AI-Powered Traffic Management System

Detailed Report

Mou Yangyi-1306078
Huang Jinfan-1236607

Wenzhou-Kean University

Dr. Hamza Djigal

May 9,2025

1. Requirement Analysis

1.1 Problem Definition

The increasing volume of urban traffic has led to chronic congestion, travel delays, and higher emissions. Traditional traffic signals with fixed timings or simple sensor triggers struggle to adapt to dynamic traffic flows, resulting in inefficiencies during peak hours. Moreover, emergency vehicles often face delays at intersections due to lack of coordinated priority systems. An AI-powered traffic management system is proposed to address these problems by using real-time data and AI-driven control to adapt signal timings dynamically. This system aims to optimize overall traffic flow, reduce waiting times, and ensure that emergency vehicles receive priority passage.

1.2 Stakeholders

The key stakeholders include:
City Traffic Authorities: Responsible for implementing and overseeing traffic control systems; they need improved traffic flow and safety.
Emergency Services (Ambulance, Fire, Police): Require prioritized routes to ensure rapid response; the system must guarantee that emergency vehicles can request and receive priority passage.
General Public (Drivers and Pedestrians): Users of the road who benefit from reduced waiting times and safer intersections; their travel experience and safety are directly impacted.
System Administrators and Operators: Technicians and traffic engineers who manage the system, monitor performance, and perform maintenance tasks.
Government and City Planners: Interested in reducing pollution and congestion for economic and environmental goals; they fund and regulate traffic infrastructure.
Sensor and Equipment Vendors: Provide hardware components (sensors, controllers, lights) and may be involved in installation and maintenance contracts.

1.3 Functional Requirements

The system must support the following core functions:
Vehicle Detection and Classification: Detect the presence of vehicles, check the action of the car(if it is illegal， if yes , update its plate to database) and identify emergency vehicles through sensors (e.g., camera, radar) at intersections.
Data Collection and Storage: Collect real-time traffic data (vehicle counts, speeds, license plate info) and store it in a database for analysis and record-keeping.
Traffic Signal Control: Dynamically adjust traffic light states (green, yellow, red) based on AI decision logic to optimize flow and minimize wait times.
Emergency Priority Handling: Allow emergency vehicles to send priority requests to the admin, triggering a sequence (admin check if the request is valid, if yes, then update the request to the databse)that gives them a green signal as quickly as possible.
Default Control Mode: Provide a baseline signal timing algorithm for normal conditions when no emergency request is active.
Administrative Overrides: Enable authorized personnel (admins) to manually change signal states or update system data (e.g., authorize emergency requests, update vehicle databases).
System Self-Monitoring: Perform regular self-checks of sensors and system components to detect faults, and handle invalid data (e.g., update or ignore false plate readings).

1.4 Non-Functional Requirements

The system must meet the following qualities:
Real-Time Performance: Signal decisions should be computed and applied within strict time constraints (e.g., sub-second latency) to respond promptly to traffic changes.
Reliability and Availability: Operate continuously (24/7) with minimal downtime; ensure fail-safe modes so that intersections default to safe signal patterns if the system fails.
Scalability: Support deployment across multiple intersections or an entire city; handle increasing traffic volumes and numbers of sensors without performance degradation.
Accuracy: Vehicle detection and classification (especially emergency identification) should be highly accurate to avoid false positives or missed requests.
Security: Protect against unauthorized access or malicious commands (e.g., ensure only legitimate emergency requests are honored, secure data storage).
Maintainability: Allow software updates and AI model retraining without major system overhauls; support diagnostic tools for maintenance.
Interoperability: Integrate with existing traffic signal hardware and city infrastructure, using standard communication protocols where possible.
Safety: Ensure that any control changes do not compromise safety; for example, avoid giving green lights to conflicting directions.
Efficiency: Optimize not only travel time but also secondary metrics like fuel consumption and emissions reduction through smooth traffic flow.

1.5 Main Method

1. Requirements-Driven Design
The system was developed based on a clear analysis of user needs and urban traffic challenges. Through identifying stakeholders and defining functional and non-functional requirements, the solution ensures practical value and applicability in real-world city environments.

2. UML-Based System Modeling
Unified Modeling Language (UML) was employed extensively to describe the system's structure and behavior. UML diagrams used include:

Use Case Diagrams to define user interactions and system functions,

Sequence Diagrams to illustrate dynamic workflows (e.g., emergency vehicle priority handling),

Class Diagrams to describe object relationships and responsibilities,

State Diagrams to capture lifecycle transitions of key components such as traffic lights and the AI controller.
These visual models facilitated better communication among team members, clearer documentation, and modular implementation planning.

3. Object-Oriented Analysis and Design (OOAD)
The system components are modeled using object-oriented principles, encapsulating real-world entities such as vehicles, sensors, and traffic lights into classes with attributes and methods. Inheritance and polymorphism are utilized, as seen in the design of EmergencyVehicle as a subclass of Vehicle.

4. License Plate Recognition (LPR) Technology
The system integrates license plate recognition to identify emergency vehicles and detect traffic violations. LPR is implemented via:

