



COMSATS University Islamabad (CUI) Attock Campus

Traffic Flow Control In Urban Cities

By:

Fakhar Ul Hassan CIIT/SP21-BCS-004/ATK

Supervisor:

Ms. Fouzia Jabeen

Bachelors of Science in Computer Science (2021-2025)

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



COMSATS University Islamabad (CUI) Attock Campus

Traffic Flow Control In Urban Cities

**A project presented to
COMSATS University, Islamabad**

**In partial fulfillment
of the requirement for the degree of**

Bachelors of Science in Computer Science (2021-2025)

**By:
Fakhar Ul Hassan CIIT/SP21-BCS-004/ATK**

DECLARATION

We hereby declare that this software, neither whole nor as a part has been copied out from any source. It is further declared that we have developed this software and accompanied report entirely on the basis of our personal efforts. If any part of this project is proved to be copied out from any source or found to be reproduction of some other, we will stand by the consequences. No Portion of the work presented has been submitted of any application for any other degree or qualification of this or any other university or institute of learning.

Fakhar Ul Hassan

CERTIFICATE OF APPROVAL

It is to certify that the final year project of **BS(CS) “Traffic Flow Control In Urban Cities”** was developed by **Fakhar Ul Hassan CIIT/SP21-BCS-004/ATK** and under the supervision of Ms. Fouzia Jabeen and that in his opinion; it is fully adequate, in scope and quality for the degree of Bachelors of Science in Computer Sciences.

Supervisor

External Examiner

Head of Department
Department of Computer Science

EXECUTIVE SUMMARY

Urban cities face increasing traffic congestion, leading to delays, fuel wastage, and environmental concerns. Our project, "Traffic Flow Control in Urban Cities," aims to provide a technology-driven solution to optimize traffic signal management using advanced computer vision techniques.

Leveraging YOLOv5, a state-of-the-art object detection model, the system identifies and counts vehicles in real time through traffic surveillance cameras. The vehicle count data determines the duration of green lights for each road segment, dynamically prioritizing roads with higher traffic volumes. This ensures smoother traffic flow and reduces congestion.

To maintain fairness and prevent extended delays on less congested roads, the system incorporates a mechanism to open all roads for 40 seconds after a predefined interval. This feature ensures residual traffic on minor roads is also cleared, promoting balanced traffic distribution. The implementation of this project enhances the efficiency of existing traffic management systems, reduces average wait times for commuters, and offers an environmentally sustainable solution by minimizing idle vehicle times and fuel consumption.

ACKNOWLEDGMENT

All praise is to Almighty Allah who bestowed upon us a minute portion of His boundless knowledge by virtue of which we were able to accomplish this challenging task.

We are greatly indebted to our project **Ms. Fouzia Jabeen** . Without his personal supervision, advice and valuable guidance, completion of this project would have been doubtful. We are grateful to him for his encouragement and continual help during this work.

And we are also thankful to our parents and family who have been a constant source of encouragement for us and brought us with the values of honesty & hard work.

Fakhar Ul Hassan

List of Abbreviations

FR Functional Requirements

PC Personal Computer

SDD Software Design Description

SRS Software Requirement Specification

Software Requirement Specification (SRS)

Personal Computer (PC)

Software Design Description (SDD)

Functional Requirements (FR)

Table of Contents

Declaration	i
Certificate Of Approval	ii
Executive Summary	iii
Acknowledgment	iv
List of Abbreviations	iv
Table of Contents	vi
List of Figures	ix
List of Tables	x
Chapter One: Introduction	2
1.1 Problem Statement	2
1.2 Objectives	2
1.3 System Limitations/Constraints	3
1.4 Project Deliverables	3
1.5 Relevance to Course Modules	3
Chapter Two: Related Work	6
2.1 Traffic Dataset and Its Analysis	6
2.1.1 Model Training:	6
2.1.2 Performance Evaluation:	6
2.1.3 Traffic Flow Analysis:	6
2.1.4 Use of YOLO in Traffic Management:	7
2.1.5 Web Application:	7
2.1.6 Traditional Traffic Light Systems	7
2.1.7 Adaptive Traffic Control Systems (ATCS)	7
2.1.8 Computer Vision-Based Traffic Management:	7
2.2 Approaches for Analyzing Image Data	8
2.2.1 Computer Vision (YOLOv5):	8
2.2.2 Deep Learning (CNNs):	8
2.2.3 Data Processing and Prediction:	8
2.2.4 Justification for the Proposed Model	8
Chapter Three: Requirement Analysis	10
3.1 User classes and characteristics	10
3.1.1 Use Case Diagram	10

3.2	Requirement Identifying Technique	13
3.2.1	Event Response Table	13
3.3	Functional Requirements	15
3.3.1	Functional Requirement 1: Traffic Monitoring and Control Setup	15
3.3.2	Functional Requirement 2: Car Detection and Counting	16
3.3.3	Functional Requirement 3: Dynamic Traffic Light Timing	16
3.3.4	Functional Requirement 4: Congestion Clearing Interval	17
3.3.5	Functional Requirement 5: System Error Handling	17
3.4	Non-Functional Requirements (NFRs):	18
3.4.1	Reliability	18
3.4.2	Usability	18
3.4.3	Performance	19
3.4.4	Security	19
3.5	External Interface Requirements	19
3.5.1	User Interfaces Requirements	20
3.5.2	Software Interfaces	23
3.5.3	Hardware Interfaces	24
3.5.4	Communications Interfaces	25
	Chapter Four: Design and Architecture	27
4.1	Modules	27
4.1.1	Module 1: Vehicle Detection and Traffic Monitoring	27
4.1.2	Module 2: Traffic Light Adjustment and Control	27
4.1.3	Module 3: Traffic Incident Detection and Alerts	28
4.1.4	Module 4: User Interface and Traffic Monitoring Dashboard	28
4.1.5	Module 5: Data Analytics and Reporting	28
4.1.6	Module 6: Notifications and Reminders	28
4.2	Architectural Design	28
4.3	System Architecture	28
4.4	Design Models	31
4.5	Data Design	37
4.6	Human Interface Design	39
4.6.1	Screen Images:	39
4.6.2	Home Screen Objects and Actions	47
4.6.3	Workout Video Upload Screen Objects and Actions	47
4.6.4	Custom Workout Plan Screen Objects and Actions	48
4.6.5	Diet Plan Screen Objects and Actions	48

4.6.6	Past Results Screen Objects and Actions	48
Chapter Five: Implementation		50
5.1	Algorithm	50
5.2	Tools and Technologies	51
5.3	External APIs/SDKs	52
5.4	User Interface	53
5.5	Deployment	61
5.5.1	Hardware Deployment	62
5.5.2	Backend Deployment	62
5.5.3	Model Deployment	62
5.5.4	Monitoring and Maintenance	62
5.5.5	Scalability and Future Updates	63
Chapter Six: Testing and Evaluation		65
6.1	Unit Testing	65
6.2	Functional Testing	65
6.3	Business Rules Testing	66
6.4	Integration Testing	66
Chapter Seven: Conclusion and Future Work		69
7.1	Conclusion	69
7.2	Future Work	69
References		70
References		70

List of Figures

Figure	Description	Page
Figure 3.1	Use Case diagram	12
Figure 4.1	Architecture diagram	30
Figure 4.2	Activity diagram	32
Figure 4.3	Class diagram	33
Figure 4.4	Sequence diagram	34
Figure 4.5	State Transition diagram	35
Figure 4.6	DFD diagram	36
Figure 4.7	Class diagram	38
Figure 4.8	Welcome Screen	39
Figure 4.9	About Screen	40
Figure 4.10	Upload Video Screen	41
Figure 4.11	Choose Video Screen	42
Figure 4.12	Uploaded Video Screen	43
Figure 4.13	Apply Mask Screen	44
Figure 4.14	Mask Result Screen	45
Figure 4.15	Video Result Screen	46
Figure 4.16	Asked Question Screen	47
Figure 5.1	Welcome Screen	53
Figure 5.2	About Screen	54
Figure 5.3	Upload Video Screen	55
Figure 5.4	Choose Video Screen	56
Figure 5.5	Uploaded Video Screen	57
Figure 5.6	Apply Mask Screen	58
Figure 5.7	Mask Result Screen	59
Figure 5.8	Video Result Screen	60
Figure 5.9	Asked Question Screen	61

List of Tables

Table	Description	Page
Table 2.1	Related System Analysis with Proposed Project Solution	8
Table 3.1	Event-Response Table for Traffic Flow Control In Urban Cities	13
Table 3.2	Event-Response Table for Traffic WebApp	14
Table 3.3	Description of FR-1 - Traffic Monitoring and Control Setup	15
Table 3.4	Description of FR-2 - Car Detection and Counting	16
Table 3.5	Description of FR-3 - Dynamic Traffic Light Timing	16
Table 3.6	Description of FR-4 - Congestion Clearing Interval	17
Table 3.7	Description of FR-5 - System Error Handling	18
Table 3.8	Description of SI-1 - Traffic Flow Control in Urban Cities App Software In- terface	23
Table 3.9	Description of HI-1 - Hardware Interface for Traffic Camera Feed Capture . .	24
Table 3.10	Description of CI-1 - Traffic Alert and Notification System	25
Table 5.1	Tools and Technologies for Traffic Flow Control in Urban Cities	51
Table 5.2	External APIs and SDKs for Traffic Flow Control in Urban Cities	52

Chapter One

Introduction

Our project, "Traffic Flow Control in Urban Cities," addresses these challenges by integrating advanced computer vision techniques with traffic signal systems. Using YOLOv5, a state-of-the-art object detection [1] model, the system captures real-time traffic data through surveillance cameras to count vehicles on each road segment. This data enables dynamic adjustment of signal timings, prioritizing roads with higher traffic volumes. Additionally, the system incorporates a fairness mechanism, opening all roads for a predefined duration after specific intervals to clear residual traffic on less busy roads. This ensures a balance between traffic flow efficiency and equity across all road users.

1.1 Problem Statement

Cities are facing a big problem with traffic jams, which cause long waiting times, more fuel usage, and pollution. Most traffic lights work on fixed timers, which don't change based on how many cars are on the road. This means busy roads stay crowded, while roads with fewer cars still get the same amount of green light time.

Because of this, people waste time and fuel, and cars stuck in traffic create more air pollution. There is a need for a smart traffic system that can count cars, give more time to busy roads, and clear traffic on all roads fairly. This will make traveling faster, reduce pollution, and save energy.

1.2 Objectives

The main objectives of the project "Traffic Flow Control in Urban Cities" are:

Improve Traffic Efficiency To reduce traffic congestion by dynamically adjusting signal timings based on real-time vehicle counts.

Prioritize High-Traffic Roads To give more signal time to roads with higher traffic density, ensuring smoother flow on busier routes.

Ensure Fairness To clear traffic on less busy roads by opening all signals for a fixed duration after specific intervals.

Reduce Environmental Impact To minimize vehicle idling time, thereby reducing fuel consumption and air pollution.

Leverage Technology To use YOLOv5 for accurate and real-time vehicle detection and counting.

1.3 System Limitations/Constraints

While **Traffic Flow Control in Urban Cities** aims to provide an accessible and efficient solution for detecting cars, there are some limitations and constraints:

Camera Dependency: Requires high-quality cameras for accurate vehicle detection

Weather Impact: Rain, fog, or poor lighting can affect detection accuracy.

Power and Internet: Needs reliable power and internet for real-time processing.

Limited Small Object Detection: May struggle with identifying small vehicles like bikes in crowded scenes.

Fixed Interval Fairness: May delay the cars which are in rush. Because the system don't know the the rushed person.

1.4 Project Deliverables

The **Traffic Flow Control in Urban Cities** project will deliver a fully functional web application with the following key features:

- **Real-Time Traffic Detection System:** This is the system that uses cameras and the YOLOv5 model to count cars on the roads in real time.
- **Traffic Signal Control [2] Algorithm:** A smart algorithm that adjusts the traffic lights based on how many cars are on each road, giving more time to busier roads.
- **User Interface:** A simple control panel or dashboard that allows users to see how the system is working and make adjustments if needed.
- **Test Results:** Data showing how the system reduces traffic.

1.5 Relevance to Course Modules

The development of **Traffic Flow Control in Urban Cities** is closely tied to several course modules studied during the Bachelor of Computer Science (BCS) program. Key concepts and skills from the following courses have been applied:

- **Computer Vision:** The project uses YOLOv5, a machine learning model, for vehicle detection, which ties to the computer vision module of the course.
- **Artificial Intelligence:** The system uses AI algorithms to adjust traffic signal timings based on real-time data.

- **Machine Learning** The project involves handling real-time data (vehicle counts) and processing it to make decisions.
- **Software Engineering:** The project follows the software development life cycle (SDLC), covering requirements analysis, design, development, and testing, all of which are covered in software engineering courses.
- **Data Structures and Algorithms:** The traffic signal control algorithm optimizes traffic flow, which relates to the algorithm design and optimization modules.

Chapter Two

Related Work

Intelligent Traffic Signal Systems: Many cities have adopted smart traffic signal systems that adjust signal timings based on real-time traffic data, like sensors or cameras. However, these systems often use traditional algorithms, which may not be as efficient as AI-based approaches. **Vehicle Detection using Computer Vision:** Previous studies have used computer vision for vehicle detection in traffic management. For example, systems that use cameras and machine learning models to count cars and detect congestion have been implemented in several smart city projects. **Adaptive Traffic Control Systems (ATCS):** Some cities have experimented with adaptive control systems that change traffic light patterns based on traffic conditions. These systems usually rely on loop detectors or radar sensors, but they can be limited in handling large or unpredictable traffic flows.

2.1 Traffic Dataset and Its Analysis

For our Final Year Project (FYP), we used **Vehicle dataset** available on Kaggle, Data Preprocessing:

Cleaning: Remove any incomplete or irrelevant data points. **Labeling:** For object detection, images need to be labeled with bounding boxes around vehicles. This dataset often comes with pre-labeled data or can be manually annotated using tools like LabelImg. **Augmentation:** To increase the dataset size and improve model accuracy, data augmentation techniques (like rotation, scaling, and flipping) can be applied.

2.1.1 Model Training:

YOLOv5 is trained using the preprocessed dataset to detect and count vehicles in each image or video frame. The model's performance is evaluated based on its accuracy (how correctly it detects vehicles) and processing speed (how fast it can analyze each frame).

2.1.2 Performance Evaluation:

Precision and Recall: Measures the accuracy of vehicle detection (precision) and the model's ability to detect all vehicles (recall). **mAP (mean Average Precision):** This metric helps evaluate the model's overall performance in detecting vehicles.

2.1.3 Traffic Flow Analysis:

The data can be analyzed to determine peak traffic times, congestion levels, and the effectiveness of adaptive traffic signal changes. By comparing traffic flow before and after implementing the system, we can measure the improvements in traffic management.

2.1.4 Use of YOLO in Traffic Management:

YOLO (You Only Look Once) has been successfully used for real-time object detection, including vehicle counting in traffic management. It's proven to be faster and more accurate compared to traditional methods like image classification or feature extraction.

2.1.5 Web Application:

Several web applications exist that assist with our project, but none are specifically focused on the detection of cars. Below is an analysis of similar systems, their limitations, and how **Eye-Doctor-Pro** addresses these weaknesses:

2.1.6 Traditional Traffic Light Systems

Weakness: These systems can lead to congestion on busy roads while underutilizing less crowded ones, causing delays and inefficiencies. They also often fail to adapt to sudden changes in traffic, such as accidents or events.

