Smart Logistic Analysis

It is a smart city subdomain which helps the transportation control, traffic flux and manage congestion.

The entities in this scenario are :

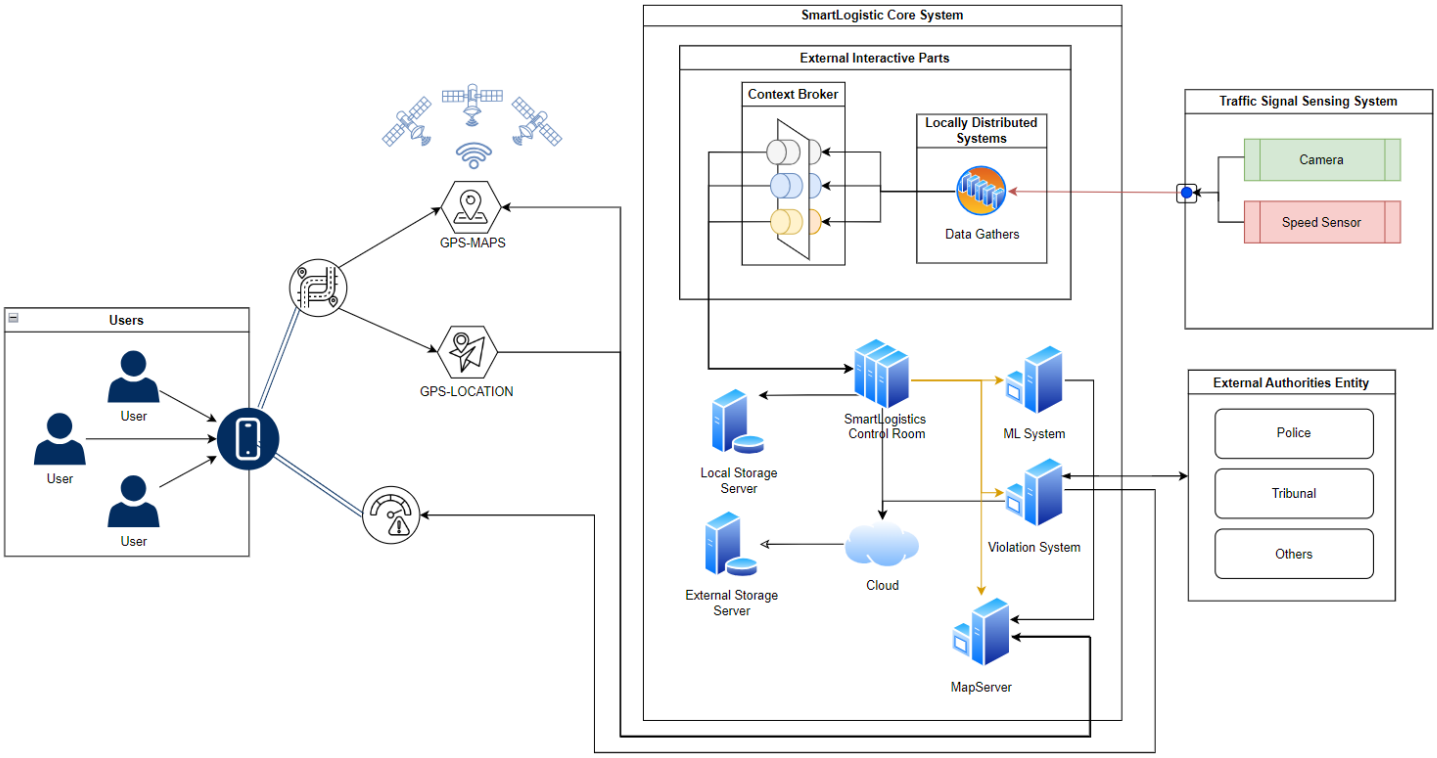
- The Smart People that:
    - Interrogate the maps (for vehicle, bicycle or pedestrian). The maps status are updated via Map-Server that computes the shortest path in order to reduces congestion and regulate flux/affluence.
    - The user position is sent (via Map-Server) in order to process the data and eventually make a more accurate traffic prediction. Moreover, the users have to choose which transport method they have to use (car, bicycle, or just walk) in order to get and transmit more accurate information.
        - [Frequency] the user location is updated once each 5 minutes
        - [Storage] None i.e. once the data has been processed, they are discarded
        - [Volume] 5 MB/day ∀ user
        - [Hypothetical structure] timestamp, session key (privacy policy), means of transport, coordinates
        - [Protocol] HTTPS - TCP
- Each traffic signals have a block sensing system (composed by cameras and radar guns) which handle an intelligent flux via traffic-regulation. The data sent from the sensing system can be of two possible kinds:
    - *Street video capture*
        - [Frequency] Real-time
        - [Storage] Hot data are weekly settled using a *weekly* roll-in roll-out approach and the cold data are kept for 1 year (for incidents/accident proof or similar).
            - Note: in this week, the video can be fast-consulted at any time for any reason.
        - [Volume] 10 GB/week ∀ Source (compressed data)
        - [Protocol] SRTP - TCP
    - *Road speed violation*
        - [Frequency] Whenever there is a violation
        - [Storage] 1 year
        - [Volume] 1 MB/week ∀ Violation
        - [Hypothetical structure] timestamp, coordinates, speed reached, frame
        - [Protocol] MQTT – TCP/IP