High-definition cameras at intersections,

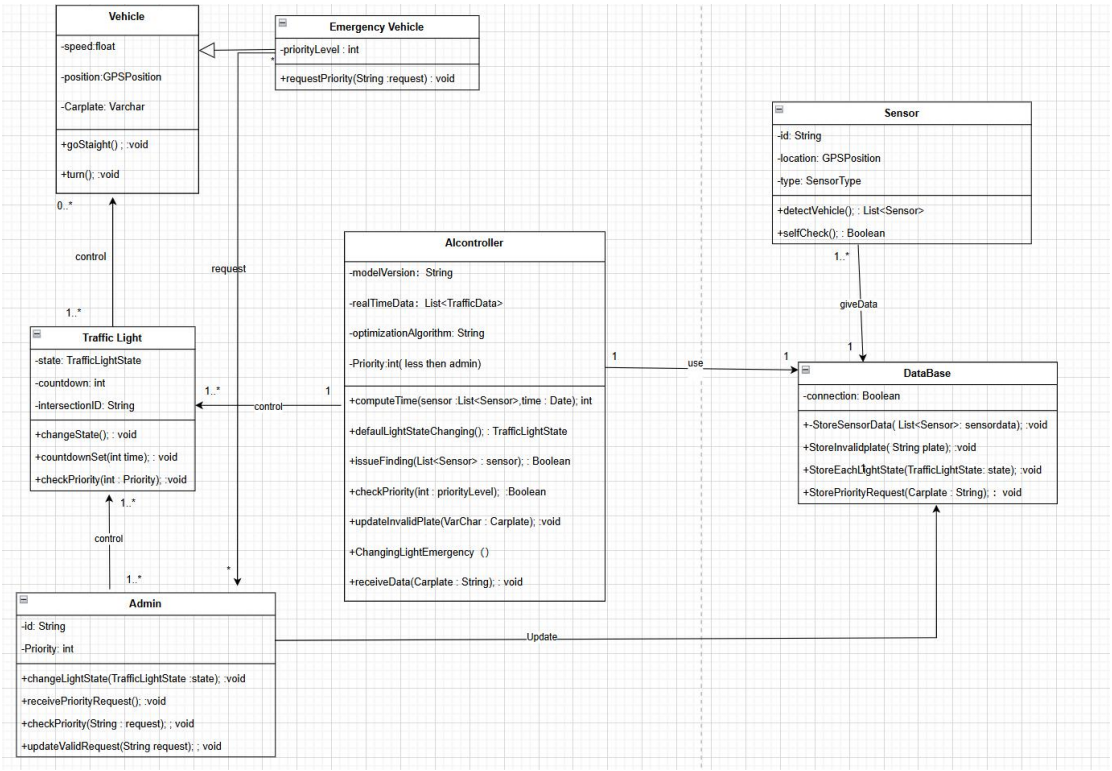
Computer vision algorithms (e.g., YOLO or OpenCV-based OCR),

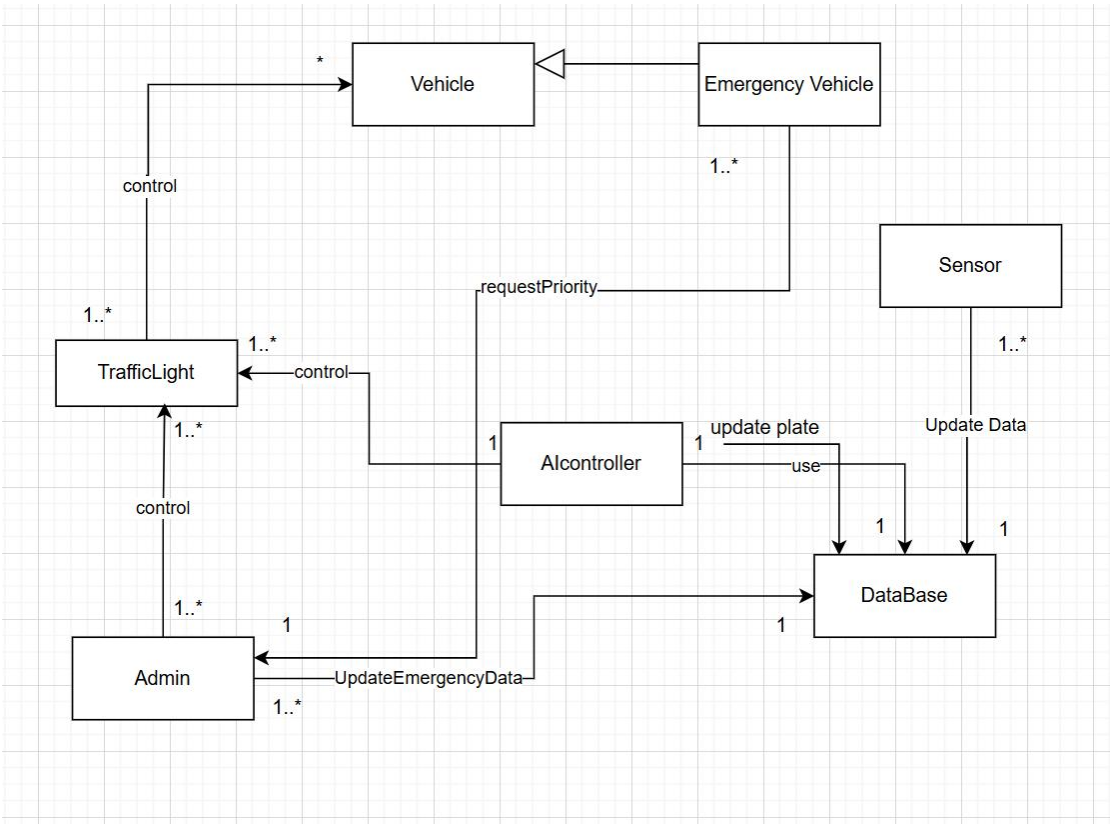
A matching mechanism with a pre-authorized emergency vehicle database.
Detected plates are validated and classified as either valid or invalid; unauthorized vehicles attempting to trigger emergency control are logged for review.

5. Event-Driven and AI-Based Decision Making
The core of the system is event-driven: traffic data or emergency requests trigger AI processing in real time. The AI controller evaluates sensor input, predicts traffic flow, and generates control

signals using intelligent decision logic. This enables dynamic adjustment of light durations and emergency prioritization.

2.System Architecture





The system employs an object-oriented architecture with several main modules, as illustrated by the provided class and component diagrams. These modules correspond to key real-world entities and their interactions. The AIController acts as the central coordinator, linking Sensors, Traffic Lights, and the Database. Below are the primary classes/modules and their relationships:
Vehicle: Represents a generic vehicle with attributes for speed, position (GPS), and a license plate. It provides methods such as goStraight() and turn(). Each TrafficLight can control multiple vehicles approaching that intersection.

EmergencyVehicle: A subclass of Vehicle that adds a priorityLevel attribute and a method requestPriority(String request). Emergency vehicles can signal to the AIController that they need priority. The class diagram indicates multiple EmergencyVehicle instances can exist and interact with the system.

