Bus Stop Monitoring

- Leveraging Data for Seamless Transit

- Project by:
- Pantelis Glentis 1066464
- Evgenia Gkagkastathi 1019191

Challenges In Public Transportation

Public transportation faces challenges impacting efficiency and passenger experience, such as:

1. **Overcrowding at Bus Stations and Buses:**
    - Long wait times leading to unpleasant passenger experience
    - Increased risk of safety incidents
2. **Illegal Parking:**
    - Disruptions in bus operations
    - Traffic congestion around stations
3. **Schedule Delays:**
    - Unreliable bus services
    - Inconvenience for passengers
    - Potentially increased operational costs for bus companies

These issues not only impact the reliability of the transit system but also lead to frustration among commuters, emphasizing the need for a solution.

The project

is an application designed to efficiently track and manage city bus traffic systems, with a primary focus on meeting the passengers needs.

Objectives:

- Provide data-driven insights for effective decision-making.
- Enhance overall safety and security at bus stations.
- Improve the reliability and efficiency of bus schedules.
- Offer passenger-centric services for a seamless transit experience.

Implemented features

Bus Schedule Monitoring

- Virtual GPS bus tracking
- Notifying the bus drivers of congestion in buses as it occurs.
- Retrieving historical data on crowd levels for bus lines at their respective stations.

Crowd Flow Monitoring

- Real-time detection of people movement in both bus stations and buses with the use of computer vision technology and virtual data.
- Monitoring station and bus occupancy levels and crowd density to effectively track congestion caused by passenger flow.
- Obtaining historical crowd level data for specific bus stations within a defined time period.

