Tara Can
Pennsylvania State University
SWENG 837
Spring II 2024, Professor Nalubandhu

Smart City Traffic Management System

Problem Statement: Develop a system that uses IoT sensors and AI algorithms to optimize traffic flow, reduce congestion, and improve public transportation.

Table of Contents

Problem Statement and Requirements

A city needs to create a smart traffic system (STS) to manage traffic flow, increase public safety, and be highly modifiable for periodic temporary events.

The STS will comply with all federal and state regulations for traffic signals and displays.

Problems and Business Goals

1. **Roadway safety.**
   Business Goal: It will improve roadway safety by performing critical and repetitive logic calculations with ease.
2. **Accident investigations.**
   Business Goal: It will eliminate bias during vehicular accident investigations using video recordings and motion sensors to analyze events that led up to the incident.
3. **Inefficient use of manpower.**
   Business Goal: It will decrease the manpower needed to modify routes for temporary events (such as short term construction events or seasonal parades).
4. **Emergency vehicle routing.**
   Business Goal: It will improve upon the current systems for responding to the highly variable and unpredictable need for emergency vehicles (such as ambulances) to get through busy intersections.
5. **Traffic patterns.**
   Business Goal: It will respond to the changing way that traffic systems work with smart cars, trains and buses to improve traffic patterns and reduce congestion during busy times of day.

Functionalities

The following requirements are broken up into two categories, the business requirements and the non functional requirements.

Business Requirements

- The system shall manage traffic flow dynamically at different times of the day
- The system shall report accidents to emergency system operators and databases
- The system shall adhere to all federal regulations as released by the Department of Transportation
- The system shall be updated in the event of hazardous weather conditions
- The system shall be updated for local temporary events

- The system shall have manual overrides to implement in the event of an IoT service failure.
- The system shall function using standard traffic symbols which are compliant with federal and state regulations
- The system shall have an accessible entrypoint for city officials and law enforcement
- The system shall be able to interact with various smart car operating systems
- The system shall alert traffic engineers about skewed data that may represent dangerous intersections
- The system shall run on a Linux operating system.
- The system shall use two factor authentication for employees to gain access.
- The system shall encrypt all recordings before they are stored in the database.

Non-Functional Requirements

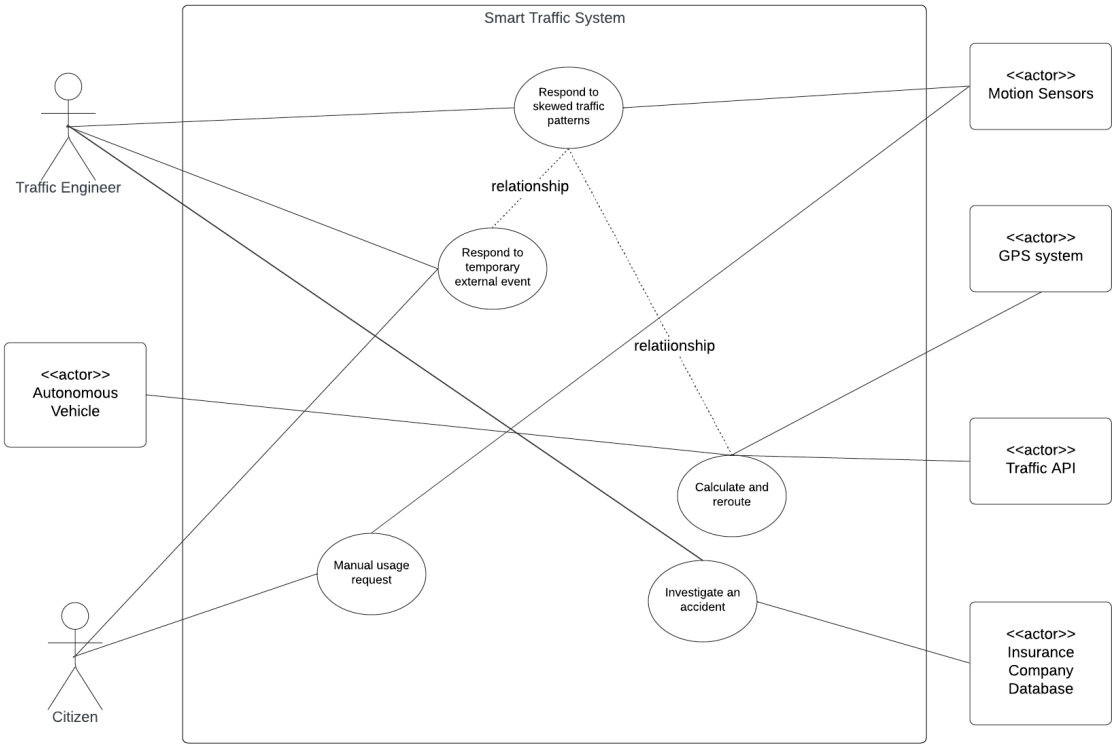
- The system shall consume no more than 5% of the city's energy grid.
- The system shall cost no more than 15% of the city's annual operating budget.
- The system shall have a user interface which employee can interact after less than 5 minutes of exploring its features.
- The system shall display traffic updates on all affected signs within 5 milliseconds of their being approved.
- The system shall have a backup power source that lasts for 60 minutes in the case of a temporary power failure.
- The system shall have a response increase of no more than 500 milliseconds per 10% increase in additional smart vehicles on the road.
- The system shall use modular coding practice.