Sensor: Detects vehicles approaching intersections. Attributes include id, location (GPS), and type. Key methods are detectVehicle() (which update the data about vehicles it senses to database) and selfCheck(). Sensors feed data into the AIController.

TrafficLight: Models a traffic signal at an intersection. Attributes include state (TrafficLightState), a countdown timer, and an intersectionID. Methods include changeState(), countdownSet(int time), and checkPriority(int priority). The AIController can command multiple TrafficLight instances (one controller to many lights).

AIController: The core intelligent controller of the system. It maintains attributes such as modelVersion, a list of realTimeData, and optimization parameters for its algorithm. Core

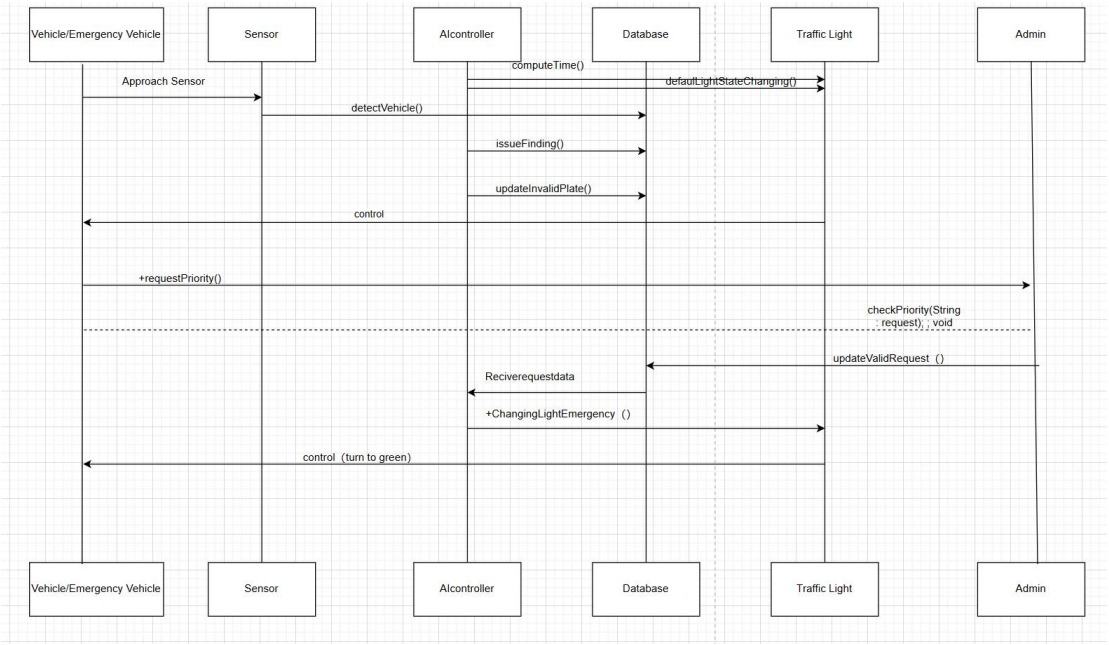
methods include computeTime(sensorList, time) to calculate optimal signal durations, defaultLightStateChanging() for normal operation, issueFinding() to detect anomalies, checkPriority(priorityLevel) to compute which emergency car is more important(when meet a lot of emergency car)and give the correct emergency control, updateInvalidPlate(carPlate) for logging invalid plates, and ChangingLightEmergency() to switch lights for emergencies. The AIController reads from and writes to the Database to log traffic data, vehicle information, and priority requests.

Database: Stores all system data. It provides methods storeSensorData(List<Sensor> data), storeInvalidPlate(String plate), storeEachLightState(TrafficLightState state), and storePriorityRequest(String carPlate). The AIController, Sensors, and Admin use the Database to store and retrieve information. For example, when updateValidRequest() is called by an Admin, the authorized emergency request is saved. The Database serves as persistent storage for traffic logs and request records.

Admin: Represents a system administrator or operator interface. Attributes include id and a priority threshold. Key operations are changeLightState(TrafficLightState state) for manual override, receivePriorityRequest() and checkPriority(String request) for handling emergency requests, and updateValidRequest(String request) to authorize valid requests. The Admin interface allows personnel to monitor the system and intervene: authorized admins can change light states or mark priority requests as valid, which updates records in the Database.

3. Data Flow

The data flow, as illustrated by the provided sequence diagram, proceeds as follows:



Vehicle Detection: A vehicle (or emergency vehicle) approaches an intersection and is detected

by the roadside Sensor (event labeled "Approach Sensor"). The sensor’s detectVehicle() method is invoked.

AI Processing: The Sensor sends vehicle data (e.g., speed, position, license plate) to the Database. AI controller read the data from Database to find the illegal car(issueFinding()),if has then use updateInvalidPlate() to update the car plate to Database. And at the same time the AI controller will compute time to find the most time-saving command run default control.

Priority Request: If the detected vehicle is an emergency vehicle, it (or the sensor) invokes the requestPriority() operation on the AIController. This notifies the system of a pending priority request. At this point, the Admin may be involved: the Admin’s checkPriority() method validates the request, and if approved, updateValidRequest() logs the authorization in the Database.

