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. Artificial Intelligence-Based Decision Making
At the core of the system lies the AIDecisionEngine, which processes real-time and historical data to make intelligent decisions on signal timing. Techniques such as machine learning (e.g.,

classification models for vehicle types, rule-based decision trees) are employed to optimize traffic light durations, detect anomalies, and handle emergency scenarios

2. System Architecture

2.1 System Layer Architecture

1. Presentation Layer
Function: Provides user interfaces for system monitoring and control.
Components:
AdminWebApp (PHP-based web application)

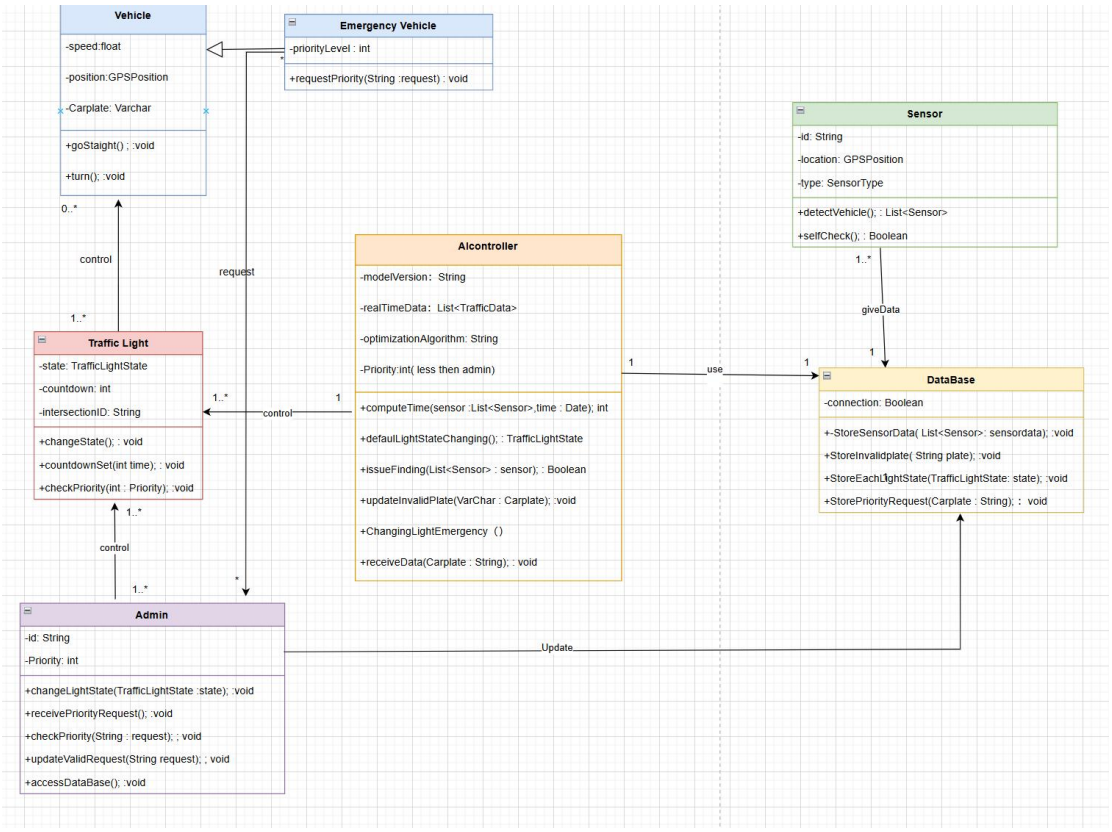
Present reports and real-time data visualization.

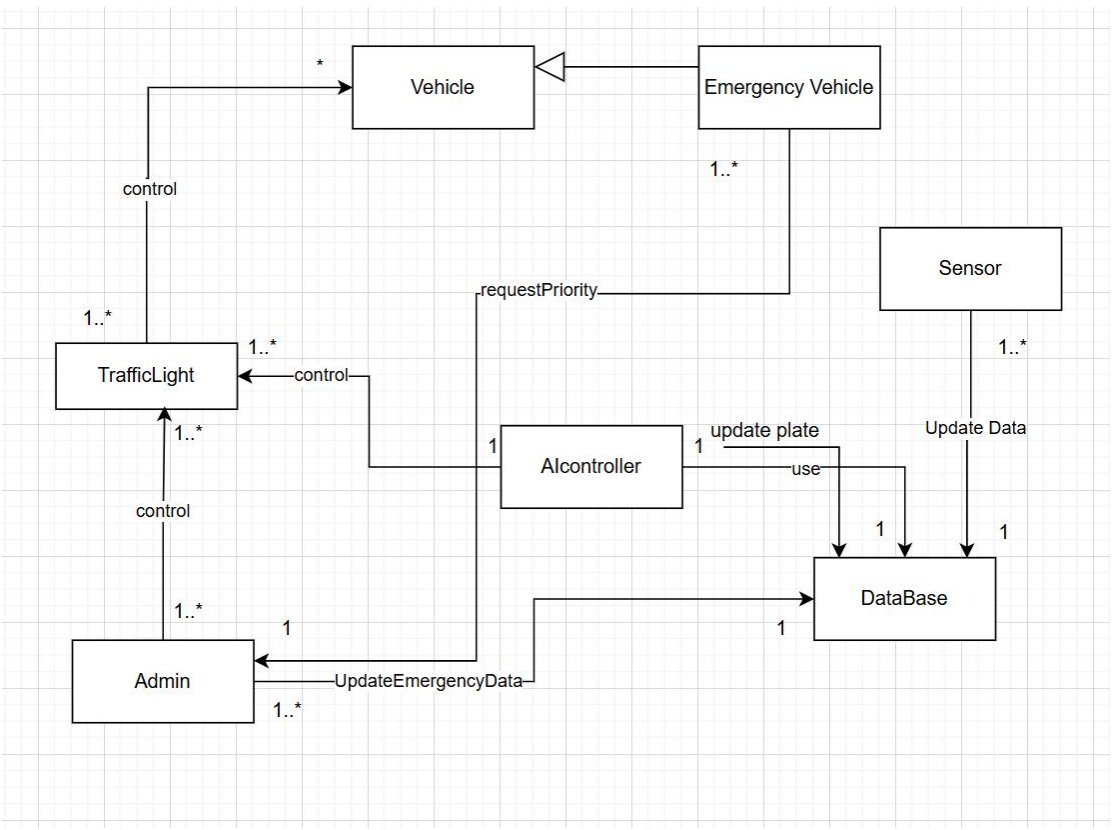
2. Business Logic Layer (AI Decision Layer)
Function: Implements core decision-making logic for traffic control.

Core methods: computeTime(), issueFinding(), checkPriority(), ChangingLightEmergency()

3. Data Access Layer
Function: Provides a unified interface for querying and updating data from the database.

2.2





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.

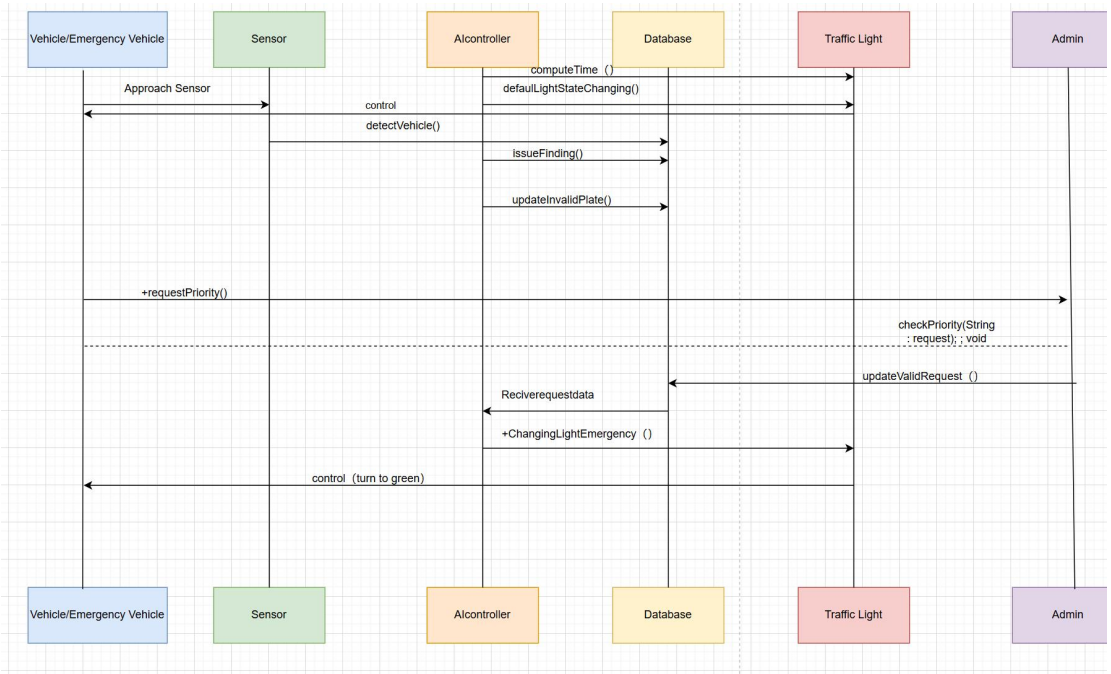
Database: Stores all system data. It provides methods storeSensorData(List<Sensor> data), storeInvalidPlate(String plate), storeEachLightState(TrafficLightState state), and storePriorityRequest(String carPlate). The AIController, Sensors, and Admin use the Database to store and retrieve information. 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.Admin can

access database by +accessDataBase();.

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()).

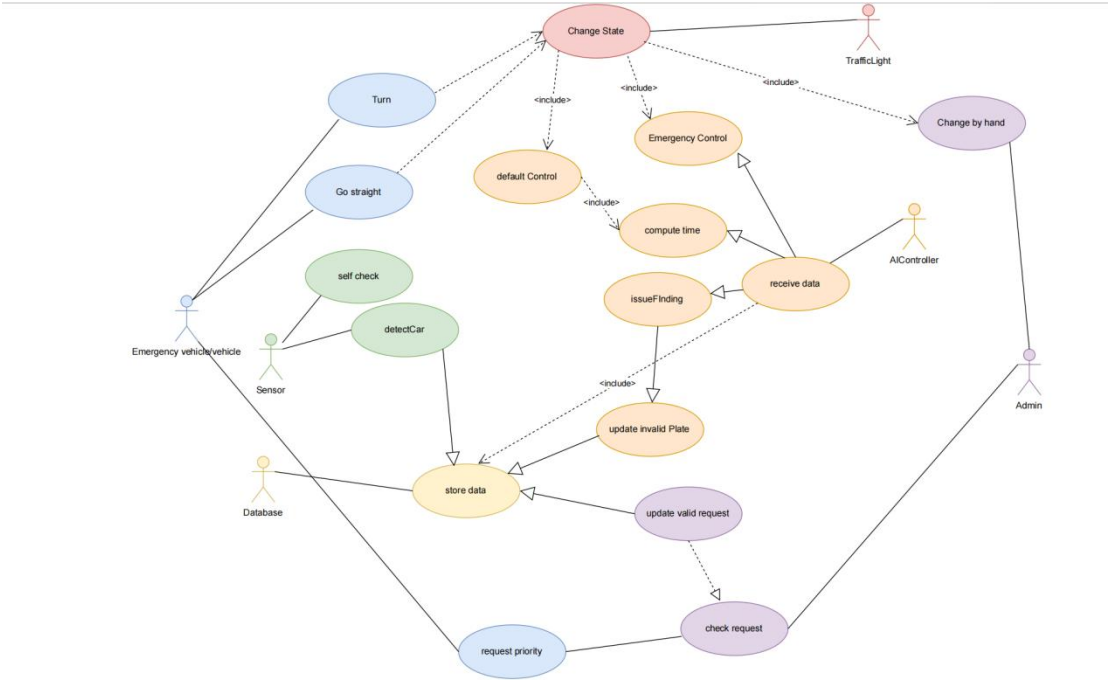
Traffic Clearance: The TrafficLight turns green on the emergency vehicle’s approach. The

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

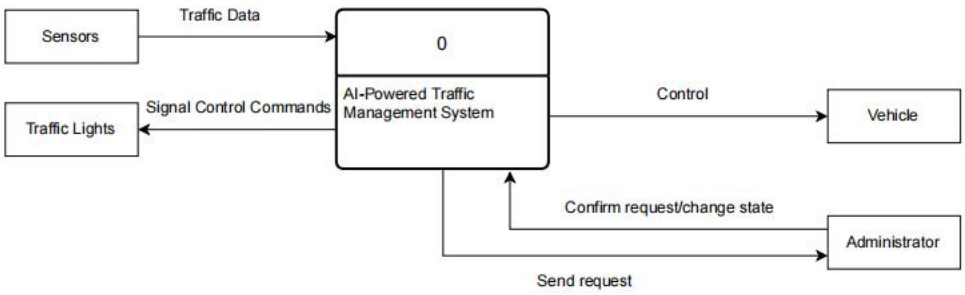
Traffic Light：

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)



The Level 0 DFD offers a high-level overview of the entire system. It treats the AI-Powered Traffic

Management System as a single process and illustrates its interactions with external entities. The

Vehicle: Sends real-time traffic information to the system through roadside sensors.

Admin: Receives emergency priority requests and can authorize control actions.

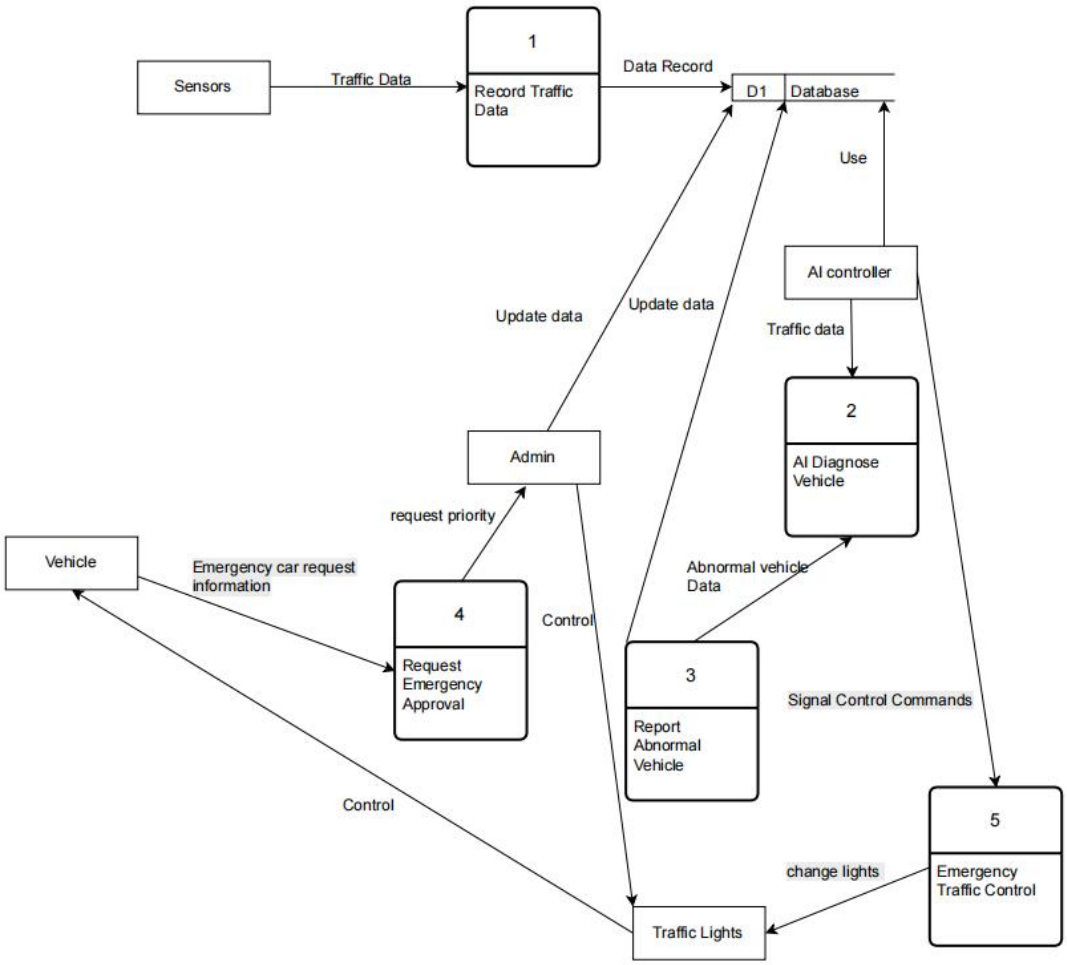
Main data flows include:

Vehicle data transmitted to the system via sensors.

Emergency requests flowing from vehicles and processed by the admin.

Signal control commands dispatched to the traffic light infrastructure.

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.

Internal processing stages

1. Record Traffic Data
   Input: Traffic data from sensors (e.g., license plate, speed, direction).
   Process: Gathers and formats vehicle data.
   Output: Data stored in the central Database (D1) and passed to the AI module for further analysis.

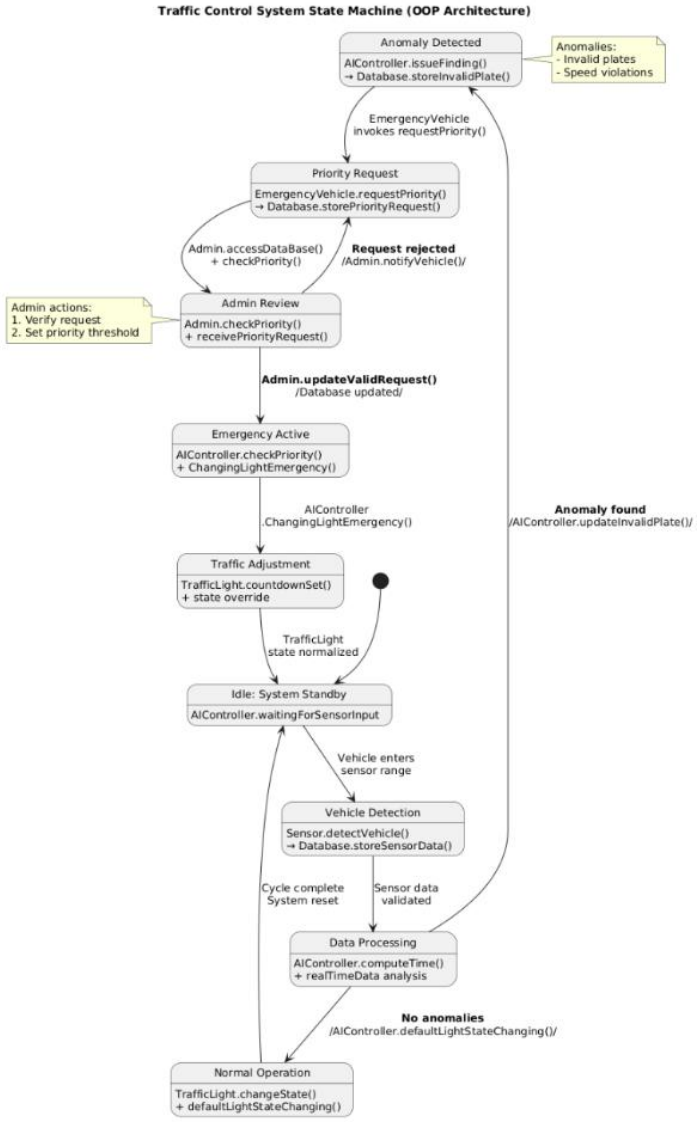
2. AI Diagnose Vehicle
   Input: Real-time traffic data from the database.
   Process: Uses AI models to identify anomalies, illegal behavior, or emergency vehicle signals.
   Output: Results sent to the abnormal reporting unit.

3. Report Abnormal Vehicle
   Input: Suspected vehicle data (e.g., invalid plate, illegal behavior).
   Process: Logs incident in the abnormal vehicle database.
   Output: Records are saved for administrative punish or educate.

4. Request Emergency Approval
   Input: Emergency vehicle request and request data.
   Process: Admin verifies request authenticity.
   Output: If valid, admin will update the vehicle status in the database and triggers emergency procedures.

5. Emergency Traffic Control
   Input: Verified emergency requests and AI control signals.
   Process: Executes emergency traffic light overrides.
   Output: Signals sent to the physical Traffic Light infrastructure.

6.State Diagram



Core Workflow
Idle State

System awaits sensor input (AIController.waitingForSensorInput)

Vehicle Detection

Sensors detect vehicles (Sensor.detectVehicle()) and log data (Database.storeSensorData())

AI Decision-Making

AIController.computeTime() analyzes traffic data

Normal Flow: Adjusts lights via TrafficLight.changeState()

Anomaly Detected: Flags violations (AIController.issueFinding())

Emergency Handling
Emergency vehicles request priority (EmergencyVehicle.requestPriority())

Admin manually verifies requests (Admin.checkPriority())

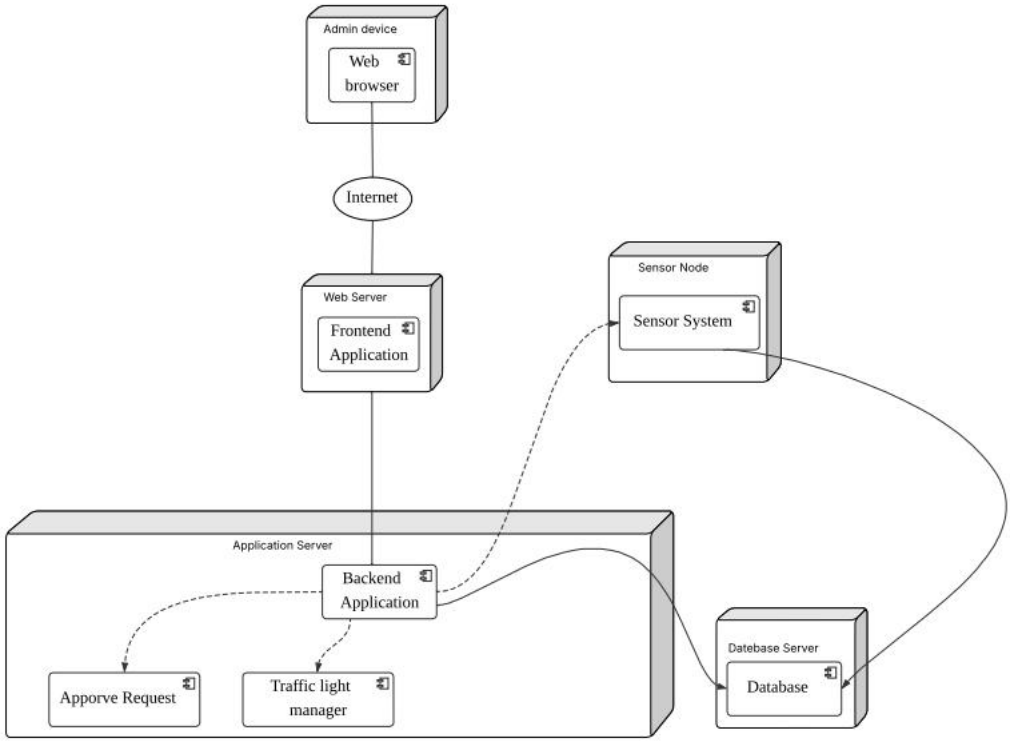
Approved requests trigger emergency protocols (AIController.ChangingLightEmergency())

Key Features
Database integration for all operations (storeInvalidPlate(), storePriorityRequest())

Admin override capabilities (Admin.updateValidRequest())

Self-resetting cycle (returns to Idle after completion)

7.Deployment Diagram



The deployment diagram illustrates the physical architecture of the AI-powered traffic management system. It outlines how software components are distributed across different hardware nodes and how these nodes communicate to achieve the system's functions. This structural view ensures clarity in deployment planning and guides the implementation of

real-world infrastructure.

The system adopts a layered deployment design composed of five key physical nodes:

1.**Admin Device**

Hosts the web browser interface used by system administrators.

Connects to the web server via the Internet to access the frontend application.

2.**Web Server**

Deploys the Frontend Application, providing a user interface for admin interactions such as emergency request handling, manual overrides, and system monitoring.

3.**Application Server**

Contains critical modules:

Approve Request: Handles admin authorization of emergency vehicle priority requests.

Traffic Light Manager: Issues commands to control traffic lights in both default and emergency modes.

4.**Sensor Node**

Deployed at roadside intersections, runs the Sensor System which detects approaching vehicles and sends traffic data to the backend.

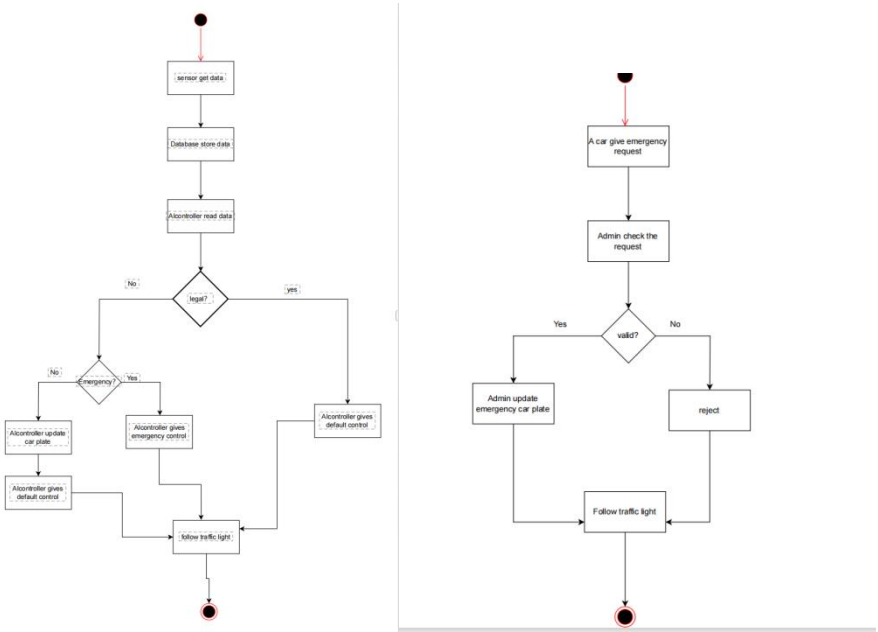
Communicates directly with both the Application Server and Database Server.

5.**Database Server**

Stores all collected and processed data including sensor inputs, AI-diagnosed anomalies, approved emergency requests, and traffic light states.

Serves as the central data source for real-time AI decisions and post-event analysis.

8.Activity Diagram



Traffic Control System Workflow

1. Emergency Request Handling

Request Initiation

A vehicle submits an emergency request

Admin Verification

Administrator reviews the request

Decision branch:

Approved: Vehicle added to emergency priority list

Rejected: Request discarded

System Response

Emergency-approved vehicles receive priority routing

Speed adjustments and traffic light overrides implemented

2. Sensor-Based Traffic Management
Data Collection

Sensors detect vehicle data → stored in Database

AI Analysis

AIController evaluates:

Legal compliance (e.g., valid plates)

Emergency status

Control Actions

Standard vehicles: Default traffic control applied

Emergency vehicles:

License plate logged (updateCarPlate)

Priority routing activated (emergency control)

Traffic Light Coordination

All vehicles ultimately follow adjusted traffic signals

Key System Features
Dual verification: Automated (AI) + manual (Admin) checks

Dynamic adaptation: Real-time light adjustments for emergencies

Data-driven: Sensor inputs trigger all control decisions

9.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.

10. 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.

11. 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.