The Analytical part of this subsystem is related to:

1) take information from the *street video capture* and:
    - [***car-count submodule] the map status is updated considering the street-intersection flux in order to gives a hint/alternative of what path a user should take for reaching its destination avoiding too much congestion on the same road for each user.
        - [System Update Frequency] Real-time analysis
        - [User Update Frequency] In order to avoid excessive users-server communication, once each 5 minutes is updated.
    - [road accident submodule] the data are weekly analyzed in order to find if some accident happened, and, in the case, a short-video is cut/taken and kept in case of request from the local authorities.
        - [Frequency] The analysis is weekly computed
        - [Information Storage] 1 year
        - [Volume generated] 100 MB/day ∀ Violation
        - [Hypothetical structure] timestamp, camera who record the accident, coordinates, short-video
- [violation module] Take information from the and process, from *road inflation speed* and:
    - [plate recognition submodule] process the frame sent by the sensor and retrieve the plate and store the event into a database.
        - [Frequency] Real-Time
        - [Storage] 1 year (starting from signal timestamp)

        - [Volume] 1 MB/day ∀ Violation
        - [Hypothetical enriched structure] timestamp, location, frame, plate string
        - [Protocol] HTTPS – TCP/IP
    - [violation message submodule] when a violation has been committed and stored, the next step is using some authority channel to retrieve the owner of the plate and send a message directly to owner of the vehicle that committed a violation.
        - [Frequency] Real-Time
        - [Protocol] HTTPS – TCP/IP
- [path hint module] Takes information from the users' location (and *street video capture* ***), for:
    - [Channel recommendation] this makes prediction for the white, orange and red trafficability area and intersection such that it sends alternative/different path in order to avoid congestions . This information can be seen via user application when consulting maps.
        - [System Update Frequency] Real-time analysis
        - [User Update Frequency] In order to avoid excessive users-server communication, once each 5 minutes is updated.
    - [Blocking Access system] this is purely defined on that areas that are known for supporting a limit amount of people. When a certain number of peoples are located on a given area, this module can lock/advice against that particular section.
        - [System Update Frequency] Real-time analysis
        - [User Update Frequency] In order to avoid excessive users-server communication, once each 5 minutes is updated.
- [roll-in-roll-out module] Once a day, a service (via cron-job) executes the roll-in roll-out of the video in order to archive them.