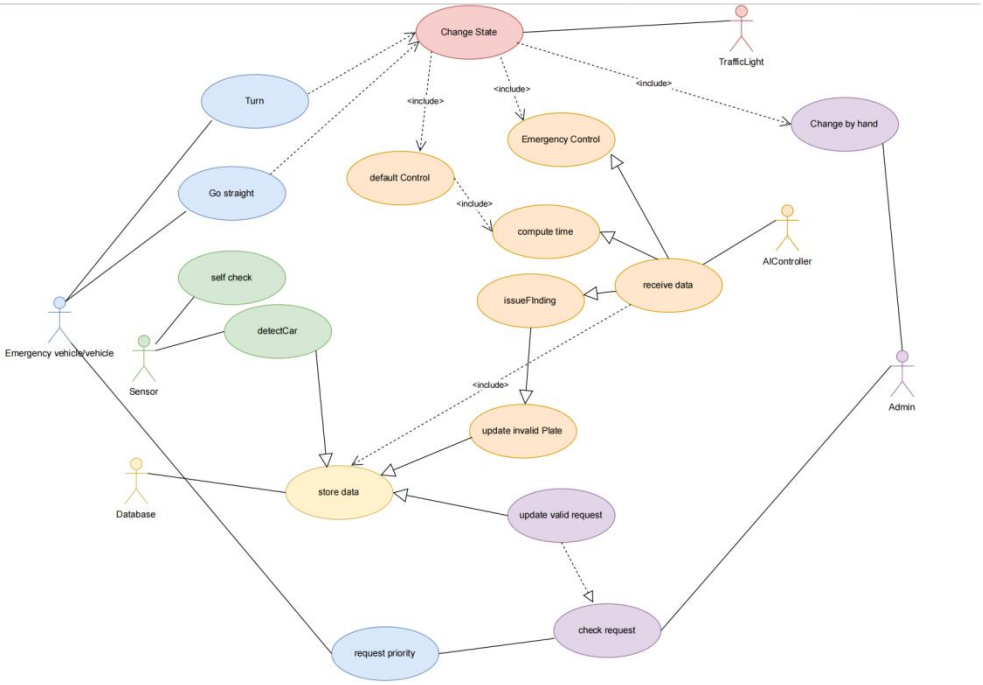
Emergency Handling: Once the priority request is validated (or automatically accepted), the AIController executes ChangingLightEmergency(), signaling the corresponding TrafficLight to switch its state to green for the emergency vehicle’s direction. The TrafficLight then changes its state via changeState() and begins a new countdown (calling countdownSet()).

Traffic Clearance: The TrafficLight turns green on the emergency vehicle’s approach. The AIController continues to monitor the vehicle’s passage (via sensor feedback) and, when the emergency event is complete, reverts control to the normal traffic cycle.

Data Recording: Throughout this sequence, the Database is updated with sensor readings, vehicle info, signal states, and any priority requests. Each component (Sensor, AIController, TrafficLight, Admin) writes relevant data to the Database for persistence.

Fallback and Default Mode: In the absence of an emergency request, the AIController follows its default cycle (defaultLightStateChanging()) and signals the TrafficLight in routine fashion. Admins can at any time manually override signals (changeLightState()) or update request data (updateValidRequest()) to intervene.

4.Use Case



Emergency vehicle / vehicle：

1.Turn
2.Go straight
3.Request priority

Sensor：

1.Self check
2.Detect car

Database：

Store data

AIController：

1. Receive data
2. Compute time
3. Issue Finding
4. Update Invalid car plate
5. Default control
6. Emergency control

Admin：

1.Update valid request
2.Check request
3.Change(Traffic light state) by hand

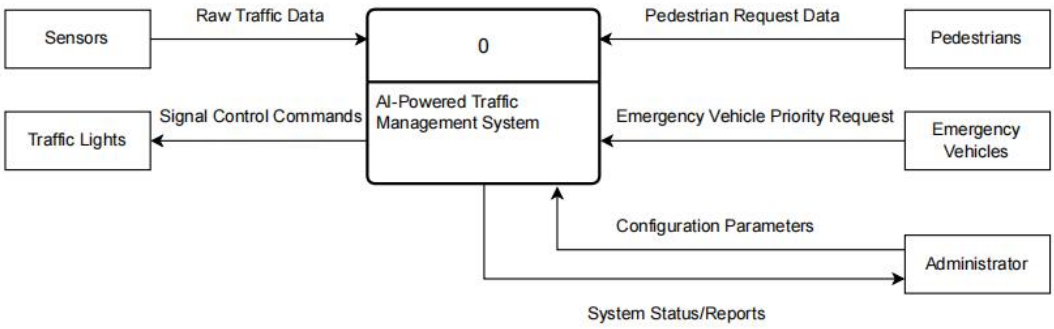
Traffic Light：

Change State

5. Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is used to visualize the flow and processing of data within the system and between the system and its external environment. The DFD for this system is presented two levels: Context Diagram (Level 0) and Level 1 DFD.

5.1 Level 0 DFD (Context Diagram)



Central Process: AI-Powered Traffic Management System. This single process represents the boundary of the entire system, encapsulating its core functionalities and interactions with the external world.
External Entities:
Sensors: Responsible for real-time detection of traffic conditions at intersections, including vehicle presence, type, speed, and pedestrian requests, and inputting this raw data into the system.
Pedestrians: Represent road users who interact with the system, typically via physical devices (e.g., push buttons), to request safe passage across an intersection.
Emergency Vehicles: Specifically refers to vehicles like ambulances, fire trucks, and police cars that require priority passage and can issue priority requests to the system.
Administrator: System operators or traffic engineers responsible for monitoring system status,

adjusting configuration parameters, manually intervening with traffic signals, validating emergency requests, and reviewing system-generated reports.

Traffic Lights: Act as the system's actuators, receiving control commands from the core system and changing their physical state (red, yellow, green) accordingly.

Major Data Flows:

Raw Traffic Data: Flows from Sensors to the AI-Powered Traffic Management System, containing unprocessed vehicle and environmental information.

Pedestrian Request Data: Flows from Pedestrians (via sensors) to the AI-Powered Traffic Management System, indicating a pedestrian's need to cross.