2.1.7 Adaptive Traffic Control Systems (ATCS)

Weakness: These systems may struggle in areas with high variability or where traffic data is not accurate. They also rely on sensors that might be expensive or difficult to maintain.

2.1.8 Computer Vision-Based Traffic Management:

Weakness: The quality of the system heavily depends on camera quality, lighting conditions, and the model's accuracy. In busy or complex intersections, there may be challenges in vehicle detection.

Table 2.1: Related System Analysis with Proposed Project Solution

Application Name	Limitations	Improvement
Traditional Traffic Light Systems	Fixed timers, inefficient for varying traffic conditions.	Dynamic signal control based on actual traffic volume.
Adaptive Traffic Control Systems	Limited accuracy with sensor data; expensive to maintain.	Cost-effective and more accurate system, adaptable to changing conditions.
Computer Vision-Based Systems	Depends on camera quality and lighting conditions; may struggle in complex intersections.	Improved vehicle detection accuracy and speed, even in difficult environments.

2.2 Approaches for Analyzing Image Data

Here are some common approaches for analyzing image data:

2.2.1 Computer Vision (YOLOv5):

YOLOv5 is used for real-time vehicle detection and counting in traffic images. This helps the system identify the number of cars on different roads and adjust traffic signals accordingly.

2.2.2 Deep Learning (CNNs):

Convolutional Neural Networks (CNNs) are used for extracting features from traffic images and detecting objects (vehicles). YOLOv5, which is a type of CNN, helps in object detection and classification.

2.2.3 Data Processing and Prediction:

The model processes vehicle count data to predict which roads need more green light time, improving efficiency and reducing congestion.

2.2.4 Justification for the Proposed Model

Given the limitations of existing systems and approaches, the Traffic web app employs a CNN model to provide an automated, accurate, and user-friendly solution for conjunctivitis detection. The model was chosen due to its proven success in image classification tasks, ensuring robustness and precision in disease detection.

Chapter Three

Requirement Analysis

The Requirement Analysis for the project focuses on ensuring the necessary hardware, software, and data components are available for efficient implementation. The hardware requirements include high-resolution cameras for real-time vehicle detection and high-performance computers to run the YOLOv5 model for processing. Additionally, traffic signal controllers are needed to interface with the system and adjust signal timings based on vehicle count. On the software side, the project requires the YOLOv5 model for object detection, along with programming languages such as Python for model implementation and C++ for hardware interfacing. Libraries like OpenCV [3] for image processing and deep learning frameworks such as TensorFlow or PyTorch will be used. A database like MySQL or MongoDB is needed to store traffic data and vehicle counts. The system also requires a labeled vehicle dataset for model training, along with real-time traffic flow [4] data for decision-making.

3.1 User classes and characteristics

The Traffic web app is designed to cater to various user classes, each with distinct needs and characteristics:

- **Traffic Control Operators:** These users monitor traffic flow and control signals in real-time. They need to have access to the system dashboard for adjusting traffic light timings if required. Ability to manually override or adjust signal timings in case of unusual situations.
- **City Traffic Management Authorities:** These users oversee the implementation and effectiveness of the traffic control system across the city. They focus on the long-term performance of the system. Access to detailed reports and analytics on traffic flow and system performance.
- **System Administrators:** These users are responsible for setting up, maintaining, and troubleshooting the system. They ensure smooth operation and perform necessary updates. Monitor system performance and ensure it operates without errors.

3.1.1 Use Case Diagram

Actors Traffic Control Operators: Monitor real-time traffic data. Adjust traffic light timings based on vehicle counts. Use manual override in emergencies. City Traffic Management Authorities: Oversee the system's overall performance. Access detailed reports on traffic flow and congestion. System Administrators: Maintain and troubleshoot the system. Perform updates

and ensure the system runs smoothly. End Users (Commuters): Experience improved traffic flow due to dynamic signal adjustments.

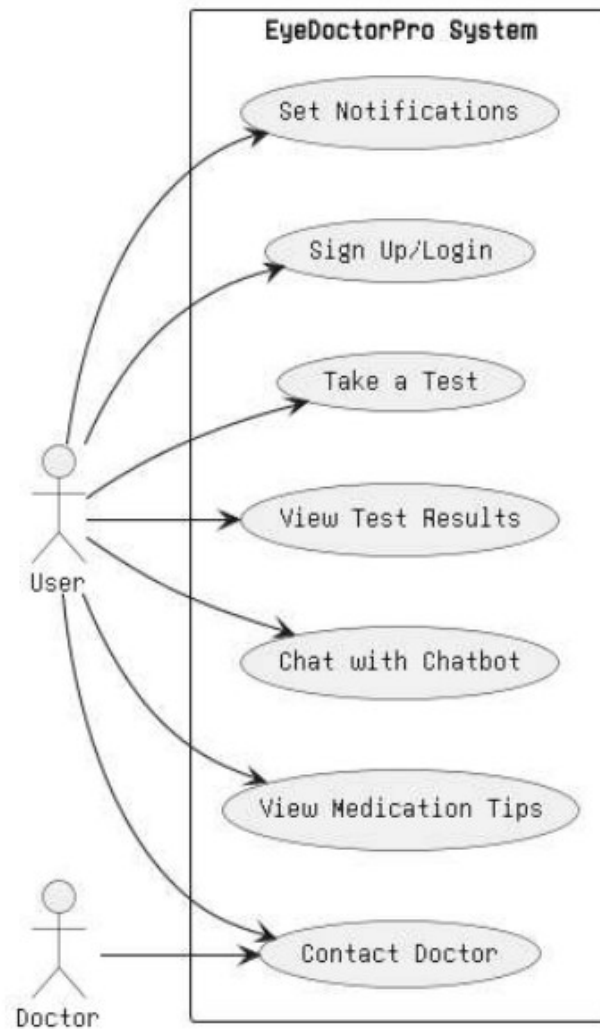


Figure 3.1: Use Case diagram

3.2 Requirement Identifying Technique

Conduct interviews with traffic management authorities and operators to understand their needs and expectations from the system.

3.2.1 Event Response Table

This Event-Response Table captures the key interactions and responses between the traffic monitoring system and the environment, ensuring that the system efficiently manages traffic flow based on real-time car counts. It outlines how the system reacts to different traffic conditions, such as prioritizing roads with more cars, dynamically adjusting traffic light timings, and clearing congestion by periodically opening all roads for a brief period. The table helps define how the system handles various events, ensuring that it operates optimally to reduce traffic jams and improve flow in urban cities.

Table 3.1: Event-Response Table for Traffic Flow Control In Urban Cities

Event	Trigger	Response	System Component
Camera detects cars	Camera captures real-time video feed	YOLOv5 model detects and counts the number of cars on each road, sending the count data to the traffic control system.	Car Detection
Traffic light timing decision	System analyzes car count data	Traffic light timings are adjusted dynamically, prioritizing roads with higher car counts and optimizing flow.	Traffic Flow Control
Interval check	Timer reaches set interval	Traffic lights are opened for all roads for 40 seconds to clear any congestion on less crowded roads.	Traffic Control Logic

Event	Trigger	Response	System Component
System malfunctions	System detects an error in car counting or light control	Error message displayed, and system attempts to auto-correct or notifies the operator of the malfunction.	Error Handling

Table 3.2: Event-Response Table for Traffic WebApp

Event	Trigger	Response	System Component
Test result displayed	System completes image analysis	System displays diagnosis results (e.g., Conjunctivitis detected or not), with follow-up recommendations.	Diagnosis
User views past results	User selects "Past Results" option	System displays a list of past test results, including dates and diagnoses.	Test History
User accesses Medication screen	User selects the "Medication" screen option	System displays tips on disease prevention and doctor contact information.	Medication Screen
User receives a notification	System triggers a scheduled event or reminder	System sends a reminder for retesting or provides health tips based on the user's test history.	Notifications

3.3 Functional Requirements

The Functional Requirements (FRs) section outlines the specific functionalities that the Traffic Flow Control system must perform to ensure efficient management of urban traffic. These requirements are designed to provide a responsive and optimized experience for drivers and pedestrians by adjusting traffic light timings based on real-time car counts. Each feature is described in detail, specifying the core functionality, expected behavior, and dependencies on other components such as car detection, traffic light control, and congestion clearing intervals. The system must dynamically prioritize roads with higher car counts, periodically open all roads to clear congestion, and ensure smooth traffic flow across the city.

3.3.1 Functional Requirement 1: Traffic Monitoring and Control Setup

This requirement focuses on setting up the environment for efficient traffic flow management, ensuring that real-time data regarding traffic conditions is securely collected, processed, and used to adjust traffic light timings. It ensures that the system is ready to prioritize roads with higher car counts and manage congestion effectively.

Table 3.3: Description of FR-1 - Traffic Monitoring and Control Setup

Identifier	FR-1
Title	Traffic Monitoring and Control Setup
Requirement	The system will be set up to collect real-time data from traffic cameras to monitor car counts on different roads. This data will be securely processed and used to adjust traffic light timings based on the number of cars, ensuring that roads with higher car counts are prioritized. The system should also handle congestion clearing by periodically opening all roads for a fixed time interval.
Source	Traffic Monitoring System
Rationale	Efficient traffic management
Business Rule (if required)	No
Dependencies	Traffic camera feeds, car detection model (YOLOv5), traffic control system
Priority	High

3.3.2 Functional Requirement 2: Car Detection and Counting

This requirement outlines the system's ability to detect and count cars using the YOLOv5 model from the video feed captured by traffic cameras, ensuring that the data is accurate and reliable.

Table 3.4: Description of FR-2 - Car Detection and Counting

Identifier	FR-2
Title	Car Detection and Counting
Requirement	The system will use the YOLOv5 model to process real-time video feeds from traffic cameras, detecting and counting the number of cars on each road segment. This data will be sent to the traffic control system to adjust traffic light timings.
Source	Camera Feed, YOLOv5 Model
Rationale	Accurate car detection and counting are crucial for optimizing traffic light control.
Business Rule (if required)	No
Dependencies	Traffic camera feeds, YOLOv5 model
Priority	High

3.3.3 Functional Requirement 3: Dynamic Traffic Light Timing

This requirement focuses on adjusting the timing of traffic lights dynamically based on real-time data from car counts. Roads with higher traffic will be given more green light time, while roads with fewer cars will have shorter green light durations.

Table 3.5: Description of FR-3 - Dynamic Traffic Light Timing

Identifier	FR-3
Title	Dynamic Traffic Light Timing
Requirement	The system will adjust traffic light timings dynamically based on the real-time car count data from the traffic camera feeds. Roads with more cars will be given longer green light durations, and roads with fewer cars will have shorter green light durations.
Source	Traffic Monitoring System

Identifier	FR-3
Rationale	Dynamic traffic light control will optimize traffic flow and reduce congestion.
Business Rule (if required)	No
Dependencies	Traffic camera feeds, car count data, traffic control system
Priority	High

3.3.4 Functional Requirement 4: Congestion Clearing Interval

This requirement ensures that after a set time interval, the system will open all roads for a fixed period to clear congestion from less crowded roads.

Table 3.6: Description of FR-4 - Congestion Clearing Interval

Identifier	FR-4
Title	Congestion Clearing Interval
Requirement	The system will trigger a congestion clearing interval every fixed period, opening all roads for 40 seconds to allow cars to move across less crowded roads and reduce overall congestion.
Source	Traffic Control Logic
Rationale	Periodically clearing congestion helps maintain an even flow of traffic across all roads.
Business Rule (if required)	No
Dependencies	Traffic light control system, timing logic
Priority	Medium

3.3.5 Functional Requirement 5: System Error Handling

This requirement specifies how the system should handle errors in detecting cars, controlling traffic lights, or any system malfunctions, ensuring that appropriate actions are taken to maintain traffic flow.

Table 3.7: Description of FR-5 - System Error Handling

Identifier	FR-5
Title	System Error Handling
Requirement	The system will detect errors in car detection, traffic light control, or any component malfunctions. In case of failure, the system will notify the operators and attempt to auto-correct or initiate a fallback process to prevent traffic congestion.
Source	Error Detection System
Rationale	Error handling ensures the system maintains traffic flow even during technical issues.
Business Rule (if required)	No
Dependencies	Car detection system, traffic control system, error detection mechanism
Priority	High

3.4 Non-Functional Requirements (NFRs):

The Non-Functional Requirements (NFRs) for Traffic Flow Control In Urban Cities are following:

3.4.1 Reliability

The system should maintain an uptime of 99.9% per month to ensure high availability and responsiveness, especially during peak traffic hours. It should handle system errors gracefully and provide real-time status updates for users and traffic operators to ensure the system is functioning correctly. Any traffic management failures should be logged and addressed promptly with automatic recovery mechanisms to minimize disruption.

3.4.2 Usability

The user interface for traffic operators should be intuitive and easy to navigate, even for individuals with minimal technical expertise. The system should provide real-time traffic data and control dashboards with clear visual indicators, such as road congestion levels and active traffic light timings. The system should be designed to support easy integration with other urban traffic

management platforms. Furthermore, it should include clear alerts for system malfunctions or errors to guide the operators in resolving issues.

3.4.3 Performance

The system must be capable of processing and analyzing real-time video feeds from traffic cameras to detect and count cars in under 2 seconds per frame. It should support the analysis and management of traffic data for at least 100 roads simultaneously. The system should also handle large spikes in traffic data (e.g., during rush hours) without a noticeable performance drop, ensuring timely updates and traffic light adjustments.

3.4.4 Security

Traffic data, including real-time car counts and traffic camera feeds, must be encrypted (e.g., AES-256) to ensure the privacy and security of the data. The system should implement secure authentication for traffic operators and administrators, with multi-factor authentication (MFA) for sensitive operations such as adjusting traffic light timings. Compliance with local and international data protection regulations (such as GDPR and CCPA) should be ensured to maintain legal adherence and protect the privacy of individuals.

3.5 External Interface Requirements

The Traffic Flow Control in Urban Cities system integrates with user interfaces, external hardware, and software for efficient traffic management. The User Interface provides traffic operators with real-time data on car counts, traffic light timings, and congestion levels. The system communicates with traffic cameras via RTSP to capture video feeds, using YOLOv5 for car detection. It adjusts traffic light timings dynamically using protocols like MQTT or REST API. Data processing is handled through cloud-based or on-premise servers, and the system integrates with existing traffic management platforms for optimized control. Robust error handling includes alerts and logs for hardware issues, ensuring smooth operation and interoperability with other systems via standard protocols and data formats.

3.5.1 User Interfaces Requirements

This section describes the logical characteristics of the user interface that the system needs to provide. The following aspects are considered:

- **Standards for Fonts, Icons, and Colors:**

- **Fonts:**

- * Headers: Bold, larger font size (e.g., 18-22pt) to emphasize key data, like car count and traffic light status.
 - * Body Text: Regular, medium font size (e.g., 14-16pt) for clear communication of real-time traffic information.

- **Icons:**

- * Use clear, simple icons to represent system functions (e.g., road icons, car icons, stoplights).
 - * Icons should be intuitive, representing roadways, traffic signals, and monitoring systems (e.g., live camera feed, traffic data).

- **Color Scheme:**

- * Primary color: Light blue (CFE7F5) for the interface background to create a calm, professional feel.
 - * Accent color: Deep blue (2070A0) for action buttons and highlighted traffic signals.
 - * Neutral color: White (FFFFFF) for background to enhance readability of data.
 - * Error color: Red (FF5C5C) to indicate issues like traffic congestion or malfunctioning cameras.