General Idea Smart Logistic

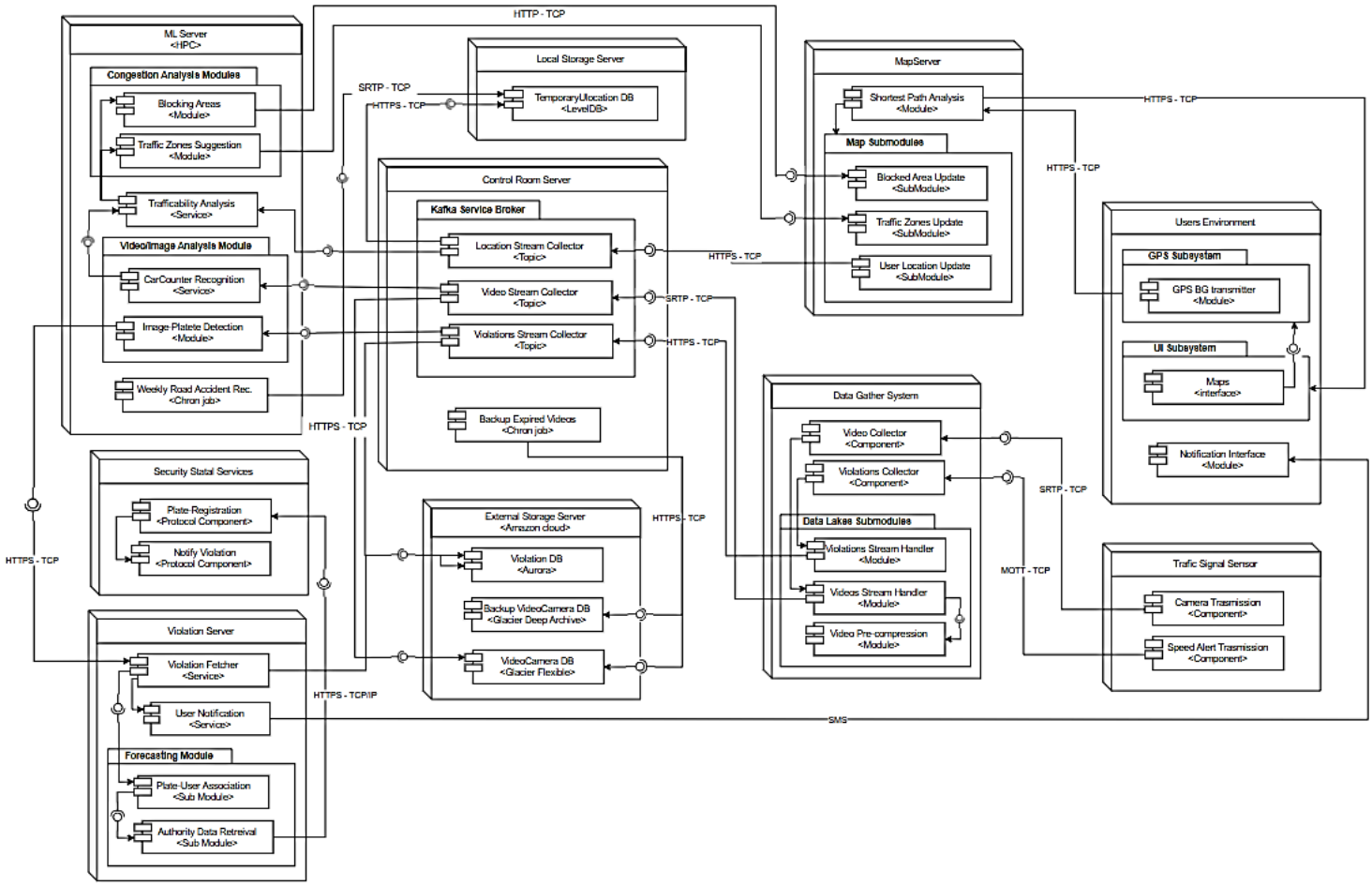


The idea behind Smart Logistic is that:

- The people, access to the application (interface via map-server) and can open the maps, which is updated with hot-point and congestion analyzed considering the means of transport chosen.
- Meantime, the map server, sends the users positions to the control room which is analyzed and processed in order to update the trafficability status of the city in combination to the camera sensor.