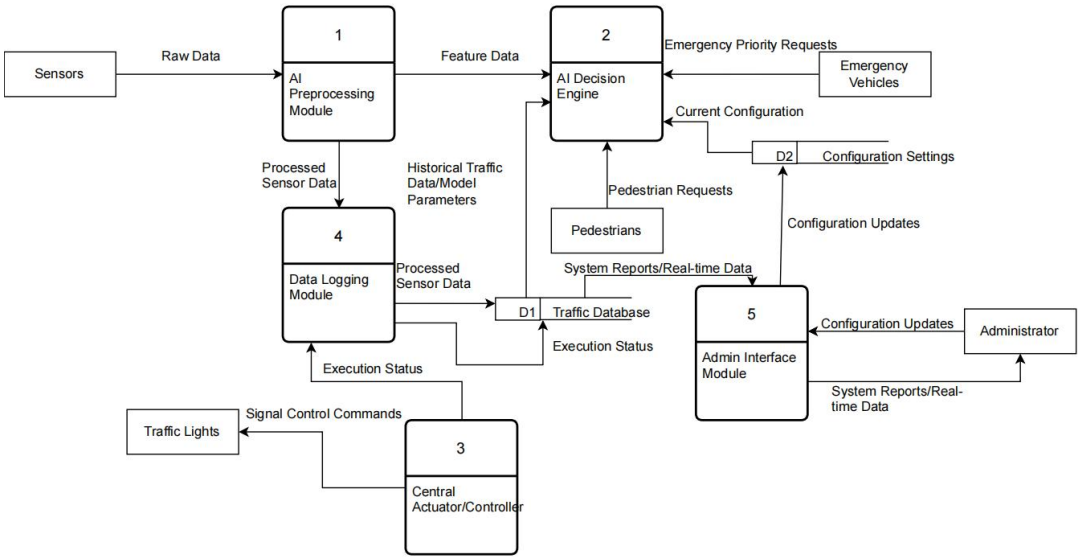
Emergency Vehicle Priority Request: Flows from Emergency Vehicles (possibly via dedicated signals or sensor identification) to the AI-Powered Traffic Management System, requesting right-of-way.

Configuration Parameters: Flows from the Administrator to the AI-Powered Traffic Management System, used to adjust system behavior, such as signal timing strategies and AI model parameters.

Signal Control Commands: Flows from the AI-Powered Traffic Management System to Traffic Lights, instructing them to change states.

System Status/Reports: Flows from the AI-Powered Traffic Management System to the Administrator, providing real-time operational status, performance metrics, and historical data reports.

5.2 Level 1 DFD



This diagram decomposes the Level 0 central process, AI-Powered Traffic Management System, into several major internal processing stages and data stores, illustrating the data transformation and flow within the system in more detail.

Major Processes:

1. **AI Preprocessing Module**: Receives Raw Data from Sensors, performs cleaning, formatting, and initial feature extraction to generate Feature Data.
2. **AI Decision Engine**: The intelligent core of the system. It receives Feature Data, Current Configuration from D2 Configuration Settings, Historical Traffic Data/Model Parameters from D1

Traffic Database, and requests from external entities Emergency Vehicles and Pedestrians. Based on these inputs, it runs AI algorithms to generate Signal Control Commands and may update information about illegal license plates in the database.

3. **Central Actuator/Controller**: Receives Signal Control Commands from 2. AI Decision Engine and translates them into specific operational commands for the external entity Traffic Lights. It also sends Execution Status to the data logging module.

4. **Data Logging Module**: Responsible for collecting operational data from various parts of the system (e.g., Processed Sensor Data from 1. AI Preprocessing Module, decision logs from 2. AI Decision Engine, Execution Status from 3. Central Actuator/Controller) and storing this data into D1 Traffic Database.

5. **Admin Interface Module**: Serves as the bridge for interaction between the Administrator and the system. It receives Configuration Updates from the Administrator to update D2 Configuration Settings or D1 Traffic Database, and extracts data from D1 Traffic Database to generate System Reports/Real-time Data for the Administrator.

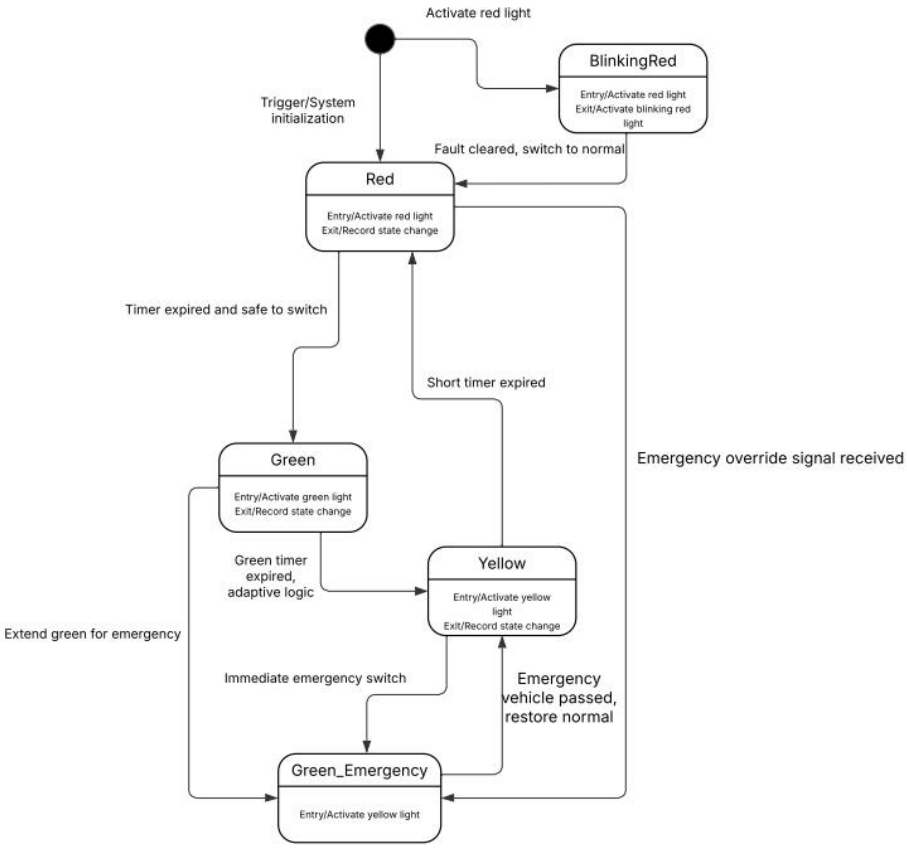
Data Stores:

D1 Traffic Database: The system's central data repository for persisting various types of data, including processed sensor data, traffic flow statistics, vehicle information (license plates, types, behavior), traffic light state history, emergency request logs, administrator action logs, system event logs, and historical pattern data AI models.

