.min.js"></script>

  <style>

    video {

      width: 600px;

      border: 2px solid #333;
```

```
            margin-bottom: 10px;
        }
        button {
            padding: 10px;
            margin: 5px;
            font-size: 16px;
            cursor: pointer;
        }
        #platesList {
            margin-top: 15px;
            font-weight: bold;
        }
        #status {
            margin-top: 10px;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1>Welcome, {{ username }}</h1>
    <form id="uploadForm"
enctype="multipart/form-data">
        <input type="file" name="video"
accept="video/*" required>
        <button type="submit">Upload and
Play</button>
    </form>
    <video id="videoPlayer" controls></video>
    <div>
        <button id="redBtn"
style="background:red; color:white;">Red
Signal</button>
        <button id="greenBtn"
```

```
style="background:green; color:white;">Green
Signal</button>
    </div>
    <div>
        <p>Current Signal: <span
id="signal">green</span></p>
        <p id="status">Idle</p>
        <p>Detected Plates:</p>
        <ul id="platesList"></ul>
    </div>
    <script>
        let detectionInterval = null;
        $('#uploadForm').submit(function (e) {
            e.preventDefault();
            const formData = new FormData(this);
            $.ajax({
                url: '/upload_video',
                type: 'POST',
                data: formData,
                contentType: false,
                processData: false,
                success: function (response) {
                    const videoName =
response.filename;
                    $('#videoPlayer').attr('src', '/video/'
+ videoName);
                    $('#videoPlayer')[0].play();
                    $('#status').text('Video playing...');
                    startDetectionLoop();
                },
                error: function () {
                    alert('Failed to upload video');
            }
          });
        });
        $('#redBtn').click(() =>
toggleSignal('red'));
        $('#greenBtn').click(() =>
toggleSignal('green'));
        function toggleSignal(signal) {
          $.ajax({
            url: '/toggle_signal',
            method: 'POST',
            contentType: 'application/json',
            data: JSON.stringify({ signal:
signal }),
            success: function (res) {
              if (res.success) {
                $('#signal').text(res.signal);
              }
            }
          });
        }
        function startDetectionLoop() {
          if (detectionInterval)
clearInterval(detectionInterval);
          detectionInterval = setInterval(() => {
            $.getJSON('/detection_status',
function (data) {
              $('#signal').text(data.signal);
              if (data.video_processing) {
                $('#status').text('Video
processing...');
              } else if (data.video_finished) {
                $('#status').text('Video
finished.');
                    clearInterval(detectionInterval);
                } else {
                    $('#status').text('Idle');
                }
                // Update detected plates list
                const platesUl = $('#platesList');
                platesUl.empty();
                data.latest_plates.forEach(plate =>
{                   platesUl.append(`<li>${plate}<
/li>`);
                });
            });
        }, 1500); // poll every 1.5 seconds
    }
    // Pause detection updates when video ends
    $('#videoPlayer').on('ended', function () {
        clearInterval(detectionInterval);
        $('#status').text('Video ended.');
    });
  </script>
</body>
</html>
```

#violations_result.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Violation Details for
{{ plate_number }}</title>
```

```
  <style>
      body { font-family: Arial, sans-serif;
padding: 40px; }
      h2 { margin-bottom: 20px; }
      table {
         border-collapse: collapse;
         width: 100%;
         max-width: 600px;
      }
      th, td {
         border: 1px solid #ddd;
         padding: 12px 15px;
         text-align: left;
      }
      th {
         background-color: #f5f5f5;
      }
      .no-violation {
         color: green;
         font-weight: bold;
      }
      a {
         display: inline-block;
         margin-top: 20px;
         text-decoration: none;
         color: #007bff;
```

```
        }
    </style>
</head>
<body>
    <h2>Violation Records for Vehicle:
{{ plate_number }}</h2>
    {% if violations and violations|length > 0 %}
        <table>
            <thead>
                <tr>
                    <th>Challan (Plate Number)</th>
                    <th>Date of Issuance</th>
                </tr>
            </thead>
            <tbody>
                {% for v in violations %}
                <tr>
                    <td>{{ v.plate_number }}</td>

<td>{{ v.violation_time.strftime('%Y-%m-
%d %H:%M:%S') }}</td>
                </tr>
                {% endfor %}
            </tbody>
        </table>
    {% else %}
        <p class="no-violation">No challans
```

```
issued for this vehicle.</p>
    {% endif %}
    <a href="/login">Back to Login/Search</a>
</body>
</html>
```

