POLITECNICO
MILANO 1863

RASD

Requirements Analysis and Specification Document

Authors:

Version: 2.0

Date: 23 November 2019

Professor:

Contents

1. Introduction

1.1 Purpose

This document focuses on *Requirements Analysis and Specification Document (RASD)* and contains the description of the main goals, the domain and its representation through some models, the analysis of the scenario with the uses cases that describe them, the list of the most important requirements and specifications that characterize the development of the software described below.

It also includes the research about the interfaces, functional and non-functional requirements and the attributes that distinguish the quality of the system.

Finally, to understand better the development of the document, it contains the history that describes how it is made, with the references used and the description of its structure.

This document has the purpose to guide the developer in the realization of the software called SafeStreets, a crowd-sourced application.

1.2 Scope

The software wants to give users the possibility of notifying authorities about traffic violations.
These alerts can be sent through pictures of the infringement.
SafeStreets offers three main functionalities:

- **Basic service**: allows users to send a report including photos, its date and time, the position or the name of the street, and the type of the violation. Then, the software takes care of running an algorithm to read the license plate, it stores the information obtained and send these to the municipality.
  In addition, users and authorities can extract the information collected, naturally in different levels of visibility (users can only see general information about violations, while municipalities can see the violations of a unique license plate).

- **Advanced function 1**: the municipality can access this service to offer its information about accidents. Therefore, the software crosses this information with its own data and identifies potentially unsafe areas. The purpose of this function is to suggest possible interventions in the areas with more risk of accidents and violations.
  In addition, users can view a map of the city in which are highlighted in different colours the areas with different level of safety.

- **Advanced function 2**: the local police can adhere to this service to take information about violations coming from the application and generates automatically traffic tickets from it. Furthermore, the information about issued tickets can be used by the software to build statistics about the more common types of violations, or SafeStreets effectiveness.

The authorities which want to benefit from the services must be subscribed to it with an authentication. Meanwhile, the users that report the violation are anonymous.

In the case of the last function, the software must ensure that the pictures received are not fake. To be certain of this it runs an algorithm which extracts metadata directly from the picture and crosses this data with the additional information received by the user. In case they are fake, data are discarded.

Moreover, for good measure the application allows users to take photos only in real time by the camera and does not accept photos from the gallery in order to assure that the pictures are not photoshopped.

If at the moment in which a user fills in a report the internet connection is absent, or some field of the report are left incomplete any field, the report and its pictures are saved in a queue.
A complete report will be sent to SafeStreets as soon as the connection come back.
Finally, users can keep track of the reports sent with a registry, in which they can view if their reports have been validated from the municipality.

1.2.1 World Phenomena

WP1	User takes a smartphone with himself
WP2	A traffic violation
WP3	User observes a traffic violation
WP4	User takes a picture
WP5	Accidents that occur on the territory of the municipality
WP6	Get a traffic ticket

1.2.2 Shared Phenomena

SP1	Receive an alert about a traffic violation
SP2	Send a traffic violation to the municipality
SP3	Receive a query to mine some information
SP4	Receive information about accidents of an area
SP5	Send a possible solution to the authorities to secure an area
SP6	Generate traffic tickets
SP7	Receive information about issued tickets
SP8	Receive a request to join an advanced service from the municipality

1.2.3 Goals

G1	Allow users to notify authorities about traffic violation
G2	Allow users to send pictures
G3	Allow users to insert the type of violation
G4	Allow users to insert geographical position in two different ways
G5	Allow users to write additional information about the violation
G6	Users and authorities can mine information of violations in an overview
G7	Authorities can mine information about individual violations
G8	Identify potentially unsafe areas
G9	Suggest possible interventions to municipality to secure an area
G10	Send key information to the municipality to generate traffic tickets
G11	Information collected in the database can be used to build statistics
G12	Allow users to see the reports sent or incomplete

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

Validation	Municipality sends a validation when confirms that the report contains a punishable violation
Secure	SafeStreets defines secure a report when the photos are clearly not fake and the metadata of the pictures match with the information written on the report
Metadata	Data directly extracted from the picture through an algorithm
Query	Synonym for request
Individual Request	Request referred to a single license plate
General Request	Request referred to a specific type of a violation or vehicle (for example the frequency)
Area Request	Request referred to a specific area of the city (for example the safety)

1.3.2 Acronyms

RASD	Requirement Analysis and Specification Document
GPS	Global Positioning System

1.3.3 Abbreviations

WPn	World Phenomenon number n
SPn	Shared Phenomenon number n
Gn	Goal number n
BS	Basic service of SafeStreets
AF1	Advanced function 1 of SafeStreets
AF2	Advanced function 2 of SafeStreets
Rn	Requirement number n

1.4 Revision history

Date	Modifications
10/11	First Version
23/11	Second Version: ○ Update UML ○ Update some signatures in the Alloy Code ○ Adding section 3.6 Additional Specifications ○ Update some details in the sequence diagrams

1.5 Reference Documents

- Specification Document: “SafeStreets Mandatory Project Assignment.pdf”
- Slides of the lectures

1.6 Document Structure

- **Chapter 1** gives an introduction about the purpose of the document and the development of the application, with its corresponding specifications such as the definitions, acronyms, abbreviation, revision history of the document and the references.
  Besides, are specified the main goals, world and shared phenomena of the software.

- **Chapter 2** contains the overall description of the project. In the product perspective are included the statecharts of the major function of the application and the model description through a Class diagram. In user characteristic are explained the types of actors that can use the application. Moreover, the product function clarified the functionalities of the application. Finally, are included the domain assumption that can be deducted from the assignment.

- **Chapter 3** presents the interface requirement including: user, hardware, software and communication interfaces. This section contains the core of the document, the specification of functional and non-functional requirements. Functional requirements are submitted with a list of use cases with their corresponding sequence diagrams and some scenarios useful to identify specific cases in which the application can be utilised. Non-functional requirements included: performance, design and the software systems attributes.

- **Chapter 4** includes the alloy code and the corresponding metamodels generated from it, with a brief introduction about the main purpose of the alloy model.

- **Chapter 5** shows the effort spent for each member of the group.

- **Chapter 6** includes the reference documents.

2. Overall Description

2.1 Product perspective

2.1.1 UML Description

The UML below describes at high-level the model of the system to be developed. It considers the basic service together with the advanced function 1 and advanced function 2 previously specified. The UML does not include every class that will be necessary to define the complete architecture of the system. SafeStreets offers different functions beyond the basic service. The municipality registers to the application providing all necessary information and can decide at a later stage to activate the distinct tasks (AF1 or AF2 or both of them). The user simply downloads the application on his/her own device to be able to use it. Here we can identify the main aspects related to SafeStreets:

- The user can create a report to illustrate a violation that wants to notify. The violation contains specific attributes and a set of pictures that clearly describes the situation. These pictures are taken using the application: whenever a picture is shot, the application labels it with the exact time, date and GPS Position of the location reported by the user's device when the photo is created. Later in the report, the user specifies the exact address of the violation.

- The user can see on the map the different areas which are characterized with different colours representing the level of safety and retrieve information about the most frequent violations in different zones of the city.

- The municipality can give information about the accidents and receives suggestions about the interventions that can be adopted to increase the well-being of the citizens.

- The local police takes information about the violations and generates traffic tickets for the offenders. SafeStreets can use these new data, together with the information already stored in its own database, to build useful statistics that measure for example the effectiveness of the application.

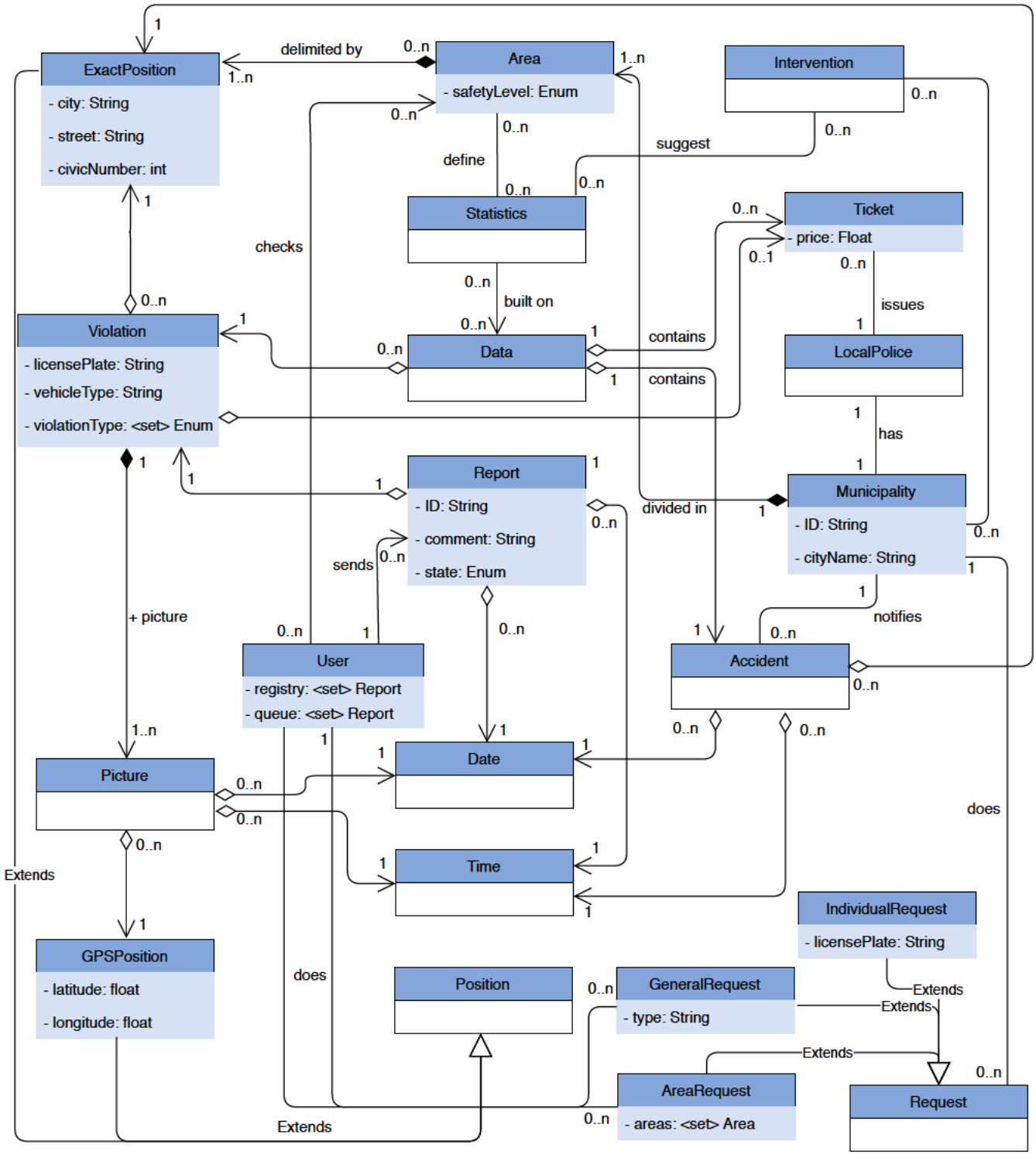


Figure 0 - Class Diagram using UML

2.1.2 State Charts

Now we are going to analyse some critical aspects of the application, modelling their behaviours and showing the evolution over time of their states through appropriate state diagrams, which are reported below.