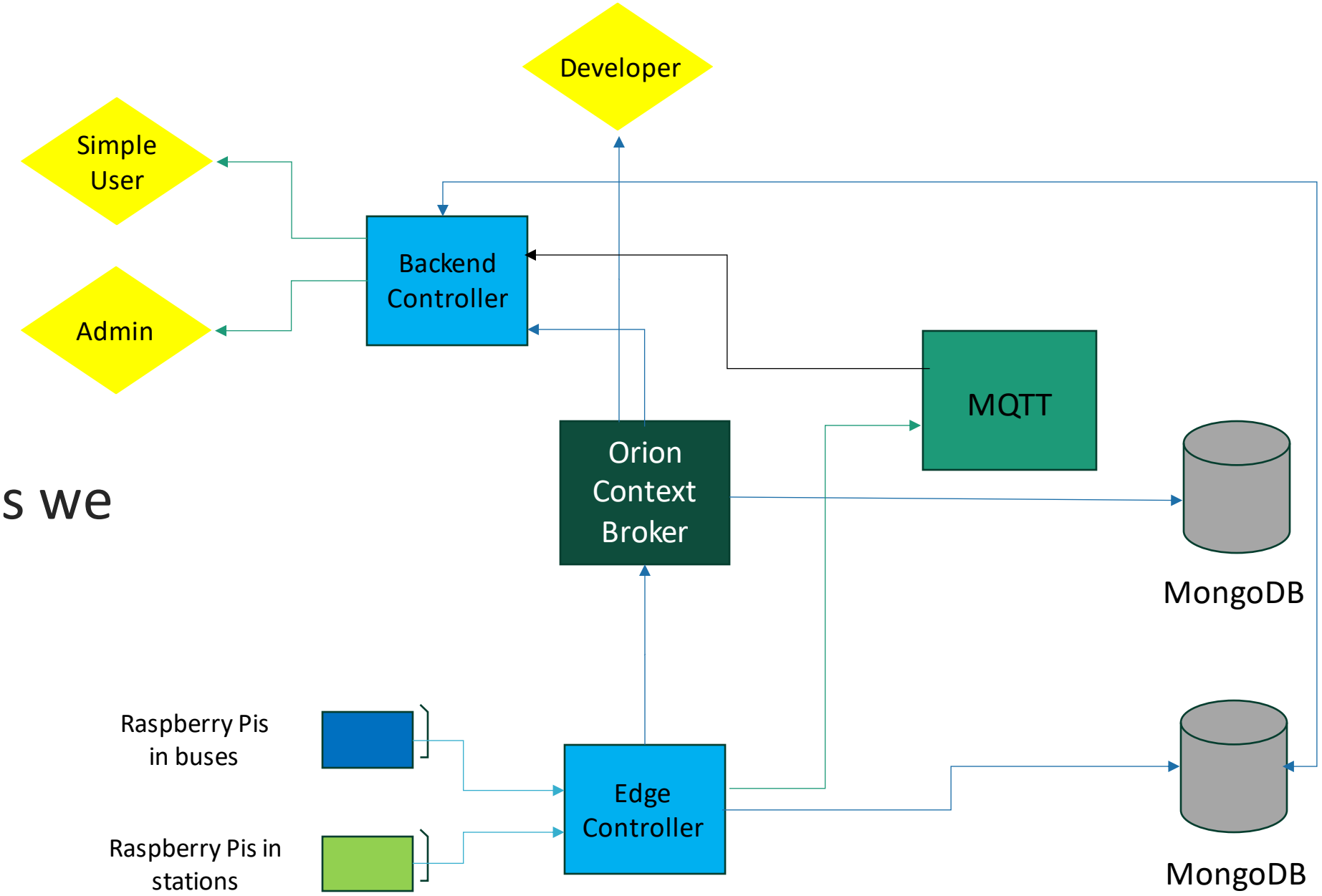
Illegal Parking Detection

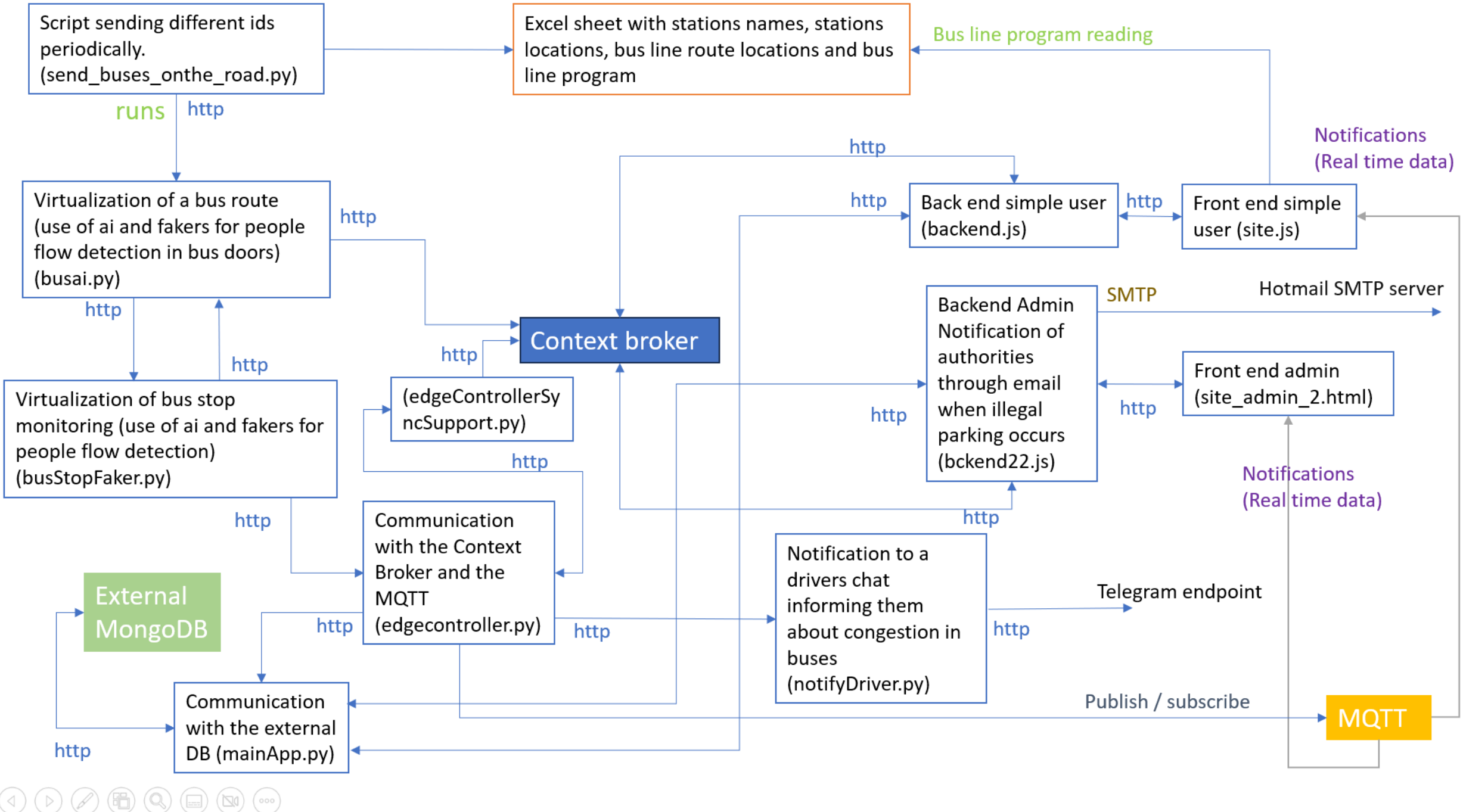
- Notifying public authorities of illegal parking on stations as it occurs.