Target users

1. **Traffic engineers** will use the system regularly responding to system alerts and external requests.
2. **Motorists** will use the system passively and be recipients of the system's broadcasts and traffic displays.
3. **Autonomous vehicles** will interact with the system in dynamic ways.
4. **City officials** will be able to submit plans for infrastructure updates that may affect roadway conditions.
5. **Emergency vehicles** will interact with the system when responding to a sudden event that requires immediate traffic routing demands.
6. **Traffic APIs** will interact with the system using its broadcasted messages and responding to its data requests.
7. **Insurance companies** will be able to make data requests in response to events that result in insurance claims.

System Design using Domain Modeling

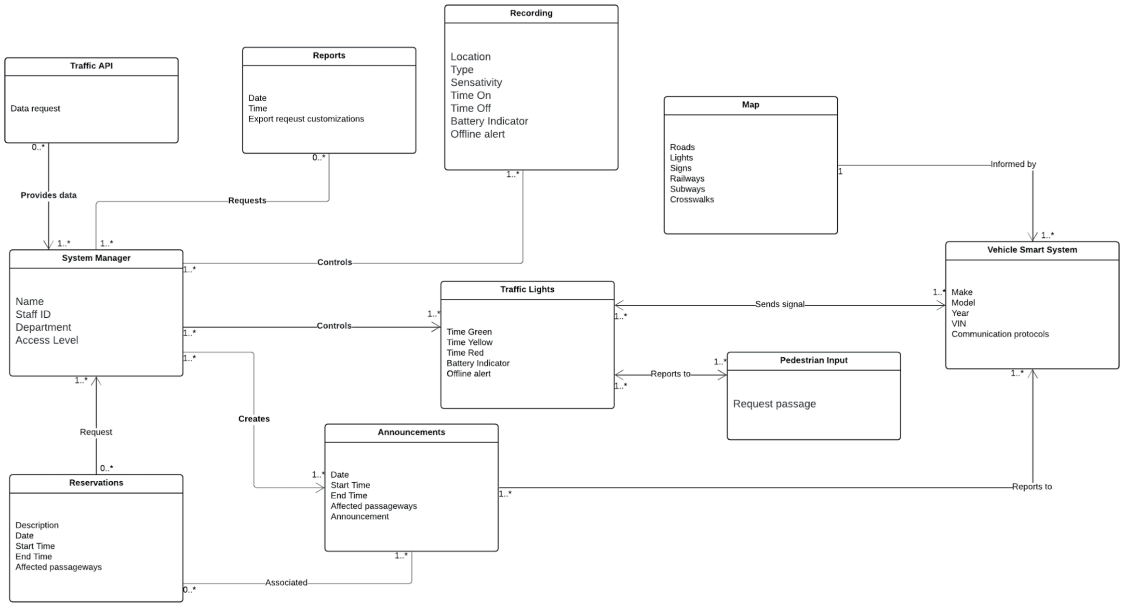
UML Use Case Diagram



Explanation: There are three main actors in the system, the traffic engineers, the autonomous vehicle systems and the city's citizens. These actors will work directly with the STS and be central to the listed use case scenarios.

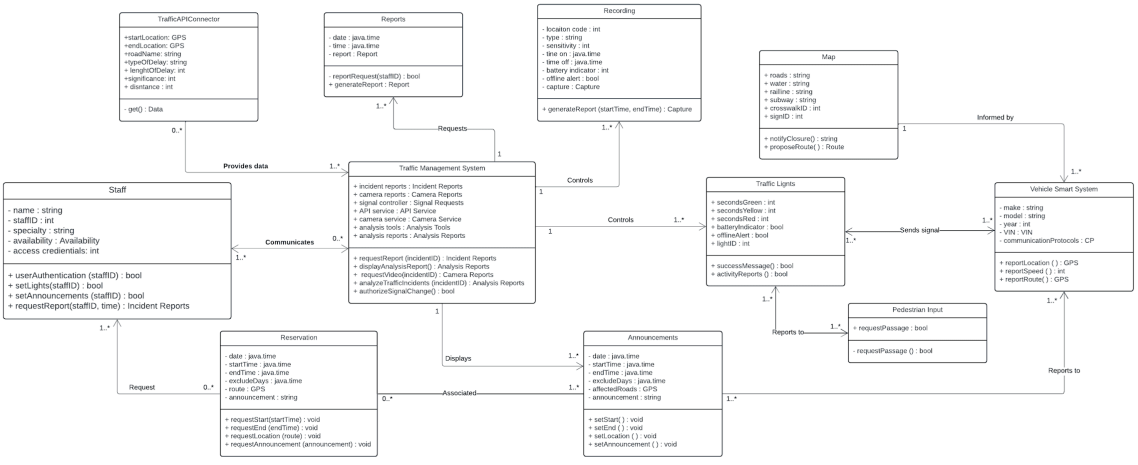
Supporting actors include the IoT devices such as motion sensors and camera recordings, external GPS systems, external traffic APIs and insurance company databases. These actors provide a service and assist in achieving the goals of the use cases.