CHAPTER 7

RESULTS

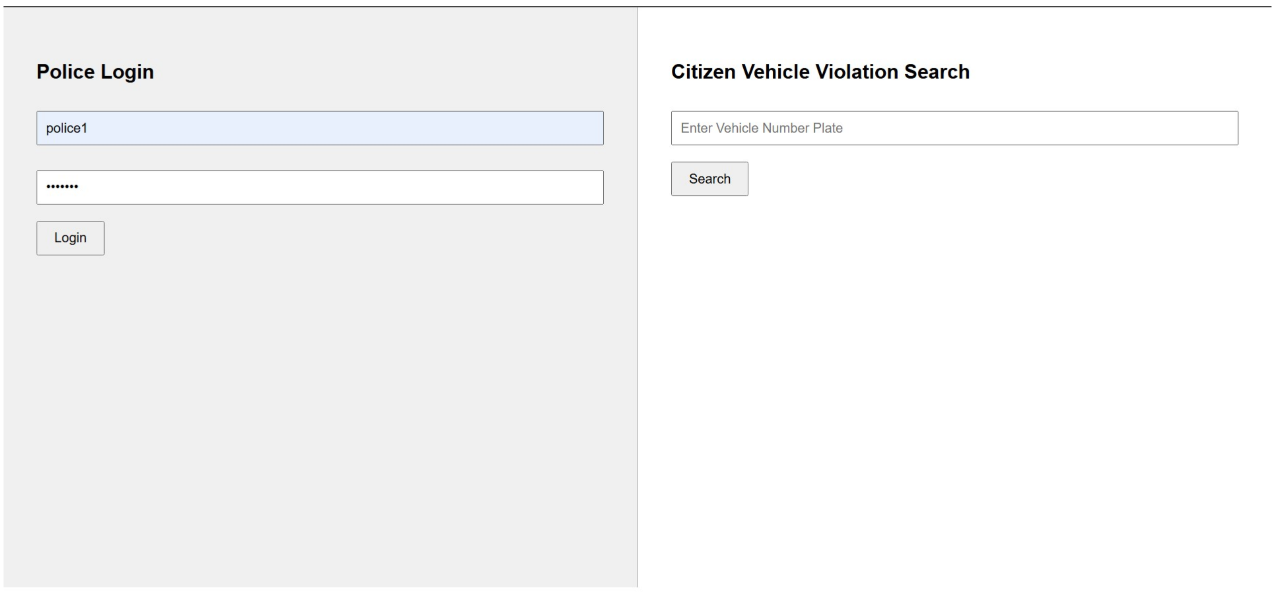
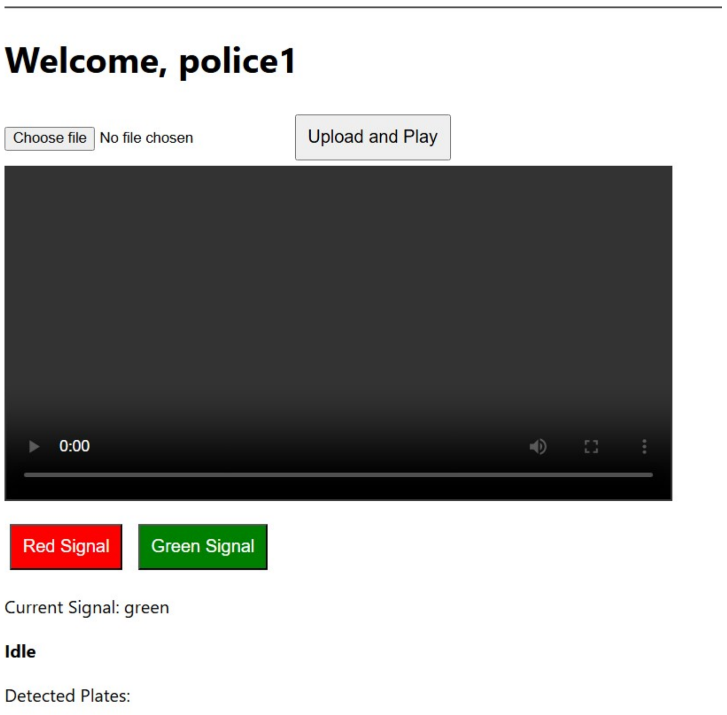