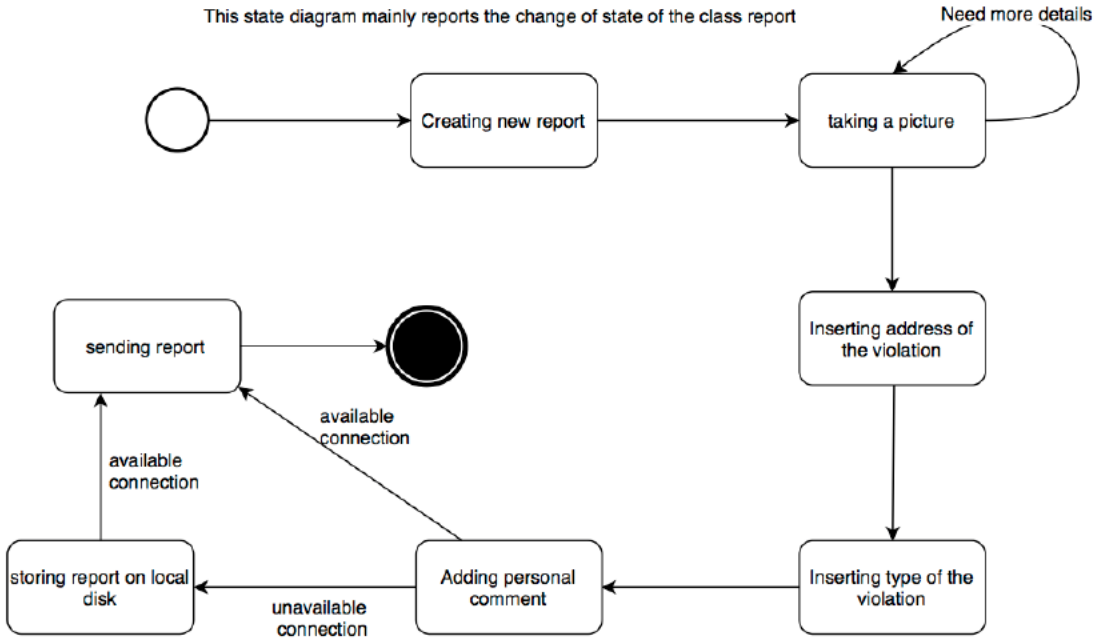


Figure 1 - State diagram 1: Creating a report and send it

In the first state diagram (figure 1), we model the creation of a new report by a user showing the main actions that he has to perform until send the report, if internet connection is available.

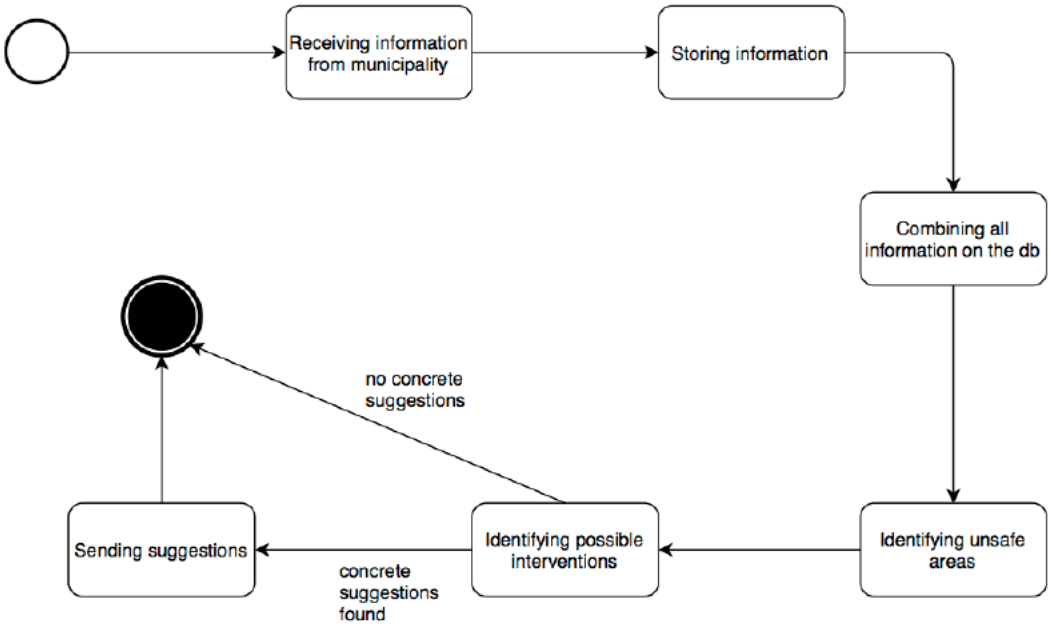


Figure 2 - State diagram 2: Identify unsafe areas

In the second state diagram (figure 2), it is modelled how SafeStreets can identify the areas that are less safe using the data received from the municipality. Then, SafeStreets can also suggest some possible interventions to reduce the number of accidents in these areas. In this case it is assumed that the municipality is subscribed to the advanced function 1.

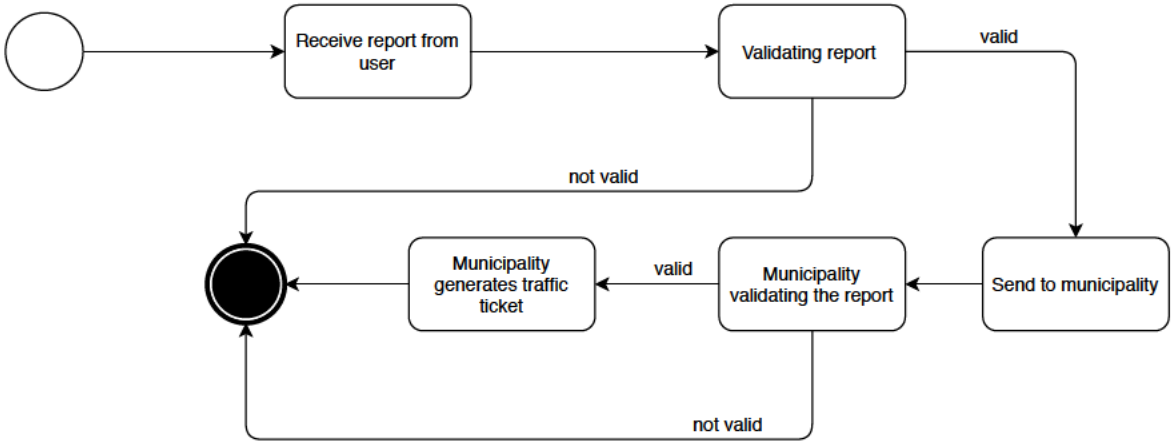


Figure 3 - State diagram 3: Municipality generates traffic tickets

Finally, in the last one (figure 3), it is explained how municipality arrives to generates traffic tickets.
It is important to note that there are two phases of validating: the first one, made by SafeStreets, in which they run an algorithm to verify if the pictures are not fake and the second one, made by the municipality, in which they check if the violation is a real infraction or not. Hence, if pass these two checks, the municipality can finally generate the traffic ticket. It is important to underline that in this case the municipality is subscribed to the advanced function 2.

2.2 Product functions

Here are described some functions of the software. Be aware that some products functions are described also in other parts of this documents, particularly in the goal parts and in the functionals requirements parts.

Report a violation

The main functionality of the system is the possibility of the user to report authorities when traffic violations occur, and in particular parking violations. In order to do that the user has to take pictures about the infraction and provide information about the type of violation, type of vehicle, additional comment describing the situation in which the infraction took place and which problem caused.
Furthermore, the user can take pictures of violation storing them in the application and then, in a second moment, can fill the report and send it to SafeStreets. Hence, if the user is online, the report is sent, otherwise the report is stored in a queue and then, when the internet connection will be available, it will be sent.
When the report is received by SafeStreets, an algorithm for the recognition of the licence plate is run, and the alphanumeric code contained in the pictures is recognized and attached to the report.
The user can keep trace of the report done in a registry and check its state (for example if the municipality considered the violation as valid).

Collection and release of information

SafeStreets stores the information provided by users, completing it with suitable metadata.
Moreover, allows both the users and the authorities to mine the information and shows the result of this operation with different levels of visibility. For instance, highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the greatest number of violations for the user, while the violations committed by a single vehicle for the authorities.

Identify unsafe areas

SafeStreets can collect data sent by municipalities about the accidents occurred in the area of competence. Periodically, it crosses the information with its own data identifying areas that are less safe and can supply possible interventions to authorities in order to prevent violations and reduce the number of accidents, trying to increase the security of different zones.
Moreover, the user can see the map of the city where the areas have different colours due to their safety level.

Provide reliable evidence for municipalities

The service helps the municipality to generate traffic tickets thanks to the reports provided. To validate the violations reported by the users, SafeStreets offers a secure method that does not let the user to send pictures that are not directly taken from the application itself, to avoid the upload of modified pictures. In addition, it crosses the information about position inserted by the user and the one retrieved from the metadata of the pictures. With the data received from the municipalities, regarding the tickets emitted, SafeStreets, periodically, builds statistics based on the data stored in the database, identifying for example the most egregious offenders or measuring the application’s impact.

2.3 User characteristics

The actors of the application are the following:

1. **User**: someone who downloads the application on his device and wants to help reporting traffic violations that occur. Furthermore, the user can obtain useful data regarding violations and accidents happened.

2. **Municipality**: it is necessary to distinguish how the municipality is meant depending on the service:

    - *Basic service* and *Advanced function 1*: the municipality is considered as the authority responsible of a well-defined territory that includes many areas. The municipality is interested about the traffic violations occurred in its territory, receives reports sent by users and, in addition, can get useful information from SafeStreets. If subscribed to the AF1 offers information concerning the accidents that took place in its territory.

    - *Advanced function 2*: the concept of municipality is extended also to the local police that can generate traffic tickets directly from the data received from SafeStreets about the violations.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

D1	One license plate for each violation
D2	The internet connection works properly
D3	Each user has a smartphone
D4	Each smartphone has a camera
D5	Each smartphone has a GPS sensor integrated
D6	GPS is active while the app is running
D7	GPS provides the exact location with an error of 5 meters at most
D8	The pictures taken are not blurry
D9	The license plate is readable
D10	The pictures sent by the user show the situation of the violation clearly and from many points of view
D11	For each picture contained in a single report appears always the same license plate
D12	In one picture at least it has to be visible both the violation and the license plate
D13	Every license plate is unique for each car owner
D14	The data sent by the municipality are correct
D15	The municipality is always available and able to receive data
D16	Every report contains at least one type of violation
D17	Every vehicle reported has a license plate

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The device in use must be able to take pictures, so it must have a camera, and save the position of the user. It also must be able to capture the date and the time in which the pictures were taken. Finally, after the device has captured this information, it must be able to send all of them to SafeStreets system. A smartphone is the suitable device that owns all these features.
The following mockups give the basic idea of what:

- the icon of the application looks like;
- the mobile app will look like in the home page, after opening the application.

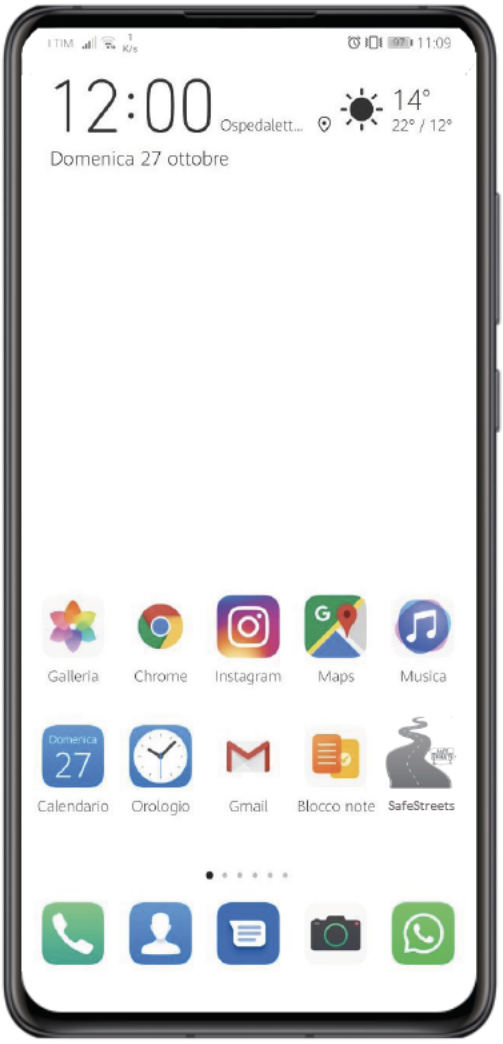


Figure 1 – Icon Mockup

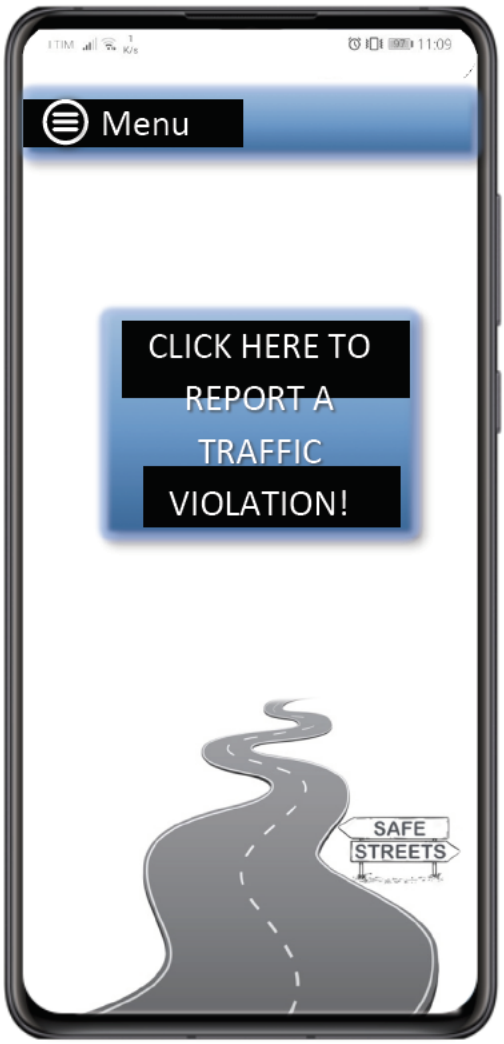


Figure 2 – Home Page Mockup

Other Mockups will be presented in the Design Document part for User Interfaces.

3.1.2 Hardware Interfaces

The software application contains three main functions and two types of actors, which require different kinds of hardware interfaces.

- Regarding **BS**:
  The users must own a smartphone with a camera in order to take pictures and send reports. They can also use a computer accessing the WebApp but in this way only the function of retrieving information can be performed.
  The authorities, instead, can access the system only through the WebApp mainly using a computer. They first have to register themselves to the system and then can receive reports or get information.

- Regarding **AF1 & AF2**:
  The municipality needs to have at least a computer in order to:
  - Join the service
  - Share information with SafeStreets
  - Receive advices
  - Generate traffic tickets

3.1.3 Software Interfaces

The system uses the following external interfaces:

- **City map**
  We assume that the system uses a public API to provide to the user:
  - His position in real time for the report compilation.
  - A map of the city in which are identified the possible unsafe areas.

- **Communication with the municipality**
  Regarding information exchanges between SafeStreets and the municipality, the system must have an interface that connects to them. It has the following functions:
  - Send the report of the users.
  - Receive validation of a report.
  - Receive query about violations.
  - Receive data about accidents.
  - Send advices about unsafe areas.

- **Extract data**
  We assume that the software uses an internal system able to:
  - Read from a photograph the license plate of a vehicle.
  - Extract metadata directly from a picture, like the date and time, and when possible, the place where it was taken.

3.1.4 Communication Interfaces

The device connects to SafeStreets system via internet connection.

3.2 Functional Requirements

3.2.1 List of Requirements

R1	Users are anonymous
R2	Authorities are certified with an authentication*
R3	Authorities should register to the application filling the form with mandatory fields*
R4	Only authorities can receive reports about violations from the system
R5	Each report will be forwarded to the related municipality only if it is subscribed, otherwise it will be sent to the closest municipality enrolled to the service
R6	Authorities can choose to adhere or not to the advanced function 1
R6.1	Authorities can choose to adhere or not to the advanced function 2
R7	After the authority subscription, reports of interest of its related territory are forwarded to it
R8	Users shall authorise the system to access the local camera and the local position
R9	Municipality is requested to send to the system validations of received reports
R10	Municipality that uses the system needs to be logged
R11	Municipality allows the system to acquire its data about accidents and issued tickets
R12	After a report is validated positively from municipality its data are stored in the database system
R13	Users cannot access personal data of offenders through any queries
R14	Authorities have no restrictions to access personal data
R15	Authorities and users have to choose from a list* of predetermined requests to extract information by queries
R16	The system provides secure reports to municipalities that join AF2
R17	The system sends to the authorities reports as soon as they arrive
R18	Users can choose the type of violation from a predefined checklist*
R19	Users should compile a report in all its mandatory fields*
R20	Users can add an optional comment to the report
R21	Users can access a queue that contains incomplete reports and the ones completed in absence of an internet connection

R22	Completed reports in the queue are sent to the system as soon as the connection is available
R23	Users can view in a registry the reports already sent and the result of the validation from the municipality
R24	Users can select from a list* of city maps to view the level of safety of specific areas characterized with different colours
R25	The system allows users to take pictures only directly from the application
R26	Pictures automatically collect the date, the time and the GPS position when they are taken
R27	The system suggests possible solutions to municipalities that join the AF1 to make their territory safer
R28	The system ensures that the pictures sent to the municipality joining the AF2 are not modified
R29	Data of municipalities enrolled in AFs are collected by the system
R30	Statistics are built considering data stored in the database

***have been specified later in the document**

3.2.2 Mapping

Goals	Domain Assumptions	Requirements
G1	D1, D2, D3, D10, D12, D13, D15, D16, D17	R1, R2, R3, R4, R5, R6, R6.1, R7, R9, R10, R12, R17, R19, R20, R22
G2	D3, D4, D8, D9, D10, D11, D12	R8, R25, R26
G3	D16	R18
G4	D3, D5, D6, D7	R8
G5	D3	R20
G6	D2	R10, R13, R15, R24
G7	D1, D2	R2, R3, R10, R14, R15
G8	D2, D14	R6, R11, R15, R24, R29
G9	D2, D14, D15	R2, R3, R6, R27
G10	D1, D2, D7, D8, D9, D10, D11, D12, D13, D15, D16, D17	R2, R3, R4, R5, R6.1, R7, R10, R11, R16, R17, R28, R29
G11	D14	R11, R29, R30
G12	D3	R21, R22, R23

G1	Allow users to notify authorities about traffic violation
D1	One license plate for each violation
D2	The internet connection works properly
D3	Each user has a smartphone
D10	The pictures sent by the user show the situation of the violation clearly and from many points of view
D12	In one picture at least it has to be visible both the violation and the license plate
D13	Every license plate is unique for each car owner
D15	The municipality is available and able to receive data
D16	Every report contains at least one type of violation
D17	Every vehicle reported has a license plate
R1	Users are anonymous
R2	Authorities are certified with an authentication*
R3	Authorities should register to the application filling the form with mandatory fields*
R4	Only authorities can receive reports about violations from the system
R5	Each report will be forwarded to the related municipality only if it is subscribed, otherwise it will be sent to the closest municipality enrolled to the service
R6	Authorities can choose to adhere or not to the advanced function 1
R6.1	Authorities can choose to adhere or not to the advanced function 2
R7	After the authority subscription, reports of interest of its related territory are forwarded to it
R9	Municipality is requested to send to the system validations of received reports
R10	Municipality that uses the system needs to be logged
R12	After a report is validated positively from municipality its data are stored in the database system
R17	The system sends to the authorities reports as soon as they arrive
R19	Users should compile a report in all its mandatory fields*
R20	Users can add an optional comment to the report
R22	Completed reports in the queue are sent to the system as soon as the connection is available

G2	Allow users to send pictures
D3	Each user has a smartphone
D4	Each smartphone has a camera
D8	The pictures taken are not blurry
D9	The license plate is readable
D10	The pictures sent by the user show the situation of the violation clearly and from many points of view
D11	For each picture contained in a single report appear always the same license plate
D12	In one picture at least it has to be visible both the violation and the license plate
R8	Users shall authorise the system to access the local camera and the local position
R25	The system allows users to take pictures only directly from the application
R26	Pictures automatically collect the date, the time and the GPS position when they are taken

G3	Allow users to insert type of violation
D16	Every report contains at least one type of violation
R18	Users can choose the type of violation from a predefined checklist*

G4	Allow users to insert geographical position in two different way
D3	Each user has a smartphone
D5	Each smartphone has a GPS sensor integrated
D6	GPS is active while the app is running
D7	GPS provides the exact location with an error of 5 meters at most
R8	Users shall authorise the system to access the local camera and the local position

G5	Allow users to write additional information about violation
D3	Each user has a smartphone
R20	Users can add an optional comment to the report

G6	Users and authorities can mine information of violations in an overview
D2	The internet connection works properly
R10	Municipality that uses the system needs to be logged
R13	Users cannot access personal data of offenders through any queries
R15	Authorities and users have to choose from a list* of predetermined requests to extract information by queries
R24	Users can select from a list* of city maps to view the level of safety of specific areas characterized with different colours

G7	Authorities can mine information about individual violations
D1	One license plate for each violation
D2	The internet connection works properly
R2	Authorities are certified with an authentication*
R3	Authorities should register to the application filling the form with mandatory fields*
R10	Municipality that uses the system needs to be logged
R14	Authorities have no restrictions to access personal data
R15	Authorities and users have to choose from a list* of predetermined requests to extract information by queries

G8	Identify potentially unsafe areas
D2	The internet connection works properly
D14	The data sent by the municipality are correct
R6	Authorities can choose to adhere or not to the advanced function 1
R11	Municipality allows the system to acquire its data about accidents and issued tickets
R15	Authorities and users have to choose from a list* of predetermined requests to extract information by queries
R24	Users can select from a list* of city maps to view the level of safety of specific areas characterized with different colours
R29	Data of municipalities enrolled in AFs are collected by the system

G9	Suggest municipality for possible interventions to secure an area
D2	The internet connection works properly
D14	The data sent by the municipality are correct
D15	The municipality is available and able to receive data
R2	Authorities are certified with an authentication*
R3	Authorities should register to the application filling the form with mandatory fields*
R6	Authorities can choose to adhere or not to the advanced function 1
R27	The system suggests possible solutions to municipalities that join the AF1 to make their territory safer

G10	Send key information to the municipality to generates traffic tickets automatically
D1	One license plate for each violation
D2	The internet connection works properly
D7	GPS provides the exact location with an error of 5 meters at most
D8	The pictures taken are not blurry
D9	The license plate is readable
D10	The pictures sent by the user show the situation of the violation clearly and from many points of view
D11	For each picture contained in a single report appear always the same license plate
D12	In one picture at least it has to be visible both the violation and the license plate
D13	Every license plate is unique for each car owner
D15	The municipality is available and able to receive data
D16	Every report contains at least one type of violation
D17	Every vehicle reported has a license plate
R2	Authorities are certified with an authentication*
R3	Authorities should register to the application filling the form with mandatory fields*
R4	Only authorities can receive reports about violations from the system
R5	Each report will be forwarded to the related municipality only if it is subscribed, otherwise it will be sent to the closest municipality enrolled to the service
R6.1	Authorities can choose to adhere or not to the advanced function 2
R7	After the authority subscription, reports of interest of its related territory are forwarded to it
R10	Municipality that uses the system needs to be logged
R11	Municipality allows the system to acquire its data about accidents and issued tickets
R16	The system provides secure reports to municipalities that join AF2
R17	The system sends to the authorities reports as soon as they arrive
R28	The system ensures that the pictures sent to the municipality joining the AF2 are not modified
R29	Data of municipalities enrolled in AFs are collected by the system

G11	Information collected in the database can be used to build statistics
D14	The data sent by the municipality are correct
R11	Municipality allows the system to acquire its data about accidents and issued tickets
R29	Data of municipalities enrolled in AFs are collected by the system
R30	Statistics are built considering data stored in the database

G12	Allow users to see the reports sent or incomplete
D3	Each user has a smartphone
R21	Users can access a queue that contains incomplete reports and the ones completed in absence of an internet connection
R22	Completed reports in the queue are sent to the system as soon as the connection is available
R23	Users can view in a registry the reports already sent and the result of the validation from the municipality

3.2.3 Use Cases

3.2.3.1 Use Cases Description

1. Registration of Authority

Name	Registration of Authority
Actors	Authority
Entry Condition	Authority has the internet connection available and has accessed the Web Browser on its device
Event Flow	1) Authority visualizes the initial page of the WebApp 2) Authority clicks on "Sign up" button 3) Authority compiles all the mandatory fields 4) Authority loads a certification document which proves that it is a real municipality 5) The system validates the certification 6) The system confirms the registration of the Authority 7) The system saves the information
Exit Conditions	Authority is successfully registered to the application and it can take advantage of BS
Exception	1) Authority is already present in the system 2) Authority is not a real municipality 3) The Authority did not fill up all mandatory fields with valid data If one of the three above situations happens, the application will throw an error message and will return to the registration form page.

2. Login of Authority

Name	Login of Authority
Actors	Authority
Entry Condition	Authority is already registered to the application service

Event Flow	1) Authority accesses the WebApp through its device 2) Authority compiles the fields “Username” and “Password” 3) Authority clicks on “Login” button 4) The system opens the application home page 5) If there are some reports to view, authority receives a notification
Exit Conditions	The authority has access to the services of SafeStreets
Exception	1) Authority enters invalid Username 2) Authority enters invalid Password If one of the above conditions is detected, the application notifies the Authority taking it back to the login screen.

3. Selection between advanced function

Name	Selection between advanced function
Actors	Authority
Entry Condition	Authority has already logged in the application and it is not subscribed to both the advanced functions
Event Flow	1) Authority clicks on “Advanced Functions” button 2) Authority chooses which of the advanced functions wants to subscribe to* 3) Authority accepts privacy terms and conditions which allow SafeStreet to receive data from the authority itself 4) The system crosses its own information with the authority data
Exit Conditions	Authority can benefit from one of the advanced functions of SafeStreets
Exception	1) Authority refuses privacy terms and conditions If the above exception is detected, the application will take back the Authority to the home page. *if authority wants to join both of the advanced function, it repeats the procedure one more time

4. Report a violation

Name	Report a violation
Actors	User, Municipality
Entry Condition	User has already installed the application on its device; the user sees a violation
Event Flow	1) User opens the application on its device 2) User clicks on "Report a Violation" button 3) User takes pictures of the license plate 4) User takes pictures of the violation from different points of view 5) User compiles the mandatory fields 6) If User wants, can write an optional comment 7) User sends the report to SafeStreets 8) The system receives the report and only in case of *AF2*: *8.1) The system runs an algorithm which extracts metadata from the pictures and crosses these with the information in the report* *8.2) The system confirms that the report is not fake* 9) The system sends the report to the municipality closest to the place of violation registered in the database 10) The municipality sends to SafeStreets the validation of the report and only in case of *AF2*: *10.1) also the information about issued ticket* 11) The system stores all the information in the database In points 3&4, the application keeps track automatically of date, time and GPS position.
Exit Conditions	A valid report is sent to the municipality
Exception	1) User fills incorrectly one or more fields If the above exception is detected, the application will take back the User to the home page. 2) User leaves one or more mandatory fields incomplete 3) The internet connection is not available If one of the above situations happens, the report is saved in a queue and, when it will be completed, will be sent to SafeStreets. 4) The municipality answers to SafeStreets with the rejection of the report (it means that the pictures do not represent any type of violation) If the above exception is detected, the User can view in his violations registry the result "Rejected".

	In case of AF2: *5) The system discover that one or more pictures are fake* *If the above exception is detected, the system discards the report.*

4.1 Report a violation for the first time

Name	Report a violation for the first time
Actors	User
Entry Condition	User has already installed the application on its device; the user sees a violation
Event Flow	1) User opens the application on its device 2) User clicks on "Report a Violation" button 3) User clicks on "Allow SafeStreets to access the camera" button 4) User clicks on "Allow SafeStreets to access the GPS position" button *everything that happens below is reported in the use case **4.** from point 3)

5. User extracts general information

Name	Extract general information
Actors	User
Entry Condition	User has the application running on his device
Event Flow	1) User clicks on the "Search" button and enters the name of the area he is interested of 2) User clicks on the button related to the specific info needed 3) SafeStreets sends updated information
Exit Conditions	The user sees on his device the results of the request
Exception	none

6. Authority extracts information

Name	Extract information
Actors	Municipality
Entry Condition	Municipality is at least subscribed to the basic service Municipality is logged in
Event Flow	1) Municipality clicks on the "Search" button and enters the name of the area of interest among those available 2) Municipality clicks on the button related to the specific info needed 3) If the area specified pertains to the municipality, it specifies the level of visibility of the info (individual or general) 4) The system gets the request and returns a reply with the information needed
Exit Conditions	Municipality displays the information on his device
Exception	none

7. Municipality does area request and takes possible intervention

Name	Identify unsafe areas and suggest possible intervention
Actors	Municipality
Entry Condition	Municipality is registered and subscribed to the AF1 Municipality is logged in
Event Flow	1) Municipality clicks on the button "Upload accidents info" 2) SafeStreets receives the information about accidents 3) SafeStreets analyses data related to the municipality identifying unsafe areas 4) Municipality is notified when data are available 5) Municipality asks for results 6) SafeStreets sends information requested 6.1) If SafeStreets also find a feasible suggestion, it is sent to the municipality
Exit Conditions	The municipality can see the map related to its area where every zone is showed with distinct colours representing the different safety levels: red for the most dangerous ones, yellow for the less dangerous and green for the safest ones*
Exception	none

It is necessary to specify that in the previous use case the action of sending information about the accidents is not always strictly related with a successive area request. The two actions can also evolve independently to each other.
*more information about this will be found in the Additional Specifications section

8. Provide data to generate traffic tickets

Name	Provide data to generate traffic tickets
Actors	User, Authority
Entry Condition	Authority joins AF2
Event Flow	1) User sends a report through the SafeStreets' application 2) SafeStreets checks that the report arrived is truthful by comparing the position provided by the user with the one extracted from the metadata of the images, thanks to the GPS 3) SafeStreets sends the report to the Authority 4) The Authority verifies the violation of the report 5) The Authority generates the traffic ticket with the information received 6) The issued ticket is submitted in the database of the system
Exit Conditions	Authority generates the traffic ticket
Exception	If the matching done from SafeStreets fails or if the violation is not a real one the report is rejected

9. Build statistics

Name	Build statistics
Actors	Municipality
Entry Condition	Municipality is subscribed to at least AF1 or AF2
Event Flow	1) Municipality sends data about accidents (case of AF1) 2) Municipality sends data about issued tickets (case of AF2) 3) The system elaborates the data received from the municipality, crossing it with its own data 4) The system builds statistics
Exit Conditions	The system builds statistics
Exception	If the system has not enough data available to generate statistics it will be generated an error

10. See registry of reports

Name	See registry of reports
Actors	User
Entry Condition	User has already installed the app
Event Flow	1) User opens the app on his device 2) User clicks on drop down menu button 3) User selects the button “Registry”
Exit Conditions	The User can see all sent reports with their relative state
Exception	none

11. See queue

Name	See queue
Actors	User
Entry Condition	User has already installed the app
Event Flow	1) User opens the app on his device 2) User clicks on drop down menu button 3) User selects the button “Queue”
Exit Conditions	The User can see the unsent reports, but also the reports that have to be completed
Exception	none

3.2.3.2 Use Cases Diagram

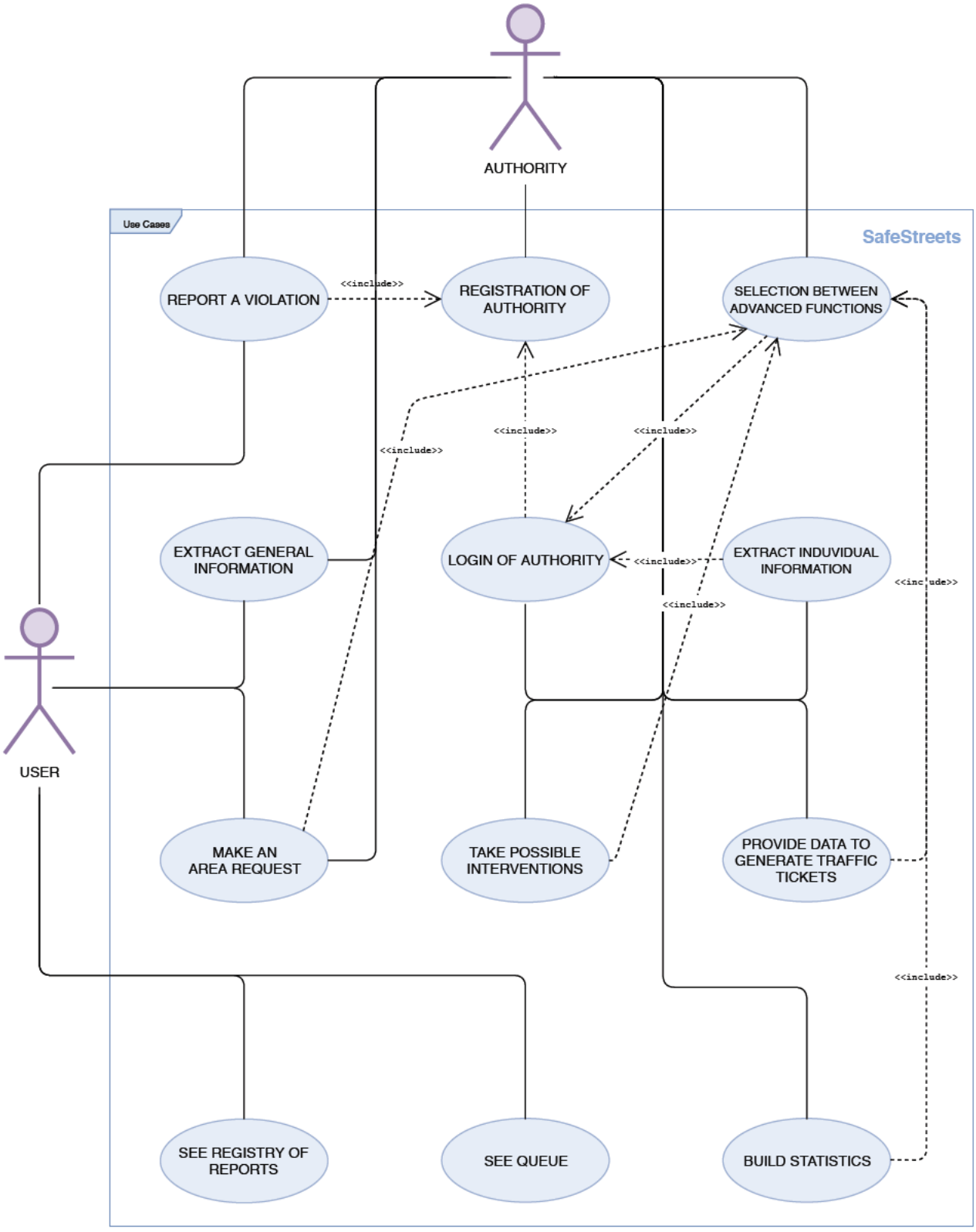
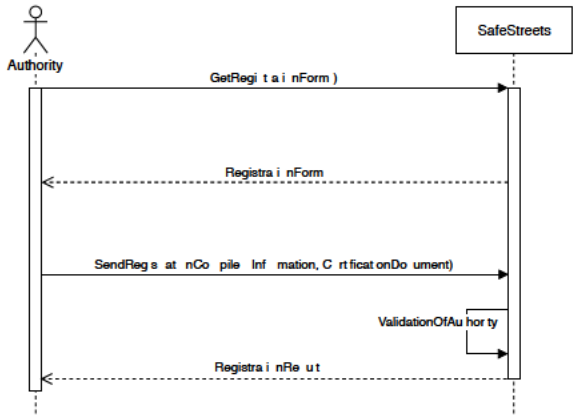


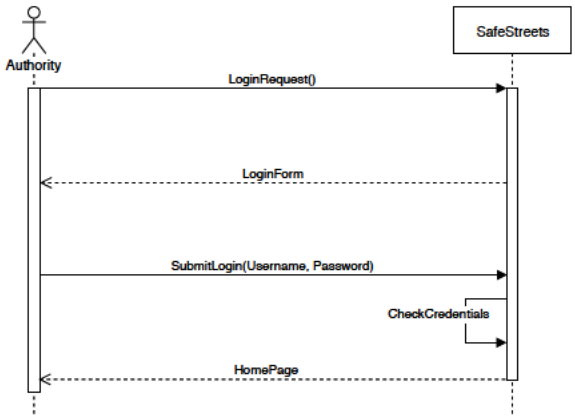
Figure 1 – Use Cases Diagram

3.2.4 Sequence Diagrams

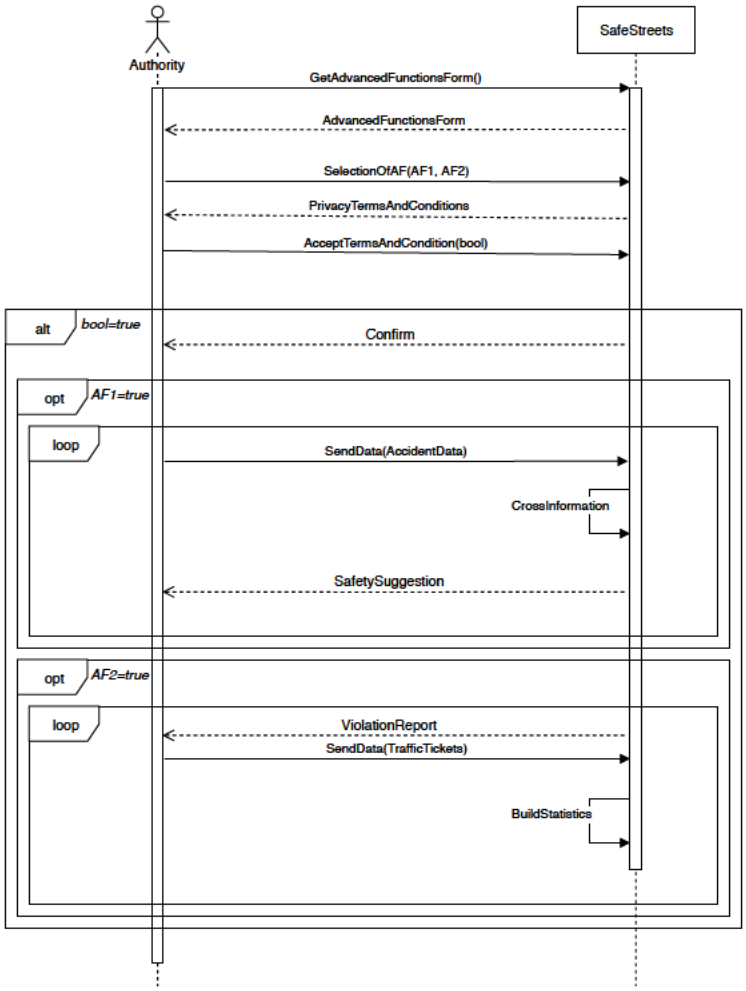
1. Registration of Authority



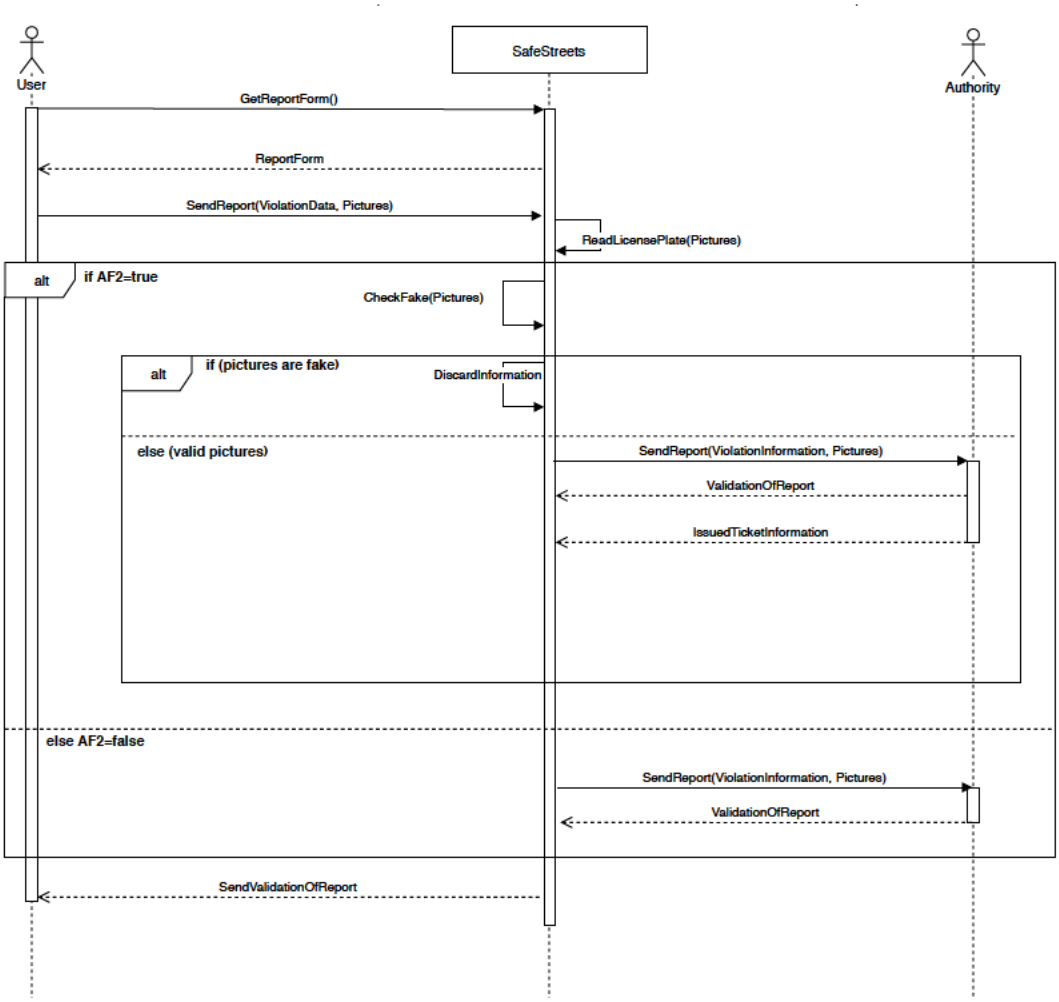
2. Login of Authority



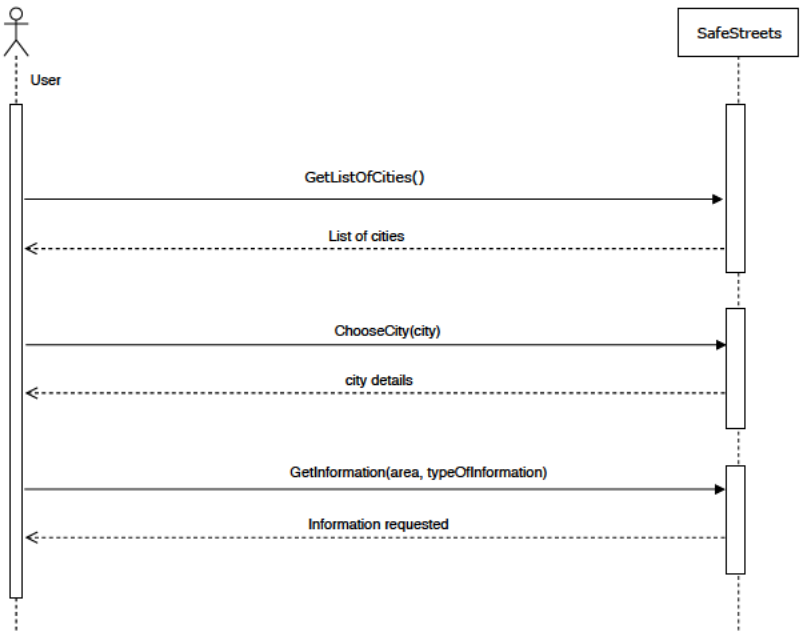
3. Selection between advanced function



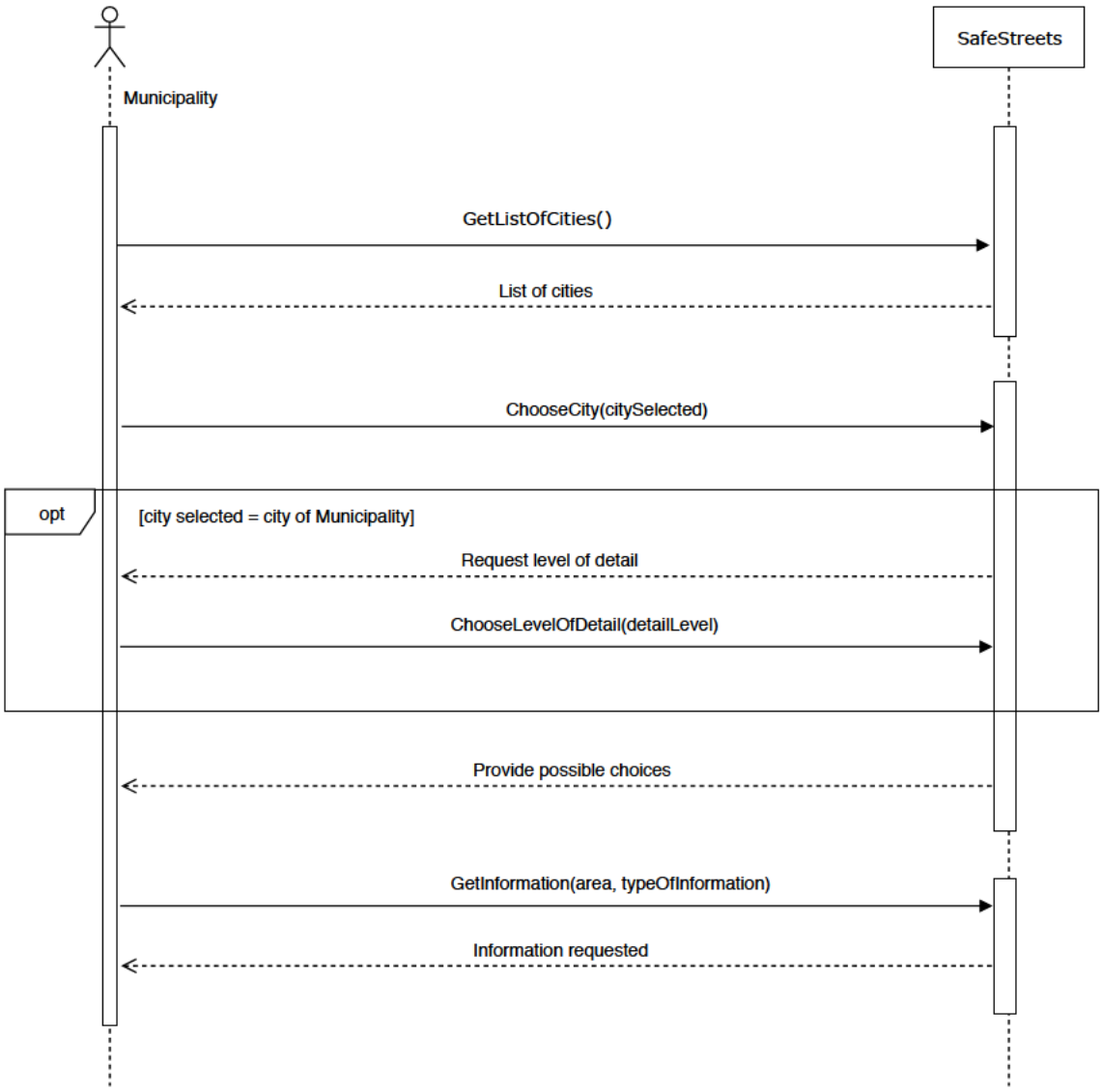
4. Report a violation



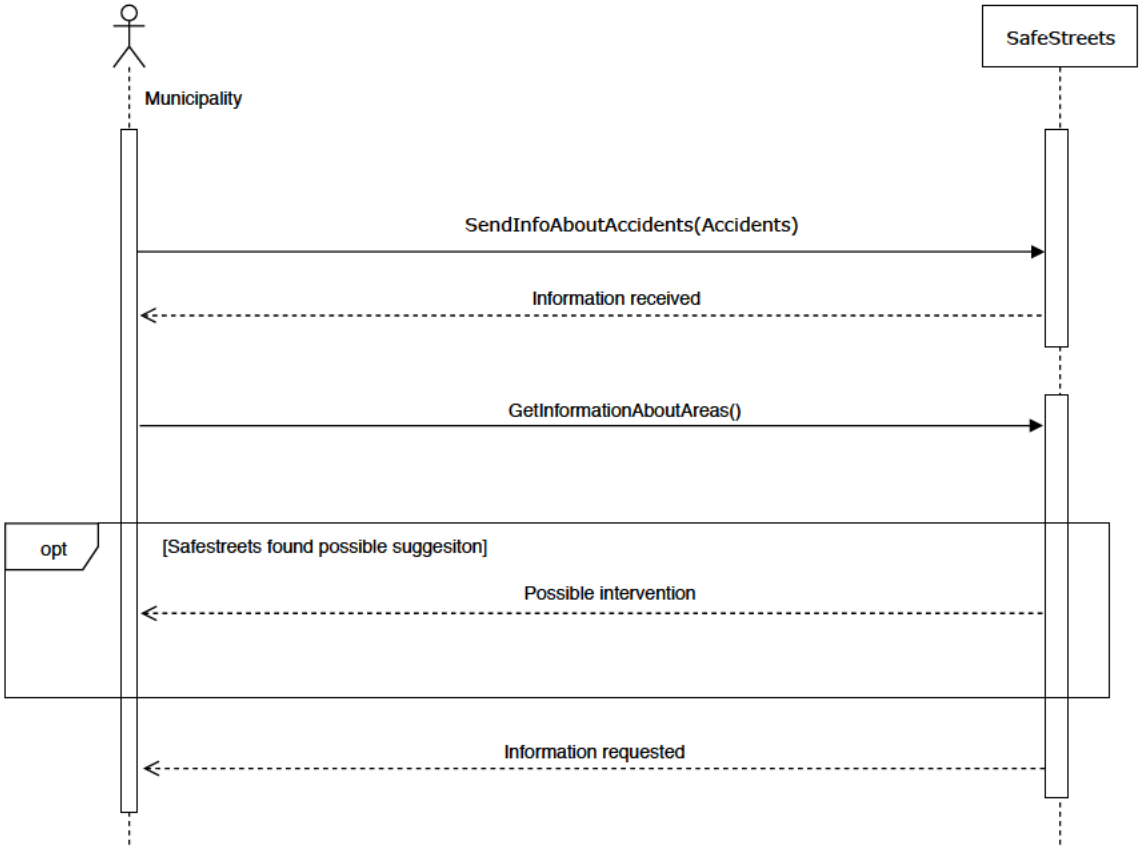
5. User extracts general information

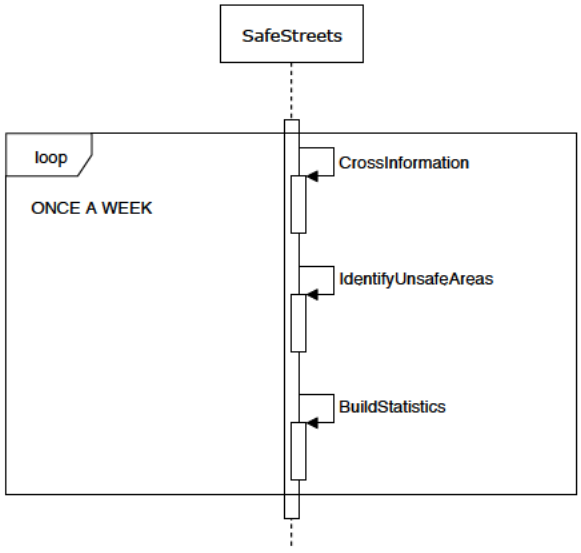


6. Authority extracts information

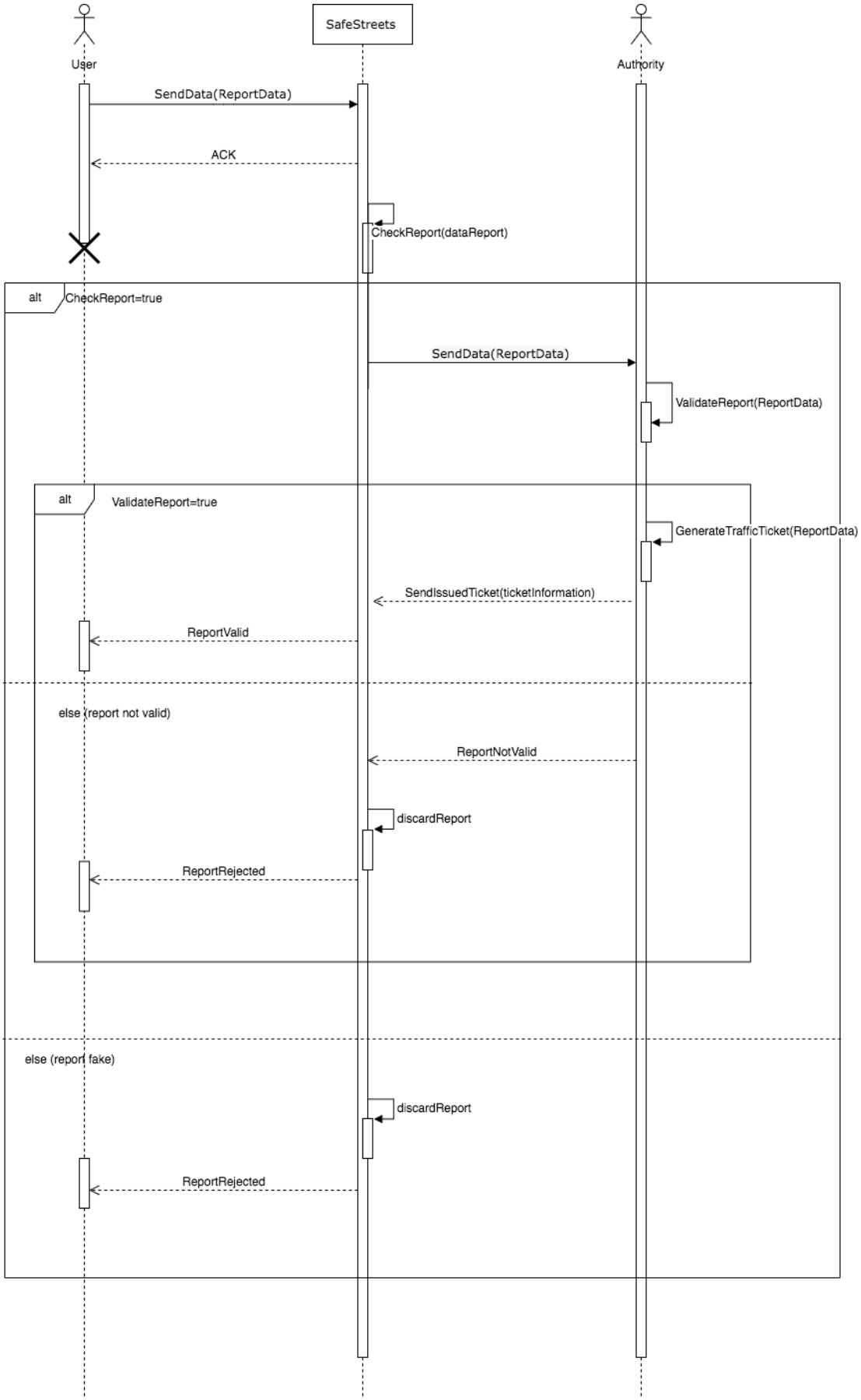


7. Municipality does area request and takes possible intervention

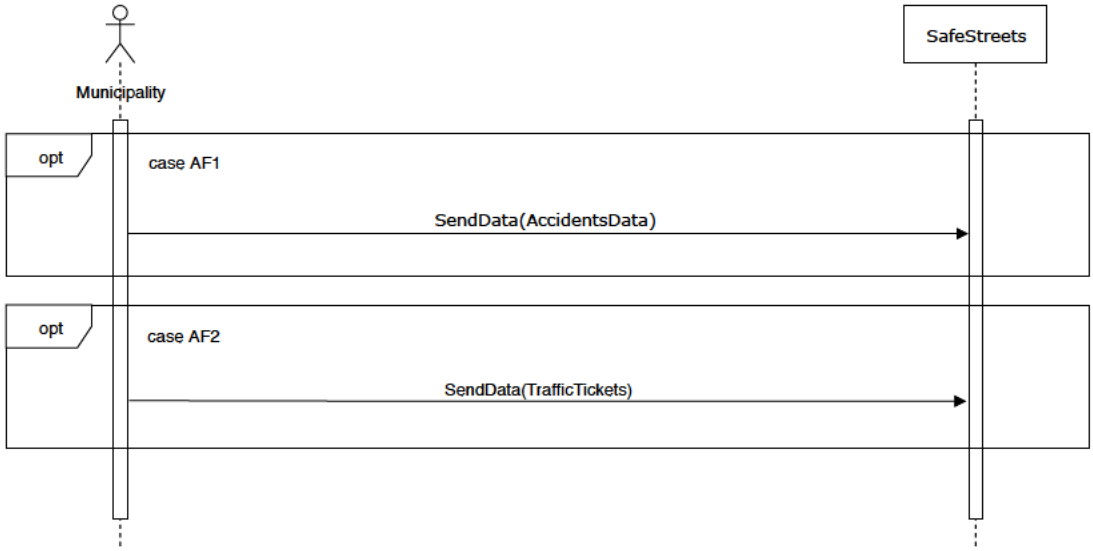


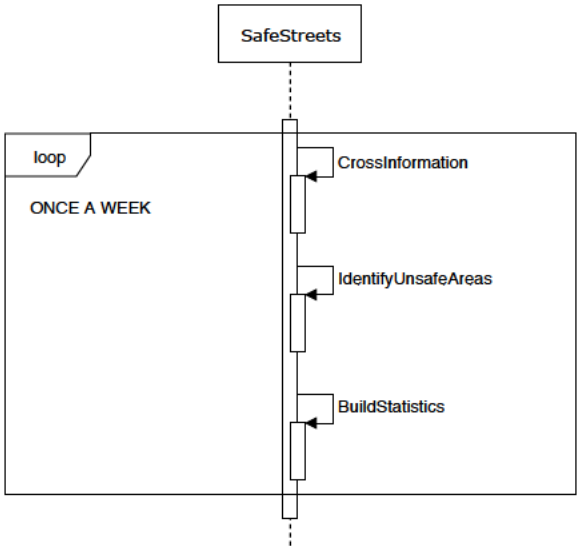


8. Provide data to generate traffic tickets

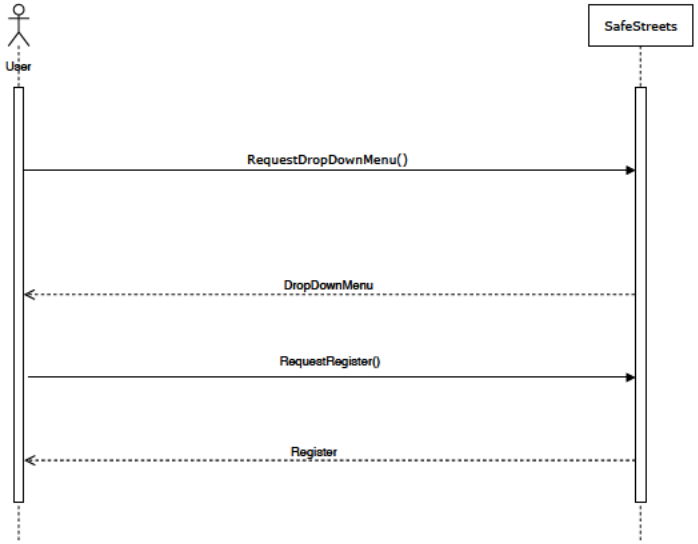


9. Build statistics

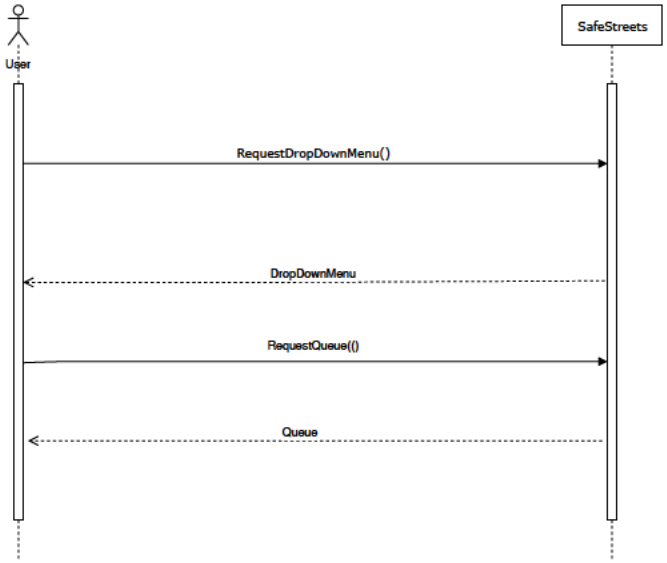




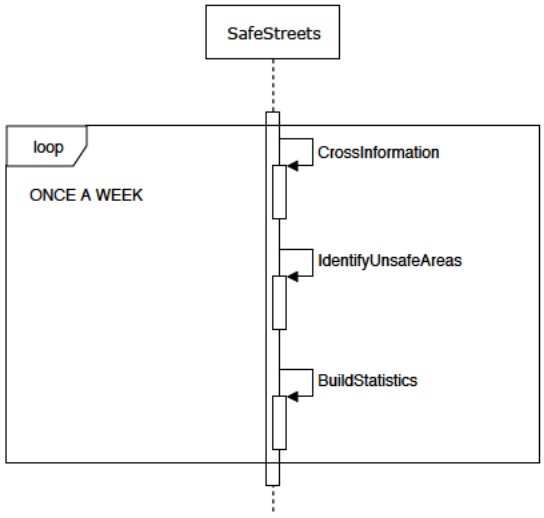
10. See registry of reports



11. See queue



The above diagrams describe specific and single events, which can be repeated more than once over time. For example, an actor can make to the system as many requests as he wants, and SafeStreets is able to receive any information at any time.