UML Domain Model



Explanation: The domain model shows the key entities and their relationships within the problem domain. These entities were pruned from a larger list, eliminating any redundancies, irrelevancies, ambiguities and classes that are implementation specific. The multiplicities, associations and attributes are included in the domain model.

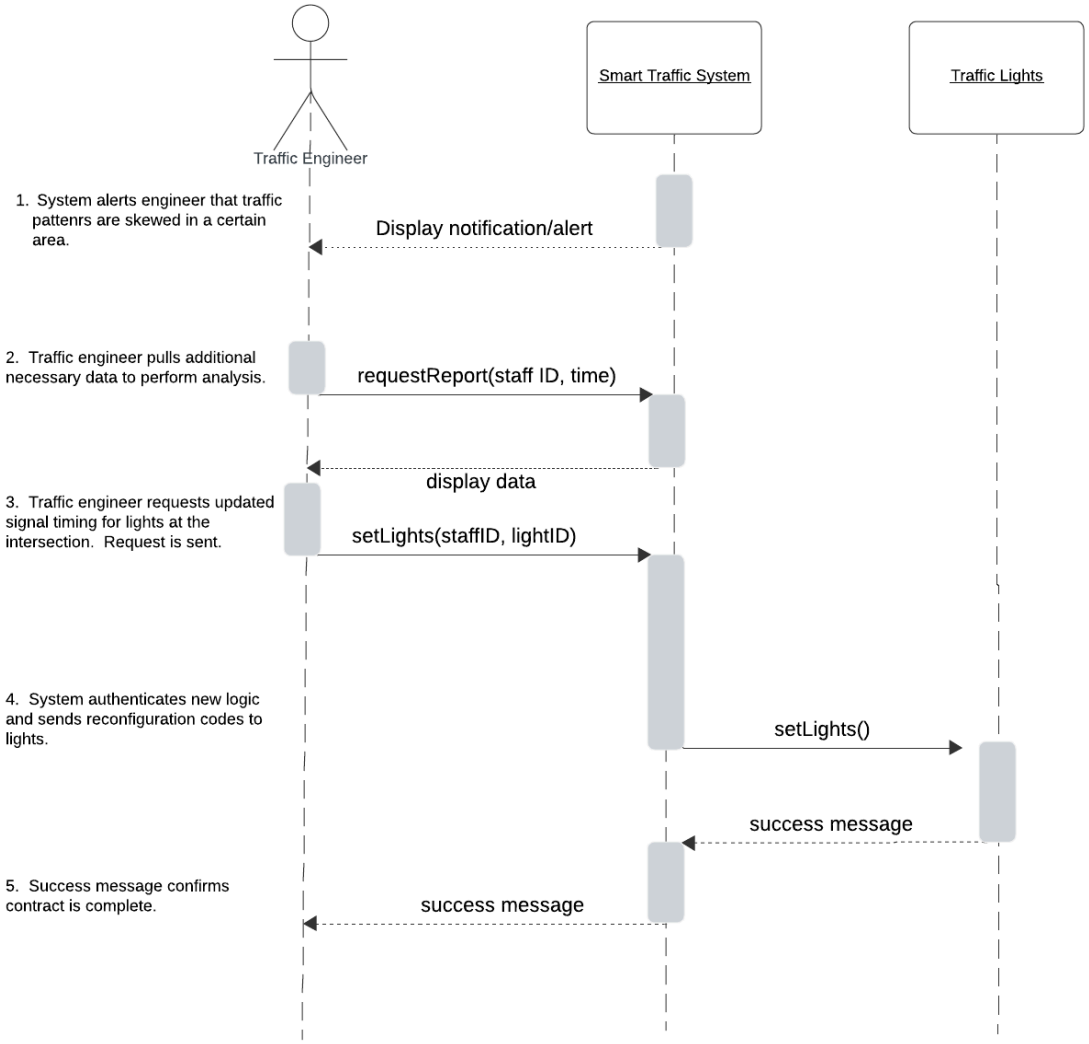
UML Class Diagram



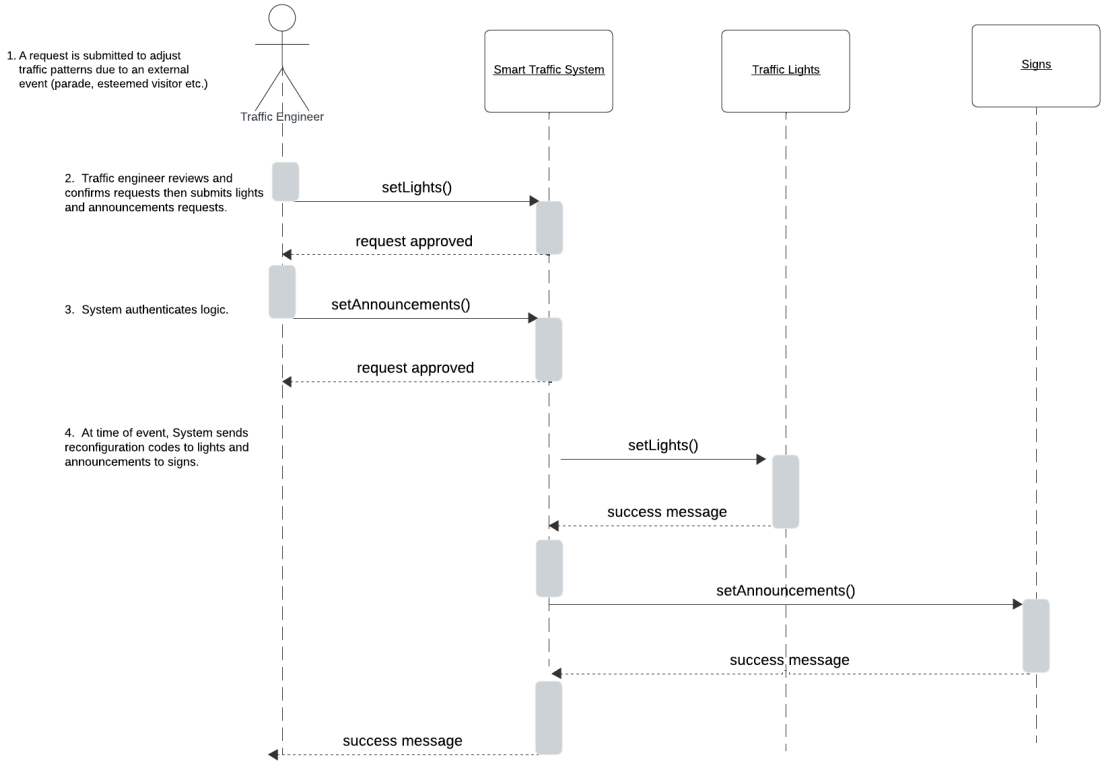
Explanation: The domain model was translated into a class diagram which includes the main classes, their attributes types, the methods and the associations.

UML Sequence Diagrams

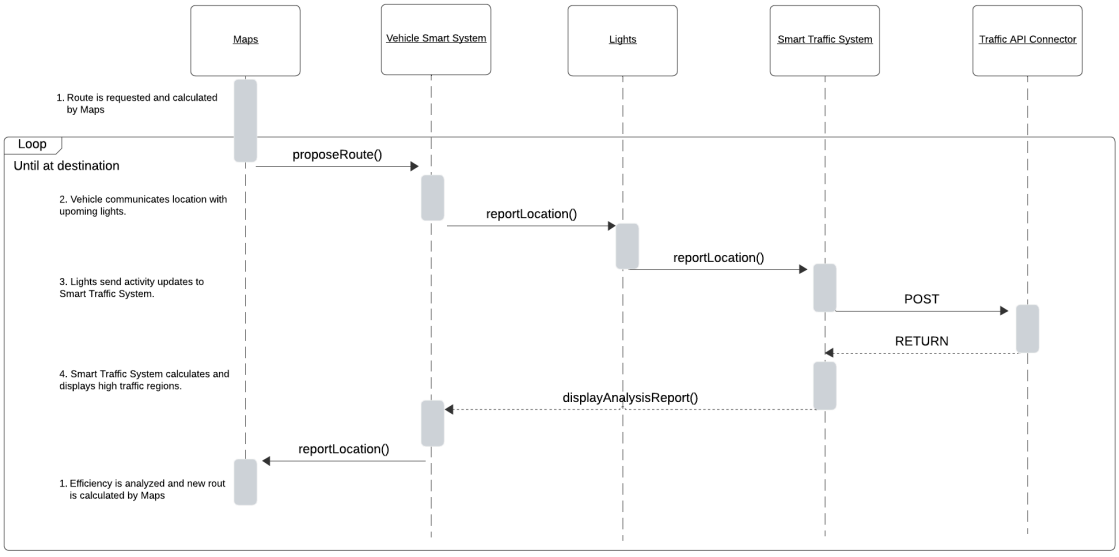
Use Case 1: Respond to skewed traffic patterns



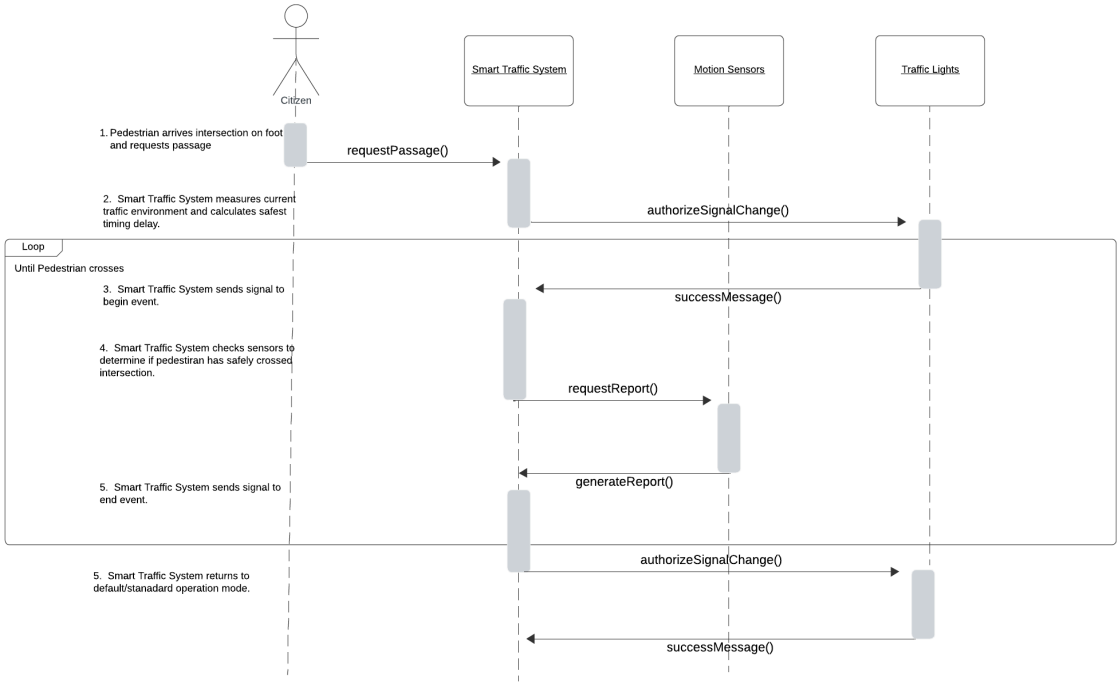
Use Case 2: Respond to temporary external event



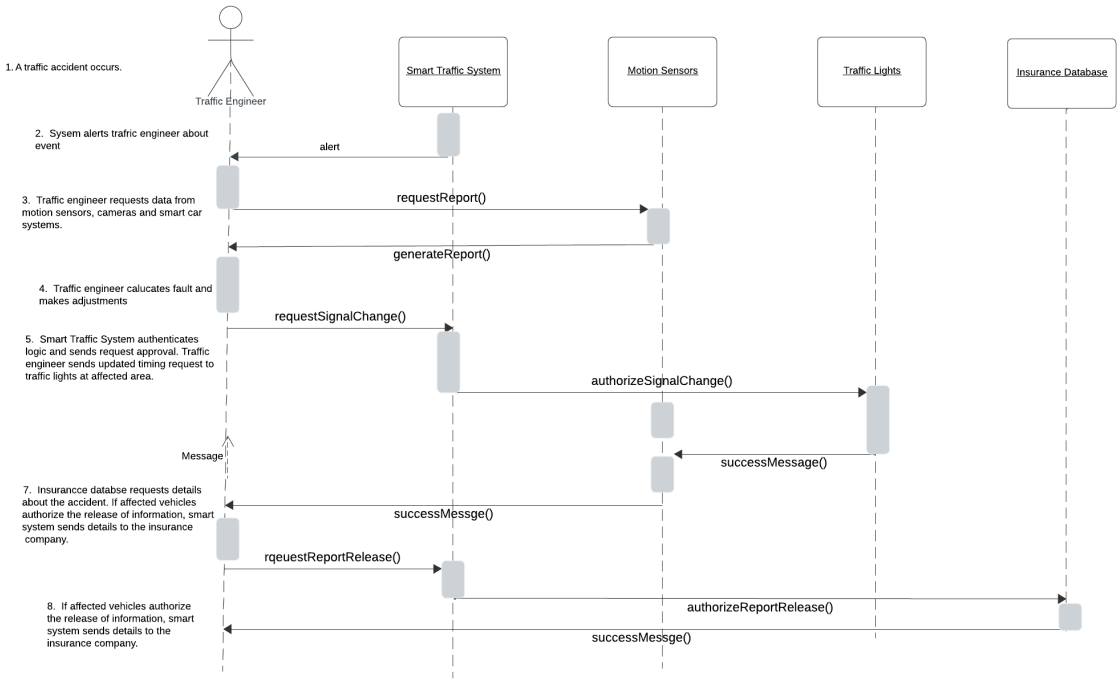
Use Case 3: Safely and efficiently navigate a route



Use Case 4: Pedestrian Passage Request



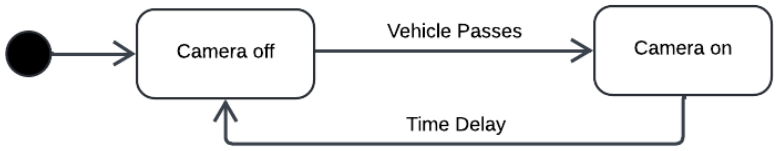
Use Case 5: Investigate an accident



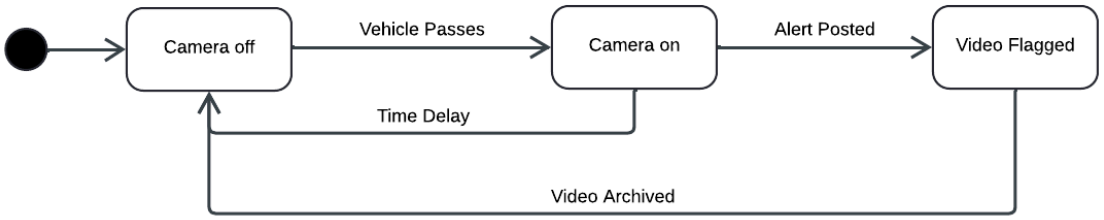
Explanation: Each use case is modeled below to include the actors involved and their role in the sequence of events to respond to and resolve the problem statements.

UML State Diagram

Recording traffic

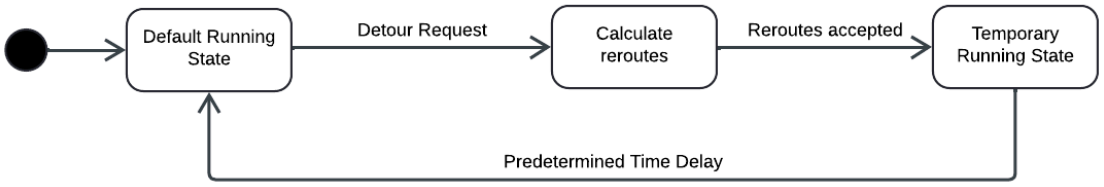


Alerting system manager



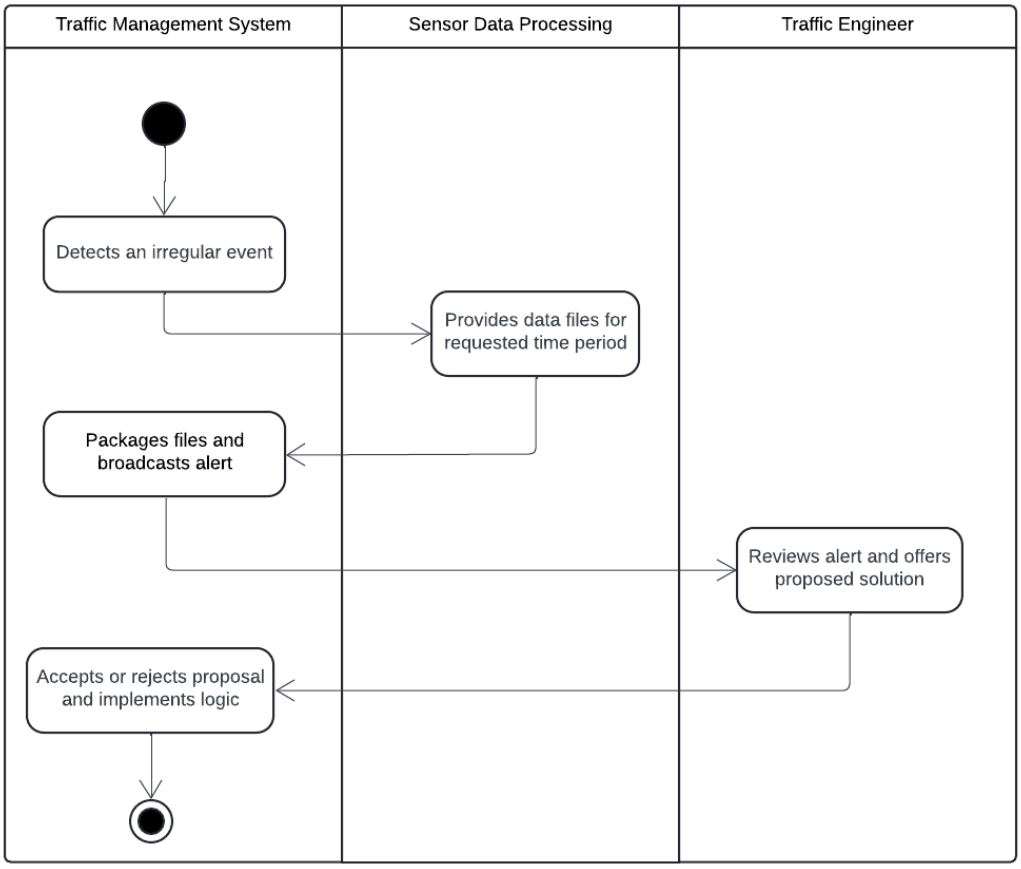
Explanation: A critical feature of the STS is its use of sensors and cameras which are placed along the roadways and responding to the way that vehicles are traveling. This feature gives real time data to the STS and can alert the system if there is some sort of unique event that may need to be reviewed by a traffic engineer or an accident investigation. To conserve energy, cameras and sensors should only be on when they are triggered by some sort of motion and they should not be running during times when there is no motion in their vicinity.

Programming detours



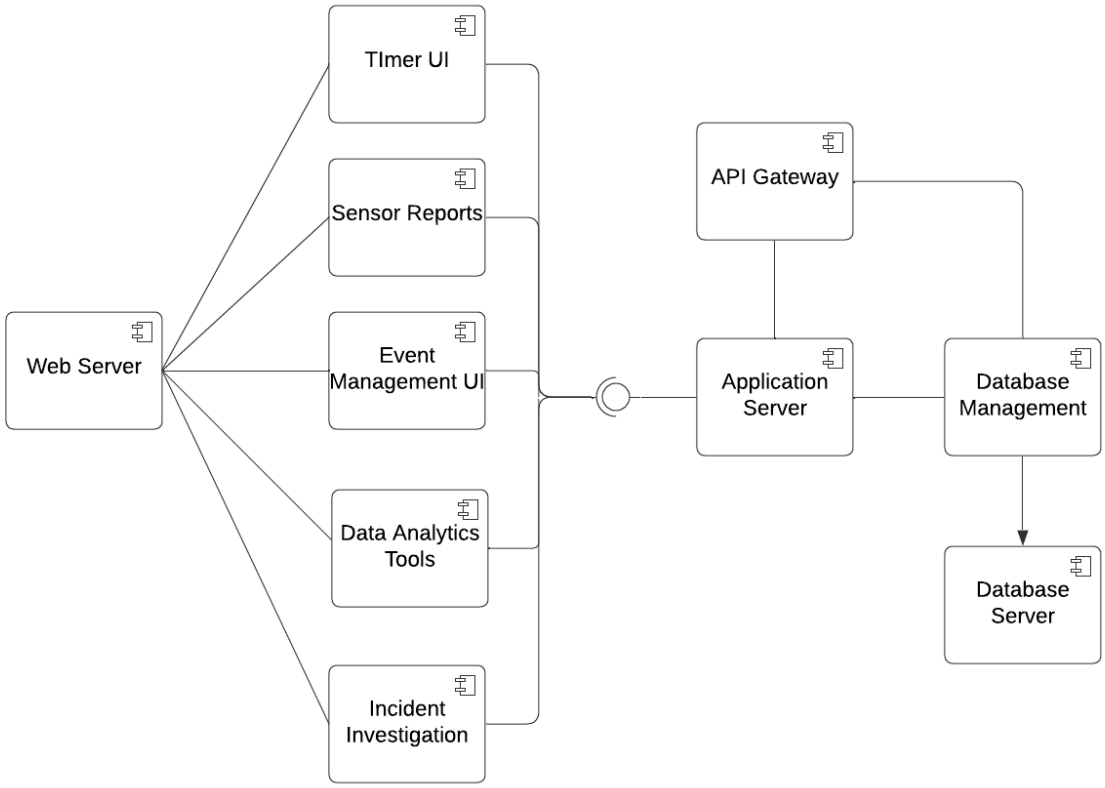
Explanation: A second critical feature of the STS is its ability to recalculate routes for temporary events such as construction delays, time of day congestion, accidents, or providing safe passage for emergency vehicles. The state diagram above demonstrates how this temporary running state is switched on and off for a predetermined amount of time.

UML Activity Diagram (Swimlane Diagram)



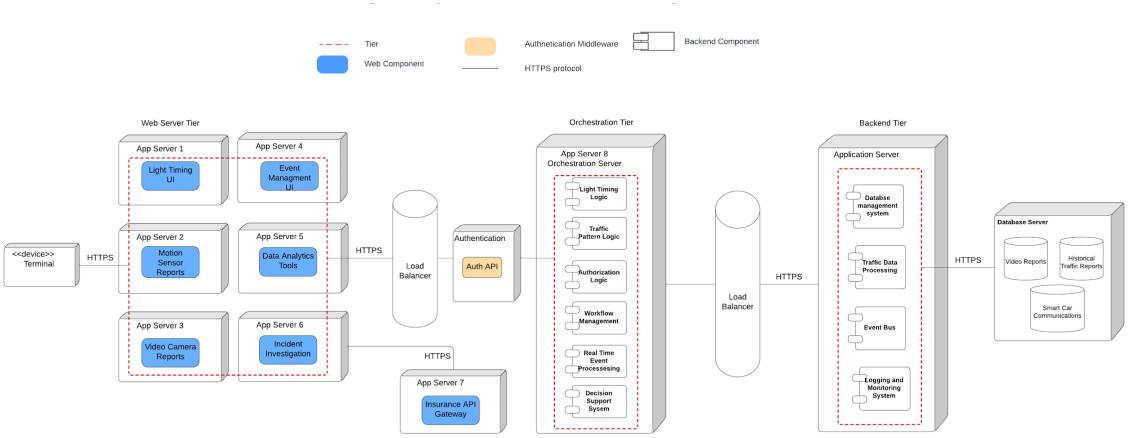
Explanation: The diagram above shows the way the STS responds to an irregularity that is detected on one of its sensors. In a case like this, there is a human proposal that must be approved by the logic standards in the STS Traffic Management class.

UML Component and Connector Diagram



Explanation: The component and connector diagram shows the way that the web server connects with the feature components of the STS. Each of these feature components is hosted on an application server which further interfaces with the external APIs and the storage databases.

Cloud Deployment Diagram



Explanation: The Web Server Tier on the cloud deployment diagram hosts numerous app server interface components which will be accessed by the people responsible for proposing changes to the system's functionality. The incident investigation interface will communicate with insurance company APIs for sending and receiving data.

Load balancers were added in front of and behind the application server in order to manage system traffic and ensure that it scales well as requests increase.

The orchestration tier hosts the logic for the system, as well as workflow management, event processing and database management.

The backend tier is made up of components which manage exchanging data and system messages.

Skeleton Classes

Traffic Management System

```
public class TrafficManagementSystem {
        public IncidentReports requestReport(String incidentID){
                //implementation to request incident report based on incident
                return incidentReports.getReport(incidentID);
        }
        public AnalysisReports displayAnalysisReport() {
                //implementation to display analysis report
                return analysisReports;
        }
        public CameraReports requestVideo(String incidentID){
```

```
                //implementation to request video based on incident ID
                Return cameraService.requestVideo(incidentID);
        }
        public AnalysisReports analyzeTrafficIncidents(String incidentID){
                //implementation to analyze traffic incidents based on incident ID
                return anaylsisTools.analyzeTrafficIncidents(IncidentID);
        }
        public boolean authorizeSignalChange(){
                //implementation logic to authorize signal change
                return signalController.authorizeSignalChange();
        }
}
```

Traffic Lights

```
public class TrafficLights{
        public TrafficLights(int secondsGreen, int secondsYellow, int secondsRed, boolean
        batteryIndicator, boolean offlineAlert, int lightID){
                this.secondsGreen = secondsGreen;
                this.secondsYellow =secondsYellow;
                this.secondsRed = secondsRed;
                this.batteryIndicator = batteryIndicator;
                this.offlineAlert = offlineAlert;
                this.lightID = lightID;
        }
        boolean successMesssage(){
                //implementation to send success message
                return true;
        }
        boolean activityReport() {
                //implementation to generate activity reports
                return true;
        }

}
```

Tables Definition

Video Reports

```
CREATE TABLE VideoReports(
        recording_id INT PRIMARY KEY,
        device_type VARCHAR(50),
        location_code INT,
        start_time DATETIME,
        end_time DATETIME
        capture BLOB
)
```

Historical traffic reports

```
CREATE TABLE Reports(
        incident_id INT PRIMARY KEY,
        location_code INT,
        Incident_type VARCHAR(50),
        report_time DATETIME,
        Description TEXT
)
```

Smart Car Communications

```
CREATE TABLE SmartCarCommuncations(
        smartcar_id INT PRIMARY KEY,
        location_code int,
        timestamp DATETIME,
        speed_kph INT
        description TEXT
)
```

Design Patterns

To achieve high cohesion, the GRASP design pattern of Pure Fabrication is implemented in the use of a Persistent Storage class that is solely responsible for storing the various files generated by the motion sensors and cameras on the traffic devices at intersection. This Persistent Storage class can store any file time and transfer it to the correct database.

The use of the Persistent Storage class also implements the SOLID design best practice of Single Responsibility by encapsulating file storage.

The Indirection design pattern is utilized in the system by assigning the responsibility for API calls to an intermediate class. This results in the system's ability to leverage additional features such as caching logging and error handling, without upsetting the responsiveness or availability of the STS.

The Law of Demeter practice from the SOLID design principles can be observed in the entire domain model, with each class only talking to its immediate neighbors. This can be seen in the way that the Reports class only talks to the Traffic Management System, despite the fact that the reports are collected within the Traffic Lights class. Setting the domain model up in this way achieves loosely coupled classes.

Data generated from the recording mechanisms (cameras and motion sensors) will be indexed and partitioned for faster retrieval of moments where a device captured an atypical event (such as a car accident).