- The traffic flux is regulated with the smart signal system which:
    - Sends real-time video of the interception in order to be:
        - Real-time analyzed by the control room which updates the congestion status over the interception channels i.e. update the trafficability status.
        - Weekly block-analyzed such that identify road accident in order to store the video and make it available for further investigations.
    - Moreover, the traffic signals are equipped with speed sensor that sends data about the cars speed violations. When the control room got this kind of messages, it analyzes the snapshot taken for retrieving via plate-recognition the owner in order to send a violation message directly to him/her. This data are also shared to the local authorities (e.g. police department) in order to be proceed with law processes.
- Also, in this case the control room works with two different storage environments: one local, one on cloud.
- All the analytical and recognition part are handled with an apposite machine learning methods which are preformed using a HPC server.
- Since the large number of sensors available on the city, all the communication are done with various data gather system and actuator spread throughout the city that collect, preprocess and send the data in order to speed up the elaboration.

C.D.D. & D.F.D. Smart Logistic



The following system reflects the general idea proposed:

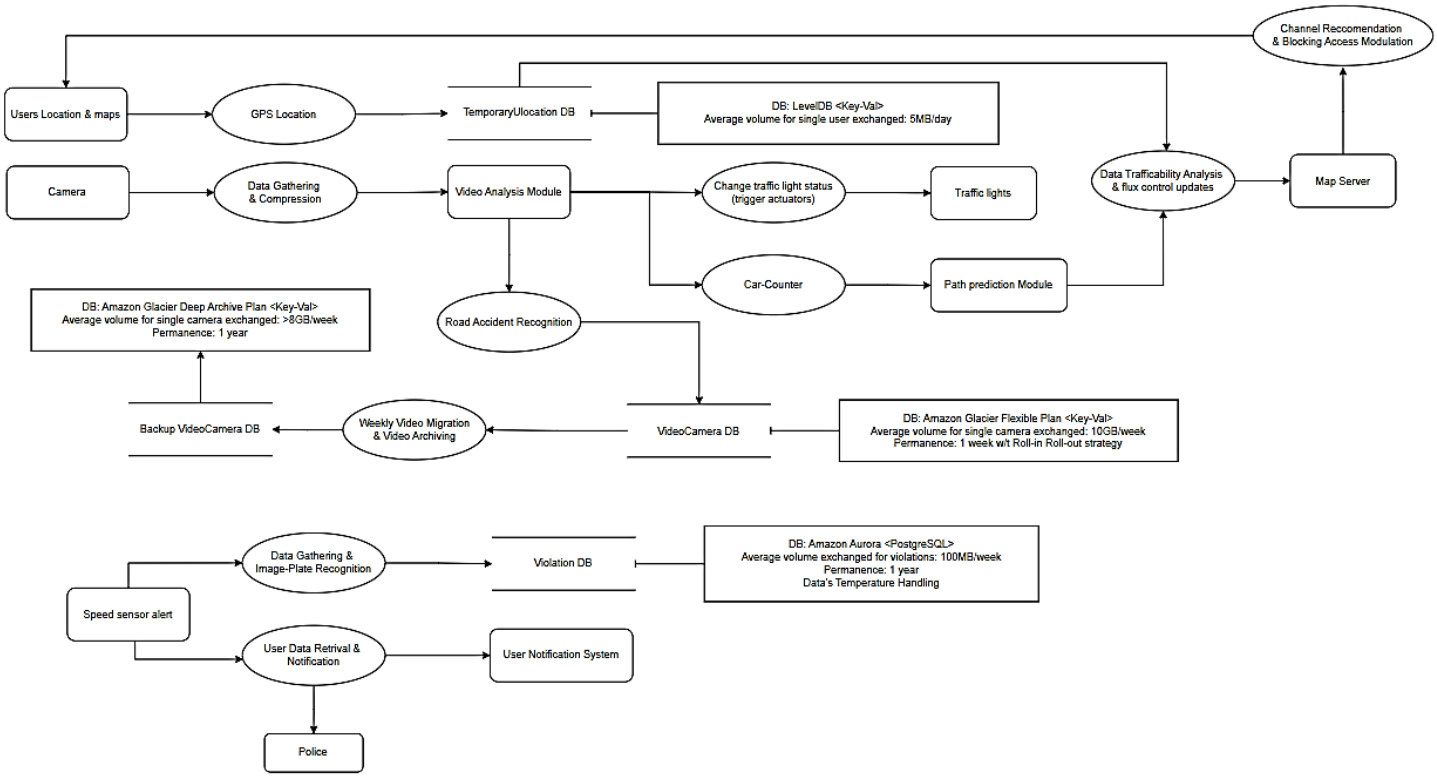
- The **Users Environment** has three different subsystems:
    - the first part sends the user position (the GPS HW is supposed to be preinstalled into the smartphone)
    - the second part is the UI application which enable the user to consult the maps after choosing the means of transport (like Google Maps).
    - the third part is the message subsystem that is just responsible to show the (violation) messages.
- The **Traffic Signal Sensor** which can be seen as a unique block sensor composed by various components:
    - a camera that sends the real-time video.