Client services

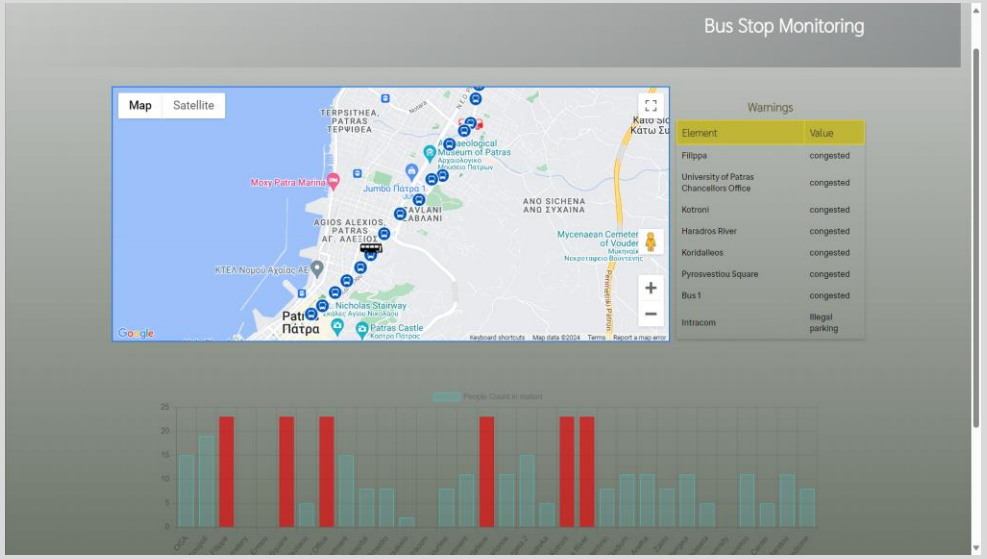
- Website interface providing bus schedule information and real time data
- Personalized station selection based on client location using real-time data.