D2 Configuration Settings: Stores configurable parameters of the system, such as AI model version numbers, optimization algorithm parameters, default signal timing plans, and emergency event handling rules.

6. State Diagram

A State Diagram is used to describe the various states an object or component may go through during its lifecycle, as well as the events and conditions that trigger transitions between these states.



6.1 Described Object: Traffic Light

This state diagram details the various operational states of a physical traffic light under the control of the AI system, responding to real-time traffic conditions and specific events like emergency vehicle requests.

Major States:

Red: Instructs vehicles to stop. Upon entering this state, the system activates the physical red light; upon exiting, it may log the duration or related events.

Green: Instructs vehicles that they may proceed. Activates the green light upon entry; logs state information upon exit.

Yellow: A warning state preceding the red light, prompting drivers to prepare to stop. Activates the yellow light upon entry.

Green_Emergency: A special green light state triggered in response to a validated emergency vehicle request, ensuring priority passage for the emergency vehicle by forcing the signal for the corresponding direction to green.

BlinkingRed: A fail-safe or special warning mode. For instance, in case of a system malfunction or during low-traffic night hours, the signal might enter this state, prompting vehicles from all directions to proceed with caution.

Major Transitions and Triggering Conditions:

System initialization or a specific trigger (Trigger/System initialization) places the traffic light into

an initial Red state.

From the **Red state**, when a preset red light timer expires and the system deems it safe to switch (Timer expired and safe to switch), it transitions to the Green state.

From the **Green state**, when the green light duration (dynamically calculated by AI), or when adaptive logic determines the green phase should end (Green timer expired, adaptive logic), it transitions to the Yellow state.

From the **Yellow state**, after a short yellow warning period ends (Short timer expired), it transitions to the Red state, completing a normal signal cycle.

Upon receiving a valid emergency vehicle priority request signal (Emergency override signal received or Immediate emergency switch), the system can transition from Red, Green, or Yellow states directly to the Green_Emergency state to clear the path for the emergency vehicle. If transitioning from Green, it might also manifest as extending the current green time (Extend green for emergency).

After the emergency vehicle has safely passed the intersection and the system signals to resume normal control (Emergency vehicle passed, restore normal), the traffic light transitions from the Green_Emergency state back to Yellow (then to red following normal logic) or directly integrates into the next normal signal cycle.

During system startup or upon detecting a critical, non-recoverable fault, the traffic light might first activate the red light (Activate red light) and then enter the BlinkingRed state as a safe-degraded mode.

When a system fault is cleared and the system is ready to switch back to normal automatic control (Fault cleared, switch to normal), the traffic light transitions from the BlinkingRed state back to the Red state, awaiting the next normal signal cycle.

6.2 (Suggested Addition) Described Object: AI Controller's Operational Modes

As the intelligent core of the system, the AI Controller also operates through different internal states or modes to handle various inputs and tasks.

Conceptual States:

Idle: The AI Controller is in this state when there are no new processing tasks or data inputs, awaiting activation.

DataProcessing: Entered upon receiving new data from the sensor network. In this state, the controller cleans, preprocesses, and extracts features from the raw data, identifying vehicle types (including emergency vehicles), pedestrian requests, etc.

NormalOperationAnalysis: In the absence of emergency events, the controller analyzes the current traffic situation based on processed traffic data and built-in AI models (e.g., traffic flow prediction models, reinforcement learning models), evaluating the potential effects of different signal timing plans.

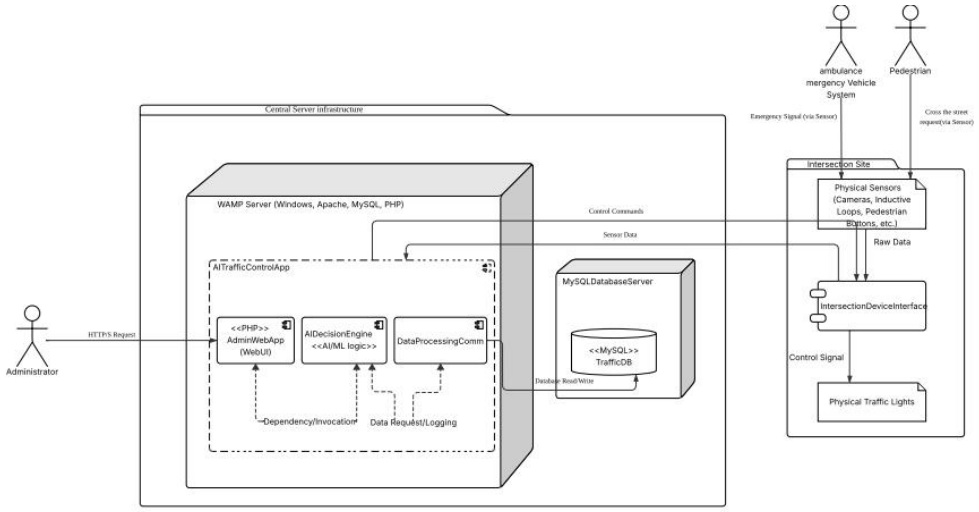
DecisionMakingAndControl: Based on the analysis results, the AI Controller selects the optimal signal timing strategy and generates specific control commands to be sent to the traffic light actuators.

EmergencyPriorityHandling: Upon confirming a valid priority request from an emergency vehicle (possibly after administrator validation), the AI Controller immediately enters this state. It interrupts or adjusts the current normal signal timing, forces the signal for the emergency vehicle's path to green, and coordinates relevant signals to ensure its rapid and safe passage.

IllegalBehaviorDetection: As per the functional requirements (e.g., issueFinding() and updateInvalidPlate()), if the AI Controller detects illegal vehicle behavior (e.g., red-light running, illegal lane changes, or license plate anomalies) during data processing, it may enter this sub-state or trigger a corresponding process, such as logging the information to the database.
(Optional) **SystemMonitoring/SelfCheck**: The AI Controller might periodically or under specific conditions enter this state to check its own health status and that of associated sensors, actuators, and other components to ensure stable system operation.

Conceptual Transitions:
From the **Idle** state, upon arrival of a new sensor data packet, transitions to DataProcessing.
After data preprocessing is complete, if there are no emergency events, transitions to NormalOperationAnalysis.
After traffic situation analysis is complete, transitions to DecisionMakingAndControl to generate and issue commands.
After control commands are successfully issued, the system may return to Idle to await the next cycle or continue monitoring and analysis based on configuration.
In any applicable state (DataProcessing, NormalOperationAnalysis, DecisionMakingAndControl, etc.), if a priority request from an emergency vehicle is received and validated, the system immediately transitions to EmergencyPriorityHandling.
After the emergency event is handled, the system transitions from EmergencyPriorityHandling back to the previous normal operational flow or to the Idle state.
During **DataProcessing** or **NormalOperationAnalysis**, if illegal vehicle behavior is identified, it may trigger entry into an IllegalBehaviorDetection sub-process or state, returning to the main flow after completion.

7. Component Diagram



The Component Diagram provides a modular view of the system's architecture, showing the main software and hardware building blocks, their responsibilities, and how they interact and collaborate through interfaces and dependencies.

Major Deployment Areas/Packages:

Intersection Site: This package encapsulates all hardware devices deployed at a physical intersection and related local interface software components.

Central Server Infrastructure: This package represents the system's backend processing center, containing servers that run core application logic and the database system.

Major Nodes (Deployment Environments):

WAMP Server (Windows, Apache, MySQL, PHP): This is a physical or virtual server node deployed within the Central Server Infrastructure. It is responsible for hosting and running the core AITrafficControlApp, including its web application part (based on Apache and PHP) and AI logic processing part.

MySQLDatabaseServer: Also deployed within the Central Server Infrastructure, this is a dedicated database server node for running and managing the TrafficDB.

Major Components and their Responsibilities:

Located at the Intersection Site:

Physical Sensors (Cameras, Inductive Loops, Pedestrian Buttons, etc.) (Physical Artifact): These are the actual physical sensing devices deployed at the intersection, responsible for capturing raw traffic data (e.g., vehicle images, inductive loop signals) and pedestrian crossing request signals in real-time. They are the source of the system's data input.

IntersectionDeviceInterface (IDI) (Component): This is a crucial hardware/software interface component deployed at the intersection. It acts as a bridge between the physical sensors/traffic lights and the central server application. Its primary responsibilities include: aggregating raw data from multiple Physical Sensors, possibly performing initial formatting or aggregation, and then transmitting it over a network to the central server; concurrently, it receives control commands from the central server and translates them into specific operational signals for the Physical Traffic Lights.

Physical Traffic Lights (Physical Artifact): This is the physical hardware device that actually displays red, yellow, and green signals at the intersection, serving as the final executor of the system's control commands.

Located on the WAMP Server Node (as part of AITrafficControlApp):

AITrafficControlApp (Main Application Component): This is the core software application deployed on the WAMP server. It is an encompassing entity that encapsulates the system's main business logic, user interaction interface, and intelligent decision-making capabilities. It is internally composed of the following key sub-components:

AdminWebApp (WebUI) (Internal Component, stereotyped <<PHP>>): This is a web application module developed using PHP and running on an Apache web server. It provides a graphical user interface for the system Administrator to monitor real-time traffic conditions, view system-generated reports, adjust system configuration parameters, manually intervene in traffic signal control, and (as per document description) validate and process priority requests from

emergency vehicles.

AIDecisionEngine (Internal Component, stereotyped <<AI/ML logic>>): This is the intelligent core and "brain" of the system. It is responsible for processing real-time and historical traffic data obtained from sensors (via IDI and DPC modules), running advanced AI and machine learning models (e.g., potentially using YOLO for vehicle detection and classification), dynamically analyzing traffic flow patterns, predicting traffic congestion, and accordingly generating optimal signal timing strategies. Furthermore, it implements the priority passage logic for emergency vehicles (e.g., the ChangingLightEmergency() method mentioned in the document) and initial identification of illegal vehicle behavior (e.g., issueFinding()).

DataProcessingComm (Internal Component): This module serves as a crucial intermediary layer and data hub between other internal modules of AITrafficControlApp (like AdminWebApp and AIDecisionEngine) and external resources (like IntersectionDeviceInterface and TrafficDB). Its primary responsibilities include: parsing raw or preliminarily processed sensor data incoming from the IDI and transforming it into a format usable by the AI engine; formatting control commands generated by the AI engine for transmission to physical traffic lights via the IDI; and encapsulating all database interaction operations with TrafficDB (such as executing SQL queries, storing data, updating records), providing a unified data access interface for upper-layer modules.

Located on the MySQLDatabaseServer Node:

TrafficDB (Database Artifact, stereotyped <<MySQL>>): The Traffic Management Database. It utilizes the MySQL relational database management system for data storage and is responsible for persisting all system-related data. This includes, but is not limited to: real-time and historical traffic flow data, detected vehicle information (e.g., license plate numbers, vehicle types, speeds, GPS locations), pedestrian request records, emergency vehicle priority requests and their processing status, administrator action logs, system configuration parameters, traffic light state change history, and historical data samples required for AI model training.

Major Actors:

Administrator: Represents the system operators or traffic engineers who interact with the system via the AdminWebApp (WebUI) to perform monitoring, configuration management, manual control, and emergency response functions.

Emergency Vehicle System: Represents the emergency vehicles themselves or their equipped signal-transmitting devices. It acts as an external entity interacting with the system via Physical Sensors (or directly with IDI) to request priority passage.

Pedestrian: Represents pedestrians wishing to cross the intersection safely. They interact with the system via Physical Sensors (such as pedestrian push buttons) installed at the intersection.

Key Dependencies and Data Flows:

The Administrator interacts with the AdminWebApp (WebUI) deployed on the WAMP server via secure HTTPS Requests to access management functionalities.

Service requests from the Emergency Vehicle System (emergency signals) and Pedestrian (crossing requests) are captured by Physical Sensors through their respective physical interaction methods (e.g., dedicated transmitters, push buttons) and are passed as data input to the IntersectionDeviceInterface (IDI).