Figure 7.1 Login / Vehicle search page for police / citizen

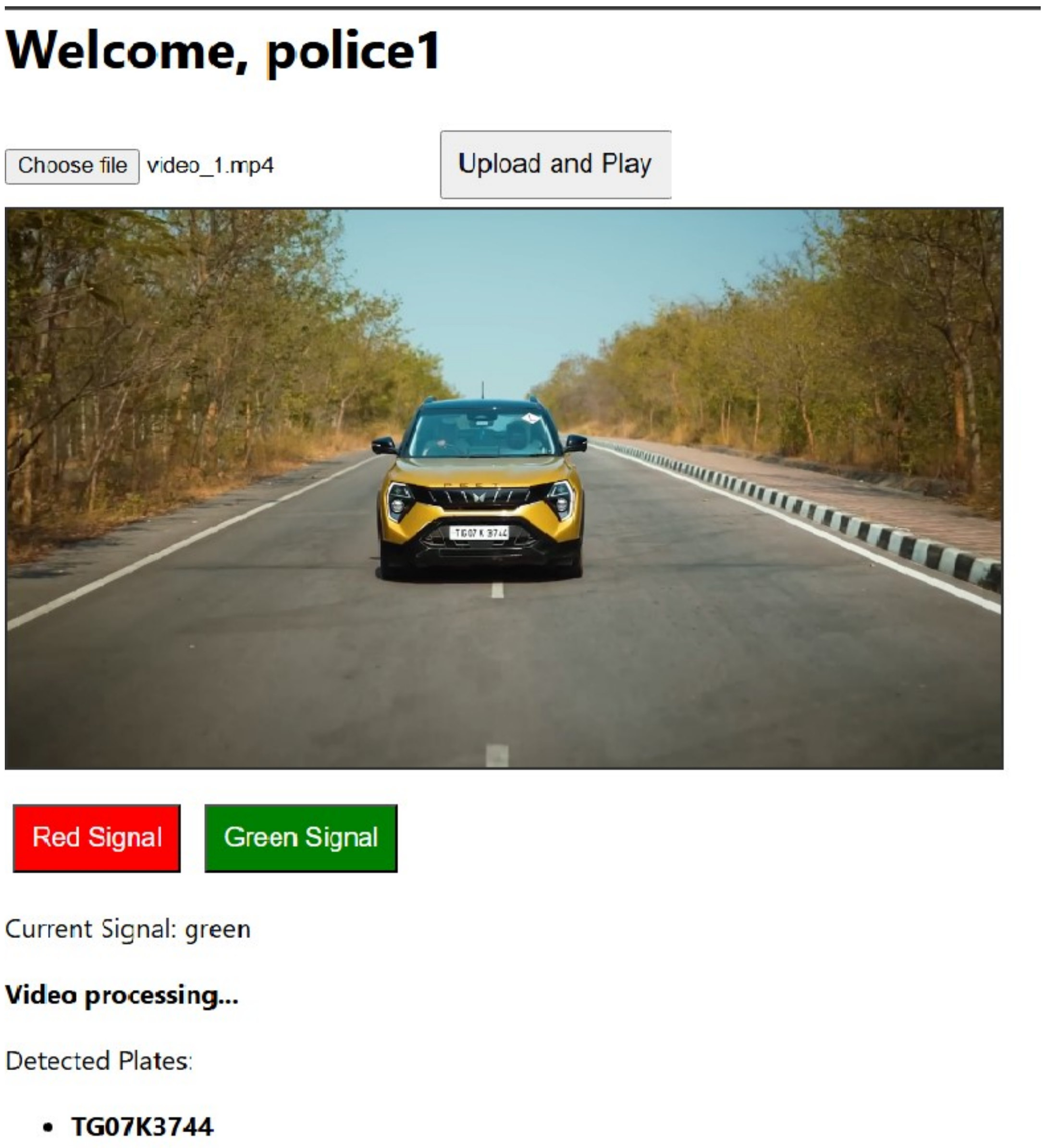


Current Signal: green

Idle

Detected Plates:

Figure 7.2 Police dashboard



Welcome, police1

video_1.mp4

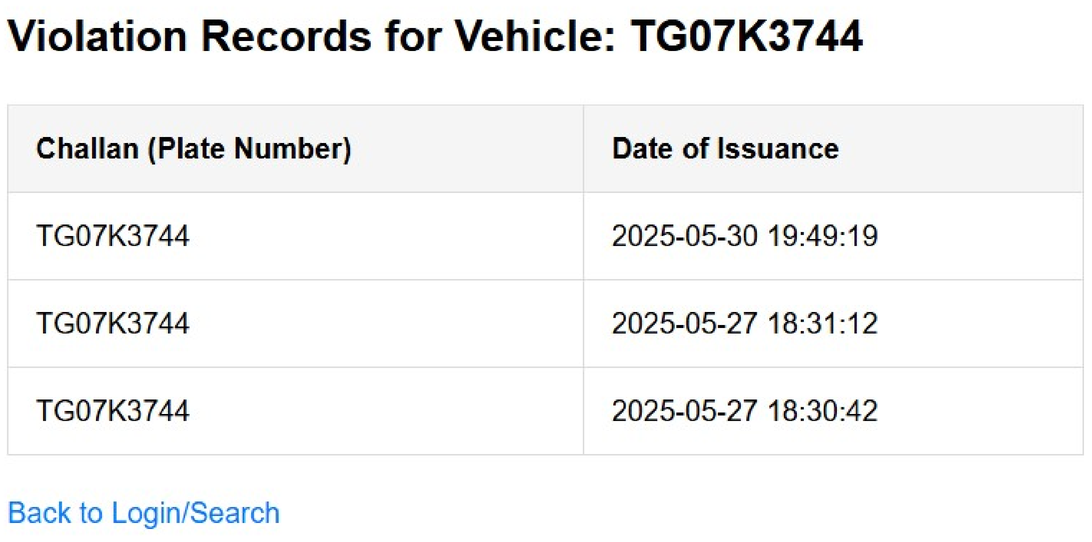
Current Signal: green

Video processing...

Detected Plates:

- **TG07K3744**

Figure 7.3 Processing a video in the police dashboard

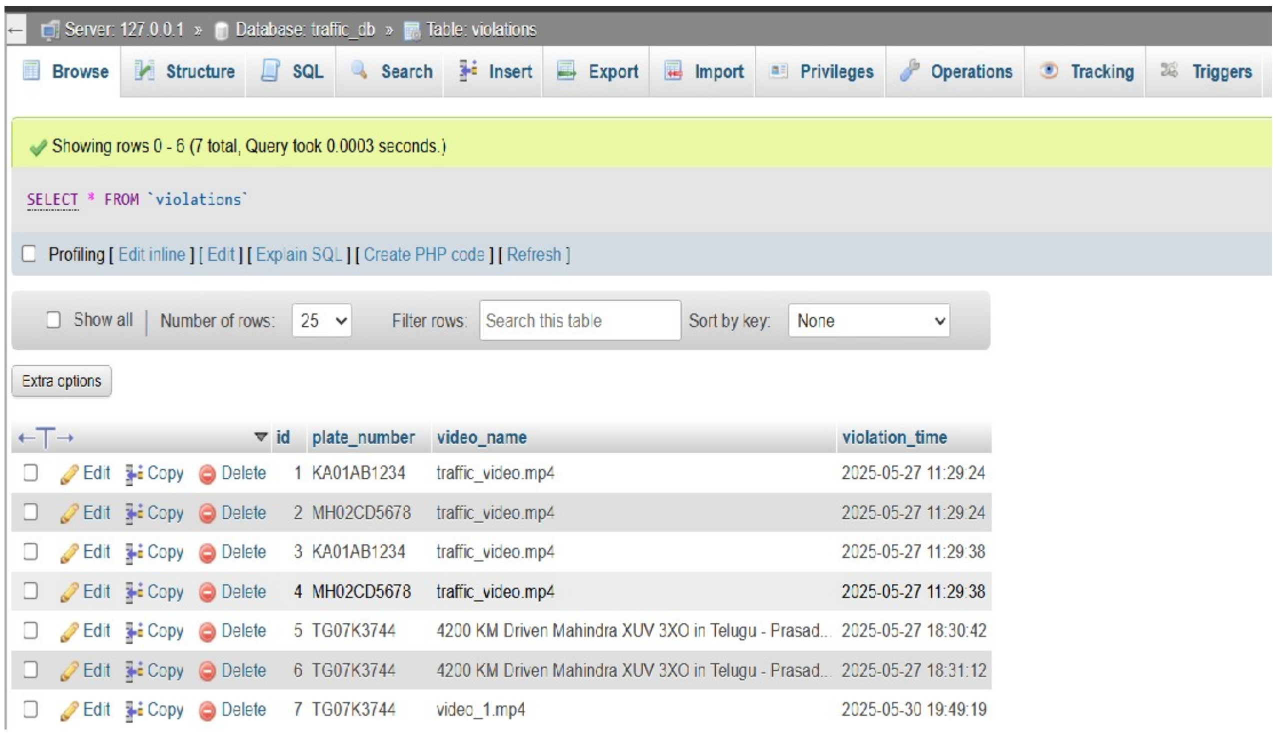


Violation Records for Vehicle: TG07K3744

Challan (Plate Number)	Date of Issuance
TG07K3744	2025-05-30 19:49:19
TG07K3744	2025-05-27 18:31:12
TG07K3744	2025-05-27 18:30:42

