POLITECNICO
MILANO 1863

SafeStreets

Requirements Analysis and Specification Document

Davide Cocco - 944122
Marco Gasperini - 944922

A.Y. 2019/2020 - Prof. Di Nitto Elisabetta

Contents

1 INTRODUCTION

1.1 Purpose

This document will contain the `Requirements Analysis and Specification Document` for SafeStreets, a crowd-sourced application that will allow users to notify traffic violations to authorities to assist them in law enforcement. Through a data mining system, users and authorities will receive relevant information in real time to help them with their job, and fining offenders will be accelerated through an automatic ticket emission functionality. The software will not be designed to substitute the existing system, but rather to improve its efficiency.

1.2 Scope

The software will be dedicated to civilians and authorities: through a mobile application, civilians will have the possibility to report a violation by submitting pictures, position, date and the license plate of the offender to the authorities in an effort to improve their communities, while officers can use the data mining system, which updates a map periodically with new reports and highlights areas with high violation frequency, to facilitate the process of law enforcement by locating offences and releasing tickets. Authorities will be supplied with a web application with the ability to add new officers, and will also have access to the aforementioned system. A data warehouse will be thus implemented, and repeated reports of violations could also lead to automated traffic tickets being released at the discretion of the authorities. The system will be designed to avoid any kind of voluntary or unpredictable behaviour which may result in invalid reports or wrong fines being released, with a vigorous effort to completely prevent harmful usage of the S2B rather than correcting it a posteriori.

1.2.1 Phenomena table

Phenomenon	Shared	Who controls it?
Occurrency of traffic violation	No	World
Filing a violation report	Yes	World
Registration and login of any type of user	Yes	World
Selection of correct or wrong reports made by authority	Yes	World
Arrival of an officer at the site of the violation	No	World
Confirmation or rejection of a report by the officer	Yes	World
Requesting of bans	Yes	World
Creation of reports composed by photo, date and location	No	Machine
Creation of traffic tickets	No	Machine
Collection and processing of data	No	Machine
Presentation of relevant data about violations and offenders	Yes	Machine
Issuing of bans	Yes	Machine
Officer corfirming the checking of a report	Yes	Machine

1.2.2 Goals

- [**G1**]: Allow future users to easily register and login.
- [**G2**]: Allow users to notify authorities of traffic violations through the use of the camera.
- [**G3**]: Store relevant info about the violation in the data warehouse.
- [**G4**]: Assist authorities in the process of law enforcement:
  - [**G4.1**]: allow automatic ticket compilation at the discretion of the authorities;
  - [**G4.2**]: show processed data relevant to the purpose.
- [**G5**]: Guarantee security:
  - [**G5.1**]: allow discarding of invalid reports;
  - [**G5.2**]: avoid unauthentic officer registrations by design;
  - [**G5.3**]: block people who came in possession of an officer's device to cause any damage;
  - [**G5.4**]: ban users who abuse of the violation reporting system.

- [**G6**]: Guarantee privacy to users.
  - [**G6.1**]: only allow authorities to visualize relevant data for law enforcement purposes.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- `Violation | Offence:` we will strictly refer to any kind of static traffic violation, especially parking violations.
- `Authority:` a law enforcement authority which manages traffic violations, it could be the local police, the traffic wardens etc. We will refer with this term also to the personnel which operates the web application in the authorities' headquarters.
- `Officer:` law enforcement personnel with the authorization to handle traffic violations in the legislation where the application will be deployed. For instance, in Italy these would be Vigili Urbani, Polizia Municipale etv.
- `User:` refers to the users of the mobile application, that is officers and civilians.

1.3.2 Acronyms

- `S2B:` software to be.
- `HFVZ:` high violation frequency zone.
- `GPS:` Global Positioning System.
- `GDPR:` General Data Protection Regulation.
- `DW:` data warehouse.
- `VLA:` Vehicle Licensing Authority.
- `GUI:` Graphical user interface.
- `ETL:` Extract, transform, load.

1.3.3 Abbreviations

- [Gn]: n-goal.
- [Rn]: n-requirment.
- [Dn]: n-domain assumption
- App: application.

1.4 Revision history

- **1/12/2019**: Alloy section revisited and improved.

1.5 Reference Documents

- Specification Document "Mandatory Project Assignment A.Y. 2019/2020"
- 830-1993 - IEEE Recommended Practice for Software Requirements
- "Appunti di Sistemi Informativi per il Settore dell'Informazione" A.Y. 2018/2019
- Alloy Language Reference "http://alloytools.org/download/alloy-language-reference.pdf"
- UML diagrams: https://www.umlÂdiagrams.org/
- "Notes for the Distributed Systems course" A.Y. 2019/2020

1.6 Document Structure

1. `Introduction`: the first section is a general description of the system's scope and its goals. It also includes the revision history of the document and its references. Definitions and abbreviations used along the paper will be provided.
2. `Overall Description`: this section includes shared phenomena, requirements and domain assumptions. It also clarifies users' needs.
3. `Specific Requirements`: this section includes all the requirements, both functional and non functional.
4. `Formal Analysis Using Alloy`: it includes the Alloy model of the described system.

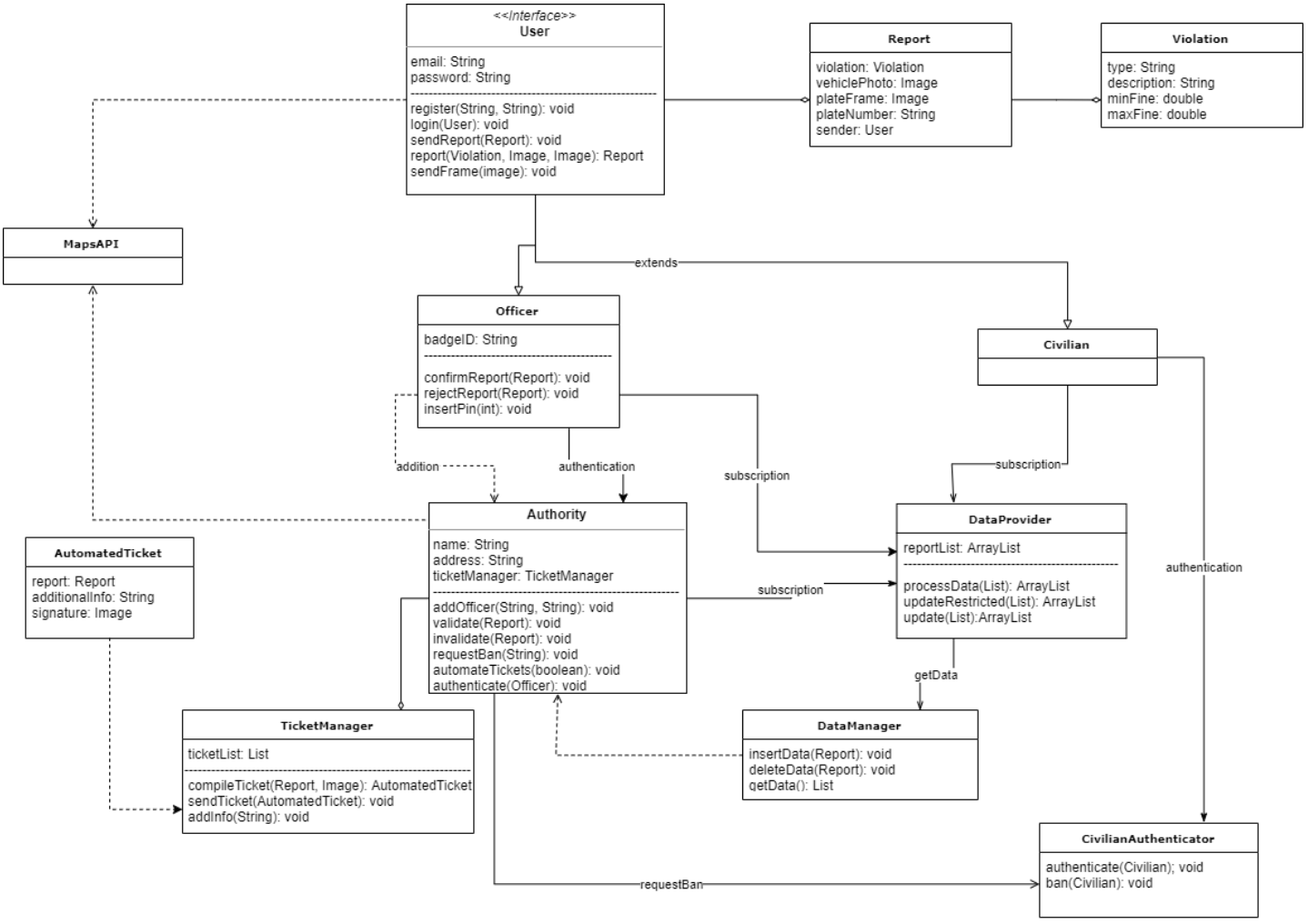
5. `Effort Spent`: this section includes information about the hours spent to compile this document.

6. `References`: this section includes references about papers/documents used to support this document.

2 OVERALL DESCRIPTION

2.1 Product perspective

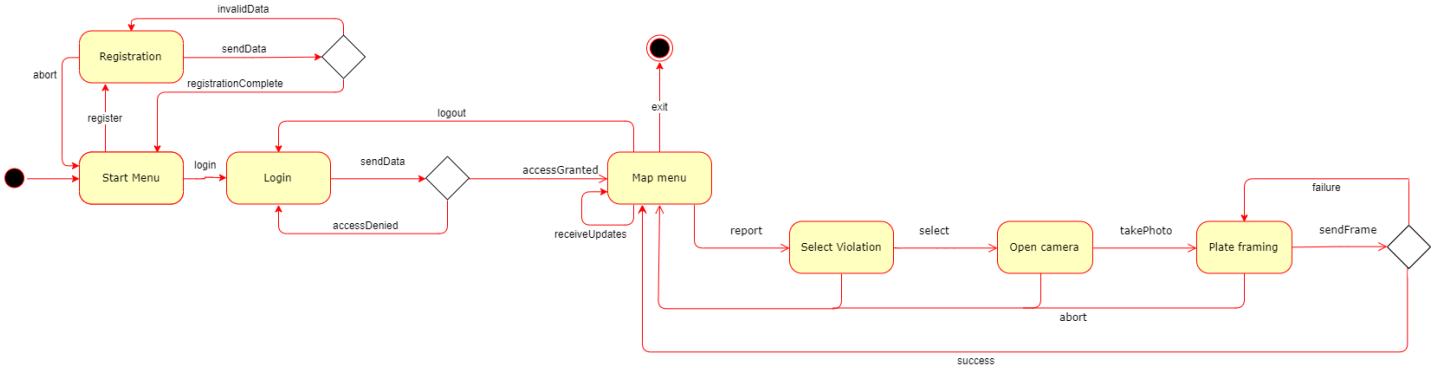
This class diagram shows a bird's eye view of the system. We decided to include more details such as methods to immediately give an idea of how every component will work in the S2B, but this diagram will in no way be the actual implementation of the system, rather a simplified representation:



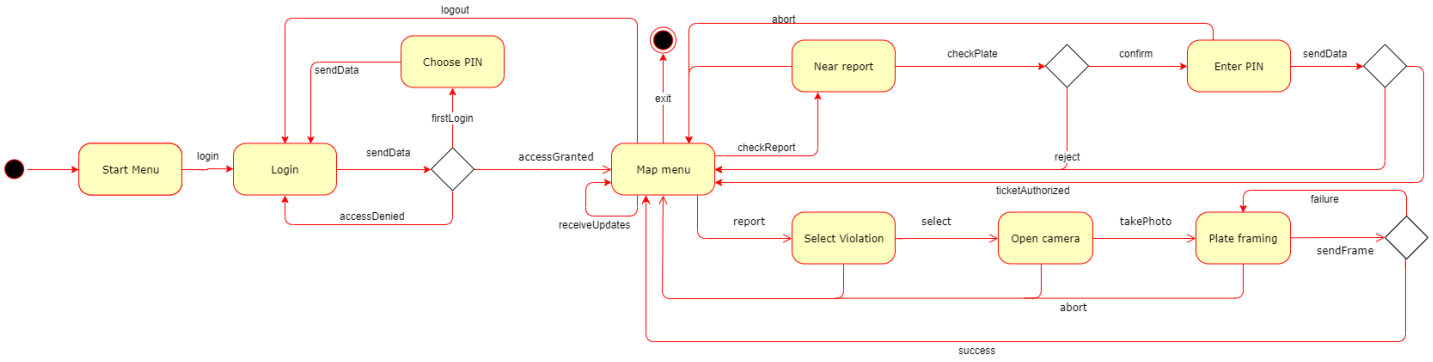
2.1.1 State Diagrams

These state diagrams describe the possibilities of each actor and how their bevahiour will be influenced by the system and external factors.

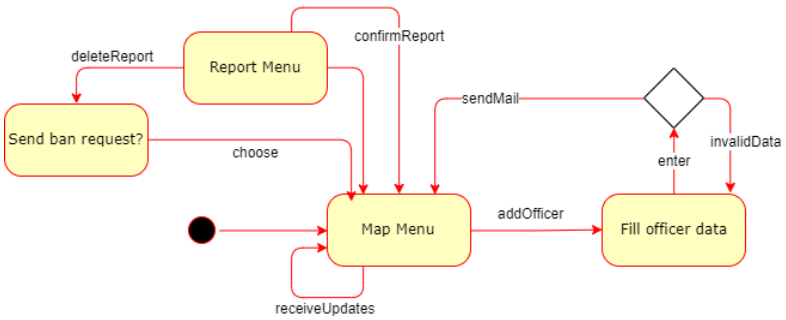
Civilian



Officer



Authority



2.2 Product functions

The goals to be accomplished require the following functions:

- `Registration and login management:` while civilians will be able to register and login in standard fashion through email and password, officers will be added by authorities through their client: an automatically generated password will be sent to their email, and they will be able to access through their badge ID and such password. This way it is impossible for civilians to log in as an officer, as they would have to know both the badge number and the password contained in the officer's personal email.

- `Violation reporting:` allow users (both civilians and officers) to choose between a set of offences and send various pictures representing it, with the corresponding date and location (via GPS and system timestamping) to the authority which will process them and discard incoherent or wrong ones. Photos can only be taken in real time and not uploaded to avoid manipulations.

- `Ban requesting:` if authorities receive undescriptive photos or in any shape or form useless reports, they can send a request from the authorities' client to the SafeStreets HQ's server to ban the abusing user.

- `Data storage, mining and visualization:` store all the relevent data from the reporting procedure to process it through ETL and present it to the authority to assist them in law enforcement. The authority will subsequently take the role of spreading the updates to mobile application when fetched. The most important functionality will be periodically updating the map in the officers' and authorities' GUIs with new violations and hightlightings of HFVZs, and also listing repetead offenders. Also civilians will have access to a restricted version of the mined data.

- `Automated ticket emission:` officers can locate violations through the map, and through a unique PIN sent to the authority, sign an automatically compilated ticket which will be emitted and sent via email to the Vehicle Licensing Agency. The PIN is to make sure that even if the officer's mobile is stolen while he's logged in, the thief cannot emit tickets.

2.3 User characteristics

- `Visitors:` users who haven't yet registered or logged in the mobile application, they are only able to register as a civilian or log as a civilian (through email and password) or as an officer (through Badge ID and password issued by the authority). They interact with:
  - Registration and login management.
- `Civilians:` users unaffiliated to law enforcement authorities who are interested in improving their community, they are only able to report violations but have access to a restricted version of the mined data. They're required to be registered and log in the mobile application so they can eventually be excluded in case of security violations. They interact with:
  - Registration and login management;
  - Violation reporting;
  - Data storage, mining and visualization: restricted version of the data.
- `Officers:` they're registered to the mobile application through their authority listing by their badge ID them through the web application and receiving a password. They're able to see the mined data, report violations, and sign the automated tickets after investigating the reports on the map. They interact with:
  - Registration and login management;
  - Violation reporting;
  - Data storage, mining and visualization;
  - Automated ticket emission: as they're the entity which checks the violation and authorizes the emission of the ticket.
- `Users:` we will use this name only as a general way to refer to both registered officers and civilians when we're describing actions they are both able to perform in the same way (like compiling violation reports).
- `Authorities:` they're manually registered by SafeStreets personnel, and their work is managed in office by an officer using the SafeStreets web application. They can add new officers and they're able to see the processed data, enable automatic ticket emissions, receive violation reports and thus discriminate between valid and invalid ones. They

can request bans which the SafeStreets server will take care of issuing and can see every officer connected to the application. They interact with:

- Registration and login management: for officer registration and authentication;
- Violation reporting;
- Data storage, mining and visualization;
- Automated ticket emission;
- Ban requesting.

2.4 Domain assumptions

- [**D1**]: The officer's Badge ID is assumed to be unique.
- [**D2**]: Users are assumed to provide a valid email.
- [**D3**]: At least one of the city's authorities' server is always online to process reports and officer logins.
- [**D4**]: Users' devices support the Mobile Application
- [**D5**]: Users' devices camera and GPS work correctly.
- [**D6**]: Email from SafeStreets containing the confirmation link for registration is always received correctly.
- [**D7**]: Email from the authority containing the officer's password is always received correctly.
- [**D8**]: Authorities' devices support the web application.
- [**D9**]: Authorities correctly receive the great majority of reports.
- [**D10**]: The web application at the authorities' location is operated by an assigned officer most of the time.
- [**D11**]: Officers only sign correct tickets and always check the license plate matches with the reported one.
- [**D12**]: Location is calculated correctly by the device with at most 5 meters range from the user's position.
- [**D13**]: Only the officer knows his personal pin that allows him to proceed in ticket compilation.
- [**D14**]: The personnel operating the Authority's client discards most of the invalid reports.
- [**D15**]: Every registered officer's signature is stored at the authority in an image file.

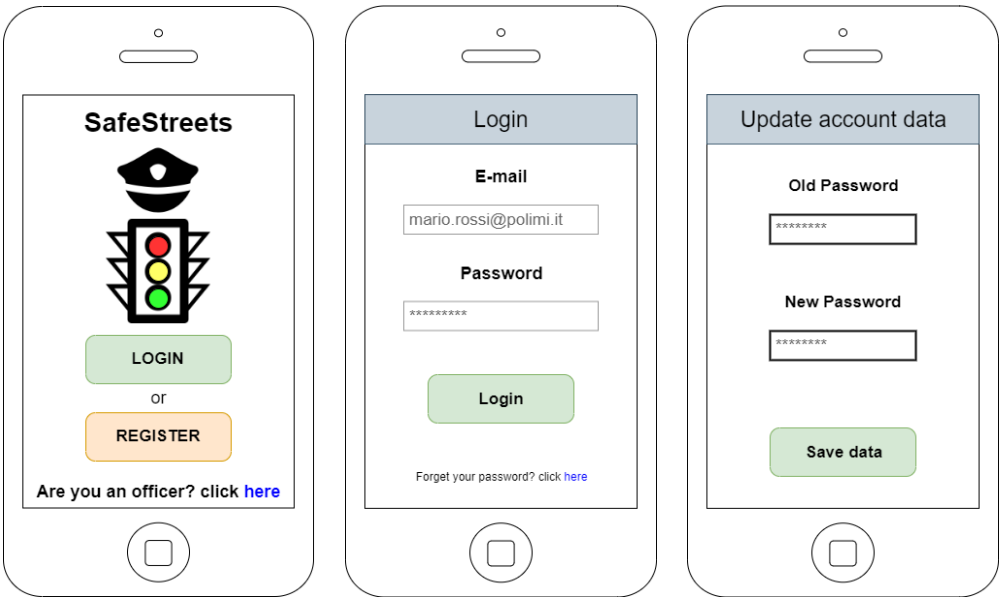
3 SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 User Interfaces

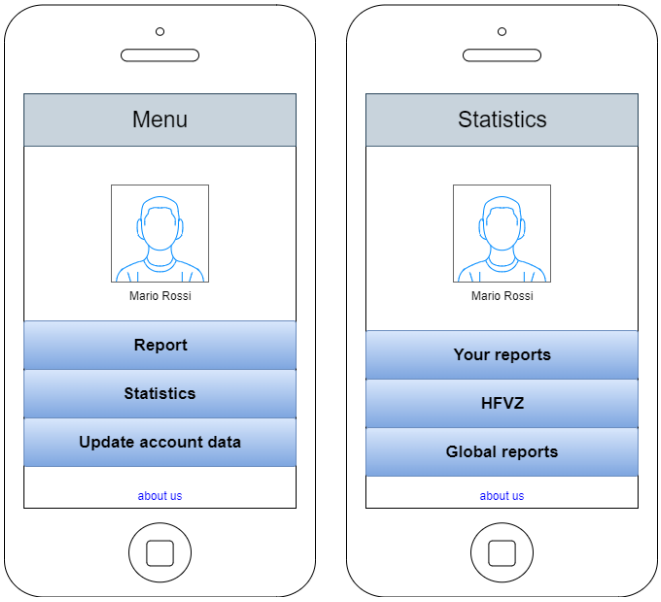
The following mockups give a basic idea of how the Mobile Application and the Web Application are supposed to look like.

- **Civilian**: they can access SafeStreets functionalities through the mobile application.



(a) *Main Page* (b) *Login* (c) *Data update*

Figure 1: Login, Registration

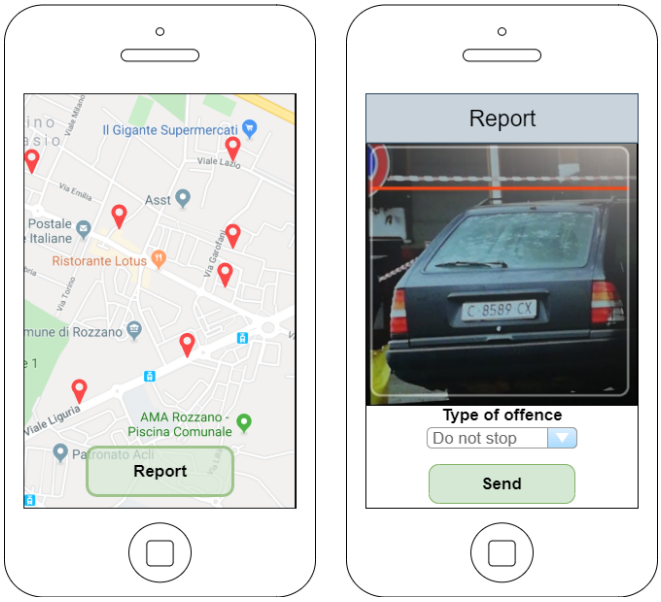


(a) *Main menu* (b) *Statistics menu*

Figure 2: User's Menu

(a) *Your reports* (b) *HFVZ* (c) *Global Reports*

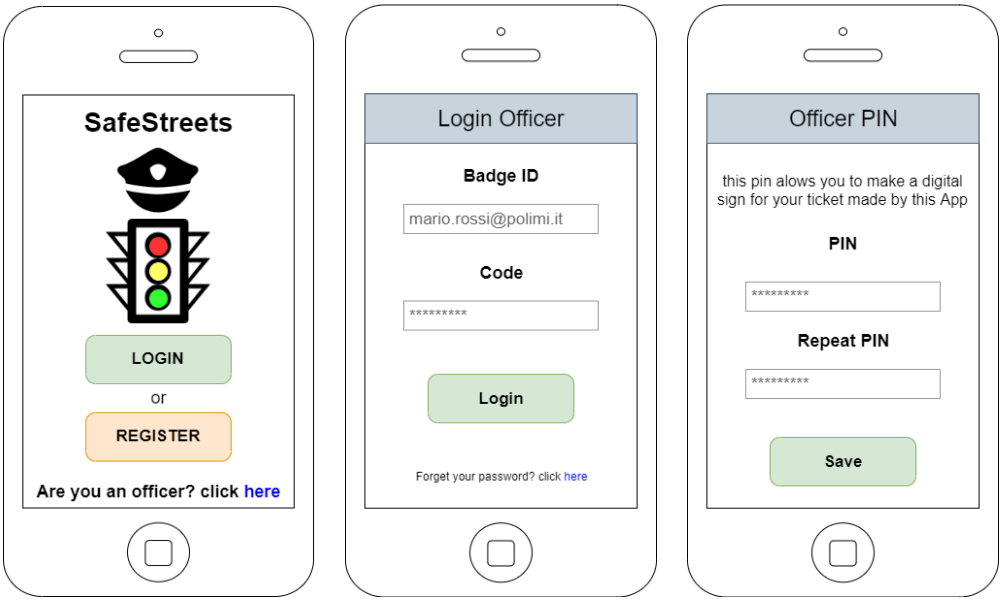
Figure 3: User's Statistics



(a) *Map report* (b) *Make report*

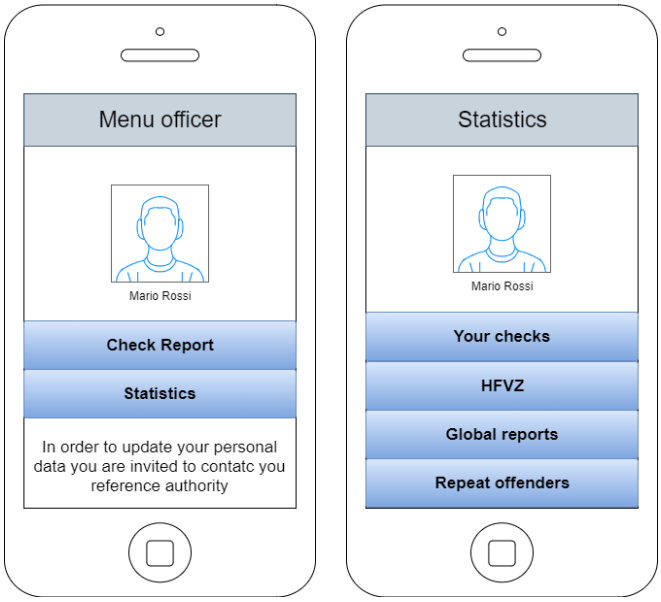
Figure 4: Report

- **Officer**: over than civilian functionalities, they can also investigate the reports and sign automated tickets.



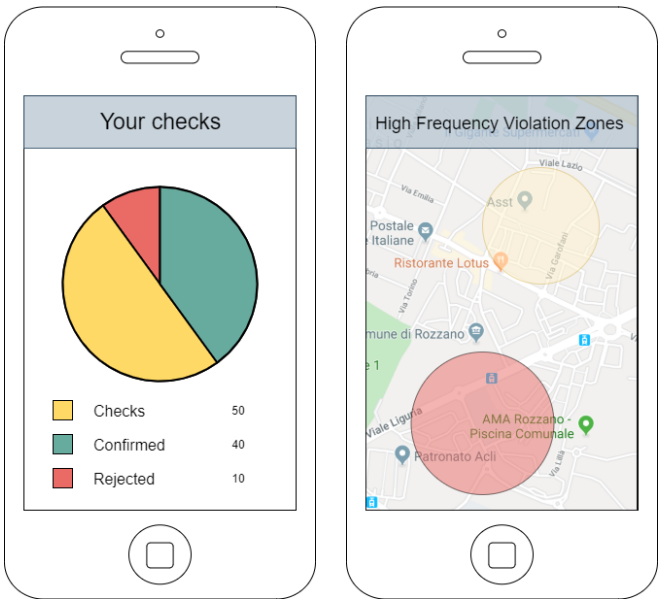
(a) *Main Page* (b) *Login* (c) *Officer PIN*

Figure 5: Login, Registration, PIN



(a) *Main menu* (b) *Statistics menu*

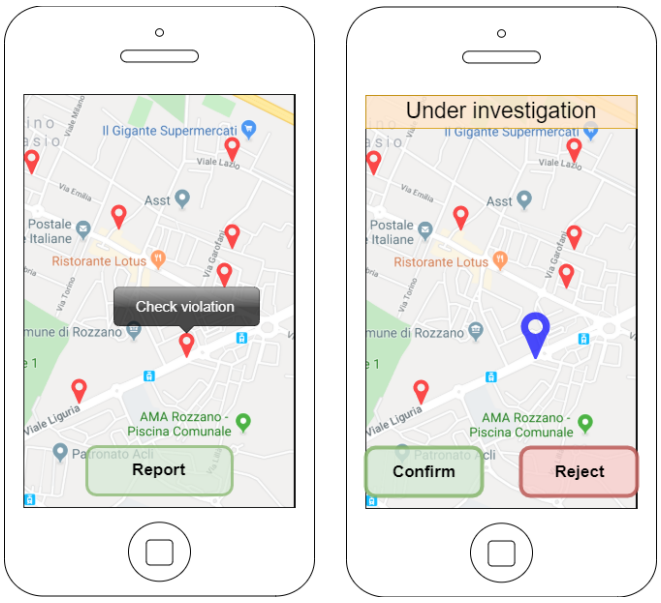
Figure 6: Officer's Menu



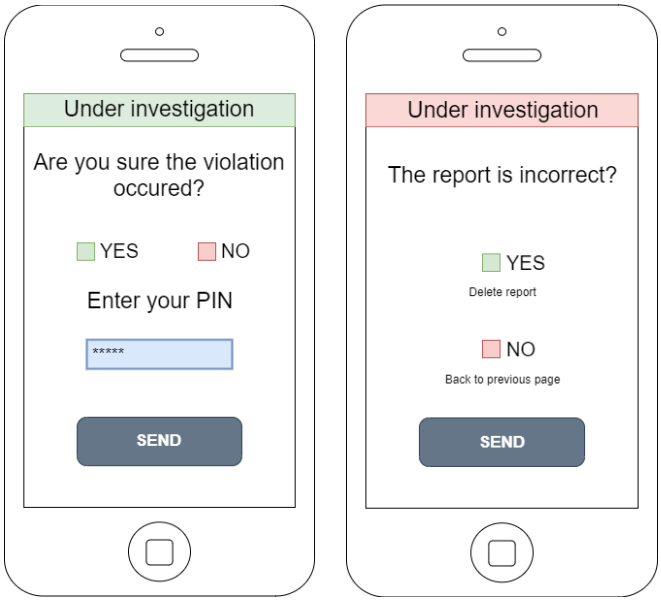
(a) *Your checks* (b) *HFVZ*

(c) *Global Reports* (d) *Repeat Offenders*

Figure 7: Officer's Statistics



(a) *Check violation* (b) *Under investigation*



(c) *Violation confirmed* (d) *Violation Rejected*

Figure 8: Investigation

- **Authority**: SafeStreets provides a Web Application for law enforcement authorities where they can validate reports, request bans, toggle the automated ticket emission functionalities on or off, and most importantly add new officers.

Figure 9: Web App

3.1.2 Hardware Interfaces

Since the application must run over the Internet, all the hardware shall require to connect network will be an hardware interface for the system, both server and client side.

- Server-side: e.g. Modem, WAN - LAN, Ethernet Cross-Cable.
- Client-side: e.g. Wi-Fi 802.11ac+ antenna, 3G/4G antenna.

3.1.3 Software Interfaces

SafeStreets will provide a very basic API for authorities to customize data about tickets and fines (so that the functionality can fit various legislations) and other minor parameters such as report expiration time. The `Google Maps API` will be used to implement the map and geolocalization services. `Hibernate` will be used to handle database accesses in all components, thus providing a way to interact with the chosen DBMS. Also, some API will be programmed to allow for email compilation and sending through some mailing service when emails are needed. More details will follow in the `Design Document`.

3.1.4 Communication Interfaces

HTTP in a RESTful client-server paradigm will be used for communication between the nodes of the network along with the most recent TLS version to secure messages containing sensitive data.

3.2 Scenarios

This section describes some of the scenarios in which SafeStreets may be used. Scenarios omit steps already described in previous ones:

3.2.1 User registration

Giancarlo saw the advertisement of SafeStreets and decided to download the mobile application to report a car that always parks on the sidewalk outside his house and blocks the passage of pedestrians. After downloading and launching the new app, he is asked to fill a registration form in which he must insert his email address and a password. After clicking the "Sign In" button he receives an email with a link to confirm his registration, which he clicks to finalize his registration. From now on he can login and report offences with the "Report" button in the main screen.

3.2.2 Correct violation reporting

Mary just logged in as a civilian to report a double-parked car. Since it's her first use of the application, she's asked to allow SafeStreets to access her GPS and camera, to which she complies. She clicks "Report" on her main screen and chooses a violation from a dropdown menu. The app then opens her camera asking to take a photo of the vehicle showing the offence being committed. Then she's asked to frame the license plate with her camera, and the supervised learning algorithm scans the various frames looking for an interpretation for the image. The app then associates the report with a timestamp and a GPS position, and sends it to one of the authorities' servers. After a couple of minutes, Mary receives a notification informing her the report was correctly accepted and processed.

3.2.3 Wrong violation reporting

Simon decides to use the app to report his neighbour's vehicle because he doesn't like him. He sends a photo which in fact doesn't show any violation being committed. Arthur, a policeman who's working with the SafeStreets web application at his barracks, receives the report and notices there's clearly no violation being committed in the photo, and clicks on the "Request of Ban" button which alerts the SafeStreets server of an invalid report, and discards the report immediately after. After a couple of minutes, Simon receives a notification telling him he's been forbidden from using the reporting function for a month.

3.2.4 Officer registration

Billy is a policeman whose barrack has just installed the SafeStreets application and data server. He received an email notifying him that he may be enabled to utilize the service. After a while, he receives an email from his barrack's SafeStreet client containing an automatically generated code. By entering his badge ID and this code in the officer login section of the mobile application, Billy is now able to login when he desires by using the code as password. Once logged in, Billy will have to enter a 5-digit pin that will allow him to make a traffic ticket signed by him when he receives a report. Bill now sees the map of his surroundings, periodically updated with new traffic offence reports and with HFVZ highlighted.

3.2.5 Correct Ticket Emission

John was on patrol this morning and checked on the SafeStreets App if there was any report in his vicinity. As soon as he logged in he noticed a reported violation in the map, about 250 meters from his position. John clicks the "Accept" button on the tooltip to confirm to other active officers he's checking the authenticity of the violation, and avoiding any other one of replicating this task. He now sees the photos of the reporting and the type of offence. Once he's near the vehicle, a "Confirm" button pops up. John checks that the vehicle is effectively committing a violation and the license plate matches the one in his screen. Once John clicks it and enters his secret 5 digit pin his authority's server will be able to place his signature into an automatically generated ticket that will be sent to the Vehicle Licensing Authority. Now the violation tooltip can disappear and John goes on with his duties.

3.2.6 Wrong report discarded by an officer

Gabriele, who is an officer, is walking towards an offence reported from SafeStreets. As soon as he arrives there, he realizes that the report involved a car parked on the pedestrian crossing which is no longer there. So he decides to click on the "Delete" button which subsequently communicates to the authority's server to remove it from the map.

3.3 Requirements

Here we include a list of both functional and non-functional requirements which will allow the efforts of the team to be concretely directed to the goals.

3.3.1 G1: Allow future users to easily register and login.

- **R1**: Separate civilian and officer registration and login functionality;
- **R2**: Data has to be correctly queried server-side everytime, to make sure no duplicate accounts are created or no unregistered visitors log in;
- **R3**: Send confirmation emails within at most one minute after the registration confirmation receipt (or the new officer being added);
- **R4**: Confirm user registration within at most one minute after the email link has been clicked.

- D1: The officer's Badge ID is assumed to be unique;
- D2: Users are assumed to provide a valid email;
- D3: At least one of the city's authorities' server is always online to process reports and officer logins.
- D4: Users' devices support the Mobile Application;
- D6: Email from SafeStreets containing the confirmation link for registration is always received correctly.
- D7: Email from the authority containing the officer's password is always received correctly.

3.3.2 G2: Allow users to notify authorities of traffic violations through the use of the camera.

- **R5**: Force the user to allow the S2B to access the device's camera and GPS;
- **R6**: Implement a function in the authority's client which allows personnel to validate or invalidate reports;
- **R7**: Implement a supervised learning algorithm server-side which scans the multiple frames sent by the user to recognize the license plate;
- **R8**: Automatically discard the report if another one with the same plate has been received within an hour to avoid duplicates;
- **R9**: The supervised learning algorithm must be accurate at least 99% of the times;
- **R10**: Acknowledgments from the authority's server must be received within 10 seconds.
- D3: At least one of the city's authorities' server is always online to process reports and officer logins.
- D4: Users' devices support the Mobile Application;
- D5: Users' devices camera and GPS work correctly.
- D8: Authorities' devices support the web application.
- D9: Authorities correctly receive the great majority of reports.

- `D10`: The web application at the authorities' location is operated by an assigned officer most of the time.
- `D12`: Location is calculated correctly by the device with at most 5 meters range from the user's position.

3.3.3 G3: Store relevant info about the violation in the data warehouse.

- **R11**: Store the report info in the DW everytime it is confirmed to be valid;
- **R12**: Make the data warehousing and processing completely invisible to human actors to avoid manipulations;
- `D3`: At least one of the city's authorities' server is always online to process reports and officer logins.
- `D8`: Authorities' devices support the web application.
- `D10`: The web application at the authorities' location is operated by an assigned officer most of the time.
- `D12`: Location is calculated correctly by the device with at most 5 meters range from the user's position.
- `D14`: The personnel operating the Authority's client discards most of the invalid reports.

3.3.4 G4: Assist authorities in the process of law enforcement.

- `G4.1`: allow automatic ticket compilation at the discretion of the authorities:
  - **R13**: implement functionality which allows (through the authority's client) to toggle automatic tickets ON or OFF;
  - **R14**: force newly logged officers to choose a 5 digit pin to release tickets;
  - **R15**: implement functionality which allows officers' near a report to confirm both the report itself and the ticket emission through the pin;

- **R16**: implement functionality which uses report data and the officer's signature to compile a ticket and send it to the VLA via email.

`G4.2`: show processed data relevant to the purpose:

- **R17**: update every actor's map with the HFVZs every 30 seconds;
- **R18**: update only the authorities' and officers' maps with the new reports and repeat offenders lists in real time;
- **R19**: show repetead offenders in real time in the authorities' GUI.

- `D3`: At least one of the city's authorities' server is always online to process reports and officer logins.
- `D4`: Users' devices support the Mobile Application;
- `D8`: Authorities' devices support the web application.
- `D11`: Officers only sign correct tickets and always check the license plate matches with the reported one.
- `D12`: Location is calculated correctly by the device with at most 5 meters range from the user's position.
- `D14`: The personnel operating the Authority's client discards most of the invalid reports.
- `D15`: Every registered officer's signature is stored at the authority in an image file.

3.3.5 G5: Guarantee security.

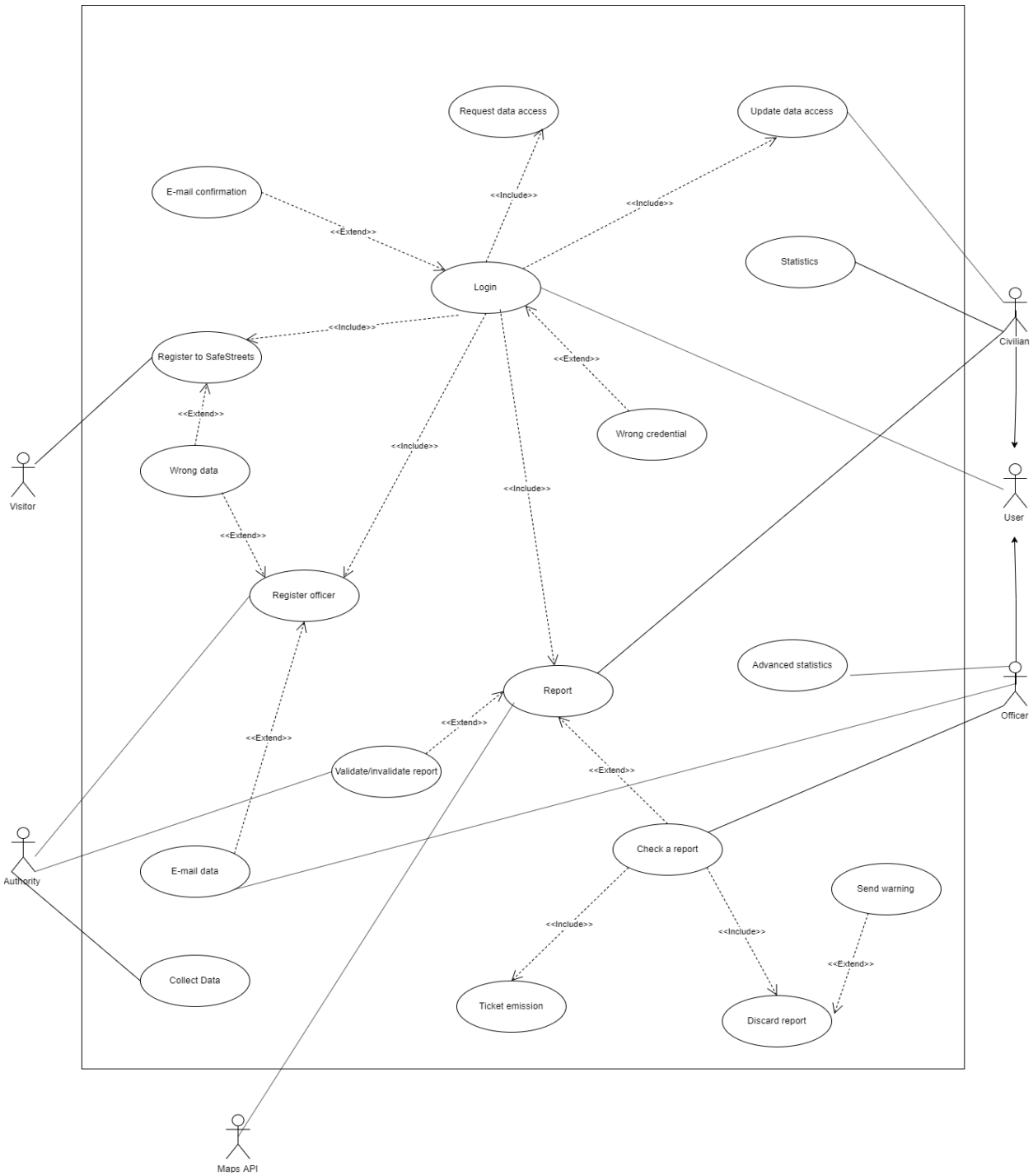
- `G5.1`: allow discarding of invalid reports:
  - **R6**: Implement a function in the authority's client which allows personnel to validate or invalidate reports;
  - **R7**: Implement a supervised learning algorithm server-side which scans the multiple frames sent by the user to recognize the license plate;
  - **R8**: Automatically discard the report if another one with the same plate has been received within an hour to avoid duplicates;
  - **R9**: The supervised learning algorithm must be accurate at least 99% of the times.

- `G5.2`: avoid unauthentic officer registrations by design:
  - **R1**: Divide civilian and officer registration and login functionality;
  - **R20**: Make sure officers can only be added through the Authority's client;
  - **R21**: Officers can only log into the app through a combination of Badge ID and a password received via email by the Authority.
- `G5.3`: block people who came in possession of an officer's device to cause any damage:
  - **R21**: Officers can only log into the app through a combination of Badge ID and a password received via email by the Authority.
  - **R22**: Officers can only allow automated ticket emission by entering their secret pin;
  - **R23**: Officers are automatically required to login every 6 hours and their access data can't be autofilled.
- `G5.4`: ban users who abuse of the violation reporting system:
  - **R24**: A ban request can be issued through the Authority's client after receiving an invalid report.
- `D3`: At least one of the city's authorities' server is always online to process reports and officer logins.
- `D4`: Users' devices support the Mobile Application;
- `D8`: Authorities' devices support the web application.
- `D13`: Only the officer knows his personal pin that allows him to proceed in ticket compilation.
- `D14`: The personnel operating the Authority's client discards most of the invalid reports.
- `D15`: Every registered officer's signature is stored at the authority in an image file.

3.3.6 G6: Guarantee privacy to users.

- `G6.1`: only allow authorities to visualize relevant data for law enforcement purposes.
  - **R18**: Update only the authorities' and officers' maps with the new reports and repeat offenders lists in real time;
  - **R25**: The S2B must comply with GDPR rules;

3.4 Use case Diagram



3.5 Use cases

These use case tables describe examples of how the actors may interact with the system to perform the previously listed tasks. An use case about the data collection, processing and representation system has been omitted as the functionality is fully automatized thus doesn't include any actor.

3.5.1 Civilian Visitor Registration

NAME	Civilian registration
ACTOR	Visitor
GOALS	[G1]
ENTRY CONDITIONS	The Visitor has installed the Application on his/her Mobile device
EVENTS FLOW	1. Fill all mandatory fields in the Civilian Registration Form 2. The visitor clicks on "Register" button 3. The system checks data has not been used already 4. The system sends an email as confirmation 5. The system saves Users' data
EXIT CONDITIONS	The Visitor has clicked in the email's link
EXCEPTIONS	1. The email is already registered 2. Some mandatory fields are not filled

3.5.2 Civilian Login

NAME	Civilian Login
ACTOR	Civilian
GOALS	[G1]
ENTRY CONDITIONS	The civilian is in the Civilian Login page
EVENTS FLOW	1. The civilian enters his email 2. The civilian enters the password 3. The civilian clicks on "Login" button 4. The system checks the correctness of the credentials
EXIT CONDITIONS	The civilian has successfully logged in
EXCEPTIONS	1. The civilian is not registered 2. The email is wrong 3. The password is wrong 4. Some mandatory fields are not filled

3.5.3 Officer Registration

NAME	Register to SafeStreets
ACTOR	Authority
GOALS	[G1]
ENTRY CONDITIONS	1.The authority has installed the web application 2. The authority has a correctly configured server
EVENTS FLOW	1. Launch the web application 2. The authority personnel clicks on the "Officers Registration" button 3. The authority adds the officer's badge ID and email 4. The server checks the data has not been used already 5. The server checks if a signature for that badge ID is present in the database 6. The server sends an email with an automatically generated password to the officer 7. The system saves the officer's data
EXIT CONDITIONS	The email has successfully been sent to the officer
EXCEPTIONS	1. The badge is already registered 2. The email is already registered 3. A signature is not present 4. Some mandatory fields are not filled

3.5.4 Officer First Login

NAME	Officer Login
ACTOR	Officer
GOALS	[G1] , [G.5.2]
ENTRY CONDITIONS	1. The officer has downloaded the mobile application 2.The officer has seen the password in his email 3.The officer is in the Officer Login page
EVENTS FLOW	1. The officer enters his badge ID 2. The officer enters his password 3. The officer clicks on "Login" button 4. The system checks the correctness of the credentials 5. The officer chooses a 5 digit pin to emit tickets
EXIT CONDITIONS	The officer has successfully logged in
EXCEPTIONS	1. The officer is not registered 2. The badge ID is wrong 3. The password is wrong 4. Some mandatory fields are not filled

3.5.5 Violation reporting

NAME	Violation reporting
ACTOR	User
GOALS	[G2] ,[G4], [G.5.1]
ENTRY CONDITIONS	1.The user is logged in 2. The user has allowed the app to use the camera and GPS
EVENTS FLOW	1. The user clicks on "Report" 2. The user chooses a violation from the dropdown menu 3. The user takes a pic of the vehicle and the violation being committed
	4. The user frames the license plate 5. The system computes the plate number 6. The system checks if the report is a duplicate 7. The system adds date and timestamp and sends the report to the authority
EXIT CONDITIONS	The report has successfully been sent
EXCEPTIONS	1. The plate can't be recognised 2. The report is a duplicate 3. GPS is turned off 4. Camera can't be used for any reason

3.5.6 Ticket confirmation

NAME	Ticket confirmation
ACTOR	Officer
GOALS	[G2] ,[G4], [G.5.3]
ENTRY CONDITIONS	1. The Officer is logged in 2. The officer is near a reported violation
EVENTS FLOW	1. The officer checks the plate number matches the one contained in the report 2. The officer inserts his pin 3. The officer clicks on the "Send" button 4. The system checks the pin's correctness 5. The system sends a notice to the authority 6. The system deletes the report from the map
EXIT CONDITIONS	The confirmation has successfully been sent
EXCEPTIONS	1.The pin is wrong 2. The plate doesn't match

3.5.7 Report validation

NAME	Report validation
ACTOR	Authority
GOALS	[G2] ,[G3], [G.5.1]
ENTRY CONDITIONS	1. The Authority is online 2. The authority received a violation report
EVENTS FLOW	1. The authority checks the photo correctness 2. The authority clicks "Validate" 3. The system adds the report to the data warehouse 4. The system adds the report to the map 5. The server sends notifications to active officers.
EXIT CONDITIONS	The report has been validated
EXCEPTIONS	None

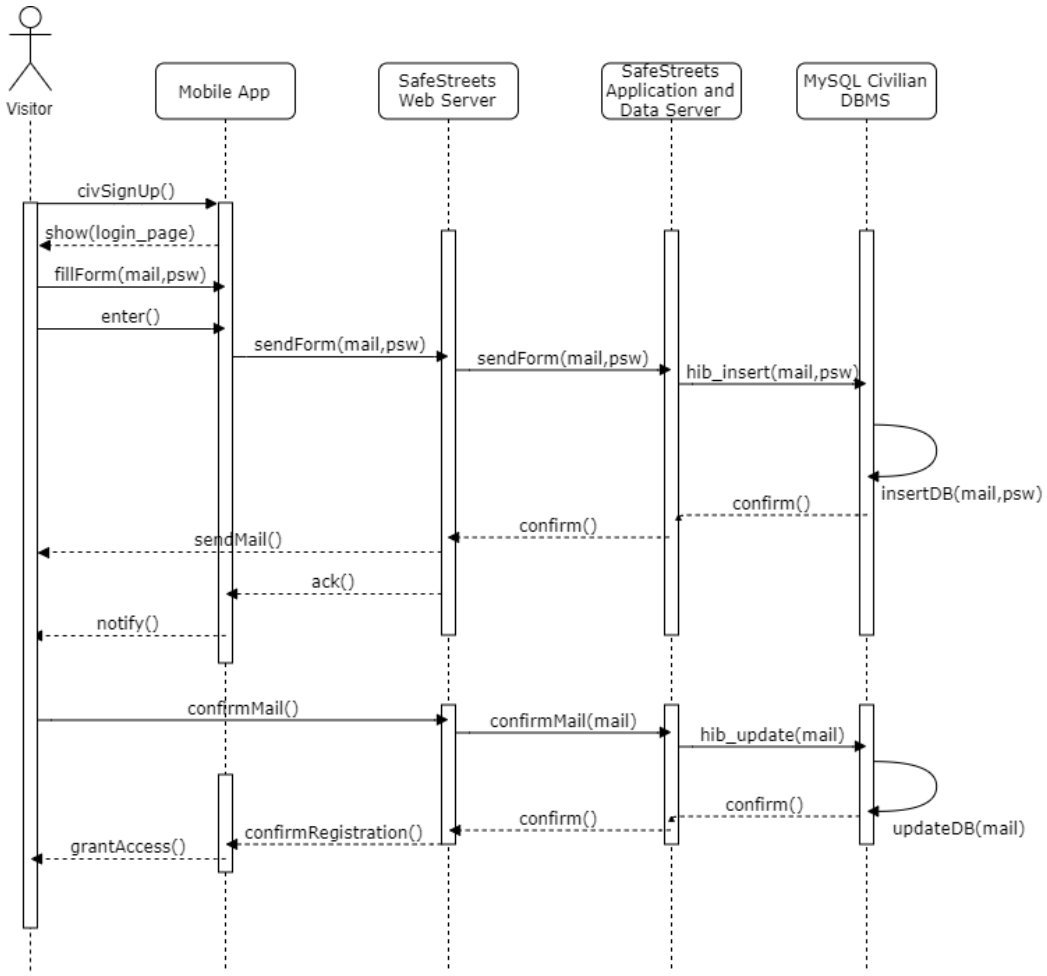
3.5.8 Report invalidation with warning

NAME	Report invalidation
ACTOR	Authority
GOALS	[G2] ,[G3], [G.5.1]
ENTRY CONDITIONS	1. The Authority is in the main menu 2. The authority received a violation report
EVENTS FLOW	1. The authority checks the photo correctness 2. The authority clicks "Validate " 3. The authority clicks on "Request of Ban" 4. The server sends a ban request to the SafeStreets server
EXIT CONDITIONS	The report has been invalidated
EXCEPTIONS	None

3.6 Sequence diagrams

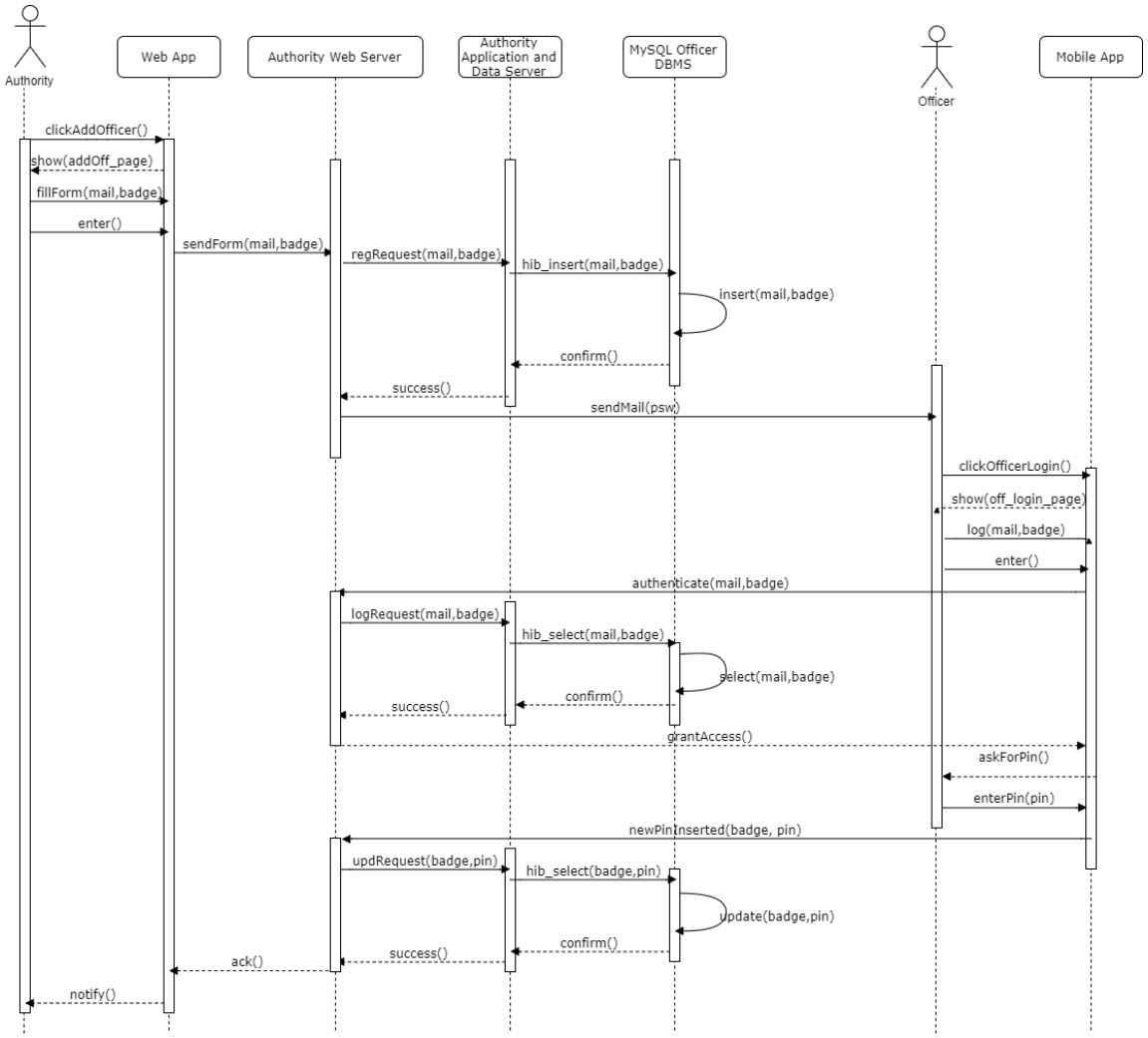
These diagrams will be better detailed in the `Design Document`, they're included here to better provide a model to describe the interactions between software components and actors.

3.6.1 Visitor to Civilian Registration

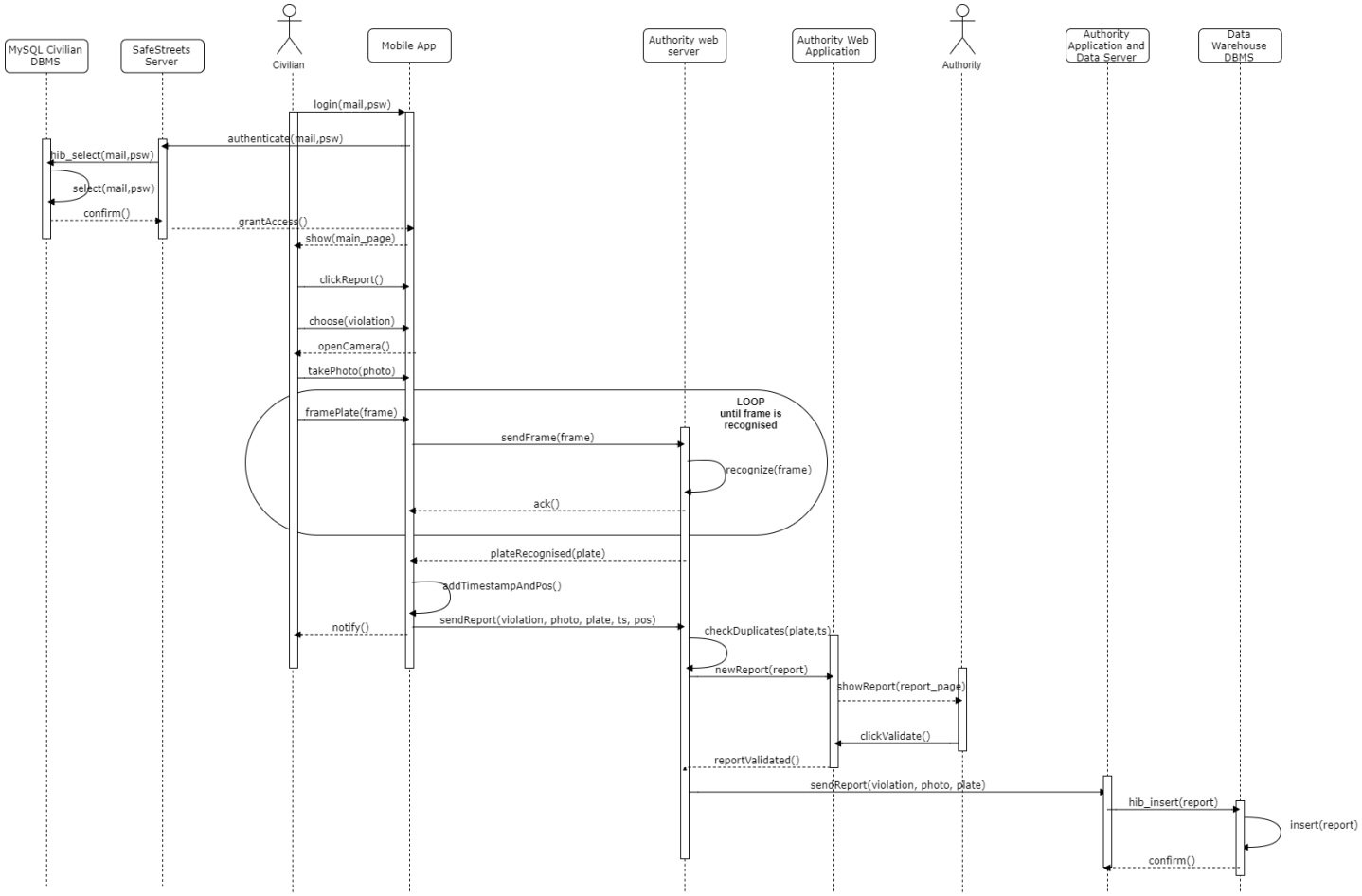


3.6.2 Officer Registration

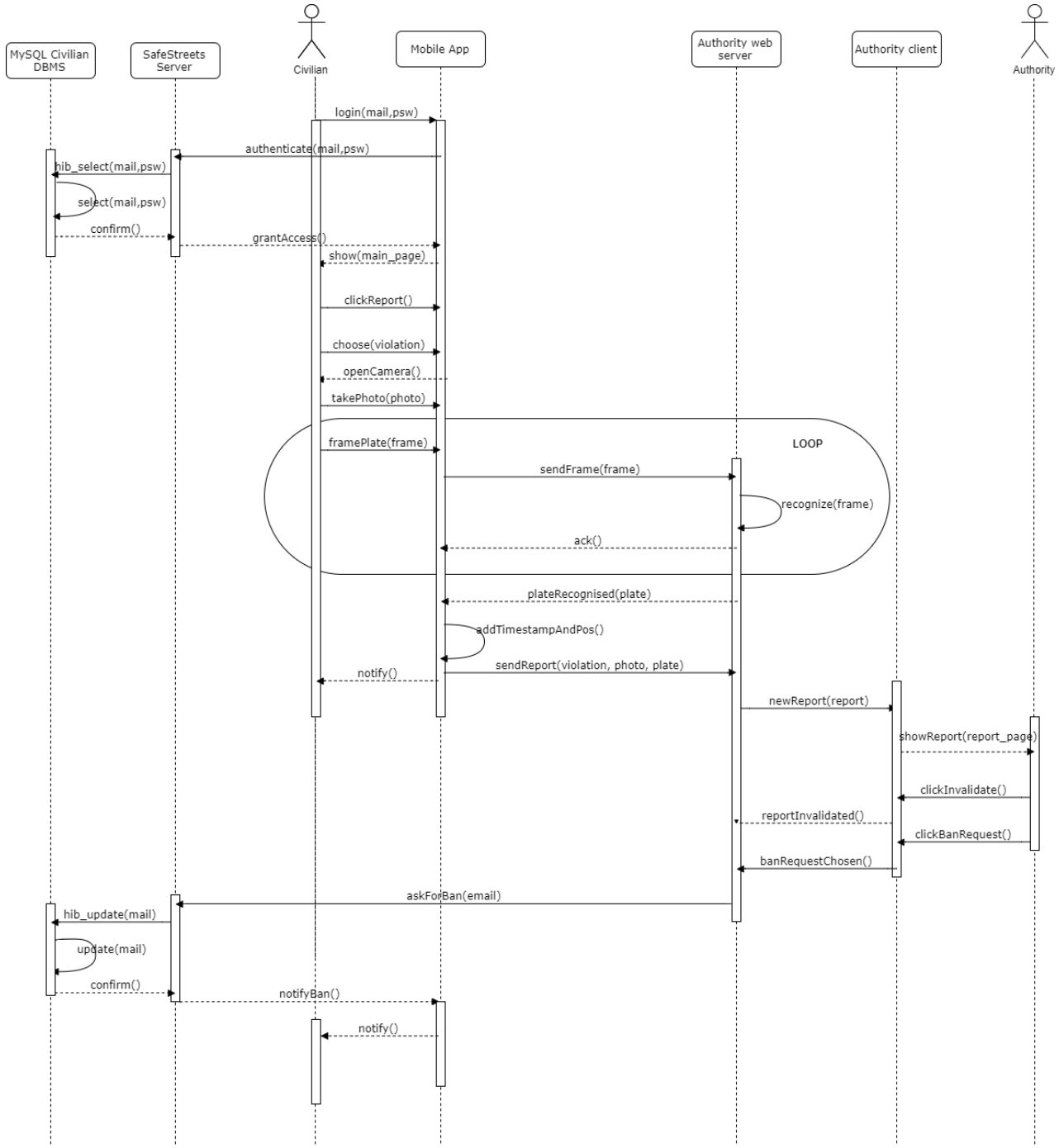
Also includes the first login in which the 5 digit pin is compiled, to complete the registration sequence.



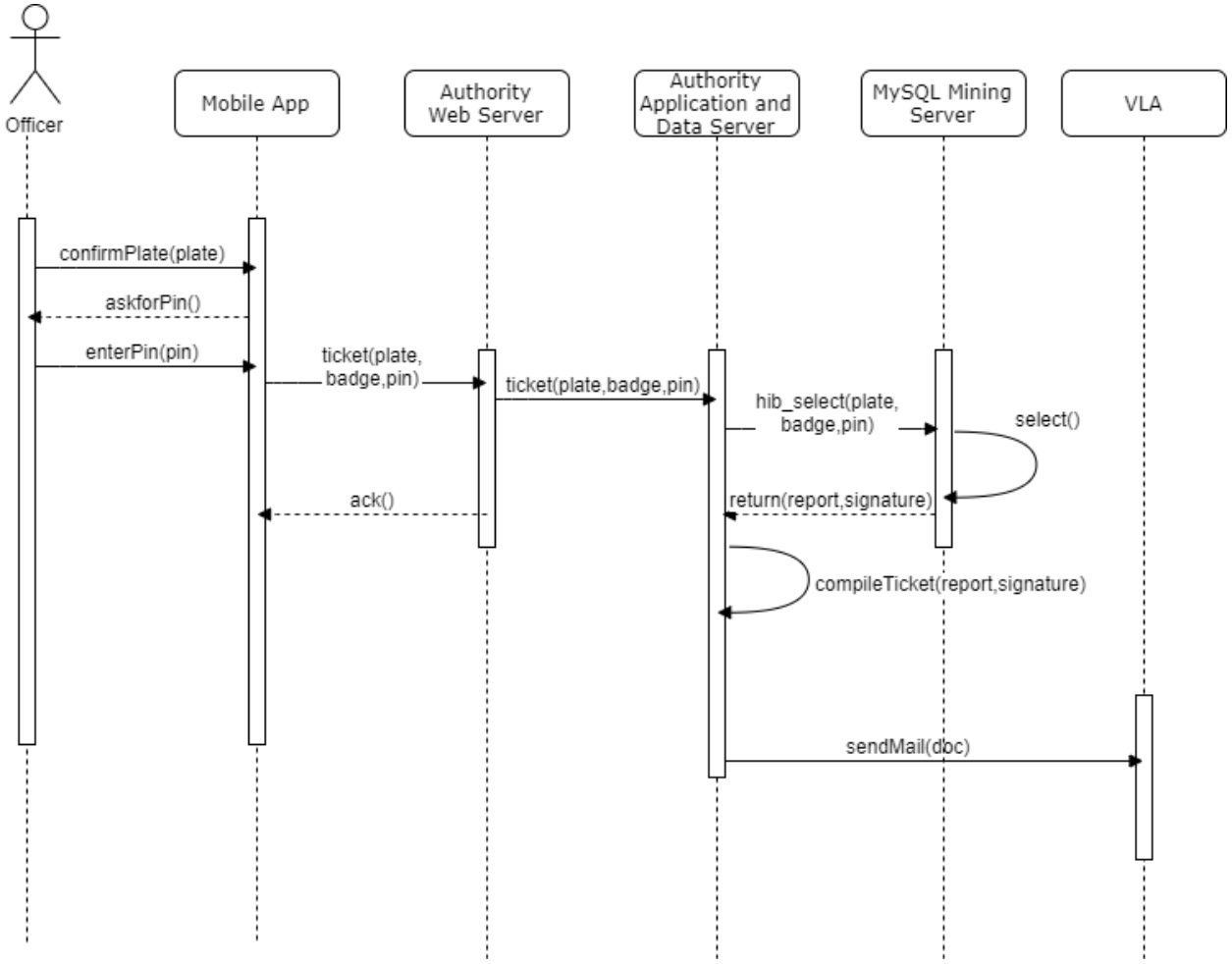
3.6.3 Civilian login and correct violation report



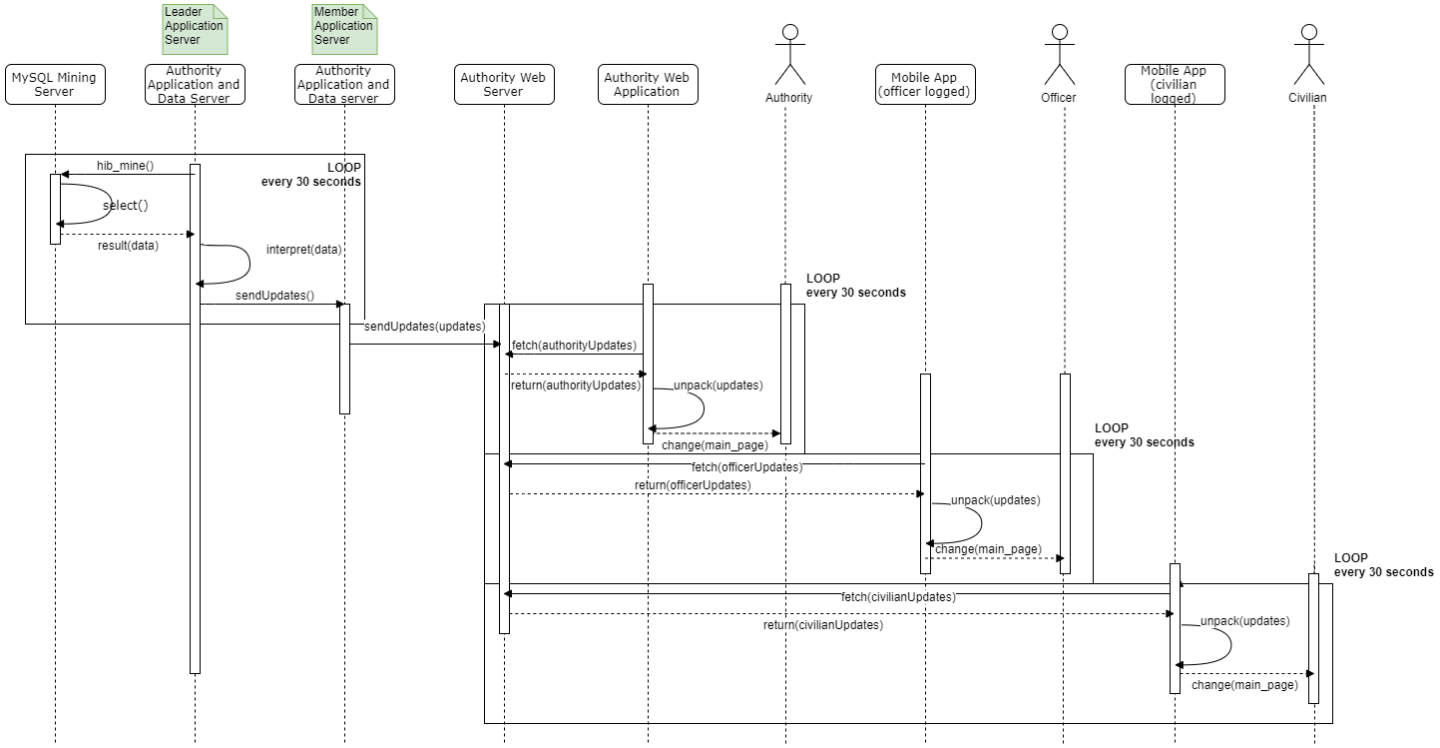
3.6.4 Invalid violation with subsequent banning



3.6.5 Automated Ticket



3.6.6 Update sending



3.7 Performance Requirements

In general, the software should be an helpful tool for authorities and thus its efficient use strongly depends on their ability to use it readily. By assuming reliable links and decent performance of network protocol communications, mailing services etc. the throughput of the system mostly depends on the personnel operating the web application and the officer (as reports cannot be converted into fines without their validation). In the presence of multiple reports and multiple authorities processing them, like what would possibly happen in big cities, the system has nonetheless to guarantee the time bounds specified in the requirements for the sending of new processed data, and especially swiftly notify new reports. Reports must be guaranteed to expire after a time limit has expired, to make sure that no effort is wasted by officers in checking old ones.

3.8 Design Constraints

3.8.1 Hardware Limitations

- `Mobile App`:
  - Smartphones: Android or iOS, Internet Connection, GPS, Camera.
- `Web App`: modern web browser (e.g. Google Chrome / Safari) able to support a Rich Internet Application, Internet connection.

3.8.2 Standards compliance

The system uses the International System of Units as it system of measurements and the sensitive data will be treated according to the General Data Protection Regulation rules.

3.9 Software System Attributes

3.9.1 Reliability

The system should always be able to process reports and store them in the data warehouse, but according to G1 the priority is assisting law enforcers and not substituting the existing procedures, thus transient and occasional deviations can be accepted. Reports can be processed by multiple authorities' servers if the previous one fail (e.g. every police barrack has a server), so replication will be used to guarantee both availability and reliability.

3.9.2 Availability

The system must fullfil all the registration, login requests and reports whenever needed, only a small percentage of requests missed (around 0.1%) can be tolerated.

3.9.3 Security

Officers information is stored in the authorities' data server and not in SafeStreets', which only hosts civilian accounts. Through the badge-password authentication and the fact that only authorities can add officers, it will be impossible for civilians to log in as officers, and even if in some way someone who is not an officer is able to access to an officer account, he will not be able to do any substantial harm because of the pin required to sign tickets. A timeout will also automatically log the officer out. Authorities can see in

real time every active officer through their client, so they also may be able to notice inconsistencies. Invalid reports will be discarded at the authority's client, and users will not be allowed to upload images to make sure every report photo is taken in real time and not manipulated. Similarly, the license plate will be scanned through an accurate supervised learning algorith run on the client so that it can not be input by a fraudolent user: officers will have the duty to later check the correctness of the plate number when signing the ticket. The system will also guarantee to not spread any kind of data to third parties to accomplish goal G6.

3.9.4 Maintainability

Best practices and design patterns of Object Oriented Programming along with documentation upkeep will be thoroughly followed to make sure the application is easily customizable and modifiable by SafeStreets' developers.

3.9.5 Scalability

The architecture must be simply scalable as the number of users grows over time. Except civilian account management, which is handled by SafeStreets' server and should not require too much complexity, functions are mostly provided by the authorities' servers which grow in number with the population of the considered city (e.g. the bigger the city the more numerous the barracks, and therefore the number of servers), thus the architecture will be scalable by design. In any case, the number of reports per day should never exceed a few thousands even in heavily populated cities. Various hardware and software methodologies to make the system more scalable are discussed in the `Design Document`.

4 FORMAL ANALYSIS USING ALLOY

4.1 Purpose of the analysis

In this section we provide a formalization of some constraints on how the entities of the system will interact, and the formalization of some non-functional requirements:

- how reports are accepted or rejected due to duplicates not being allowed and the authority manually discarding them;
- how map updates are controlled by the reports and how the reports they contain expire;
- how bans restrict the actions of civilians;
- how the report checking done by officers happens and how that controls ticket emission;
- various constraints on users' data.

The images are commented and self-descriptive on their own, thus we felt it wasn't needed to detail them any further.

4.2 Analysis

4.2.1 Signatures

```
//SIGNATURES

sig Email{} --Email of Civilian

sig Psw{}    --Password to do the login

sig Plate{} -- Plate of a vehicle

sig Violation{} --Violation type

sig Civilian{
   email : one Email,   --Unique identifier
   password : one Psw,
   rep: set Report,      --Reports do by a civilian
   ban: one Bool, --ban is equal to 1 if civlian is banned, equal to 0 if he isn't banned
   timeBan: one Int --Timestamp of the ban
}

sig Officer{
   badgeId : one Int,   --Unique identificative number of an officer
   password : one Psw,
   pin: one Int,    --identificative number for signature of an officer
   authority: one Authority,
   doRep: set Report,   --Reports done by officer
   controlRep: one Report --Report under investigation by officer
}

sig Authority{
   officers: set Officer, --officer that belog to an authority
   reports : set Report, --reports to be controlled by anauthority
   acceptedRep: set Report, --reports ready to officer investigation
   rejectedRep: set Report, --reports rejected for data discrepancy
}
```

```
one sig SS_Server{
   bannedCivilian: set Civilian --civilians who have been banned
}

sig Report{
   user: one Civilian,
   officer: lone Officer, --Officer who control that report
   violation: one Violation,
   time: one Int,  --Timestamp of the report
   isAccepted: one Bool,   --1 if authority accepted this report
   isConfirmed: one Bool, -- 1 if officer has confirmed the violation
   plate: one Plate
}{
   time >= 0
}

sig Ticket{
   rep: one Report,
   pin: one Int,
   fine: one Int,
   violation: one Violation,
   plate: one Plate,
}{
   fine > 0
}

one sig Map{
   reports: set Report,
   timeStamp: one Int --Current timestamp taken by the system
}{
   timeStamp >= 0
}
```

4.2.2 Facts

```
// FACTS

fact emailAreUnique{
   --All email of civilians are unique
   no disjoint c1, c2: Civilian | c1.email = c2.email
}

fact badgeIdAreUnique{
   --All Badge ID of officers are unique
   no disjoint o1, o2: Officer | o1.badgeId = o2.badgeId
}

fact authorityAcceptReport{
   --Only one authority can accept or reject a report
   no disjoint a1, a2: Authority | (all r: Report | (r in a1.reports and r in a2.reports))
}

fact eitherRejectedOrAccepted{
   --A report is either accepted or rejected
   no r: Report | (one a: Authority | (r in a.acceptedRep and r in a.rejectedRep))
}

fact officerControlReport{
   --Only one officer can control the validity of a report
   no disjoint o1, o2: Officer | o1.controlRep = o2.controlRep
}

fact reportIntegrity{
   --All reports controlled by authority are in reports did by civilians or officers
   all r: Report| (one o: Officer, c: Civilian|(r in Authority.reports iff (r in o.doRep or r in c.rep)))
}

fact reportIntegrity2{
   --Banned civilians can't compile reports
   no c: Civilian | (some r: Report | ( r.time >= c.timeBan and c.ban = True and r in c.rep))
}

fact unbannedCivilians{
   all c: Civilian | (c.ban = False implies c.timeBan = -1)
}

fact bannedCivilianList{
   all c: Civilian | (one s: SS_Server | ( c in s.bannedCivilian iff c.ban = True))
}
```

```
fact reportAccepted{
   --Consistent reports are accepted by authority
   no r: Report | one a: Authority | (r in a.acceptedRep and r.isAccepted = False)
}

fact reportRejected1{
   --Unconsistent reports are rejected by authority
   no r: Report | one a: Authority | (r in a.rejectedRep and r.isAccepted = True)
}


fact reportRejected2{
   --No rejected reports are under investigation by officer
   all  o: Officer, a: Authority, r: Report | ( r in a.rejectedRep implies r not in o.controlRep)
}

fact reportRejected3{
   --Duplicate reports, which such if they are on the same plate within the same hour, are rejected by authority
   no disjoint r1,r2: Report | one a: Authority | (r2 in a.acceptedRep iff (r2.plate = r1.plate and r2.time-r1.time < 4))
}

fact ticketIntegrity{
   --A ticket can only made by a confirmed report
   all r: Report | one t: Ticket | (r.isConfirmed = True implies t.rep = r)
}

fact ticketIntegrity2{
   --the pin in ticket is equal to officer that has confirmed that report
   --the violation on ticket is the same of the report
   --The plate in ticket is the same of the report
   all t: Ticket | (t.pin = t.rep.officer.pin and t.violation = t.rep.violation and t.plate = t.rep.plate and t.rep in t.rep.officer.controlRep)
}

fact reportControl{
   --A report can be in controlRep set only if it is accepted by authority
   all r: Report | (one o : Officer | (one a: Authority | (r in o.controlRep iff r in a.acceptedRep)))

}

fact mapConstraints{
  --A report is in the map only if accepted by authority and if its not older than 15 mins
   all r: Report | ( one a: Authority | (r in Map.reports iff ( Map.timeStamp - r.time < 1 and r in a.acceptedRep)))
}
```

4.2.3 Assertions

```
// ASSERTIONS

assert bannedCivilian{
   --a banned civilian can't do reports
   all c: Civilian | no r: Report | (c.ban = True and r.time >= c.timeBan and r in c.rep)
}
check bannedCivilian

assert noEqualCivilian{
   all c1, c2: Civilian | (not(c1 = c2) implies not (c1.email = c2.email ))
}
check noEqualCivilian

assert noEqualOfficer{
   all o1, o2: Officer | (not(o1 = o2) implies not (o1.badgeId = o2.badgeId))
}
check noEqualOfficer

assert noReportsRejectedUnderInvestigation{
   --Officers can't check a rejected report
   no r: Report | all o: Officer | (r.isAccepted = False and r in o.controlRep)
}
check noReportsRejectedUnderInvestigation

assert noRejectedReportsInMap{
   --no rejected reports can be shown in the map
   no r: Report | all a: Authority |
   (r in Map.reports and r in a.acceptedRep)
}

assert mapUpdatesExpire{
   --no map updates older than 15 minutes
      no r: Report |
   (r in Map.reports and Map.timeStamp-r.time >= 1 )
}
check mapUpdatesExpire
```

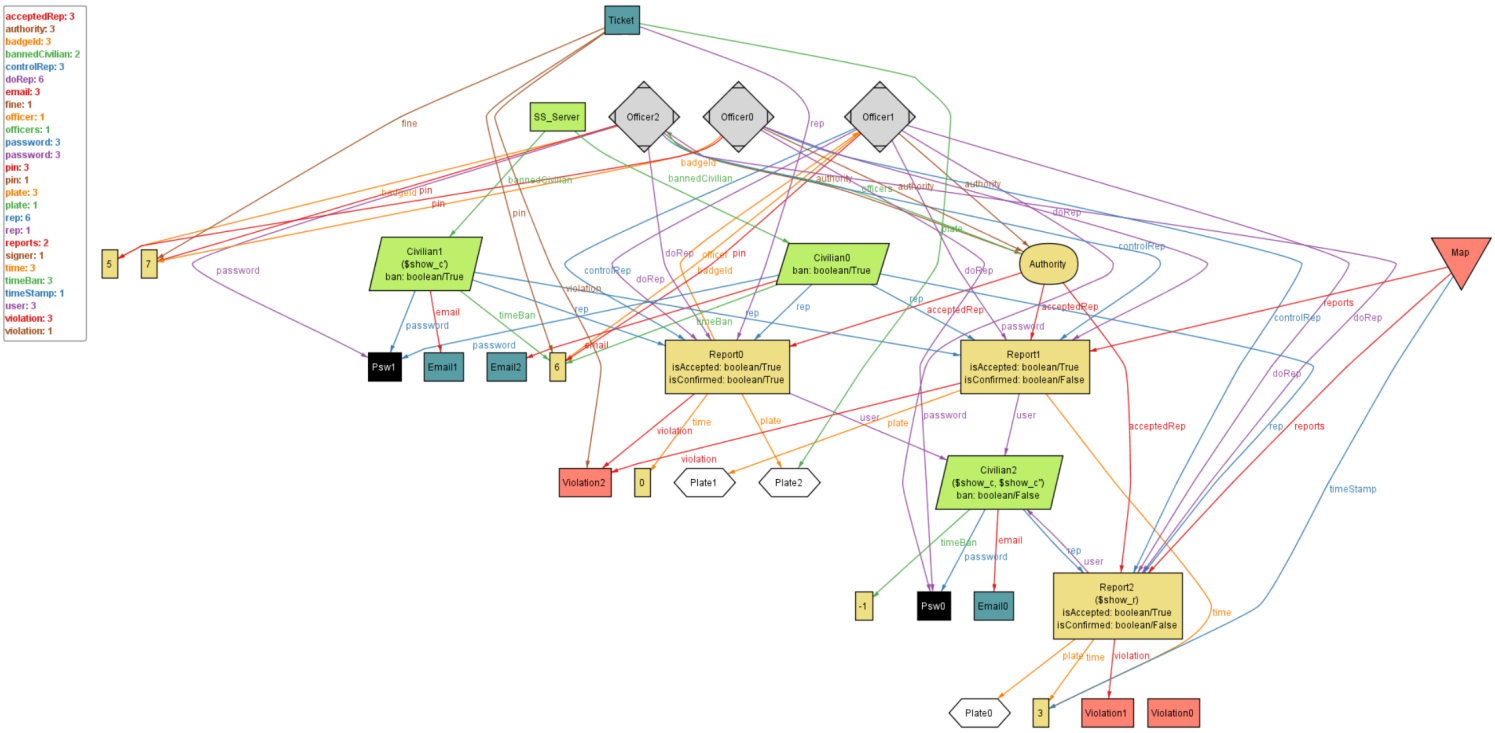
4.2.4 Predicates

```
// PREDICATES

pred show[r : Report]{
#Civilian > 1
#Authority = 1
#Ticket = 1
#Officer > 1
#Report = 3
r in Map.reports
some c: Civilian | c.ban = False
some c : Civilian | r in c.rep
one a : Authority | (r in a.acceptedRep)
one o: Officer | r in o.controlRep
some c: Civilian | ( c.ban = True and c in SS_Server.bannedCivilian)
}
run show for 3
```

4.2.5 Results

```
Executing "Check bannedCivilian"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7885 vars. 532 primary vars. 19332 clauses. 78ms.
  No counterexample found. Assertion may be valid. 47ms.

Executing "Check noEqualCivilian"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7492 vars. 532 primary vars. 18117 clauses. 63ms.
  No counterexample found. Assertion may be valid. 15ms.

Executing "Check noEqualOfficer"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7622 vars. 532 primary vars. 18507 clauses. 250ms.
  No counterexample found. Assertion may be valid. 16ms.

Executing "Check noReportsRejectedUnderInvestigation"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7457 vars. 529 primary vars. 18022 clauses. 62ms.
  No counterexample found. Assertion may be valid. 16ms.

Executing "Check mapUpdatesExpire"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7651 vars. 529 primary vars. 18656 clauses. 62ms.
  No counterexample found. Assertion may be valid. 31ms.

Executing "Run show for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  7631 vars. 538 primary vars. 18403 clauses. 63ms.
  Instance found. Predicate is consistent. 109ms.

6 commands were executed. The results are:
  #1: No counterexample found. bannedCivilian may be valid.
  #2: No counterexample found. noEqualCivilian may be valid.
  #3: No counterexample found. noEqualOfficer may be valid.
  #4: No counterexample found. noReportsRejectedUnderInvestigation may be valid.
  #5: No counterexample found. mapUpdatesExpire may be valid.
  #6: Instance found. show is consistent.
```

4.2.6 Generated Worlds



This is the world generated by the predicate `show`. We decided to limit the number of authorities and tickets to a single one, as a predicate with multiple authorities would create a too big result. We can thus say that this world represents a small city with a single authority. Things that can be noticed:

- the world shows three Civilians, two of them are banned and we can see the reports they made are anterior to their bans, and they're both marked as banned at the SafeStreets server;
- the ticket has been checked and signed by the officer who possesses the same pin as the ticket itself. Moreover, the ticket corresponds to the only report of the three which has been confirmed (by the same officer);
- the map only shows the reports with timestamps not older than one time unit of fifteen minutes;
- every officer investigated at most one report;
- all the reports accepted have been confirmed by the authority.

5 EFFORT SPENT

- Davide Cocco

Day	Subject	Hours
15/10/2019	Purpose, Scope, Goals, Product functions, User charateristics	2.5
18/10/2019	Domain assumpions, Scenarios, Software attributes, Use cases, Functional requirements and updates	6
20/10/2019	Requirements	2
21/10/2019	Sequence Diagrams	2
20/10/2019	State Diagrams	2
24/10/2019	Class Diagram	2
29/10/2019	Document revision and filling minor sections	1
9/11/2019	Alloy	5
1/12/2019	Alloy	3
3/12/2019	Final revision	5
Total		30.5

- Marco Gasperini

Day	Subject	Hours
15/10/2019	Purpose, Scope, Goals, Product functions, User charateristics	2.5
18/10/2019	Domain assumpions, Scenarios, Software attributes, Use cases, Functional requirements and updates	6
20/10/2019	Use case diagrams	2
22/10/2019	Mockups	4
9/11/2019	Alloy	5
1/12/2019	Alloy	3
3/12/2019	Alloy	7
Total		29.5