- **Screen Layout or Resolution Constraints:**

- The design should be responsive to adjust seamlessly to various screen sizes (smartphones, tablets, and desktop monitors).
 - The system should have a vertical layout for mobile devices, with a clean design for ease of navigation and viewing traffic data.
 - Target resolution: 1080x1920 pixels (or scalable with a 16:9 ratio).

- **Standard Buttons, Functions, and Navigation Links:**

- Navigation bar: The bottom bar should provide easy access to essential sections like:

- * Dashboard (Traffic Overview), Camera Feed, Traffic Analytics, Settings.
- Every screen should have:
 - * A "Back" button to return to the previous screen for quick navigation.
 - * A "Help" or "Info" button, accessible via the settings icon.
- Buttons should maintain consistency:
 - * Font size: 16pt, bold for action clarity.
 - * Rounded corners with color feedback when pressed for a responsive interface.
- **Shortcut Keys:**
 - Implement gesture shortcuts:
 - * Swipe left/right to toggle between different traffic zones or road data views.
 - * Long press on traffic data statistics to refresh or update the information.
 - Optional keyboard shortcuts for devices with physical keyboards:
 - * Enter: Confirm actions (e.g., confirm traffic light adjustments).
 - * Tab: Navigate between settings or dashboard panels.
 - * Esc: Close or cancel active system alerts.
- **Message Display Conventions:**
 - Success messages (e.g., "Traffic light timings updated") should be shown in green and placed at the bottom of the screen.
 - Error messages (e.g., "Camera feed failure") should appear in red near the issue or alert section.
 - Toast notifications (e.g., "System restart in 5 minutes") should appear at the top of the screen for 3-5 seconds.
- **Layout Standards for Localization:**
 - All text should be stored in external language files to support multi-language interfaces (e.g., for international deployment).
 - The system should dynamically resize UI elements to accommodate longer text strings for languages like German or Arabic.
 - Full support for right-to-left (RTL) languages to ensure proper alignment (e.g., Arabic or Hebrew).

- **Accommodations for Visually Impaired Users:**

- Text Size Adjustment: Provide an option to adjust font size to meet user preferences.
- Contrast Mode: High-contrast dark mode to enhance visibility for visually impaired users.
- Screen Reader Support: Ensure descriptive alt text is provided for all buttons, icons, and traffic data visuals.
- Touch Targets: Ensure that all interactive elements are large enough (minimum 48x48dp) for easy tapping.

- **Other Design Considerations:**

- Tabbing Sequence: Controls like settings and data input fields should follow a logical order, top to bottom, left to right.
- Common Controls:
 - * Dropdown menus for selecting roads or setting traffic priorities.
 - * Toggle switches for enabling automatic traffic light adjustments.
- Dialog Box Layouts:
 - * Centered dialog boxes for system confirmations (e.g., "Restart Traffic System") with clearly distinct "Cancel" and "Confirm" buttons.

3.5.2 Software Interfaces

Table 3.8: Description of SI-1 - Traffic Flow Control in Urban Cities App Software Interface

Identifier	SI-1
Title	Traffic Flow Control in Urban Cities App
Requirement	The app will interface with computer vision libraries and APIs to analyze traffic data and optimize traffic flow using real-time video feeds. Specifically, it will integrate with the YOLOv5 model to detect vehicles and adjust traffic light timings accordingly.
Source	App Development Team
Rationale	The integration of real-time video analysis improves traffic management efficiency and reduces congestion, enhancing the overall flow of vehicles in urban environments.
Business Rule (if required)	No
Dependencies	YOLOv5 (latest version), OpenCV (Version 4.5), TensorFlow [5] (Version 2.10), Flask [6] (Version 2.1), Firebase (Version 9.6)
Priority	High

3.5.3 Hardware Interfaces

The Traffic Flow Control in Urban Cities app interfaces with the smartphone hardware to capture video feeds, store data, and enable functionalities related to traffic analysis and optimization:

Table 3.9: Description of HI-1 - Hardware Interface for Traffic Camera Feed Capture

Identifier	HI-1
Title	Hardware Interface for Traffic Camera Feed Capture
Requirement	The app will utilize the smartphone or connected device's camera (or external traffic cameras) to capture real-time video feeds of traffic on various roads. The camera resolution and features should support at least 1080p resolution to ensure clear and accurate vehicle detection for traffic analysis.
Source	App Development Team
Rationale	Leveraging the smartphone's built-in camera or connected external cameras enables real-time monitoring of traffic conditions, helping to optimize traffic light timings and improve traffic flow in urban areas.
Business Rule (if required)	No
Dependencies	Smartphone camera hardware, External traffic cameras (if applicable), Operating System (Android or iOS), Internet connection for cloud integration
Priority	High

3.5.4 Communications Interfaces

The Traffic Flow Control in Urban Cities app requires several communication functions to facilitate real-time traffic updates, system alerts, and synchronization with traffic control systems:

Table 3.10: Description of CI-1 - Traffic Alert and Notification System

Identifier	CI-1
Title	Traffic Alert and Notification System
Requirement	The app shall send push notifications to notify users about traffic updates, including traffic light changes, road congestion, system errors, and emergency alerts. Additionally, the app will send notifications to traffic control centers to update traffic light timings based on real-time conditions.
Source	App Development Team
Rationale	Push notifications and alerts keep users informed about traffic conditions in real-time, helping them plan routes effectively. Additionally, the app's integration with traffic control centers ensures timely adjustments for optimizing traffic flow.
Business Rule (if required)	No
Dependencies	Push Notification Service, Traffic Control Center Integration, Internet Connection, User Settings for Notification Preferences
Priority	High

Chapter Four

Design and Architecture

The Traffic Flow Control in Urban Cities system is designed with a client-server architecture, consisting of several interconnected components to optimize urban traffic management. The system uses real-time data from traffic monitoring cameras equipped with YOLOv5 object detection to count vehicles at intersections, adjusting traffic light timings based on vehicle volume. The backend processes this data to prioritize roads with higher traffic, ensuring smooth traffic flow. A mobile application serves as the user interface, providing city operators with real-time updates, alerts, and system reports, while users can view traffic conditions to plan their routes. Push notifications, email, and SMS alert both traffic operators and users about system performance, malfunctions, or traffic incidents. All traffic data is stored in a centralized database for analysis, and APIs facilitate communication between the mobile app, traffic control center, and backend servers. The entire system is supported by a secure communication network that ensures real-time data transmission for efficient traffic management.

4.1 Modules

This section outlines the core functional modules of the **Traffic Flow Control in Urban Cities** system, each designed to optimize urban traffic management. The system is divided into several key modules that work together to ensure real-time traffic monitoring, data processing, and decision-making. Each module provides essential features to enhance traffic flow, improve road safety, and reduce congestion.

4.1.1 Module 1: Vehicle Detection and Traffic Monitoring

FE1 The system uses camera feeds to capture real-time traffic data at intersections.

FE2 YOLOv5 is used to detect vehicles and count their numbers at different traffic zones.

FE3 Data collected from cameras is processed and sent to the backend system for traffic light adjustments.

4.1.2 Module 2: Traffic Light Adjustment and Control

FE1 The system adjusts traffic light timings based on the number of vehicles detected.

FE2 Roads with higher vehicle counts receive longer green light durations, while less congested roads are given shorter durations.

FE3 The system automatically re-adjusts the timing at regular intervals to clear congestion and balance traffic flow.

4.1.3 Module 3: Traffic Incident Detection and Alerts

FE1 The system detects traffic incidents such as accidents or congestion based on vehicle patterns and speed analysis.

FE2 Alerts are sent to traffic control operators for immediate attention and necessary actions.

FE3 The system also provides alerts to users via the mobile app, informing them of traffic incidents or disruptions.

4.1.4 Module 4: User Interface and Traffic Monitoring Dashboard

FE1 The mobile app provides users with real-time traffic information, including congestion levels, estimated travel times, and route suggestions.

FE2 A dashboard for traffic operators displays vehicle counts, light status, and incident reports.

FE3 Users and operators can manually adjust traffic light timings or prioritize certain roads if needed.

4.1.5 Module 5: Data Analytics and Reporting

FE1 The system collects historical traffic data for analysis and reporting.

FE2 Traffic patterns, incident frequencies, and road efficiency are analyzed to identify trends and areas for improvement.

FE3 Reports are generated regularly for city planners and traffic management authorities.

4.1.6 Module 6: Notifications and Reminders

FE1 Users receive notifications regarding significant traffic incidents, detours, or accidents affecting their route.

FE2 Traffic operators receive automated alerts if a road experiences significant congestion or if traffic light malfunctions are detected.

4.2 Architectural Design

4.3 System Architecture

The system architecture for **EyeDoctorPro** follows a modular design that integrates key components and defines their relationships to ensure the seamless functionality of the app. The architecture encompasses multiple modules, such as the user interface, image processing, backend services, and communication systems, to ensure efficient handling of user inputs, image analysis, and notifications. Each component is designed to interact with others through well-defined

APIs and communication protocols, ensuring that the system remains cohesive, scalable, and maintainable.

The **EyeDoctorPro** app architecture consists of several key components:

- **Image Upload and Pink Eye Detection:** Allows users to upload eye images and analyzes them using a machine learning model (Convolutional Neural Network) for pink eye detection.
- **Chatbot Assistance:** Offers real-time conversational support to users regarding symptoms, prevention tips, and disease management.
- **Disease Prevention Tips:** Provides personalized health tips and advice for users diagnosed with conjunctivitis.
- **Past Results Management:** Enables users to view and track their past eye health results.
- **Notifications and Reminders:** Sends push notifications and email alerts to users, reminding them to retake tests or follow up with their eye care.

The high-level architecture diagram below illustrates these main subsystems and how they interact within the overall system. This diagram showcases how each module is connected, ensuring that the system is responsive and efficient in processing data from multiple sources.

- Initially, a Box and Line Diagram is used for a simplified representation of the system's core interactions.
- After selecting an architecture style (e.g., MVC, Client-Server, or Microservices), detailed mappings of components to their respective architectural layers are created. This includes mapping the image processing, database management, and notification services to appropriate system layers for scalability and ease of maintenance.

Architecture Diagram:

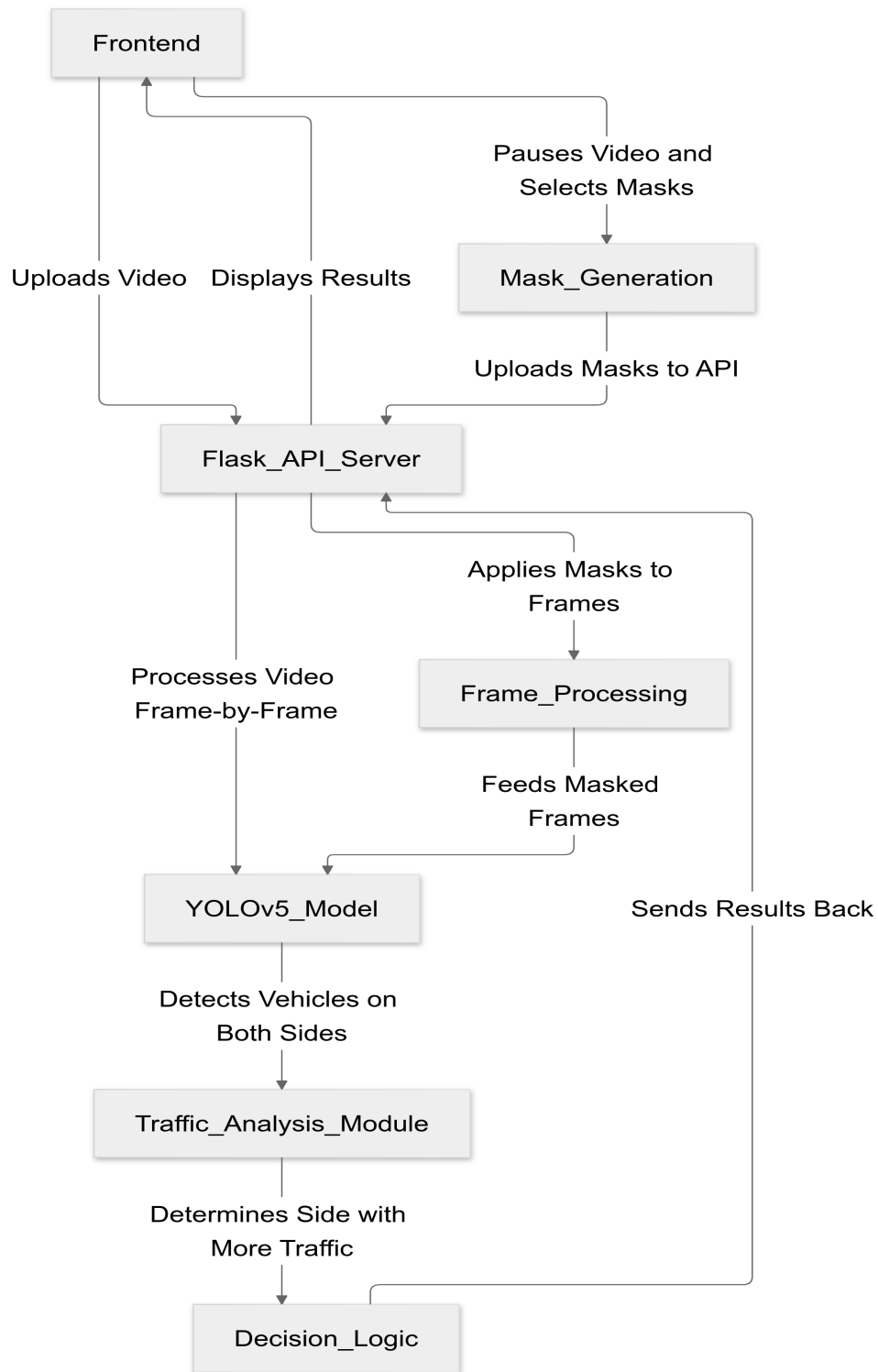


Figure 4.1: Architecture diagram

4.4 Design Models

The design models for the Traffic Flow Control in Urban Cities system outline key components and interactions, focusing on vehicle detection, traffic light control, and real-time processing. The system uses cameras and AI models, including a CNN for vehicle detection and YOLOv5 for real-time tracking. Traffic light timings are adjusted dynamically based on traffic density, prioritizing busy roads. Additional modules support system monitoring and user notifications, ensuring optimal traffic management and improved urban mobility.

Design Models for Object-Oriented Development Approach

In the Traffic Flow Control in Urban Cities project, the object-oriented design models encompass three primary views. The Class Model defines classes for components such as traffic cameras, traffic lights, and traffic zones, encapsulating their attributes and behaviors. The Interaction Model depicts the interactions between traffic cameras, AI models, and traffic light controllers to efficiently manage traffic flow. Lastly, the State Model describes the various states of traffic lights and the transitions between these states based on real-time vehicle density data and system inputs, ensuring optimal traffic control.