    - a speed radar which sends the information about speed violations.
- The **Data Gather System** is responsible for getting the vast amount of data coming from the cameras and the speed sensor violations and applies a first layer of data preprocessing (compression) before sending it to the control room.
- The **Control Room** handles the various communications channels i.e. topic using a Kafka service broker module. This module relies on three parts:
    - the first part is the user location that gathers the users' data location.
    - the second and third part is dedicated to the video collection.
    - the third part is dedicated for the violations.

  Moreover, the control room communicates with a ML server that:
    - sends the user position in order to predict the people affluence.
    - sends the video-capture in order to process the channel's interceptions congestion.
    - sends the violation such that via image recognition can automatically discern the vehicle plate.
- The **ML Server** is designed to be an HPC which purpose is the:
    - video and image recognition and detection.
    - traffic analysis.
    - weekly road accidents detection.

  Moreover, it directly communicates to the Map Server in order to updates the city's street information flus, saturation and congestion.
- The **Map Server** is designed to:
    - Handles user via shortest weighted path. The "weighted" is due to the fact that each intersection and street is updated with the flux conditions directly discerned from the ML Server.
    - For each user connected to it, the mapserver communicate each 5 minutes the user position to the control room, that updates the real-time ML prediction system and re-updates the affluence and congestion.
        - Auto-regulation strategy.
- The **Local Storage Server** handles the data coming from the user location (which has to be fast processed and after eliminated).
- The **External Storage Server** (Amazon) handles the storage of violations, cameras (and their backups) in different solution plan.

The D.F.D. is shown below



Analysis of costs

In order to build this kind of system the following choices are made for a polyglot DB environment:

1- For the temporary storage we prefer the usage of local LevelDB which is an efficient open-source on-disk key-value store written by Google in order to handle an amount of data of 5 MB each user. If we think that 100'000 users are contemporary connected, the system must have the capacity of maximum 500GB but since once each 5 minutes the data from users is elaborated and discarded (for privacy reason) using a sort of roll-in roll-out strategy on 5 minutes window. Long story short: we expect the volume never exceed that 500GB. Moreover, the system will be able to discard expiry location position.

   a. Check information about on this link: LevelDB
   b. Also check the public repo: google/leveldb.