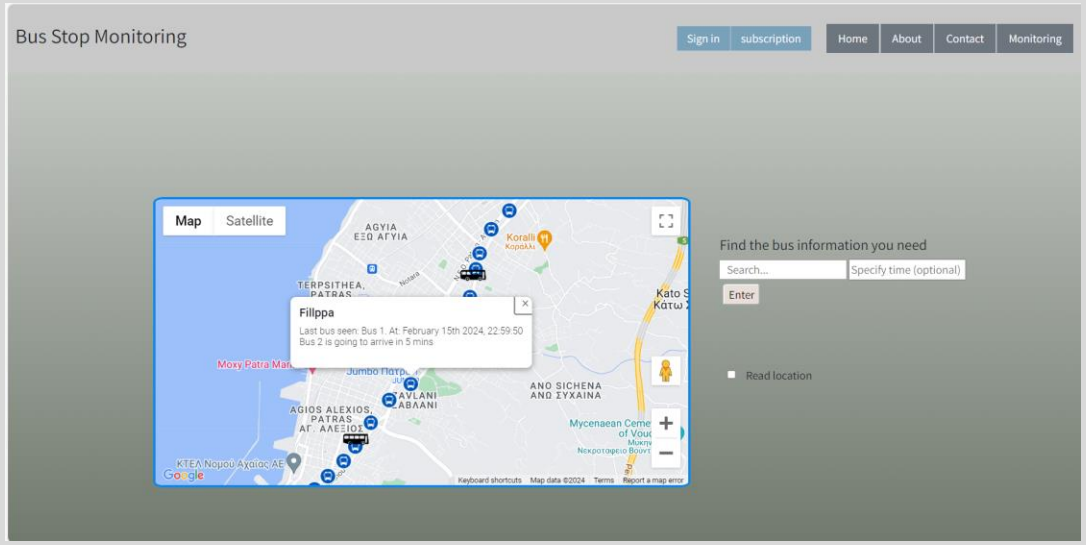
The architecture as we implemented it

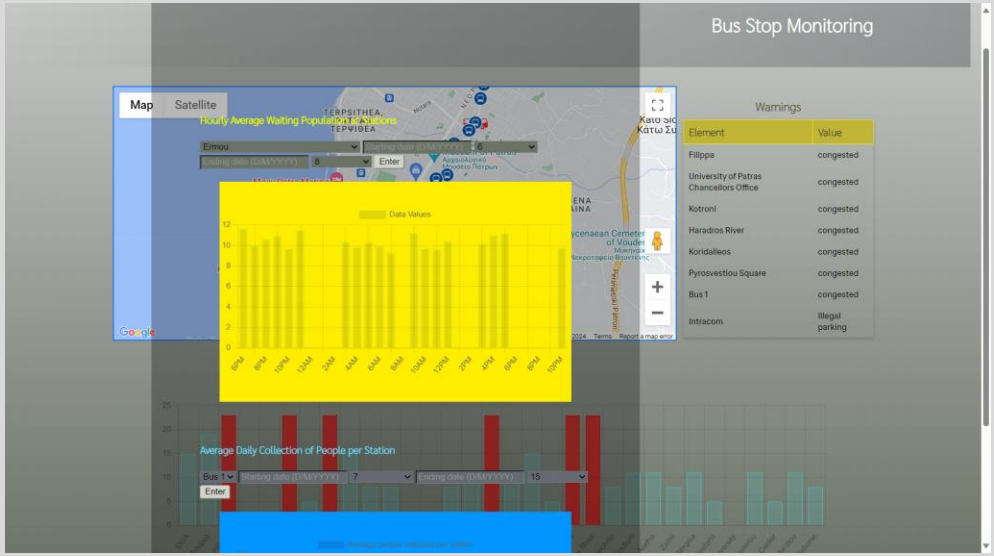


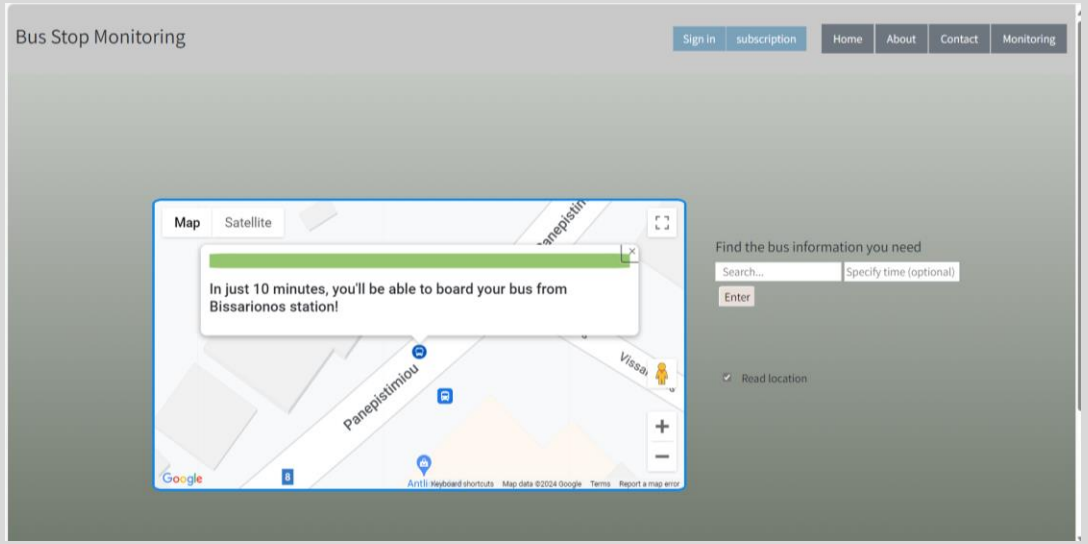

Script sending different ids periodically. (send_buses_onthe_road.py)
runs
http
Excel sheet with stations names, stations locations, bus line route locations and bus line program
Bus line program reading
Notifications (Real time data)
Virtualization of a bus route (use of ai and fakers for people flow detection in bus doors) (busai.py)
http
Back end simple user (backend.js)
Front end simple user (site.js)
Context broker
Virtualization of bus stop monitoring (use of ai and fakers for people flow detection) (busStopFaker.py)
(edgeControllerSyncSupport.py)
Backend Admin Notification of authorities through email when illegal parking occurs (bckend22.js)
SMTP
Hotmail SMTP server
Front end admin (site_admin_2.html)
Communication with the Context Broker and the MQTT (edgecontroller.py)
Notification to a drivers chat informing them about congestion in buses (notifyDriver.py)
Telegram endpoint
External MongoDB
Communication with the external DB (mainApp.py)
Publish / subscribe
MQTT

Front End pages









Following Demo Presentation

How we worked

Project

(Pantelis)
Edge Controller

- External database set-up (Mongo)
- Methods receiving data from edge devices and crafting payloads
- Methods extracting historical data from external database
- Posting to Context Broker
- Publishing to MQTT Broker
- Posting to external database
- Posting to telegram endpoint for bus driver alerts

(Evgenia)
BackEnd Controllers - Front end pages

- Backend requesting information from context broker according to get requests from the front end.
- Notifications to authorities about illegal parking in stations.
- Front end making get requests to backend.
- Subscription to mqtt entities.
- Charts and data tables presenting real time data provided by MQTT notifications.
- Use of Google APIs that facilitates the delivery of personalized services to users.

Edge Devices (Fakers)

(Pantelis)
Bus Station Devices

- Synchronizing bus and bus station data
- Asynchronous posting of crowd flow in stations every 5 seconds
- Posting Illegal parking detection
- Bus Station data update when bus has arrived or illegal parking detected
- Pausing related fake devices when AI is running. AI implementation detecting crowd flow in station with testing video

(Evgenia)
Bus Devices

- First script reading bus location from an Excel document and updating bus location information.
- First script notifying bus stops that a bus has arrived in a bus station.
- First script asynchronous posting of crowd flow and bus location information to edge controller.
- First script using ai for people flow detection from bus doors.
- First script generating virtual crowd flow data in stations.
- Use of a second script for running the first providing different entity ids.