- **Activity Diagram:**

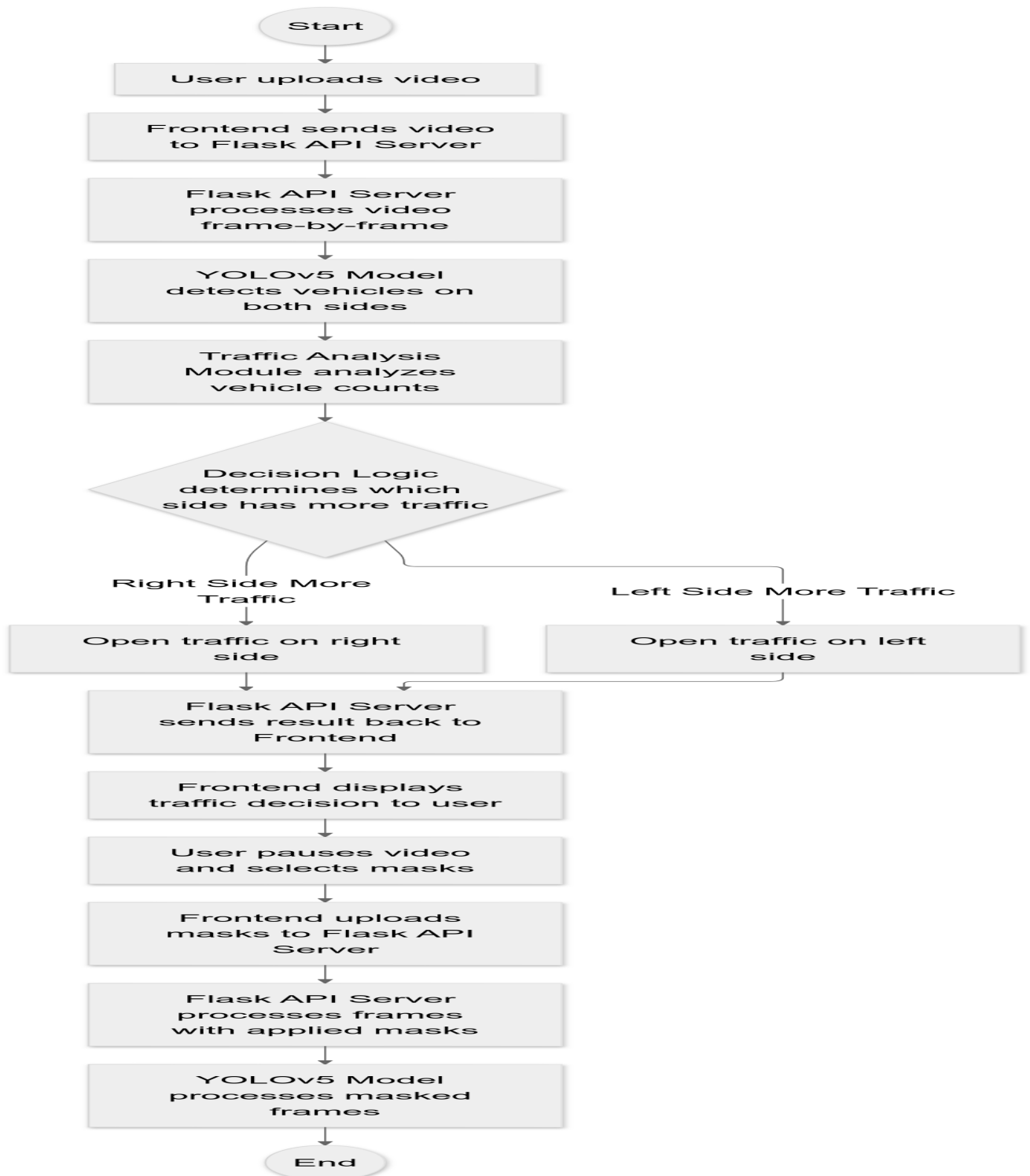


Figure 4.2: Activity diagram

- **Class Diagram:**

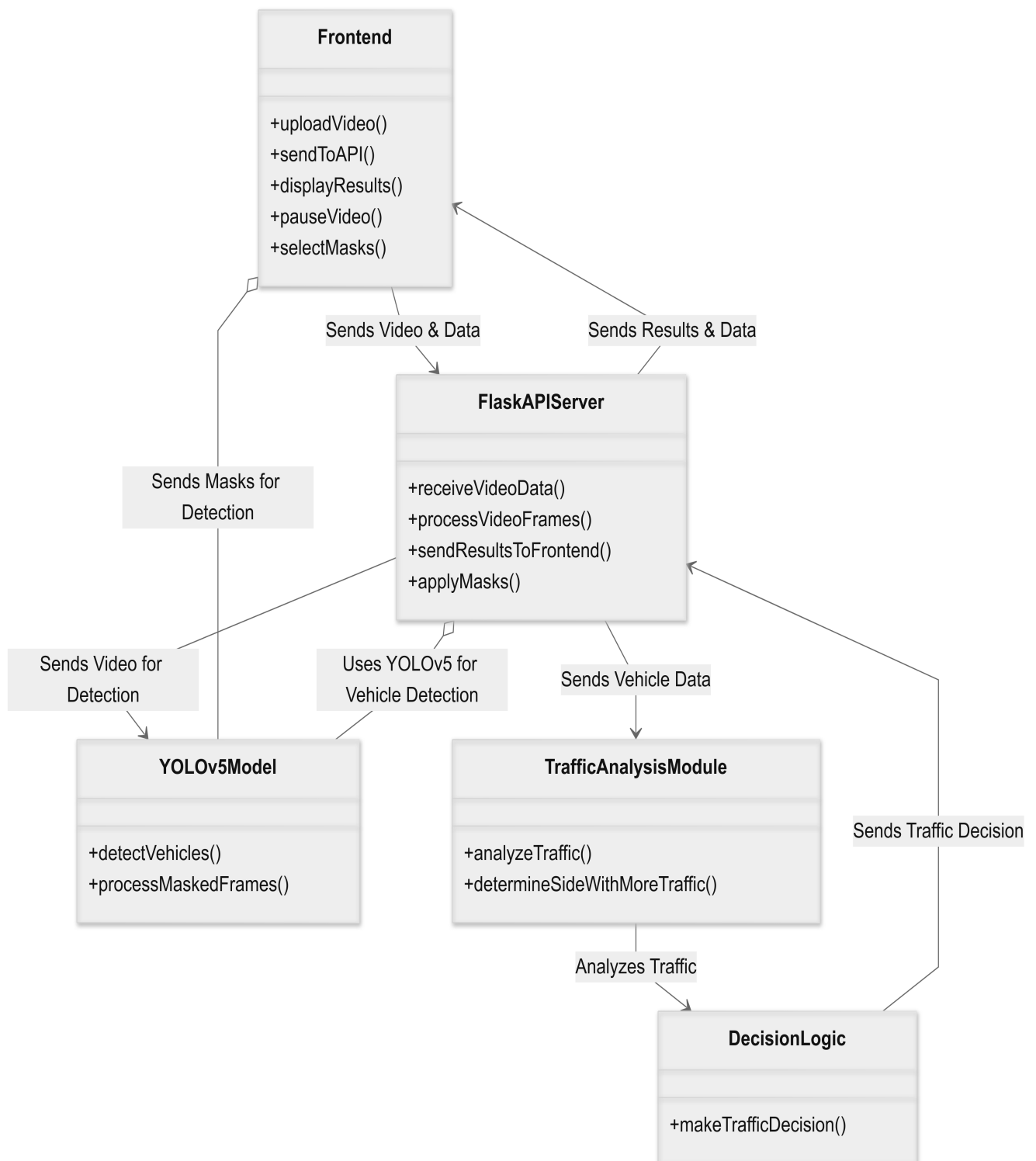


Figure 4.3: Class diagram

• **Sequence Diagram:**

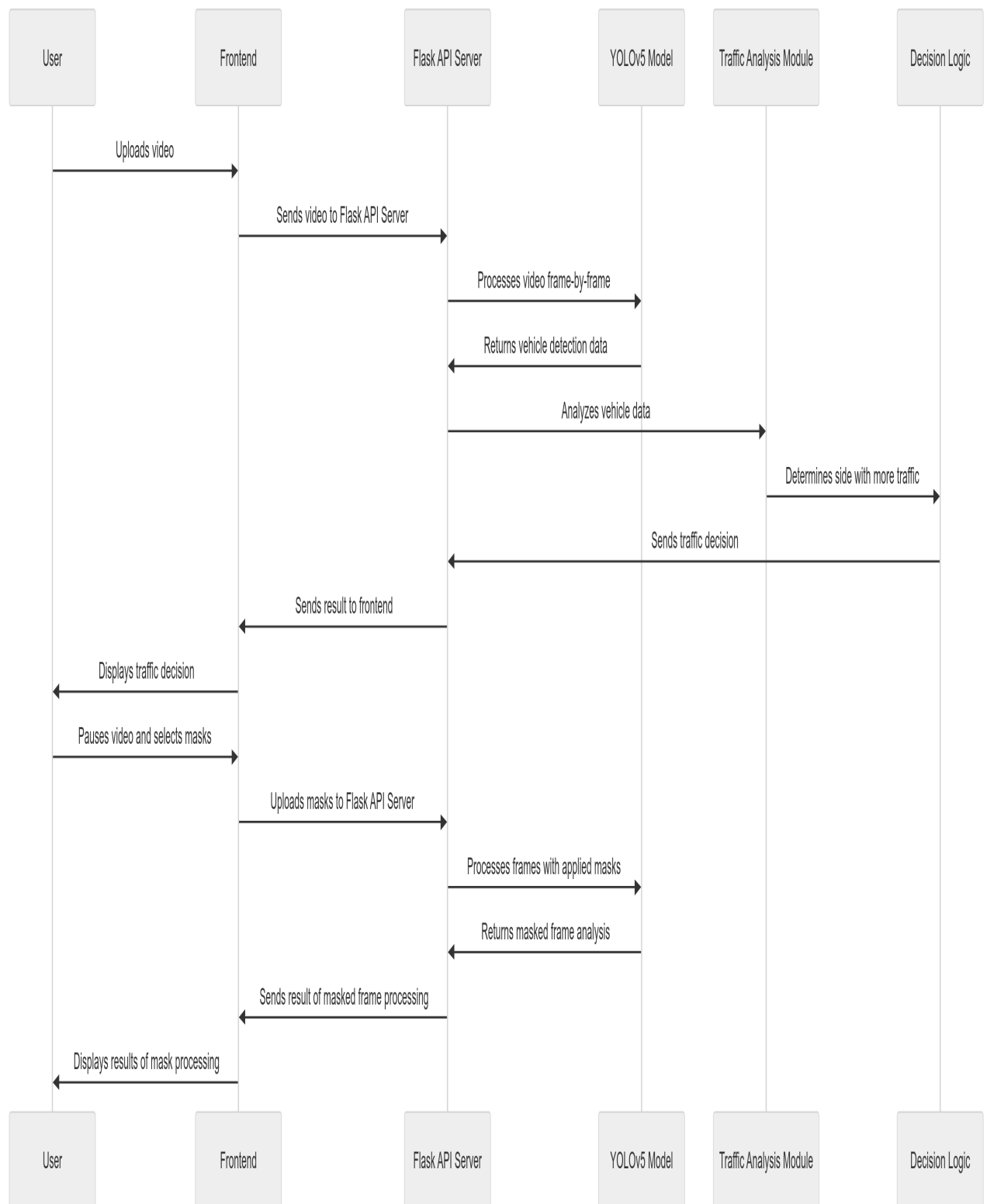


Figure 4.4: Sequence diagram

- **State Transition Diagram:**

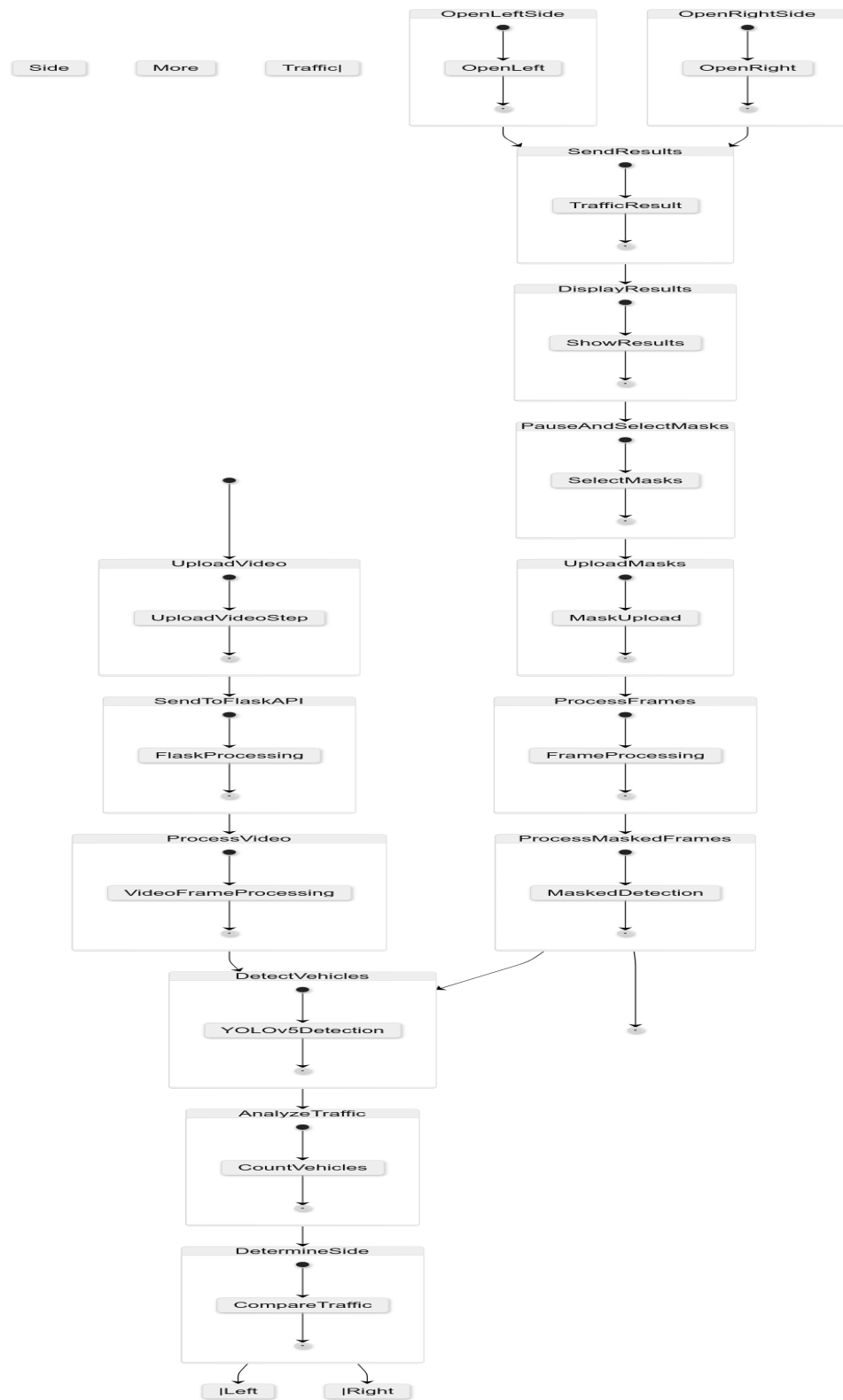


Figure 4.5: State Transition diagram

Design Models for Procedural Approach

For systems following a procedural development approach, the following design models are applicable:

- **Data Flow Diagram (DFD):**

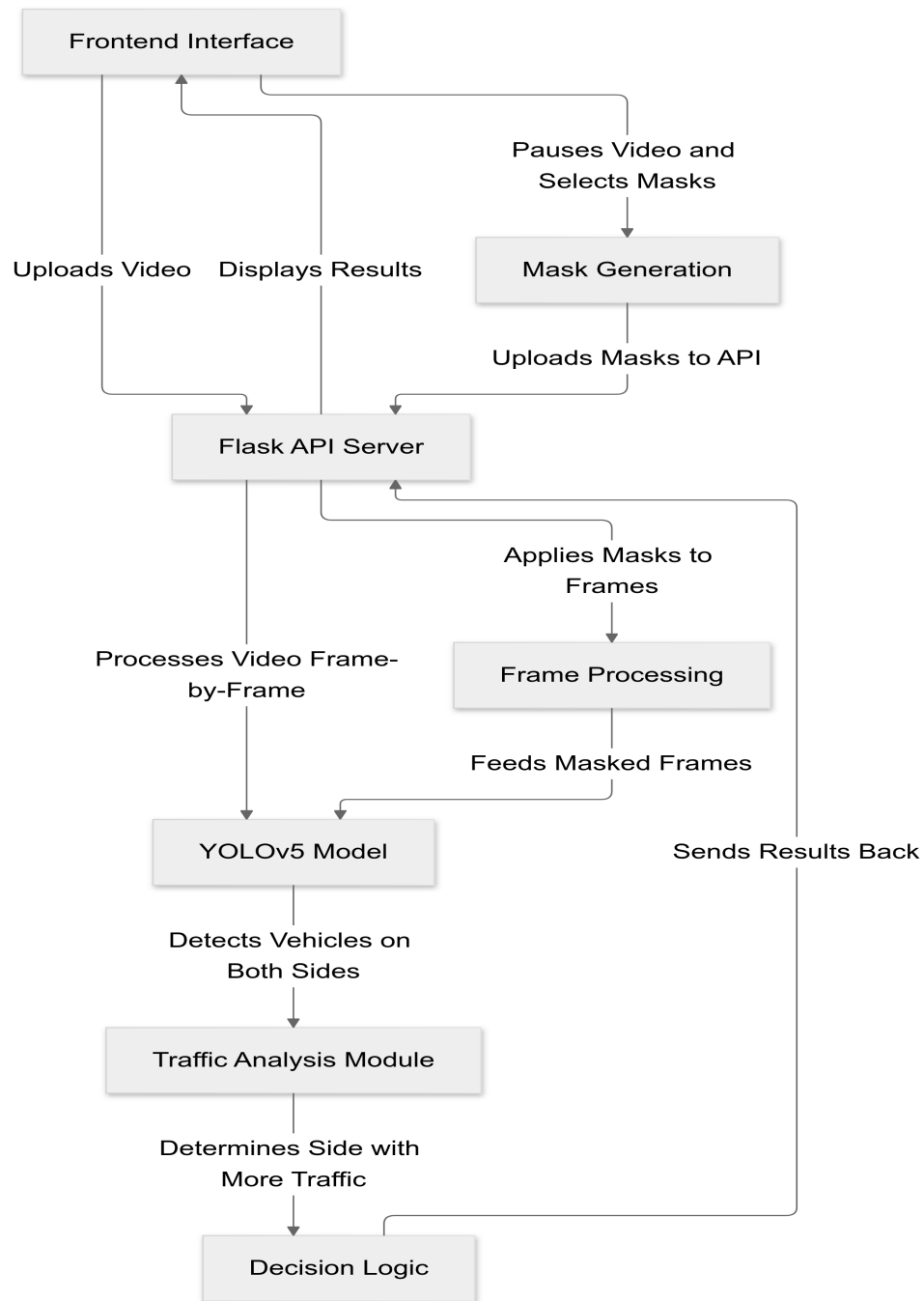


Figure 4.6: DFD diagram

4.5 Data Design

The system's information domain is transformed into structured data formats that facilitate storage, processing, and retrieval. This section includes an overview of how the primary data entities in the system are organized and manipulated to support system functionalities. For **EyeDoctorPro**, the primary data entities include user data (username, password, contact info), image data (for eye tests), test results, and chatbot conversation logs. Additionally, information about disease prevention tips and doctor contact details is stored for easy retrieval.

Data Dictionary

In the Traffic Flow Control in Urban Cities project, the Data Dictionary is crucial for understanding the entities and their relationships within the system. The entities involved include Traffic Cameras, which capture real-time vehicle data; Traffic Lights, which change states based on traffic density; Traffic Zones, representing different sections of the city; and AI Model, which processes camera data to adjust traffic light timings. The Entity Relationship Diagram (ERD) visually illustrates how these entities interact, ensuring smooth traffic management and prioritization based on real-time data inputs.

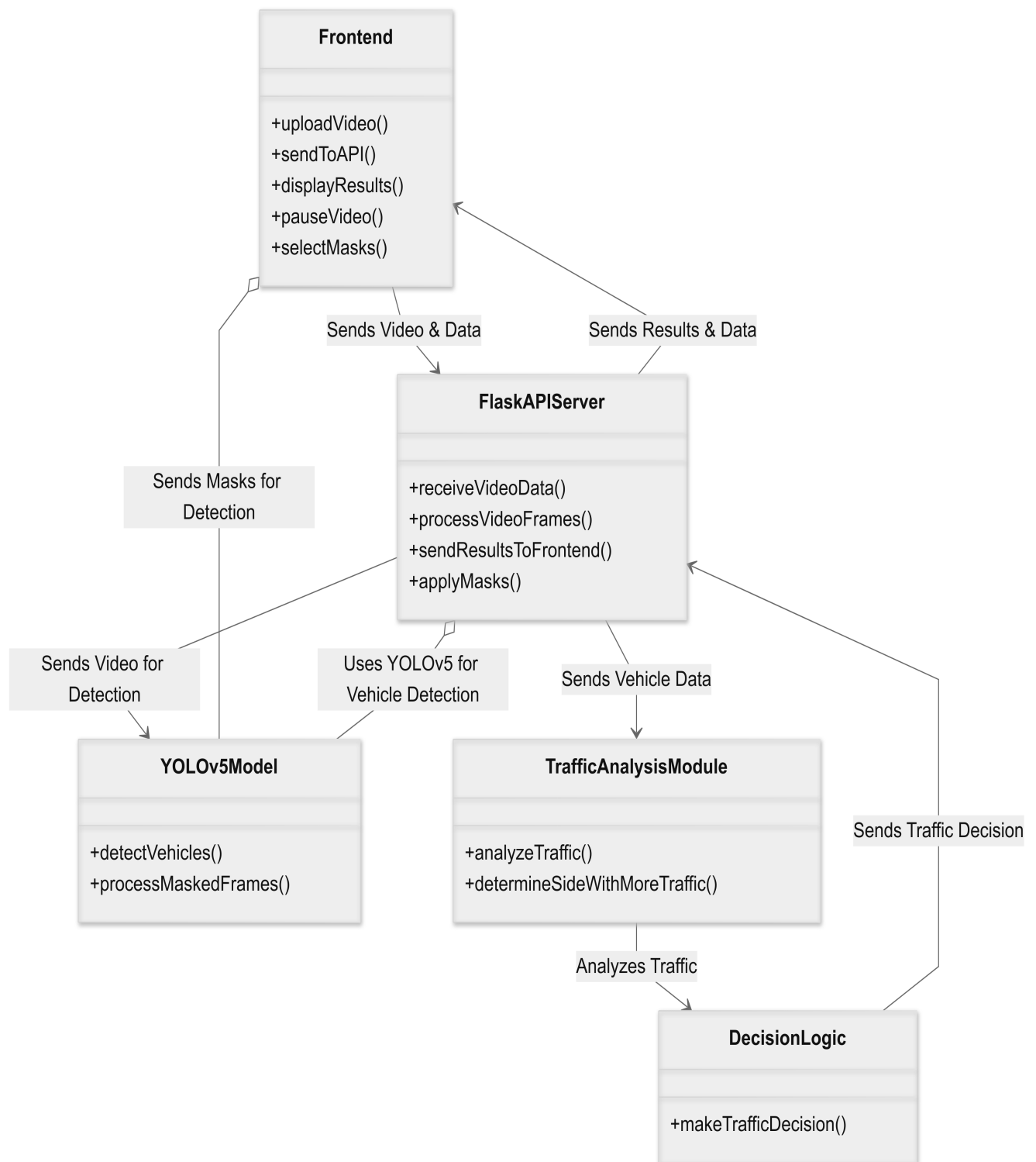


Figure 4.7: Class diagram

4.6 Human Interface Design

In the Traffic Flow Control in Urban Cities project, the user interface is designed to be intuitive and easy to navigate for city traffic operators. Users can access real-time traffic data through a dashboard that displays the status of traffic lights and vehicle counts in different traffic zones. The system allows operators to monitor live feeds from cameras, adjust traffic light timings, and view detailed analytics for traffic flow. When the system detects congestion, it automatically adjusts the traffic lights to prioritize heavily trafficked roads. Feedback is provided in the form of visual indicators (e.g., green for optimal traffic flow, red for congestion) and real-time alerts, ensuring that users can take prompt action. The interface also includes a manual override option for traffic light control and provides system status updates to inform users about the operational health of the system.

4.6.1 Screen Images:

Welcome Screen



Figure 4.8: Welcome Screen

About Screen



About

Welcome to our Smart Traffic Management System, designed to optimize urban traffic flow using real-time vehicle detection. Our system uses advanced algorithms to monitor traffic on both sides of the road and dynamically adjusts signal timings based on traffic volume. Roads with more vehicles are given priority, ensuring smoother flow and reducing congestion.

With our technology, traffic signals automatically adapt to changing conditions, giving higher priority to busier lanes. After a set interval, all roads are briefly cleared to prevent bottlenecks, promoting a balanced distribution of traffic across intersections. This approach not only enhances road safety but also minimizes travel time and emissions, contributing to smarter, greener cities.

Join us in reshaping urban mobility by making traffic management more efficient and responsive. Our goal is to create a seamless and hassle-free commuting experience for everyone.

Figure 4.9: About Screen

Upload Video Screen

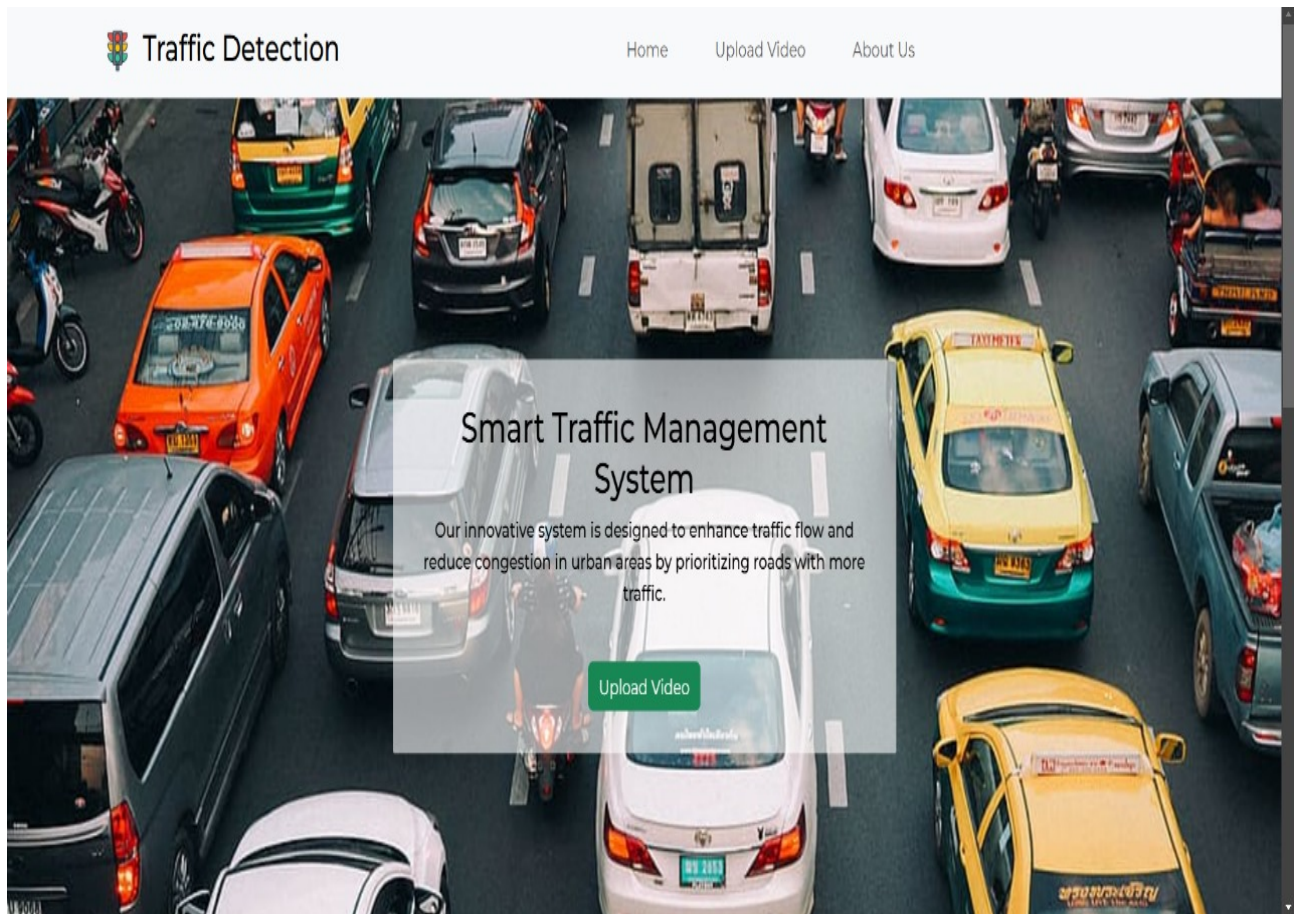



Figure 4.10: Upload Video Screen

Choose Video Screen


 Traffic Detection

[Home](#) [Upload Video](#) [About Us](#)

Car Counter with YOLO Masks

Upload Video

Drop your video file here or click to browse



Choose a video file
or drag and drop it here

Supported formats: MP4, WebM, MOV (max 100MB)

© 2024 Fakhar Corporation. All rights reserved.

Figure 4.11: Choose Video Screen

Uploaded Video Screen



© 2024 Fakhar Corporation. All rights reserved.

Figure 4.12: Uploaded Video Screen

Apply Mask Screen

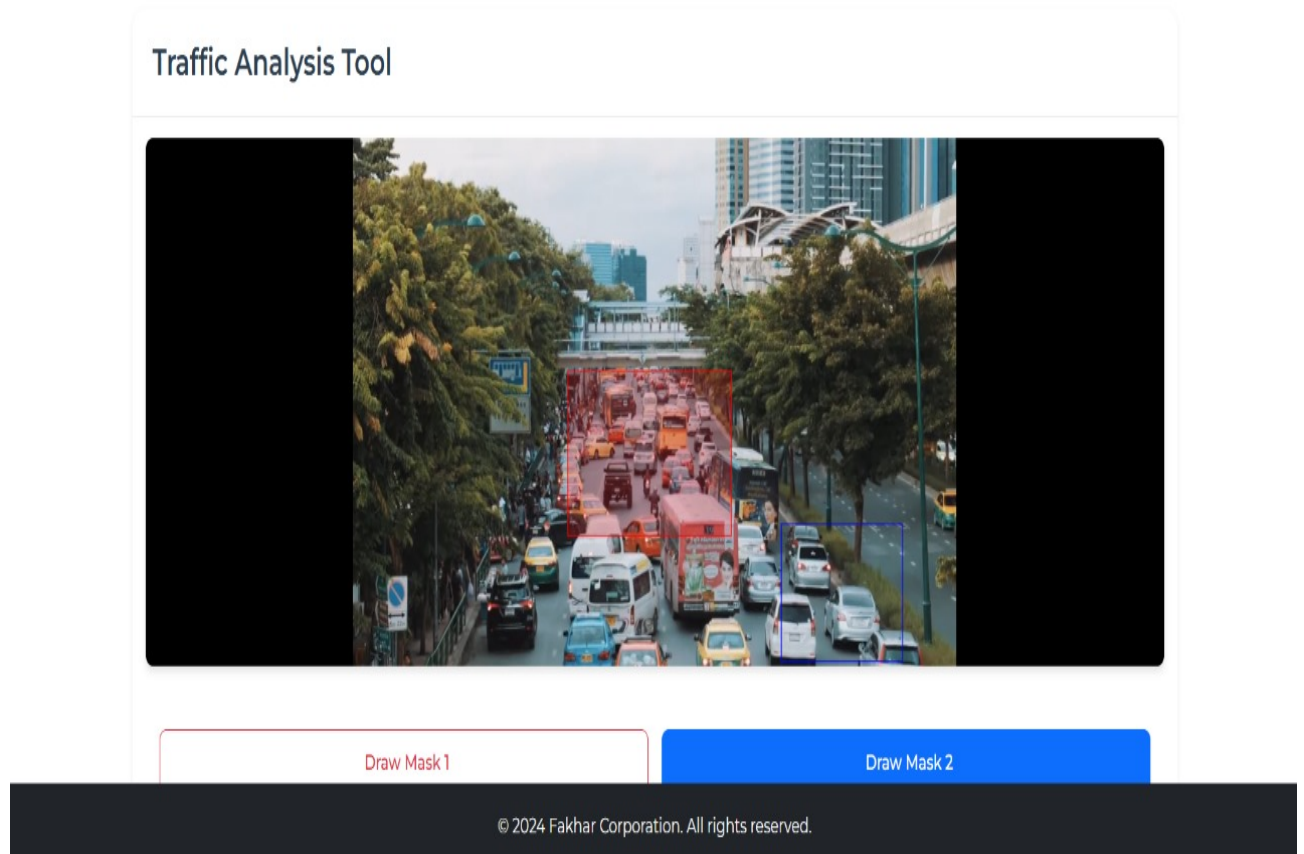
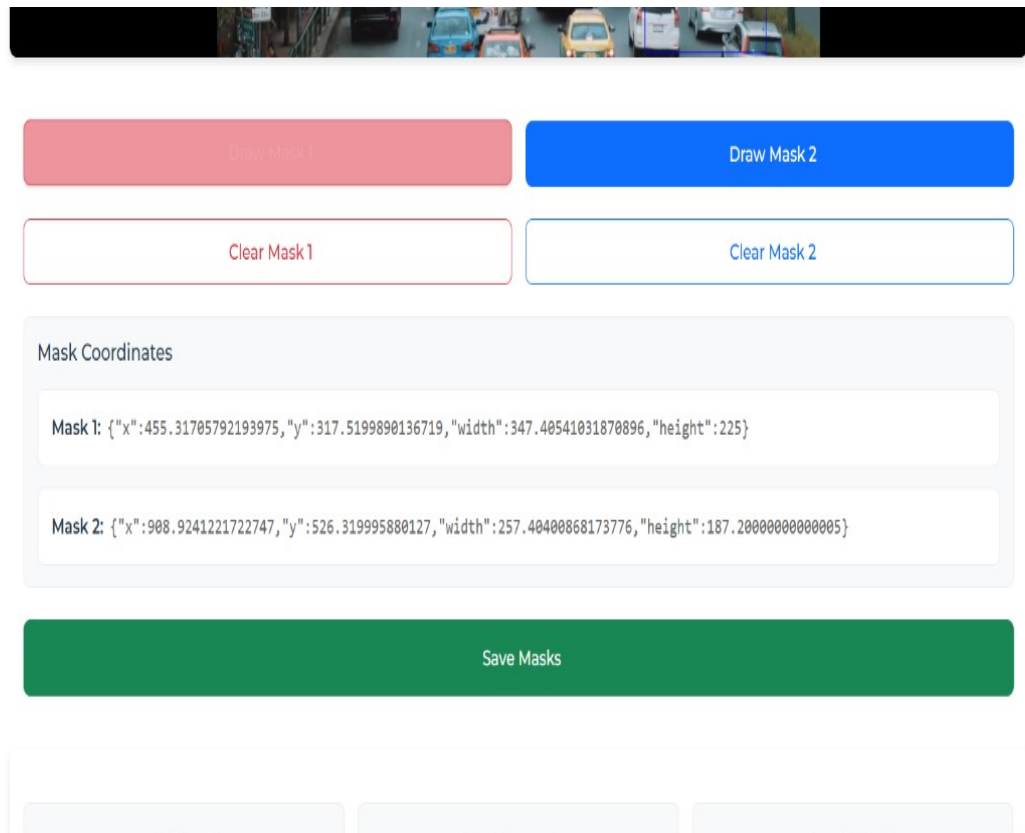


Figure 4.13: Apply Mask Screen

Mask Result Screen



The image shows a web application interface for mask management. At the top, there is a horizontal video feed showing a street scene with several cars. Below the video feed, there are four buttons arranged in a 2x2 grid: 'Draw Mask 1' (red), 'Draw Mask 2' (blue), 'Clear Mask 1' (white with red border), and 'Clear Mask 2' (white with blue border). Below these buttons is a section titled 'Mask Coordinates' which contains two text boxes. The first text box displays the coordinates for Mask 1: {"x":455.31705792193975,"y":317.5199890136719,"width":347.40541031870896,"height":225}. The second text box displays the coordinates for Mask 2: {"x":908.9241221722747,"y":526.319995880127,"width":257.40400868173776,"height":187.20000000000005}. Below the text boxes is a large green button labeled 'Save Masks'. At the bottom of the interface, there are three empty light gray rectangular boxes.

Draw Mask 1

Draw Mask 2

Clear Mask 1

Clear Mask 2

Mask Coordinates

Mask 1: {"x":455.31705792193975,"y":317.5199890136719,"width":347.40541031870896,"height":225}

Mask 2: {"x":908.9241221722747,"y":526.319995880127,"width":257.40400868173776,"height":187.20000000000005}

Save Masks

© 2024 Fakhar Corporation. All rights reserved.

Figure 4.14: Mask Result Screen

Video Result Screen

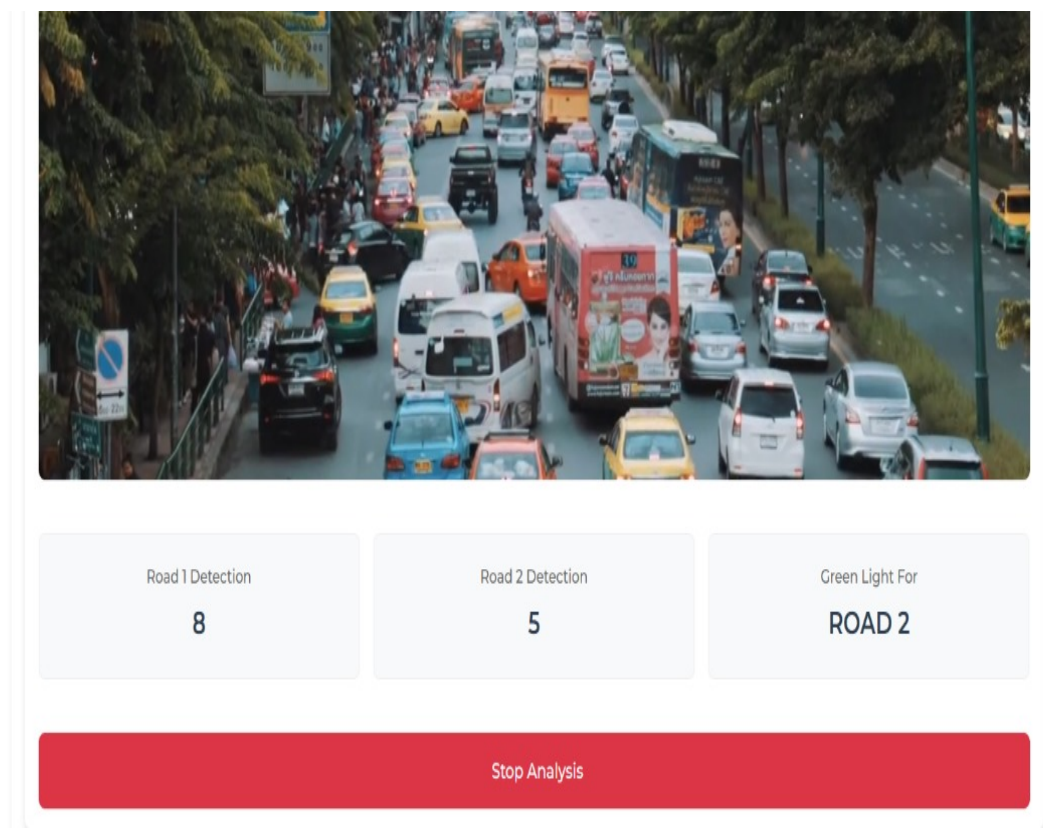


Figure 4.15: Video Result Screen

Asked Question Screen



Through our technology, traffic signals adapt to the changing conditions of the roads. By briefly clearing all roads at certain intervals, we help ensure a balanced distribution of vehicles at intersections, leading to reduced travel times and fewer traffic bottlenecks.

Frequently Asked Questions

How does the system prioritize traffic?	^
The system monitors traffic in real-time and adjusts signal timings to prioritize roads with heavier traffic. It dynamically adapts based on the number of vehicles on each road.	
Can the system handle traffic across multiple intersections?	v
Is the system eco-friendly?	v

© 2024 Fakhar Corporation. All rights reserved.

Figure 4.16: Asked Question Screen

4.6.2 Home Screen Objects and Actions

- **Navigation Buttons:** The home screen features buttons such as "Upload Workout Video," "Custom Workout Plan," "Diet Plan," and "Past Results."
- **Action:** Clicking on any button navigates the user to the corresponding screen for each feature. For example, "Upload Workout Video" leads to the video upload screen, while "Custom Workout Plan" opens the workout plan screen.

4.6.3 Workout Video Upload Screen Objects and Actions

- **Image/Video Capture Button:** Users can either take a picture or upload a workout video.
- **Action:** After uploading or capturing a video, the app analyzes the angles and provides feedback on whether the user's form is correct or incorrect. If the form is incorrect, it

suggests corrective actions or shows a reference video.

4.6.4 Custom Workout Plan Screen Objects and Actions

- **Text Blocks and Dropdowns:** This screen includes fields where users can select their fitness level (e.g., beginner, intermediate, advanced) and specific goals (e.g., strength, endurance).
- **Action:** After filling in the selections, the app generates a customized workout plan tailored to the user's fitness level and goals.

4.6.5 Diet Plan Screen Objects and Actions

- **Text Input Fields:** Users can enter their dietary preferences, allergies, and goals (e.g., weight loss, muscle gain).
- **Action:** The app uses this information to generate a personalized diet plan. It also displays nutrition facts and meal suggestions for each day.

4.6.6 Past Results Screen Objects and Actions

- **Text Blocks and Graphs:** This screen displays historical workout and diet data in graphical or tabular form.
- **Action:** The user can scroll through past performance, view progress over time, and access detailed results for each workout session or meal plan.

Chapter Five

Implementation

The implementation of the project "Traffic Flow Control in Urban Cities" involves leveraging YOLOv5, a state-of-the-art object detection model, to accurately count vehicles on different roads. The system dynamically prioritizes roads with higher traffic, allocating longer green-light durations proportionate to the vehicle count. To ensure equitable traffic clearance, the system incorporates a periodic override mechanism, opening all roads for 40 seconds at set intervals. This approach ensures efficient traffic flow management while addressing congestion on all routes.

5.1 Algorithm

Algorithm: Vehicle Counting and Prioritization

Algorithm 1: VehicleCountingAndPrioritization

Input: cameraFeeds (list of video streams)

Output: roadPriorities (list of road timings)

```
1 roadData ← [] // To store vehicle counts for each road
2 foreach feed in cameraFeeds do
3   | vehicleCount ← YOLOv5CountVehicles(feed);
4   | roadData.append(vehicleCount);
5 end
6 SortRoadsByVehicleCount(roadData);
7 roadPriorities ← AllocateTimeBasedOnCount(roadData);
8 return roadPriorities;
```

Algorithm: Dynamic Traffic Signal Control

Algorithm 2: TrafficSignalControl

Input: roadPriorities (list of road timings)

Output: None

```
1 foreach road in roadPriorities do
2   | OpenRoad(road, road.time);
3 end
4 OpenAllRoads(40) // Clear residual traffic for 40 seconds
5 NotifyCameras("Cycle Complete");
```

5.2 Tools and Technologies

Table 5.1: Tools and Technologies for Traffic Flow Control in Urban Cities

Tools and Technologies	Tools	Version
YOLOv5	6.0	For real-time vehicle detection and counting
OpenCV	4.5.5	For processing video feeds from traffic cameras
Python	3.9	For backend development, algorithm implementation, and integration
TensorFlow	2.9	For implementing advanced computations and optimizing models
Flask	2.1.2	For developing the backend API for traffic signal control
Raspberry Pi	4 Model B	For local processing of camera feeds at intersections
RESTful API	n/a	For enabling communication between the backend and traffic signal systems

5.3 External APIs/SDKs

The project utilizes several third-party APIs and SDKs for its implementation, as outlined below:

Table 5.2: External APIs and SDKs for Traffic Flow Control in Urban Cities

API/SDK	Version	Purpose
Google Maps API	n/a	Used for real-time map integration to visualize traffic flow and vehicle locations.
Twilio API	n/a	For sending notifications or alerts about traffic status and signal changes.
OpenWeather API	n/a	For providing weather data that can influence traffic flow and signal optimization.
Stripe API	n/a	For integrating payment functionality (if applicable for toll systems or traffic-related services).
TensorFlow Lite	n/a	For running the trained YOLOv5 model on edge devices such as Raspberry Pi.
PyTorch	n/a	For model training and potential experimentation with alternate machine learning models.
Road Traffic API	n/a	Used for collecting live data from traffic cameras and traffic flow statistics.

API/SDK (Continued)	Version	Purpose
Flask API	2.1.2	Used for setting up the backend and handling requests for traffic signal control.

5.4 User Interface

This section describes the user interfaces for different subsystems (such as Mobile App, Web App, Client App, and Admin App). Each user interface includes descriptions and detailed screen layouts.

Welcome Screen

The welcome screen introduces users to the "Traffic Flow Control in Urban Cities" system, showcasing its goal to optimize urban traffic management using AI and IoT technologies. Users can access features like vehicle detection, traffic prioritization, and real-time signal adjustments, as shown in Figure 5.9.



Figure 5.1: Welcome Screen

About Screen

The about screen highlights the project's purpose of enhancing traffic efficiency in urban areas through YOLOv5-based vehicle counting, real-time data processing, and intelligent traffic signal control. It emphasizes sustainability and reduced congestion for smarter cities, as shown in Figure ??.



About

Welcome to our Smart Traffic Management System, designed to optimize urban traffic flow using real-time vehicle detection. Our system uses advanced algorithms to monitor traffic on both sides of the road and dynamically adjusts signal timings based on traffic volume. Roads with more vehicles are given priority, ensuring smoother flow and reducing congestion.

With our technology, traffic signals automatically adapt to changing conditions, giving higher priority to busier lanes. After a set interval, all roads are briefly cleared to prevent bottlenecks, promoting a balanced distribution of traffic across intersections. This approach not only enhances road safety but also minimizes travel time and emissions, contributing to smarter, greener cities.

Join us in reshaping urban mobility by making traffic management more efficient and responsive. Our goal is to create a seamless and hassle-free commuting experience for everyone.

Figure 5.2: About Screen

Upload Video Screen

This screen allows users to upload a video file for analysis. Users can browse and select a video from their device, which will then be processed for traffic monitoring, as shown in Figure ??.

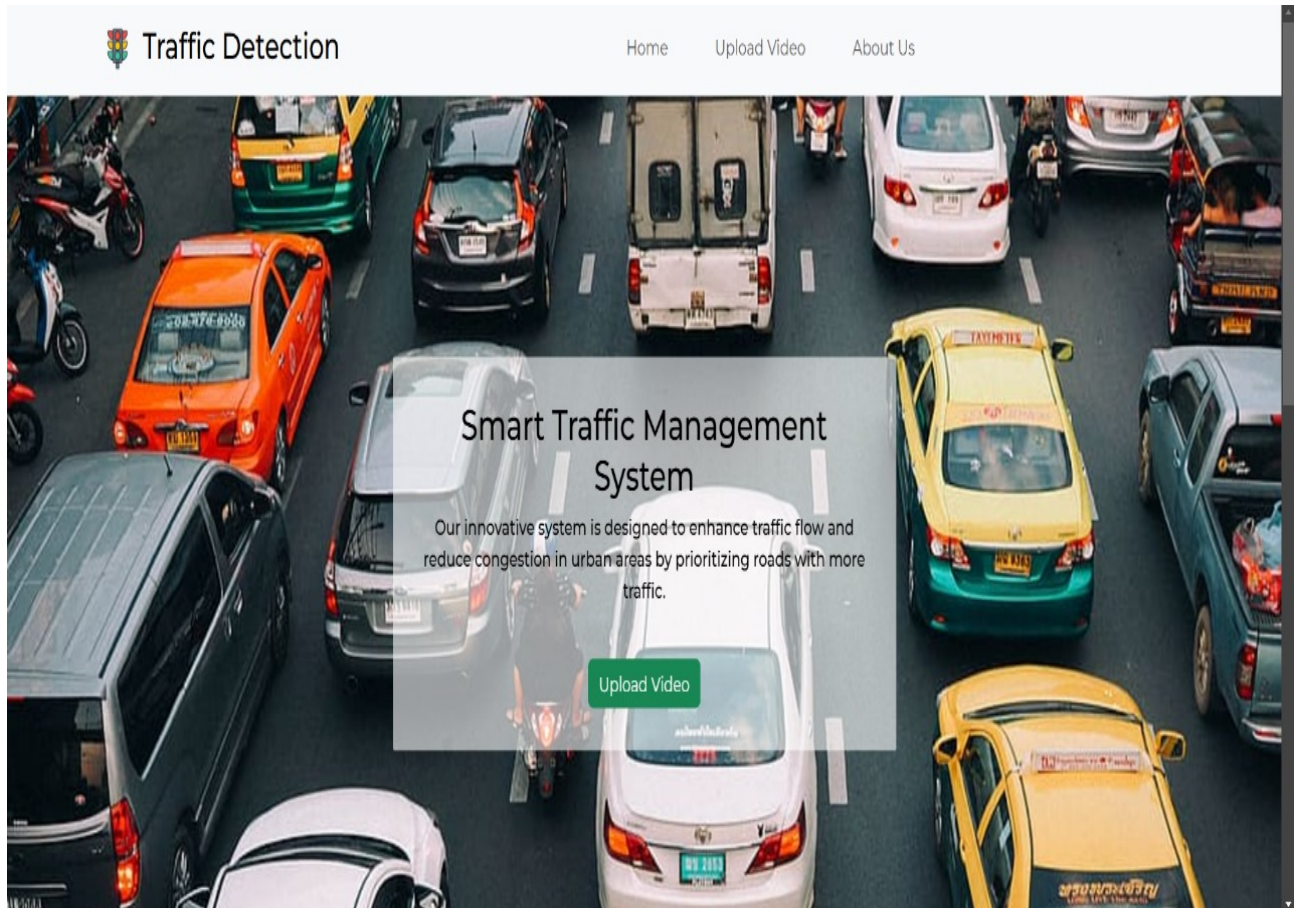


Figure 5.3: Upload Video Screen

Choose Video Screen

This screen enables users to select a pre-recorded video from a predefined library or database for analysis. It ensures flexibility by offering quick access to relevant traffic videos, Figure ?? shows the interface for this screen.

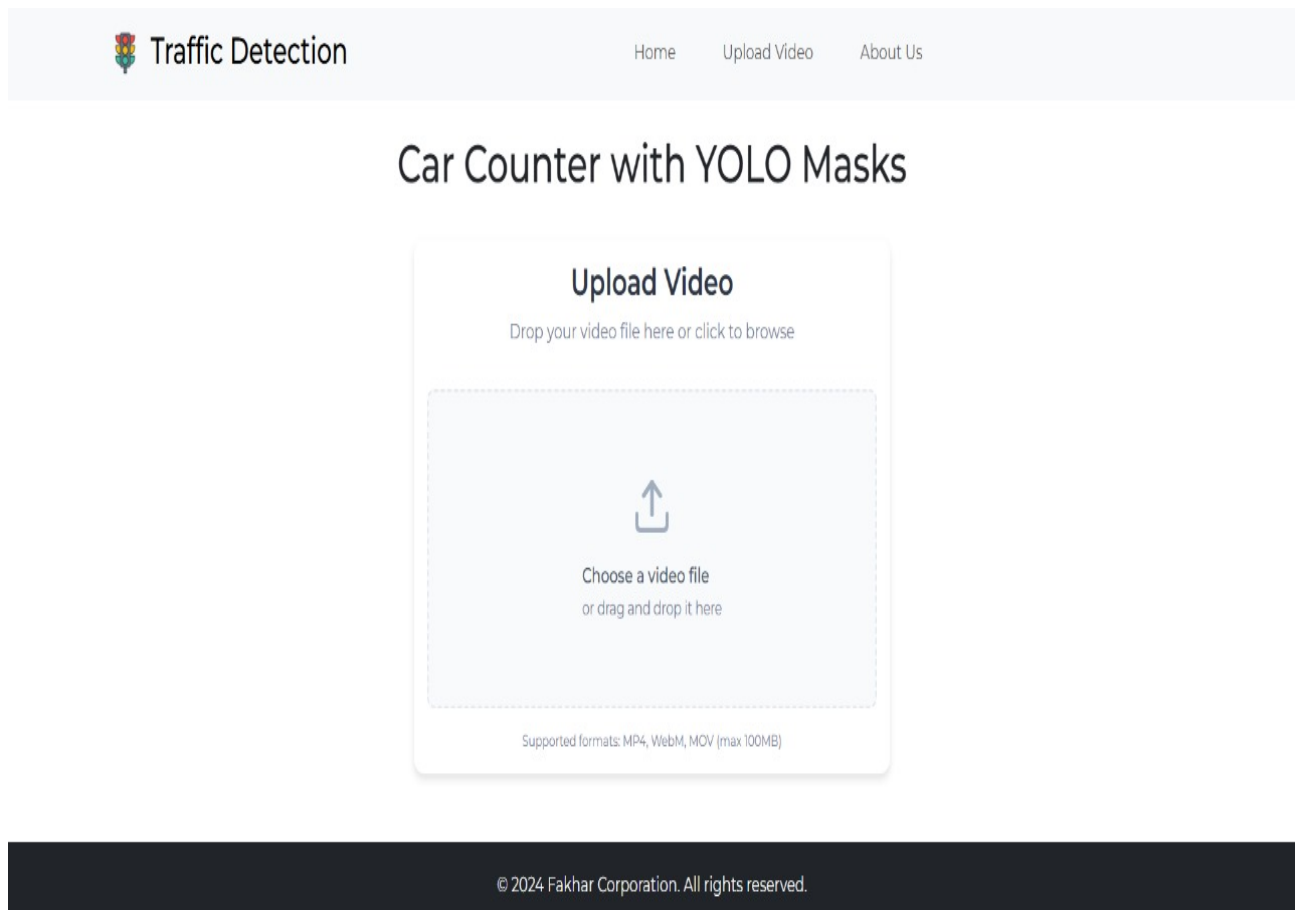
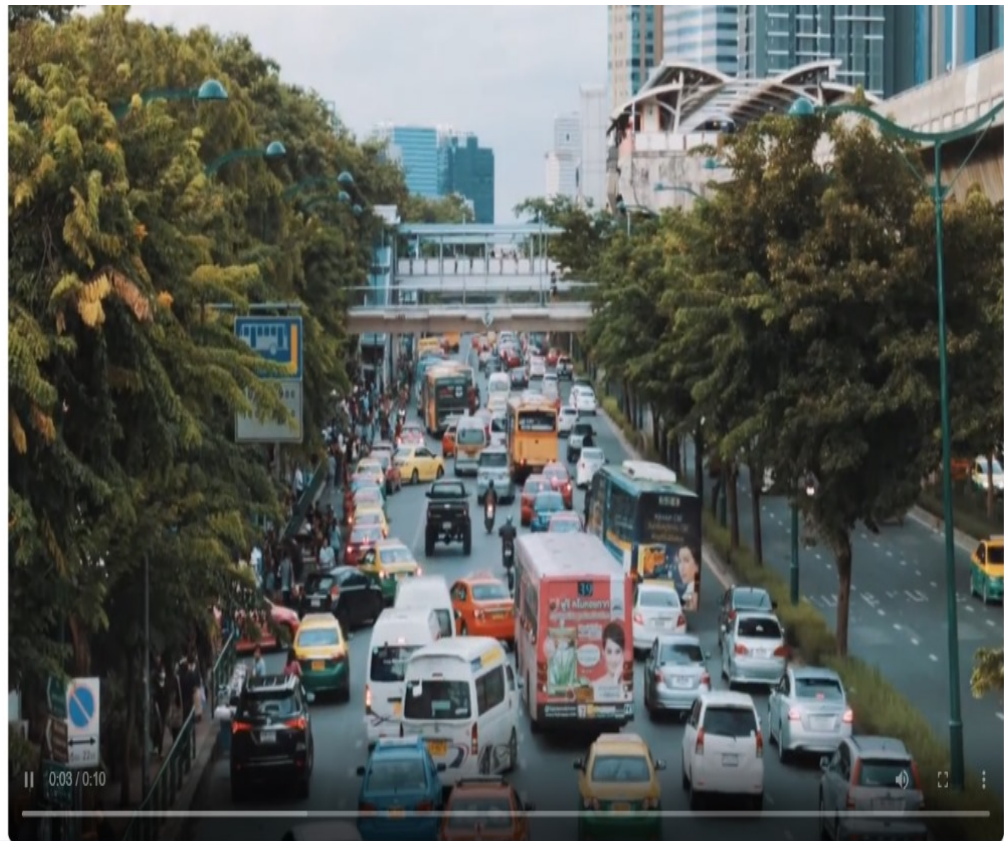


Figure 5.4: Choose Video Screen

Uploaded Video Screen

This screen provides a preview of the video selected by the user. It allows verification of the uploaded file to ensure the correct video is being analyzed, as shown in Figure ??.



© 2024 Fakhar Corporation. All rights reserved.

Figure 5.5: Uploaded Video Screen

Apply Mask Screen

This screen enables users to apply a mask on the video, highlighting specific areas of interest for precise traffic analysis and accurate data extraction, as shown in Figure ??.

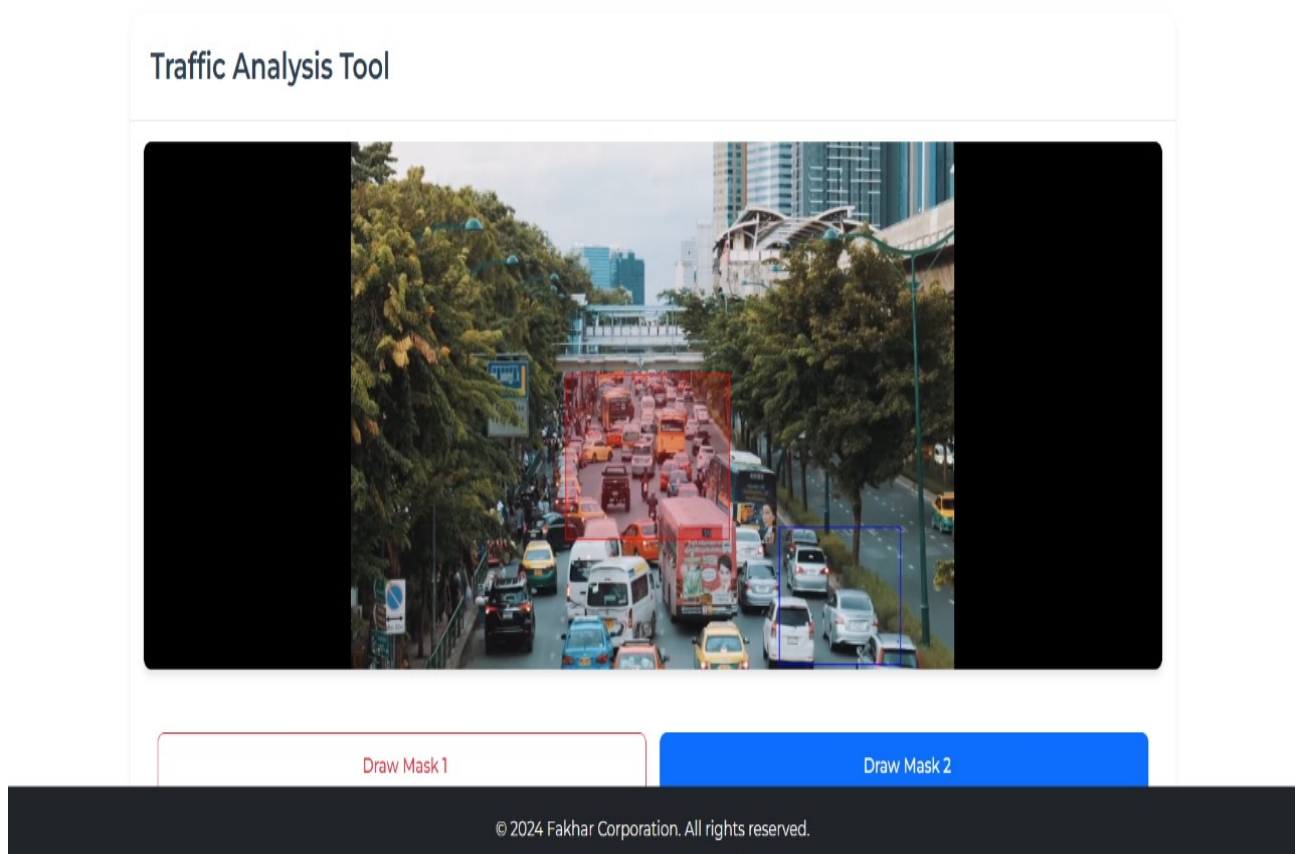


Figure 5.6: Apply Mask Screen

Mask Result Screen

This screen displays the video with the applied mask, showcasing the highlighted regions of interest and ensuring the masking process has been accurately executed for analysis, as shown in Figure ??.