2- For the video system, we have two kinds of different supports:

   a. Amazon Glacier Flexible Plan (hot/warm video-data) for "weekly video" eventually requested by institutional entities and so we need a storage system which handles a large amount of data with the possibilities to retrieve data in minutes or hour if the requested data is in week-lifespan (a use-case scenario are the accident video proof). The amount of data is supposed to be 10 GB at week for single camera. If we expect a number of sensors equals to 2000 the volume of the data is 20TB each week. Applying some data preprocessing we can save up to 2/3 of the space, so the total volume will be approximately 15TB.
   A global estimation is made Amazon Simple Storage Service (S3) (calculator.aws):

      i. Unit conversions
         1. S3 Glacier Flexible Retrieval storage: 15 TB per month x 1024 GB in a TB = 15360 GB per month
         2. S3 Glacier Flexible Retrieval Average Object Size: 15 MB x 0.0009765625 GB in a MB = 0.0146484375 GB
         3. Data retrievals (Standard): 15 TB per month x 1024 GB in a TB = 15360 GB per month

      ii. Pricing calculations
         1. 15,360 GB per month / 0.0146484375 GB average item size = 1,048,576.00 unrounded number of objects
         2. Round up by 1 (1048576.0000) = 1048576 number of objects
         3. 1,048,576 number of objects x 32 KB = 33,554,432.00 KB overhead
         4. 33,554,432.00 KB overhead / 1048576 KB in a GB = 32.00 GB overhead
         5. 32.00 GB overhead x 0.0036 USD = 0.1152 USD (Glacier Flexible Retrieval storage overhead cost for metadata)
         6. Glacier Flexible Retrieval storage overhead cost for metadata: 0.1152 USD
         7. 1,048,576 number of objects x 8 KB = 8,388,608.00 KB overhead
         8. 8,388,608.00 KB overhead / 1048576 KB in a GB = 8.00 GB overhead
         9. Tiered price for: 8.00 GB
         10. 8 GB x 0.023 USD = 0.18 USD
         11. Total tier cost = 0.184 USD (S3 Standard storage overhead cost for metadata)
         12. S3 Standard storage overhead cost for metadata: 0.184 USD
         13. 15,360 GB per month x 0.0036 USD = 55.296 USD (Glacier Flexible Retrieval storage cost)
         14. Glacier Flexible Retrieval storage cost: 55.296 USD
         15. 0.1152 USD + 0.184 USD + 55.296 USD = 55.5952 USD (Total Glacier Flexible Retrieval storage cost)
         16. 4 requests x 0.00003 USD = 0.0001 USD (PUT, COPY, POST, LIST requests to S3 Glacier Flexible Retrieval cost)
         17. 4 requests x 0.00003 USD = 0.0001 USD (Lifecycle transitions cost)
         18. 4 requests x 0.00005 USD = 0.0002 USD (Cost for Restore requests (Standard))
         19. 4 requests x 0.01 USD = 0.04 USD (Cost for Restore requests (Expedited))
         20. 4 requests x 0.00 USD = 0.00 USD (Cost for Restore requests (Bulk))
         21. 15,360 GB per month x 0.01 USD = 153.60 USD (Cost for Data retrievals (Standard))

22. 1 Provisioned Capacity Units x 100.00 USD = 100.00 USD (Cost for provisioned capacity)
23. 55.5952 USD + 0.0001 USD + 0.0001 USD + 0.0002 USD + 0.04 USD + 0.00 USD + 153.60 USD + 0 USD + 0 USD + 100.00 USD = 309.24 USD (Total S3 Glacier Flexible Retrieval cost)

Total Storage Cost (monthly) on average: 309.24 USD

b. Amazon Glacier Deep Archive (cold video-data) for "oldest videos" essentially used as semestral backups.
If we think to have more or less 15TB each week, it brings us the total semestral capability of 360TB.
A global estimation is made Amazon Simple Storage Service (S3) (calculator.aws):

i. Unit conversions
   1. S3 Glacier Deep Archive storage: 360 TB per month x 1024 GB in a TB = 368640 GB per month
   2. S3 Glacier Deep Archive Average Object Size: 16 MB x 0.0009765625 GB in a MB = 0.015625 GB