Back to Login/Search

Figure 7.4 Citizen vehicle search results / challan issuance



Server: 127.0.0.1 » Database: traffic_db » Table: violations
Browse
Structure
SQL
Search
Insert
Export
Import
Privileges
Operations
Tracking
Triggers

Showing rows 0 - 6 (7 total, Query took 0.0003 seconds.)

```
SELECT * FROM `violations`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]
Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
Extra options

	id	plate_number	video_name	violation_time
Edit Copy Delete	1	KA01AB1234	traffic_video.mp4	2025-05-27 11:29:24
Edit Copy Delete	2	MH02CD5678	traffic_video.mp4	2025-05-27 11:29:24
Edit Copy Delete	3	KA01AB1234	traffic_video.mp4	2025-05-27 11:29:38
Edit Copy Delete	4	MH02CD5678	traffic_video.mp4	2025-05-27 11:29:38
Edit Copy Delete	5	TG07K3744	4200 KM Driven Mahindra XUV 3XO in Telugu - Prasad...	2025-05-27 18:30:42
Edit Copy Delete	6	TG07K3744	4200 KM Driven Mahindra XUV 3XO in Telugu - Prasad...	2025-05-27 18:31:12
Edit Copy Delete	7	TG07K3744	video_1.mp4	2025-05-30 19:49:19

Figure 7.5 Database containing the registration numbers of vehicles that violated traffic rules

CHAPTER 8

CONCLUSION

The implementation of an AI-based Traffic Violation Detection System with Dynamic Signal Control and Number Plate Recognition offers an effective solution for modern urban traffic enforcement. By integrating YOLOv8 for vehicle detection and EasyOCR for number plate recognition, the system enables real-time identification of vehicles violating red signals. The use of a Flask web interface allows police to upload video footage and control signal states dynamically during video playback.

Detected violations are logged into a MySQL database with relevant metadata, including number plate details and timestamps. This automation significantly reduces manual effort, enhances enforcement accuracy, and ensures consistent monitoring of traffic rule violations.

The modular architecture and minimal hardware requirements make the system scalable, cost-effective, and deployable using existing surveillance infrastructure.

FUTURE ENHANCEMENTS AND DISCUSSIONS

- **Live Camera Integration**: Extend support to integrate real-time CCTV or traffic camera feeds for continuous violation monitoring.
- **Web Dashboard for Vehicle Owners**: Allow registered users to log in, view violation history, and download challan reports through a user-friendly web interface.
- **Helmet Detection and Wrong Lane Detection**: Implement additional modules for recognizing riders without helmets and detecting lane rule violations.
- **Speed Violation Detection**: Use motion tracking to detect overspeeding based on vehicle movement across frames.
- **Export Reports to Excel or PDF**: Add functionality to export violation logs into downloadable reports for legal or administrative use.
- **SMS and Email Notifications**: Automatically notify registered vehicle owners about traffic violations using integrated communication APIs.

REFERENCES

YOLOv8 by Ultralytics

Official Documentation: https://docs.ultralytics.com

GitHub Repository: https://github.com/ultralytics/ultralytics

Used for real-time vehicle detection using deep learning.

EasyOCR

Official Documentation: https://www.jaided.ai/easyocr/

GitHub Repository: https://github.com/JaidedAI/EasyOCR

Used for extracting license plate numbers from vehicle images.

Tesseract OCR

Installation Guide and Documentation: https://github.com/tesseract-ocr/tesseract

Used as the OCR backend for EasyOCR (path configured to D:\tesseract\tesseract.exe in your system).

OpenCV

Documentation: https://docs.opencv.org/

Python library used for video frame processing and image manipulation.

NumPy

Documentation: https://numpy.org/doc/

Used for numerical operations on image arrays.

Flask

Official Documentation: https://flask.palletsprojects.com/

Used to create the web interface for police officers to upload video and control traffic signals.

MySQL Database with XAMPP

XAMPP Documentation: https://www.apachefriends.org/index.html

MySQL Reference: https://dev.mysql.com/doc/

Used for storing vehicle numbers, timestamps, violation types and metadata.

Project Inspiration and Related Work

- AI-Based Traffic Management System – IEEE Project Docs
  https://ieeexplore.ieee.org (Search: "AI-based traffic violation detection")

- Number Plate Recognition using OCR and YOLO – GitHub Example
  https://github.com/hamzafaisal/ANPR

- Real-Time Red Light Violation Detection – Python Project Guide
  https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial