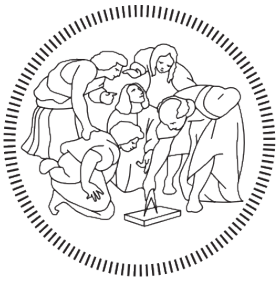


POLITECNICO
MILANO 1863

POLITECNICO DI MILANO

MASTER'S PROGRAM IN
HIGH PERFORMANCE COMPUTING ENGINEERING

Software Engineering for HPC
SustainCity Project

Group Members:

Salvatore Mariano Librici	Person Code: 11078653
Rong Huang	Person Code: 10948935
Yibo Li	Person Code: 11022291
Zhaochen Qiao	Person Code: 11021721

Academic Year:
2024/2025

UrbanLeaf

Smart Urban Mobility Management System

UrbanLeaf is a software solution designed to enhance urban sustainability through intelligent traffic management and event-aware mobility planning.

By leveraging real-time data, historical patterns, and event forecasts, UrbanLeaf supports both automated decisions and human oversight in city-wide transport optimization.

Project Code: `SustainCity` Version: 1.0 April 2025

Contents

1 The Project and Project Goals **2**
1.0.1 Overview 2
1.0.2 Project Goals 2
1.0.3 Technical Context 2
1.0.4 Reporting and Logging 3

2 Requirement Analysis **4**
2.0.1 Relevant Human and Non-Human Actors 4
2.0.2 Use Cases 5
2.0.3 Domain Assumptions 7
2.0.4 Requirements 8

3 Design **10**
3.0.1 General Description of the Architecture 10
3.0.2 Sequence Diagrams 13
3.0.3 Critical Points and Design Decisions 16

The Project and Project Goals

1.0.1 Overview

The UrbanLeaf project aims to tackle pressing global issues such as climate change and environmental sustainability, which are significantly impacted by urban transportation systems. In particular, the project focuses on reducing the negative effects of city commuting by leveraging data-driven technologies and adaptive control mechanisms.

This project is part of the Software Engineering for HPC course and is designed to apply the principles of requirement engineering and architectural design learned in the first part of the course.

1.0.2 Project Goals

The system to be designed will analyze urban traffic conditions and apply or suggest actions that optimize transportation infrastructure. The goals are divided into three categories:

- **Type 1 – Real-Time Traffic Light Adjustment:** Dynamically modify the duration of green and red lights at intersections based on live traffic conditions collected via road sensors.
- **Type 2 – Pattern-Based Optimization:** Analyze recurring traffic patterns to suggest optimizations such as changing road directions (e.g., one-way), traffic light configurations, and public transport schedules.
- **Type 3 – Event-Based Planning:** Collect information about planned large-scale public events and generate event-specific transportation plans. This includes changes in public transport, traffic light behavior, and road accessibility.

1.0.3 Technical Context

The system will integrate with several existing data sources:

- An event-based sensor infrastructure that tracks vehicle crossing times at intersections.
- A microservice providing public transport schedules, with API support for querying by street or line.

- A news feed or event channel that provides early notifications of upcoming events.

1.0.4 Reporting and Logging

The system must also:

- Automatically log all actions taken under Type 1.
- Generate daily public reports with traffic flow data and Type 1 actions.
- Generate yearly reports outlining all Type 2 and Type 3 suggestions, indicating whether they were accepted or rejected by city managers.

Requirement Analysis

2.0.1 Relevant Human and Non-Human Actors

Human Actors:

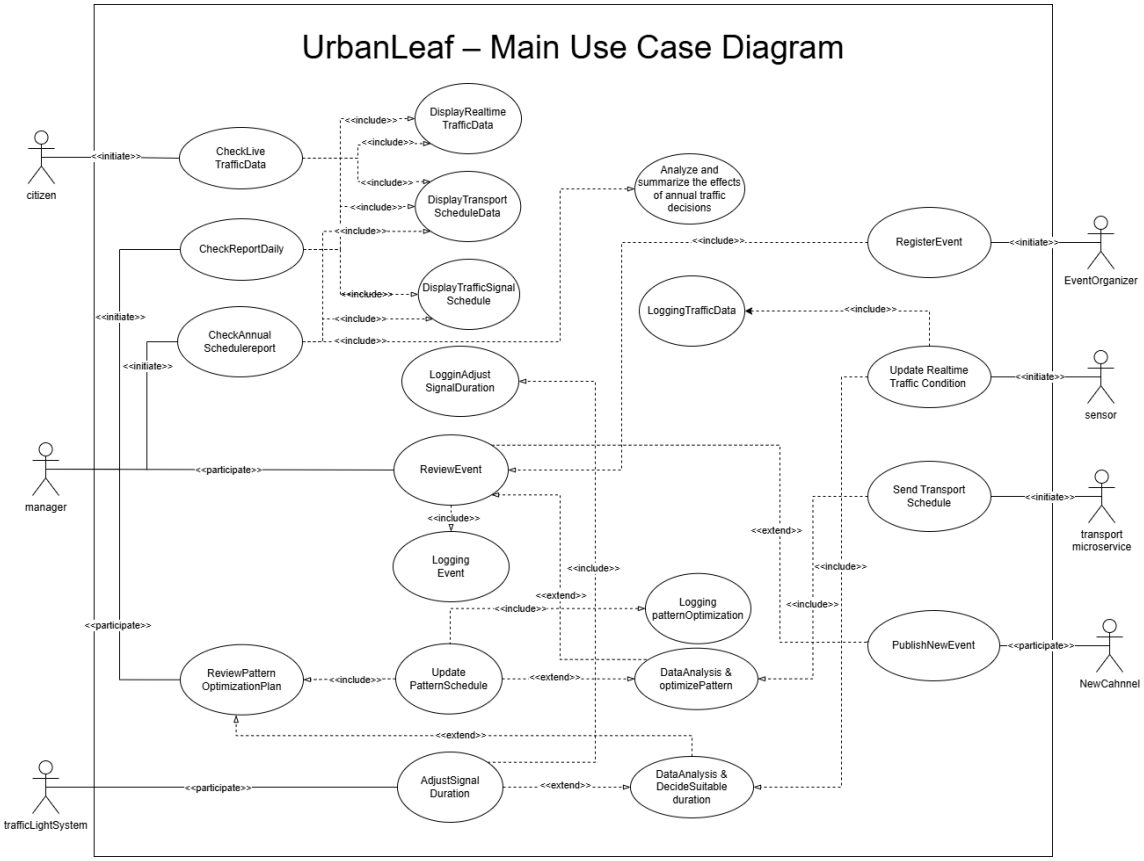
- **Traffic Managers** – Access daily reports (Type 1), review and approve/reject optimization proposals (Type 2 and Type 3).
- **Citizens** – Receive real-time traffic data (Type 1), and event-based routing suggestions (Type 2, 3) through the application.
- **Event Organizers** – Submit data about upcoming public events (Type 3).

Non-Human Actors:

- **Sensor System** – Event-based system that provides real-time traffic flow data (Type 1).
- **Public Transport Microservice** – Provides public transportation timetables (Type 2, Type 3).
- **Public Transport Schedulers** – Provide bus/tram schedules in response to accepted system recommendations (Type 2 and 3).
- **Traffic Light System** – Accepts dynamic configuration changes (Type 1 and 3).
- **News Channel** – Publishes upcoming events (Type 3).

2.0.2 Use Cases

This section describes the main use cases of the UrbanLeaf System. The use cases model the interactions between external actors (citizens, managers, event organizers, sensors, and other systems) and the functionalities provided by the application. These interactions are visualized in the use case diagram below.



Overview of Represented Use Cases

- **CheckLiveTrafficData**: Citizens access real-time traffic information through the app. This use case includes:
  - *DisplayRealtimeTrafficData*
  - *DisplayTransportScheduleData*
  - *DisplayTrafficSignalSchedule*
- **CheckReportDaily / CheckAnnualScheduleReport**: Managers view daily and yearly traffic-related reports. The annual report includes system decisions and long-term insights.
- **RegisterEvent**: Event organizers register upcoming public events, including location, time, and estimated attendance.

- **UpdateRealtimeTrafficCondition**: The sensor system sends continuous real-time traffic flow data to the application.
- **SendTransportSchedule**: The public transport microservice provides schedules for buses and trams, which are used by the system to plan responses.
- **ReviewEvent / ReviewPatternOptimizationPlan**: Traffic managers access the dashboard to review system-generated suggestions related to events or recurring traffic patterns.
- **UpdatePatternSchedule / AdjustSignalDuration**: Once suggestions are approved by managers, the system updates transport schedules or modifies signal timing accordingly.
- **PublishNewEvent**: The News Channel actor receives and publishes relevant traffic and event information for public visibility.
- **LoggingTrafficData, LoggingEvent, LoggingPatternOptimization**: All key actions are recorded for traceability, report generation, and audit purposes.

Additional Use Cases (Not Represented in Diagram)

In addition to the diagrammed use cases, the following functionalities are part of the system's behavior:

- **Notify Rerouting Information to Citizens (T1, T3)**: Citizens receive personalized routing suggestions in case of detected congestion or planned events.
- **Handle System Errors and Log Failures (All)**: The system monitors component interactions and logs any data or service-related failures.
- **Generate Optimization Suggestions (T2)**: The pattern analyzer periodically identifies trends and proposes transport or traffic optimizations.
- **Analyze Event Impact (T3)**: The event analyzer estimates the traffic impact of upcoming public events using both live and historical data.
- **Summarize Traffic and System Performance (T1, T2, T3)**: Periodic and event-triggered summaries are generated for use in daily and annual reports.

Together, these use cases represent the functional core of the system and form the basis for the requirements described in the following section.

2.0.3 Domain Assumptions

- **Traffic Infrastructure**
  - Traffic sensors are installed throughout the city and continuously provide real-time data.
  - Traffic lights can be adjusted by the system in real time.
  - A public transport infrastructure exists with modifiable and plannable schedules.
  - Historical traffic and transport data are available for pattern analysis.
- **Actors and Stakeholders**
  - Managers are authorized to approve or reject proposed traffic and transport changes.
  - Citizens use a mobile or web-based application to access real-time traffic updates.
  - Event Organizers can submit structured event information.
  - All actors interact with the system through predefined interfaces or components.
- **Communication and Data Handling**
  - Real-time and scheduled data flows through the Data Aggregator to other modules.
  - The system's App is capable of pushing notifications and updates to citizens.
  - The Logger module is always available to record actions, suggestions, and system responses.
- **System Capabilities**
  - The pattern analyzer can identify optimizations based on both historical and real-time data.
  - The analyzers are capable of generating temporary or permanent changes to schedules or signals.
  - Suggested optimizations may include schedule adjustments, signal duration, or rerouting.

2.0.4 Requirements

Functional Requirements

- FR1.1: Receive real-time traffic data from sensors every 30 seconds.
- FR1.2: Detect road congestion and dynamically adjust light durations.
- FR1.3: Log all light adjustments with time, location, and change details.
- FR1.4: Generate and publish daily reports to authorized managers via the application.
- FR1.5: Display real-time traffic updates to citizens via application.
- FR2.1: Analyze historical traffic patterns over days.
- FR2.2: Suggest light timing changes for high-traffic zones.
- FR2.3: Recommend public transport schedule changes based on congestion patterns.
- FR2.4: Present all suggestions to traffic managers and log their decisions.
- FR2.5: Generate yearly reports listing all Type 2 suggestions with status.
- FR3.1: Collect upcoming event data from Public Transport Microservice, Data Aggregator and event organizers.
- FR3.2: Assess traffic impact using historical and live data.
- FR3.3: Suggest temporary traffic configurations and transport updates.
- FR3.4: Notify users with personalized routing and event alerts.
- FR3.5: Generate yearly reports of Type 3 suggestions and approvals.

Non-Functional Requirements

- **NFR1.1**: The system shall respond to real-time sensor data updates with low latency under typical traffic conditions (Type 1).
- **NFR1.2**: Historical data analyses shall complete in a timely manner to support periodic reviews and long-term planning (Type 2).
- **NFR1.3**: Event-based suggestions shall be generated promptly upon data reception to enable early planning and action (Type 3).
- **NFR1.4**: All logged actions and decisions shall be securely stored, timestamped, and protected from tampering.
- **NFR1.5**: Reports shall be available in a standardized digital format suitable for archival and sharing (e.g. PDF).
- **NFR1.6**: The system interfaces shall be accessible, user-friendly, and optimized for use on both desktop and mobile devices.
- **NFR1.7**: Access to administrative and decision-making features shall be controlled through role-based permissions.
- **NFR1.8**: The system shall scale to support multiple simultaneous analyses and event-handling processes without loss of functionality.
- **NFR1.9**: Communication between system components shall be efficient and consistent to avoid bottlenecks and delays.
- **NFR1.10**: The system shall maintain data integrity when handling concurrent operations such as logging and configuration updates.
- **NFR1.11**: During periods of high activity, critical event-handling operations shall be prioritized over routine background tasks.

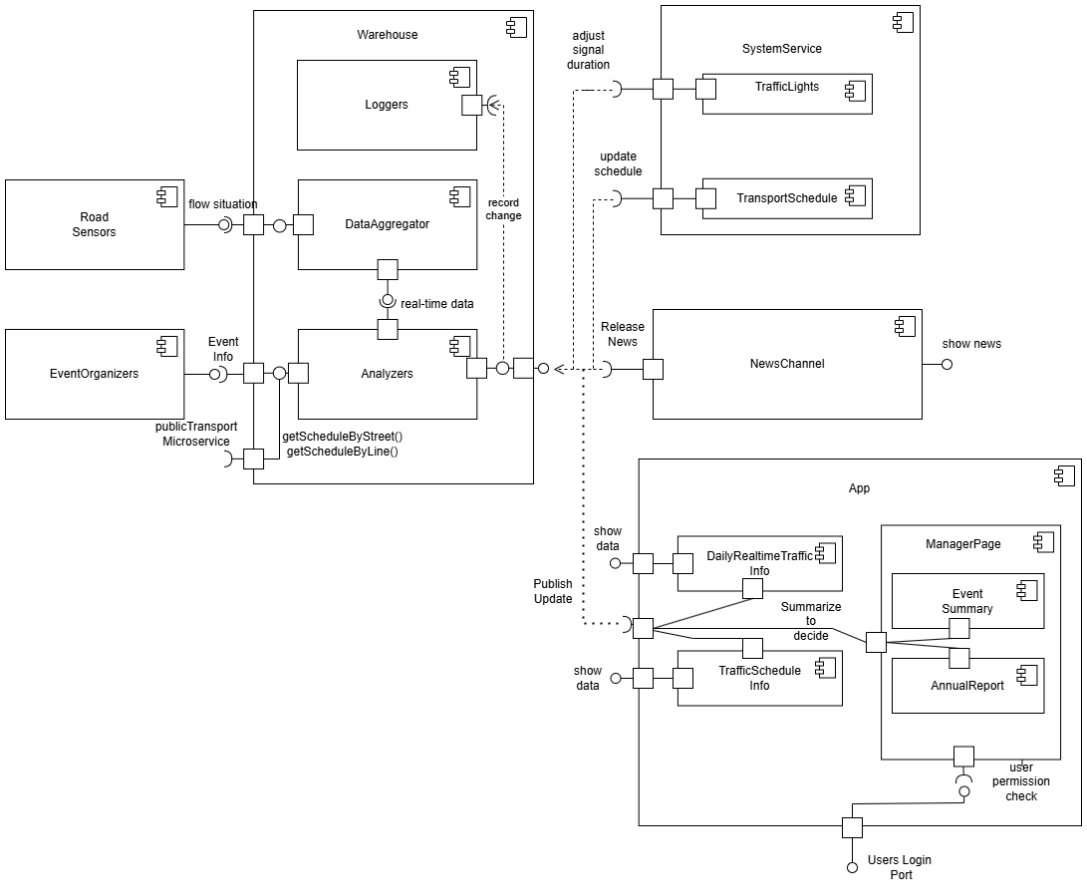
Design

3.0.1 General Description of the Architecture

The architecture of the system follows a modular, component-based design to promote scalability, maintainability, and clear separation of responsibilities. It is organized into three main subsystems:

- **Data Sources**: External components responsible for providing real-time data, event information, and public transport schedules.
- **Processing Core (Warehouse)**: Internal system modules that aggregate, analyze, and store incoming data, and coordinate responses.
- **Service and Interaction Layer**: Interfaces responsible for communicating results and actions to physical infrastructure and end users.

The diagram below represents the structure and interactions between the system components:



Component Descriptions:

- **Road Sensors**: Continuously stream real-time traffic flow data (e.g., congestion levels, vehicle counts) into the system.
- **Event Organizers**: Register events that may impact traffic. These inputs are handled by the system through the event dashboard.
- **Public Transport Microservice**: An external system providing transport timetables via API calls like `getScheduleByStreet()` and `getScheduleByLine()`.
- **Data Aggregator**: Collects raw traffic and event data from various sources and prepares it for analysis. Serves as a unified input hub.
- **Analyzers**: Responsible for processing different types of data, each dedicated to a specific functionality:
  - **Traffic Analyzer**: Handles real-time traffic data to detect congestion and trigger immediate adjustments to traffic lights (Type 1).
  - **Pattern Analyzer**: Processes historical traffic data to identify long-term trends and suggest optimizations, such as route or schedule changes (Type 2).
  - **Event Analyzer**: Assesses the potential impact of upcoming public

events on traffic and transport systems, generating temporary plans and event news.(Type 3).

- **Loggers**: Record system actions, decisions, and configuration changes for auditing, traceability, and report generation.
- **SystemService**: Composed of two internal components:
  - **TrafficLights**: Responsible for adapting signal timings based on analyzer output.
  - **TransportSchedule**: Updates public transport schedules in response to congestion or planned events.
- **News Channel**: Receives and displays information about upcoming events.
- **App**: The main interface for users, split into:
  - **DailyRealtimeTrafficInfo**: Displays live traffic data to citizens.
  - **TrafficScheduleInfo**: Shows updated public transport schedules.
  - **ManagerPage**: Restricted area for traffic managers, with:
    - **Event Summary**: Overview of past and upcoming events.
    - **Annual Report**: Summary of yearly system activity and decisions.
- **Users Login Port**: Handles authentication and role-based access control, ensuring the right interface and permissions are applied.

Communication between modules is asynchronous and event-driven. This design allows the system to process multiple workflows in parallel (e.g., analyzing traffic while handling events) and maintain high performance under varying load conditions. Each module interacts via well-defined APIs or message-passing interfaces to support scalability and future extensions.

3.0.2 Sequence Diagrams

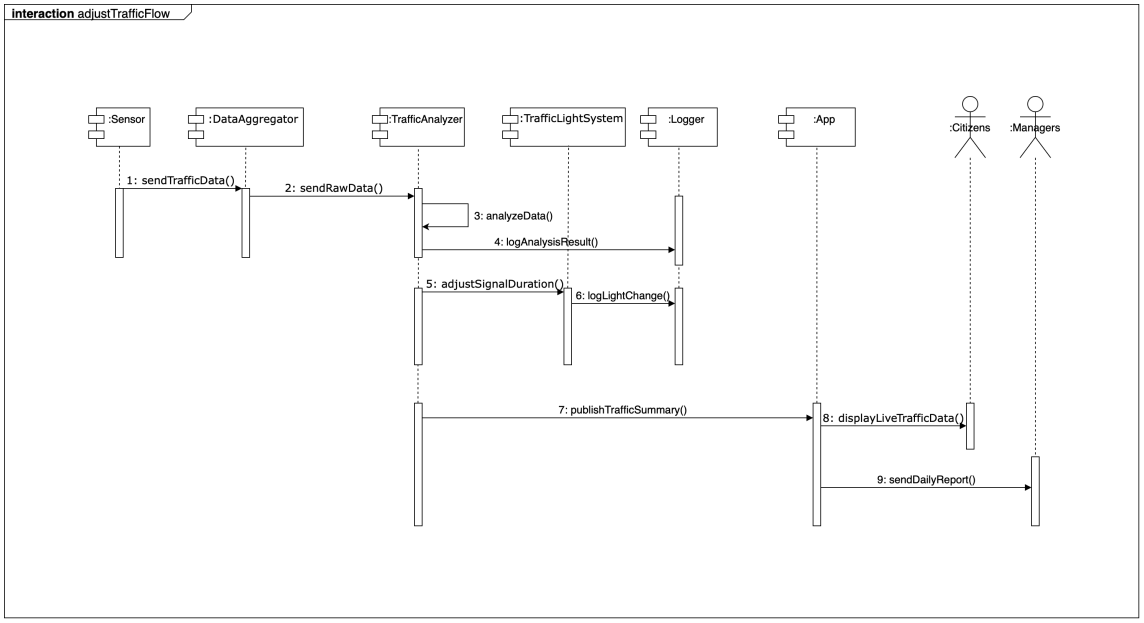
This section presents three key sequence diagrams that illustrate the dynamic interactions between system components, services, and actors. Each sequence diagram corresponds to one of the three functional categories of the system: real-time congestion handling, pattern-based optimization, and event-based traffic planning.

The diagrams highlight:

- The main flow of communication between modules.
- The triggering actions (e.g., sensor input, event registration).
- The system's behavior in processing, logging, and updating external interfaces.

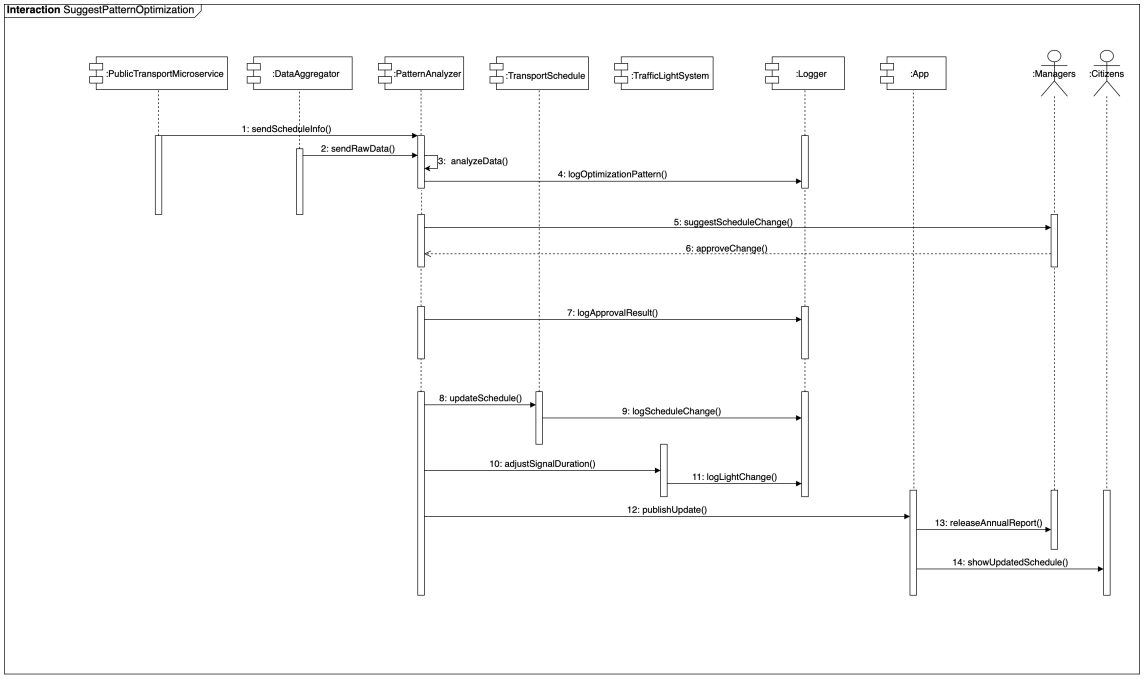
Type 1 – Adjust Traffic Flow (Real-Time Reaction)

This diagram describes how the system processes real-time traffic data sent from sensors. The data is aggregated and analyzed, triggering adjustments to traffic light durations. All changes are recorded and traffic summaries are published in the app interface for citizens and managers.



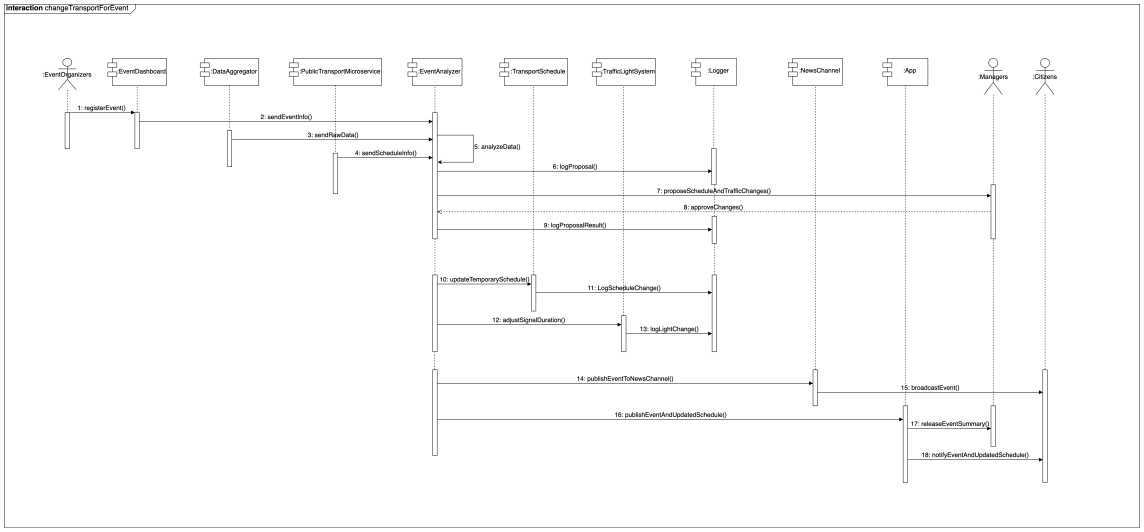
Type 2 – Suggest Pattern Optimization

This interaction shows how historical traffic and schedule data are processed by the Pattern Analyzer to generate long-term optimization suggestions. Once approved by traffic managers, updates are made to public transport schedules and traffic light settings. The results are logged and made visible in the annual report.



Type 3 – Change Transport for Event (Event-Based Adaptation)

This diagram outlines how the system reacts when a new event is registered by an organizer. The Event Analyzer uses current and historical data to suggest temporary traffic and transport changes. Upon approval, configurations are updated, and notifications are sent to citizens through the app and news channels.



3.0.3 Critical Points and Design Decisions

The following architectural and design decisions are directly based on the interactions illustrated in the sequence diagrams. They aim to ensure modularity, reliability, traceability, and scalability of the system:

- **Dedicated Analyzers per Task**: Each main functionality (real-time updates, pattern analysis, and event planning) is handled by a specific analyzer. This separation supports maintainability and allows specialized processing logic for each type of analysis.
- **Centralized Data Aggregation**: All incoming data from external sensors are collected by a shared `DataAggregator`. This avoids redundancy and ensures consistency in the data used by different system modules.
- **Event-Driven Communication**: The system reacts to incoming data asynchronously. For example, signal adjustments or schedule updates are triggered automatically after analysis, without requiring manual input. This design improves responsiveness and system autonomy.
- **Approval Workflow for Critical Changes**: In Type 2 and Type 3 flows, the system generates suggestions rather than enforcing changes directly. Traffic managers must approve each proposal, ensuring control over modifications to public infrastructure.
- **Separation of Analysis and Execution**: The process of analyzing data and generating suggestions is clearly separated from the components responsible for applying changes. This design reduces the risk of unintended behavior and improves traceability.
- **Comprehensive Logging**: All major actions such as approvals, changes to signal durations, and schedule updates are recorded by the `Logger`.
- **Role-Specific Interfaces**: The system interface (App) provides different functionalities depending on the user role. Citizens receive real-time updates and routing suggestions, as well as the incoming events, while managers access dashboards to review and check reports.
- **Scalability and Parallel Processing**: The architecture supports multiple independent data flows, as shown in the sequence diagrams. This enables the system to handle concurrent traffic conditions and events without performance degradation.