ii. Pricing calculations
   1. 368,640 GB per month / 0.015625 GB average item size = 23,592,960.00 unrounded number of objects
   2. Round up by 1 (23592960.0000) = 23592960 number of objects
   3. 23,592,960 number of objects x 32 KB = 754,974,720.00 KB overhead
   4. 754,974,720.00 KB overhead / 1048576 KB in a GB = 720.00 GB overhead
   5. 720.00 GB overhead x 0.00099 USD = 0.7128 USD (Glacier Deep Archive storage overhead cost for metadata)
   6. Glacier Deep Archive storage overhead cost: 0.7128 USD
   7. 23,592,960 number of objects x 8 KB = 188,743,680.00 KB overhead
   8. 188,743,680.00 KB overhead / 1048576 KB in a GB = 180.00 GB overhead
   9. Tiered price for: 180.00 GB
   10. 180 GB x 0.023 USD = 4.14 USD
   11. Total tier cost = 4.14 USD (S3 Standard storage overhead cost)
   12. S3 Standard storage overhead cost: 4.14 USD
   13. 368,640 GB per month x 0.00099 USD = 364.9536 USD (Glacier Deep Archive storage cost)
   14. Glacier Deep Archive storage cost: 364.9536 USD
   15. 0.7128 USD + 4.14 USD + 364.9536 USD = 369.8064 USD (Total Glacier Deep Archive storage cost)
   16. 4 requests x 0.00005 USD = 0.0002 USD (Cost for PUT, COPY, POST, LIST requests)
   17. 4 requests x 0.00005 USD = 0.0002 USD (Cost for Lifecycle transitions)
   18. 4 requests x 0.0001 USD = 0.0004 USD (Cost for Restore requests (Standard))
   19. 5 GB per month x 0.02 USD = 0.10 USD (Cost for Glacier Deep Archive Data Retrieval (Standard))
   20. 5 GB per month x 0.0025 USD = 0.0125 USD (Cost for Glacier Deep Archive Data Retrieval (Bulk))
   21. 369.8064 USD + 0.0002 USD + 0.0002 USD + 0.0004 USD + 0.10 USD + 0.0125 USD = 369.92 USD (S3 Glacier Deep Archive cost)

Total Storage Cost (monthly) on average: 369.92 USD

3- For the Violation data, since we do not expect to have much data, we prefer an on-cloud solution based on Amazon Aurora via PostgreSQL. The amount of data is supposed to be 100MB/week which is 400MB/month approximately since we must store it for one year.
Considering the smallest amount of data generated, we could also rely on internal storage form which has zero-costs. But, since it is an average estimation, the data could grows, and, moreover, it useful to have all the cloud storage advantages.
The total semestral value will be Amazon Aurora PostgreSQL-Compatible DB (calculator.aws):

a. Unit conversions
    i. Storage amount: 500 MB x 0.0009765625 GB in a MB = 0.48828125 GB
    ii. Baseline IO rate: 22 per day / (24 hours in a day x 7 days in a week) = 0.92 per hour
    iii. Peak IO rate: 22 per day / (24 hours in a day x 7 days in a week) = 0.92 per hour
b. Pricing calculations
    i. 0.48828125 GB x 0.10 USD = 0.05 USD (Database Storage Cost)
    ii. 730 hours in a month - 10 peak hours per month = 720.00 baseline hours per month
    iii. 0.92 baseline IOs x 720.00 baseline hours per month = 662.40 (Monthly Baseline IOs)
    iv. 662.40 (Monthly Baseline IOs) x 0.0000002 USD = 0.00013248 USD (Baseline IOs Cost)
    v. 0.92 peak IOs x 10 peak hours per month = 9.20 (Monthly Peak IOs)
    vi. 9.20 (Monthly Peak IOs) x 0.0000002 USD = 0.00000184 USD (Peak IOs Cost)
    vii. 0.00013248 USD (Baseline IO Cost) + 0.00000184 USD (Peak IO Cost) = 0.00 USD (IOs Rate Cost)

Total Storage Cost (monthly): 0.05 USD