Physical Sensors provide collected Raw Data to the IDI for initial processing or aggregation.

The **IDI** transmits processed (or directly forwarded) Sensor Data over the network to the AITrafficControlApp on the central server (specifically received and further processed by its internal DataProcessingComm module).

The **AITrafficControlApp** (after a decision is made by the AIDecisionEngine, via its internal DataProcessingComm module) sends generated Control Commands over the network back to the IDI at the intersection

The **IDI**, upon receiving control commands, translates them into specific Control Signals that drive the Physical Traffic Lights to change their displayed state.

Within AITrafficControlApp, there are clear dependencies and invocation relationships between its internal modules:

AdminWebApp (WebUI) depends on and invokes (stereotyped Dependency/Invocation) services удовольствие the AIDecisionEngine, for example, to request AI analysis results of the current traffic situation or to trigger specific AI optimization operations.

AdminWebApp (WebUI) depends on the DataProcessingComm module (stereotyped Data Request/Logging) to obtain data required for display to the administrator (such as statistical reports queried from the database, real-time logs) or to submit configuration changes initiated by the administrator for persistent storage.

AIDecisionEngine depends on the DataProcessingComm module (stereotyped Data Request/Logging) to obtain preprocessed and formatted input data (originating from sensors) and to store its generated decision results, model parameters, or related event logs into the database.

The **AITrafficControlApp** (through its core data interaction logic, primarily implemented by the DataProcessingComm module) performs Database Read/Write operations with the TrafficDB deployed on the MySQLDatabaseServer for persistent data storage and retrieval.

8.Implementation

5.1 IoT-Based Sensor Layer

Sensor Deployment:

Sensors are deployed at strategic intersections to detect vehicle presence, traffic density, and emergency vehicle approach. These include:

1.Infrared or radar sensors for real-time vehicle detection.

2.RFID or license plate recognition cameras to identify emergency vehicles.

5.2 AI Control Unit

Data Collection and Storage:

The AI controller receives sensor data and stores it in a centralized database. Data includes:

1.Vehicle count and type.

2.Emergency vehicle status.

3.Historical traffic patterns.

Decision-Making Algorithms:
1.The AI module processes incoming data to:
2.Compute optimal green light duration based on real-time traffic.
3.Detect anomalies or unexpected congestion (issueFinding).
4.Prioritize emergency vehicle paths by issuing requestPriority signals.

Car Plate Verification Logic:

A submodule handles:
1.Matching vehicle plates with a validated emergency vehicle list.
2.Updating the database with valid or invalid plate status.
3.Ensuring only authorized emergency vehicles trigger priority changes.

5.3 Traffic Light Control System

State Management:
1.The system supports three operational modes:
2.Default control: AI decides normal light changes.
3.Emergency control: Light state is overridden to clear paths for emergency vehicles.
4.Manual override: Admin can directly change light states if needed.

Communication and Action:
Once a control decision is made:
1.The AI sends commands to the traffic light (e.g., turn green, extend green).
2.Light state is updated in both hardware and system records.

5.4 Admin Interface:
Administrators can:
1.Monitor current system status.
2.Check and validate emergency vehicle requests.
3.Update emergency vehicle data in the database.

9. Testing and Validation

Unit Testing: Test each module in isolation. For example, verify that Sensor.detectVehicle() correctly identifies vehicles under various conditions, TrafficLight.changeState() cycles through states correctly, and the Database methods store and retrieve data as expected.

Integration Testing: Combine modules to test their interactions. For example, simulate a vehicle approaching a sensor and ensure the AIController processes the input, updates the database, and

eventually changes the traffic light. Test the end-to-end emergency request scenario, from detection to traffic light change and database logging.

Simulation of Traffic Scenarios: Use traffic simulation tools or custom scenarios to model realistic situations (e.g., rush-hour congestion, multi-intersection coordination, multiple emergency vehicles). Measure performance metrics such as average waiting time, throughput, and emergency response time to validate system effectiveness under diverse conditions.

Load and Stress Testing: Evaluate performance under high load. Simulate a large number of vehicles and sensor events concurrently to ensure the database and AIController handle the load. Verify that system response times remain within real-time requirements (e.g., sub-second) even with many simultaneous intersections or high traffic density.

System Validation: Check that the system meets all requirements. For example, verify that emergency vehicles always receive priority when requested and that the lights fall back to safe default patterns if the system fails. Perform user acceptance testing with traffic operators to ensure that the administrative interface and manual override functions behave as intended.

10. Deployment and Maintenance

Deployment Steps: Deploying in a real city involves installing sensors at intersections, equipping traffic lights with networked controllers, and setting up a central server for the AIController and database. Infrastructure must support communication between sensors, lights, and the control center (wired or wireless). A pilot phase in a limited area can validate the system before full-scale rollout, allowing calibration of the AI model and adjustment of hardware placement.

Hardware Maintenance: Regular maintenance is needed for sensors and traffic lights. This includes routine inspections, cleaning camera lenses, replacing faulty units, and calibrating sensors. Traffic light controllers and bulbs should be checked to ensure reliable operation. A preventive maintenance schedule can reduce unexpected failures and downtime.

Software and AI Model Maintenance: The AI models should be periodically retrained or updated with new traffic data to adapt to changing patterns. Software updates (bug fixes, enhancements) should be applied in a controlled manner (e.g., during off-peak hours). Regular data backups and integrity checks are necessary to prevent data loss. Monitoring tools can detect anomalies in system behavior and trigger alerts for manual intervention.

City Integration: Coordinate with city traffic management centers to integrate the new system with existing infrastructure. Ensure legal compliance (e.g., privacy protection for license plate data, adherence to emergency vehicle protocols). Train operators on the new system and establish procedures for emergencies and system overrides.

Fault Tolerance: Implement fallback strategies (for example, default fixed-time schedules if the AI system fails). The system should automatically revert to safe default signal cycles during power outages or communication loss. Using redundant components (backup controllers, spare sensors, backup power) can improve system resilience and reliability.