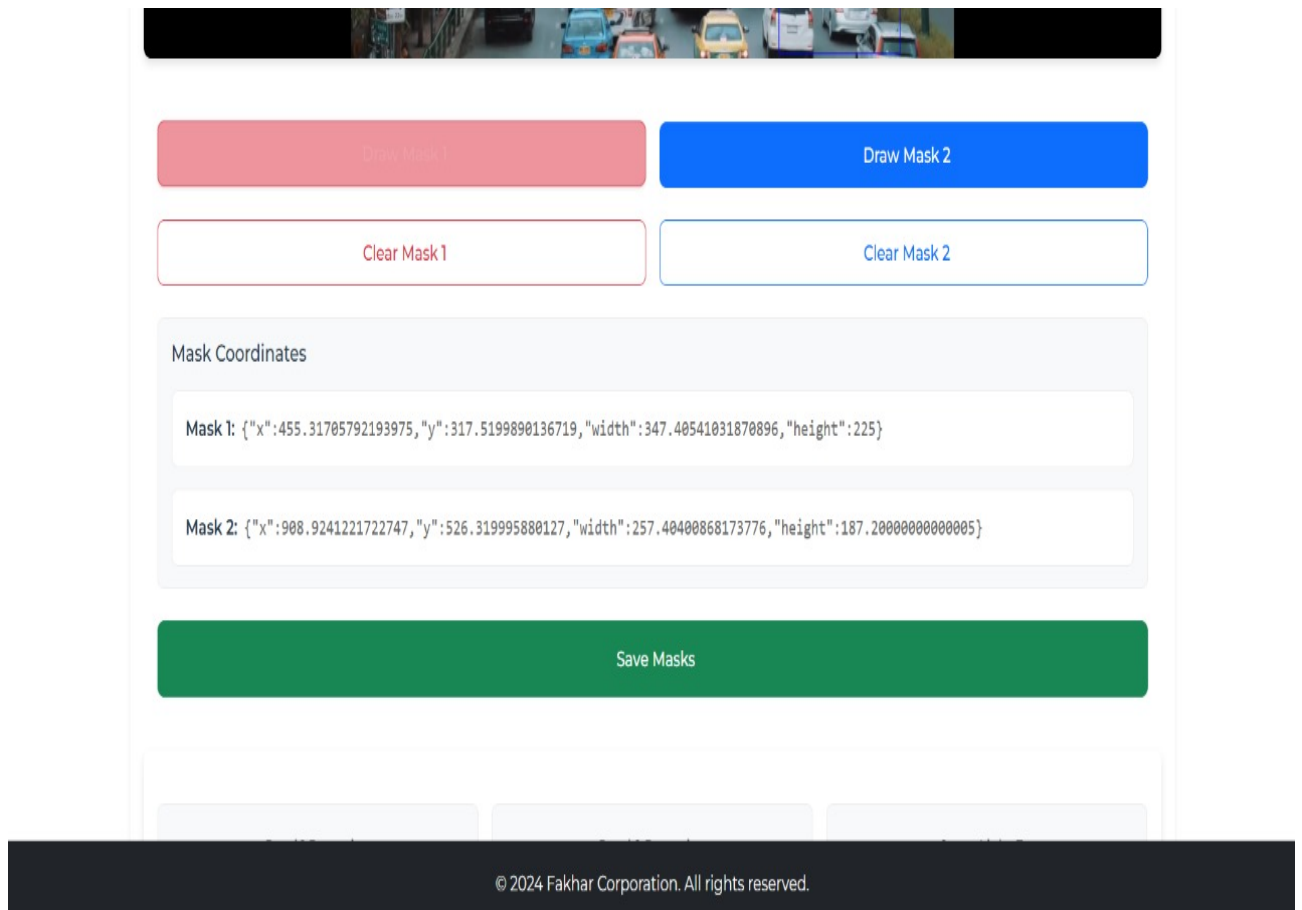
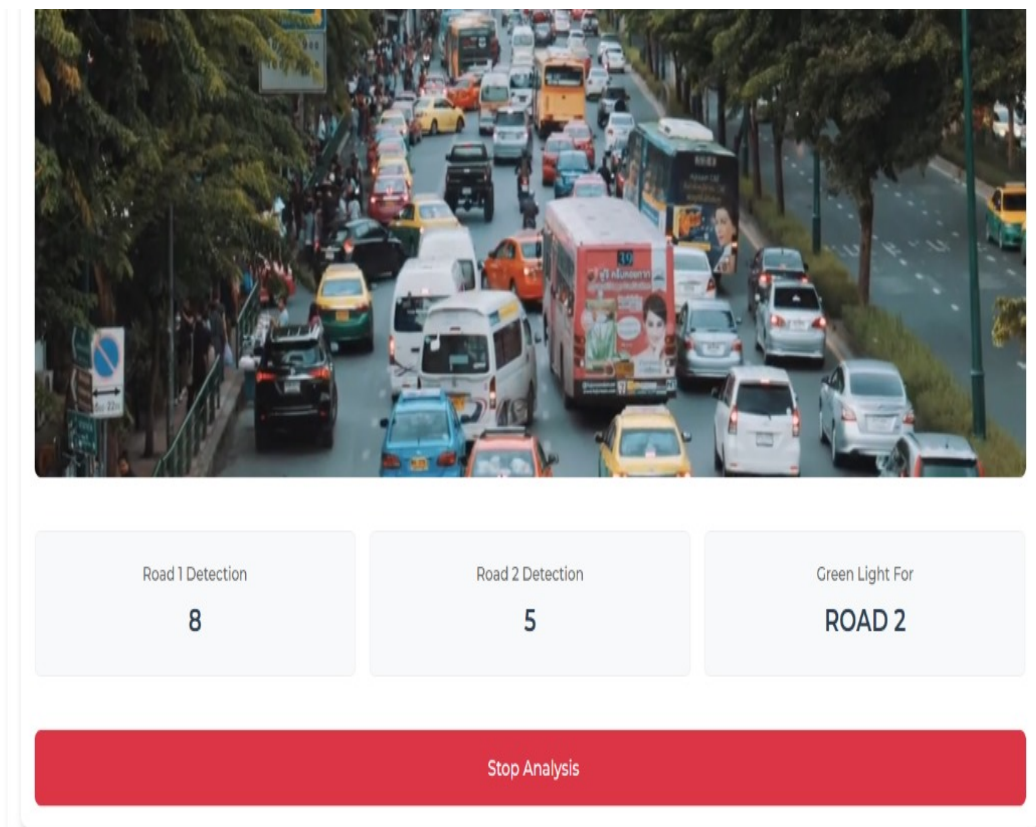


Figure 5.7: Mask Result Screen

Video Result Screen

This screen presents the processed video results, including key traffic insights such as vehicle count, density, and prioritization outcomes for improved traffic management, as shown in Figure ??.



© 2024 Fakhar Corporation. All rights reserved.

Figure 5.8: Video Result Screen

Asked Question Screen

This screen allows users to view frequently asked questions related to traffic flow management. It provides concise answers and guidance, ensuring users understand the system’s features and functionalities effectively, as shown in Figure ??.



Through our technology, traffic signals adapt to the changing conditions of the roads. By briefly clearing all roads at certain intervals, we help ensure a balanced distribution of vehicles at intersections, leading to reduced travel times and fewer traffic bottlenecks.

Frequently Asked Questions

How does the system prioritize traffic?

^

The system monitors traffic in real-time and adjusts signal timings to prioritize roads with heavier traffic. It dynamically adapts based on the number of vehicles on each road.

Can the system handle traffic across multiple intersections?

v

Is the system eco-friendly?

v

Figure 5.9: Asked Question Screen

5.5 Deployment

The deployment of the "Traffic Flow Control in Urban Cities" system involves setting up both software and hardware components, ensuring seamless integration between the backend system, edge devices, traffic cameras, and traffic signal controllers. Below are the key steps for deployment:

5.5.1 Hardware Deployment

- **Raspberry Pi Setup:** The Raspberry Pi 4 Model B will be deployed at each traffic intersection to process video streams locally. The Raspberry Pi will run the YOLOv5 model for vehicle detection and counting. Each device will be connected to traffic cameras and local signal control systems.
- **Cameras:** High-definition cameras will be set up at key intersections to capture real-time video feeds. These cameras will be integrated with the Raspberry Pi for processing and vehicle counting.
- **Traffic Signal Controllers:** The traffic signal systems will be connected to the Raspberry Pi via an API to receive timing instructions based on vehicle counts.

5.5.2 Backend Deployment

- **Cloud Server:** The backend system, built using Flask, will be deployed on a cloud server (such as AWS, Azure, or Google Cloud). The server will handle requests from the Raspberry Pi devices, process traffic data, and send instructions to traffic signal controllers.
- **Database:** MongoDB will be used to store traffic data and logs. It will be deployed in a cloud-based environment or on a dedicated server, ensuring high availability and scalability for real-time data processing.
- **API Setup:** RESTful APIs will be deployed on the cloud server to facilitate communication between the backend, Raspberry Pi devices, and traffic signal controllers.

5.5.3 Model Deployment

- **YOLOv5 Model Deployment:** The trained YOLOv5 model will be deployed on the Raspberry Pi using TensorFlow Lite or PyTorch for optimized inference. The model will continuously process the camera feeds to detect and count vehicles in real-time.
- **Model Updates:** Any updates to the YOLOv5 model (e.g., for improved accuracy or performance) will be pushed to the Raspberry Pi devices via the cloud backend.

5.5.4 Monitoring and Maintenance

- **Traffic Data Monitoring:** Real-time traffic data will be monitored via a dashboard on the backend server. This will help in analyzing traffic patterns and optimizing signal timings.

- **System Health Monitoring:** Tools like Prometheus and Grafana can be used for monitoring system health and performance, ensuring that all components (Raspberry Pi devices, cameras, and cloud server) are functioning optimally.
- **Log Management:** MongoDB will store traffic data logs, which can be used for reporting and analysis. Logs will also include system health metrics to ensure quick identification and resolution of issues.

5.5.5 Scalability and Future Updates

- **Scalability:** The system is designed to scale, with additional Raspberry Pi devices and cameras able to be deployed as needed. The cloud backend can handle an increasing volume of data and traffic signals.
- **Future Model Improvements:** Future updates to the vehicle detection model (YOLOv5 or alternative models) will be deployed through the cloud system and pushed to the Raspberry Pi devices.
- **API Integration:** The system can integrate additional APIs for weather, traffic events, or public transportation [7] systems to improve traffic management and signal control.

Chapter Six

Testing and Evaluation

The testing and evaluation of the "Traffic Flow Control in Urban Cities" system focused on assessing the accuracy of vehicle detection, real-time responsiveness, and the efficiency of traffic signal prioritization. The YOLOv5 model was tested with diverse traffic scenarios to ensure robust vehicle counting under varying conditions such as lighting and congestion levels. End-to-end system functionality was validated by simulating live traffic feeds and measuring signal control timings against predefined benchmarks. The system's performance was further evaluated using metrics like processing latency, model accuracy, and its impact on traffic flow optimization, ensuring it meets the project's goals. Once the system has been successfully developed, testing has to be performed to ensure that the system is working as intended. This is also to check that the system meets the requirements stated earlier. Besides that, system testing will help in finding the errors that may be hidden from the user. The testing must be completed before it is deployed for use.

There are few types of testing which includes the unit testing, functional testing and integration testing. You are required to perform each of these in-depth to ensure system quality.

6.1 Unit Testing

Unit Testing 1: Traffic Camera Feed Integration and Vehicle Detection

Testing Objective: To ensure that the system accurately processes live traffic camera feeds and detects vehicles correctly under various conditions, including valid and invalid input feeds.

- **Valid Inputs:** Live video streams from functioning traffic cameras. The system should correctly count vehicles in real-time.
- **Invalid Inputs:** Corrupted video files or disconnected camera feeds. The system should gracefully handle errors and log issues without crashing.

This testing validates the YOLOv5 model's reliability and the system's ability to process camera feeds as designed.

6.2 Functional Testing

Functional testing follows unit testing to ensure that the integrated modules work together seamlessly and meet the system's specifications and requirements.

Functional Testing 1: Traffic Signal Control Based on Vehicle Count

Objective: To ensure that the system prioritizes roads based on the detected vehicle count and allocates signal timings accordingly, while also implementing the default 40-second clearing interval for all roads.

- The system should accurately open roads with the highest vehicle count first, adjusting signal timings dynamically.
- All roads should be opened for 40 seconds at regular intervals to clear residual traffic, ensuring fair access for less congested roads.
- Testing ensures that the system operates as intended under different traffic conditions and maintains smooth traffic flow.

6.3 Business Rules Testing

Decision table-based testing is employed to validate the business rules in the traffic flow control system. The business rules were defined in the functional requirements and use cases. This testing technique uses a systematic approach, presenting inputs and outputs in a tabular form to model the decision logic clearly and compactly.

For the traffic control project, conditions such as vehicle counts, road congestion levels, and time intervals are tested as inputs, while the actions include assigning road priorities, allocating signal timings, and initiating the default 40-second clearing interval. The decision table ensures all possible combinations of conditions and outcomes are tested to verify the correctness of the implemented rules.

6.4 Integration Testing

Integration tests assess whether components in the traffic flow control system interact seamlessly without errors. These tests ensure that the interfaces and interactions between various modules, such as vehicle detection, traffic prioritization, and signal control, work properly. After individual unit tests have been passed, the focus shifts to validating the flow of data and control among interconnected modules. Integration testing involves a test plan with multiple scenarios developed by a team of programmers or analysts.

Integration Testing 1: Traffic Signal Prioritization

Testing Objective: To ensure that the prioritization of roads based on vehicle counts is executed correctly and that the interface between the 'Vehicle Counting Module' and the 'Traffic Signal Control Module' operates without issues.

Chapter Seven

Conclusion and Future Work

7.1 Conclusion

The Traffic Flow Control in Urban Cities system has been successfully developed to optimize traffic signal prioritization using a YOLOv5-based vehicle counting model. The system integrates several components, including vehicle detection, road prioritization, and dynamic traffic signal control, to enhance traffic management efficiency. By leveraging technologies like OpenCV, TensorFlow, Flask, and Raspberry Pi, the system processes live traffic feeds and adjusts signals dynamically. Rigorous testing and evaluation have validated the system's performance, ensuring it meets the specified requirements for urban traffic management. The project is now ready for further deployment and real-world testing in urban environments.

7.2 Future Work

While the current system provides a strong foundation for traffic management, there are numerous opportunities for future enhancements:

- **Integration with Smart City Systems:** Extend the system to integrate with broader smart city infrastructures, such as emergency vehicle management and public transportation systems.
- **Predictive Traffic Analysis:** Implement machine learning algorithms to predict traffic congestion patterns and pre-emptively adjust signal timings.
- **Scalability:** Optimize the system to handle data from a larger number of traffic intersections across multiple cities.
- **Enhanced Edge Computing:** Improve the capabilities of edge devices, like Raspberry Pi, to process more complex traffic scenarios locally, reducing latency.
- **Environmental Impact Monitoring:** Include modules to monitor and minimize vehicle emissions by optimizing idle times at traffic signals.
- **User Interface Enhancements:** Develop an intuitive dashboard for traffic authorities to monitor and control the system in real time.

By incorporating these advancements, the Traffic Flow Control system can significantly contribute to smarter, more sustainable urban traffic management, ensuring smoother commutes and reduced environmental impact.

References

- [1] J. Redmon, S. Divvala, R. B. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2016, pp. 779–788, 2016.
- [2] J. Liu, S. Zhang, and Q. Zhao, “Real-time traffic signal control with reinforcement learning: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 12, pp. 5095–5107, 2020.
- [3] N. Padhy and P. Reddy, “Vehicle detection and classification using opencv and deep learning,” pp. 122–126, 2019.
- [4] L. Zhou, X. Wang, and B. Shi, “Real-time traffic flow prediction with deep learning models,” *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 207–218, 2018.
- [5] T. Team, “Tensorflow: An open source platform for machine learning,” 2021. Accessed: 2023-11-30.
- [6] F. Team, “Flask documentation,” 2021. Accessed: 2023-11-30.
- [7] G. Garber and N. P. P. Zhou, *Intelligent Transportation Systems: Applications, Frameworks, and Technologies*. Boca Raton, FL: CRC Press, 1st ed., 2019.