Moreover, identification of unsafe areas and statistics are computed periodically, in particular once a week, using the data stored in that moment in the database. This happens because of the size of data used to perform these functions, which may require a lot of time to be analysed.
This picture appears several times in the previous diagrams which are directly connected to these two functions.

3.2.5 Scenarios

Scenario 1: GENERAL
Mara is a mum of two children: Luca and Mario. Unfortunately, the younger one, Mario, has a disability so his mum has to take him everywhere by car. But one day, while Mara is taking his son to school, she finds a disabled parking that is occupied by a car without licenses to park there. However, Mara, now, is capable of doing something about it thanks to SafeStreets. Hence, she gets her phone out, opens the SafeStreets application, makes some pictures that clearly show the violation, fills the mandatory fields (like the type of violation, the name of the street and also an optional comment) and then reports it.

Scenario 2: USER EXTRACTS INFORMATION
Alessio is a young man that lives in a small town with only one street and where everybody knows each other. He leaves his town for Milan by car due to his work, but he has never been in a big city, so he does not know what the road hazards could be. Therefore, he uses the service provided by SafeStreets to mine the information so he can discover not only what are the most common violations and where they are committed but also can identify which are the most dangerous areas. By doing so, he can avoid a lot of traffic tickets and also the unsafe areas without being expert of Milan's streets.

Scenario 3: AUTHORITIES EXTRACTS INFORMATION
John is the chief of the authorities and does not have a lot of subordinates available, but he wants to give the example to the citizens that the law must be respected. In order to do so he opens the SafeStreets' application on the computer to find out where the majority of the violations are committed, thanks to the database of the application, thus he sends his men in these streets to do as many traffic tickets as possible. So, the number of traffic tickets emitted will securely be higher than sending men randomly.

Scenario 4: NOT A TRUE VIOLATION
Giacomino, a 23 years old man, is going back home on foot from a campus party where he drank a lot. While he is walking down the street, he finds a car and he thinks that is not properly parked so he, using the SafeStreets app, shoots some photos, completes the report and sends it. Then SafeStreets forwards the report to the municipality, but the last one clearly sees that the violation is not a real violation because there are no infractions committed and, therefore, the car is parked fine. Due to this reason the municipality rejects the report and then notifies SafeStreets of this, which in turn notifies Giacomino that his report has been rejected.

Scenario 5: NEW INTERVENTION FOR BENEVENTO
The municipality of Benevento, thanks to its jobs' perspective, has lately noticed a great number of people moving to the city, but this increase in population goes together with an augmenting number of accidents. Thanks to the subscription to SafeStreets, all data about violations and accidents of the municipality are analysed by the system: the results show that the problems are mainly focused in some residential areas. SafeStreets also advises to extend the residential parking area because many accidents are caused by parking violations that occur due to overcrowding. Finally, the municipality can take measures and enhance the safety of these areas.

Scenario 6: DISCOVERING ACCIDENTS
Alessandro chooses to go to Latina for the weekend to meet his girlfriend that works there. He will get there by car crossing the city of Rome and decides to use the SafeStreets application to look up if there are any accidents along the road. He finds out that an accident occurred on the planned route and there are many cars stalled: the expected delay is of three hours. Knowing that, Alessandro chooses a different road avoiding every problem, so as to spend more time with his beloved.

Scenario 7: ATTEMPT TO CHEAT
Camilla is a young businesswoman that hates her colleague that tried to seduce her boyfriend. Camilla is very angry and wants revenge. She creates a report and takes some pictures of the colleague's car parked near a sidewalk but specifying as the position of violation a place that looks very similar to the one in the picture but in which parking is not allowed. SafeStreets receives the report and runs the algorithm to provide reliable data to the municipality and compares the GPS Position of the pictures with the position inserted by the user, discovering that the two do not match. SafeStreets does not forward the violation to the municipality and Camilla receives a notification stating that her report has been discarded.

Scenario 8: MISTAKES HAVE TO BE PAID
Giacomina is a young woman concerned about environmental issues and is used to go to work by bicycle given that the office is not so far and there are good bike lanes. She is recently experiencing some problems due to many car drivers parking in the middle of bike lanes "for just few minutes" to go grocery shopping in the near store, avoiding paying the regular parking ticket. Giacomina cannot stand this behaviour and takes pictures of the situation and reports through SafeStreets all these violations. Municipality is notified and measures are taken by writing traffic tickets. After some days of active reporting by Giacomina, the violations drastically decrease and the young woman also notices much more bikes parked in front of the store. Thanks to SafeStreets, bike lanes are safer and the use of healthier means of transport has been promoted.

3.3 Performance Requirements - Non-Functional Requirements

The system should be able to send a report to the authorities not after 10 seconds since it has been received.
Moreover, it must be able to guarantee the simultaneous connection of 100.000 individuals.
Finally, it should be able to send a response to a query and run its algorithm on metadata in less than 5 seconds.

3.4 Design Constraints - Non-Functional Requirements

3.4.1 Standards compliance

The code should follow the requirements contained in this document. Furthermore, its comments should be clear and focused.

3.4.2 Hardware limitations

The software application requires a mobile device able to capture the position and to take pictures. In alternative, the authorities, that don't need to take pictures, can use a computer to receive messages by the application and send to it some query.
Both devices can be able to send data to the software via internet connection.

3.4.3 Any other constraint

A user has a limitation of at most 10 pictures to send in a single report.

3.5 Software System Attributes - Non-Functional Requirements

3.5.1 Reliability and availability

The reliability of the system depends on the services of the system.
The system should be up for a 99% of time. That means that the average time between the occurrence of a fault and service recovery (MTTR or downtime) should be contained around at 3.65 days per years. It does not need to provide a higher availability because the service is not strictly related with emergency situations. Moreover, the service is not fully automized because has to rely on the municipality's employees that follow office hours.
In order to guarantee this time of downtime the system must have an appropriate infrastructure with a fully backup system located in different office that replicates the core services for covering general failure of the main system. Appropriate personnel must guarantee adequate recovery time of the hardware that stop to work properly. In case the system is not working properly users must be aware about the failure within 10 minutes with a specific message addressed to the devices. An equivalent message should be sent when the system goes back to work properly.

3.5.2 Security

The data stored in the system has to be encrypted and also the password of the municipality has to be hashed before stored. Moreover, in case of password recovery this should never be sent in clear. The system should be protected against intrusion from agent that are not authorized to access it.

3.5.3 Maintainability

The system must be written in java and guarantee a high level of maintainability. Code must be written with good standard with high level of abstractions without hard-code as well. Code must be written with large use of comments that cover all aspect of code itself. Code must provide testing routine that covers at least 70% of the entire code, excluding software interface.

3.5.4 Portability

The software must run in different platforms like Windows operating system, Linux operating system and Mac operating system. Also, must support Android and iOS operating systems for mobile devices.

3.5.5 Scalability

Because the new business is very helpful to cities and many citizens could exploit this service, the system must be scalable without the necessity of reformulating core part of the software for a rising number of individuals until 100,000 (10 time the base). In particularly the system must guarantee the requested level of reaction time (10 seconds) for 100,000 of individuals as users.

3.6 Additional Specifications

3.6.1 Mandatory Fields

At the moment of the registration, an authority will have to compile the following mandatory fields:

- City, Province, CAP, State
- Certification document, which proves the fact of being a real authority
- Phone number
- E-mail

3.6.2 Types of Violations

During the compilation of a report, the user can choose from a list of predefined types of violation, which is specified below:

- Parking on the crosswalk
- Parking on the sidewalk
- Parking in front of a driveway
- Parking in a restricted area
- Parking on spaces reserved for residents
- Parking near public transportation stops
- Parking on spaces reserved for disabled
- Parking within 5 meters from an intersection
- Parking in front of a ramp reserved for disabled
- Double parking
- Parking on bike lines
- Other

3.6.3 Types of Query

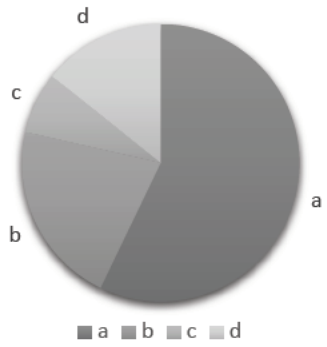
Municipalities which wants to extract information about an individual can insert a license plate on the dedicated field and then will receive all the violations (and tickets) regarding this one.
Users and Municipalities can choose between the following types of query to extract general information:

- A map which contains information about the accidents occurred in a specific territory. In this map the areas are coloured with three different colours: Green (SAFE, 0 accidents), Yellow (LESS SAFE, less than 5 accidents), Red (UNSAFE, more than 5 accidents). In addition, the positions of past accidents are marked with red points.

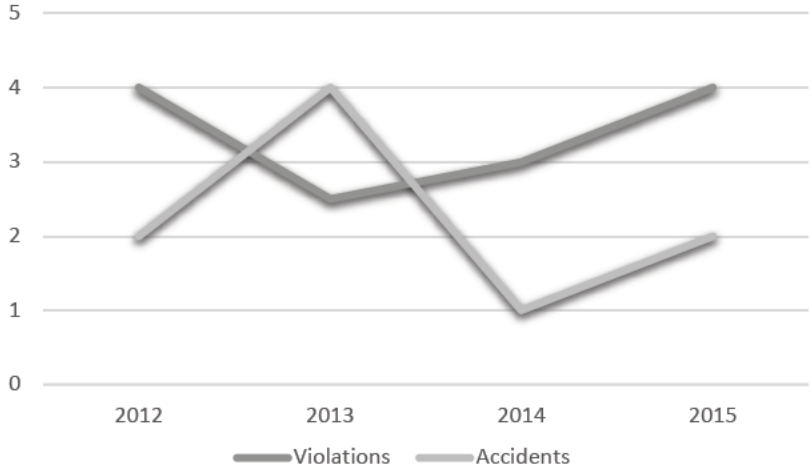
- A map which contains information about the streets and its correspondent number of violations, highlighted with three different colours: Green (no violation), Yellow (less than 10 violation), Red (more than 10 violation).

The above functionalities provide to the user/authority to choose the granularity of the territory between the entire city or one single district.

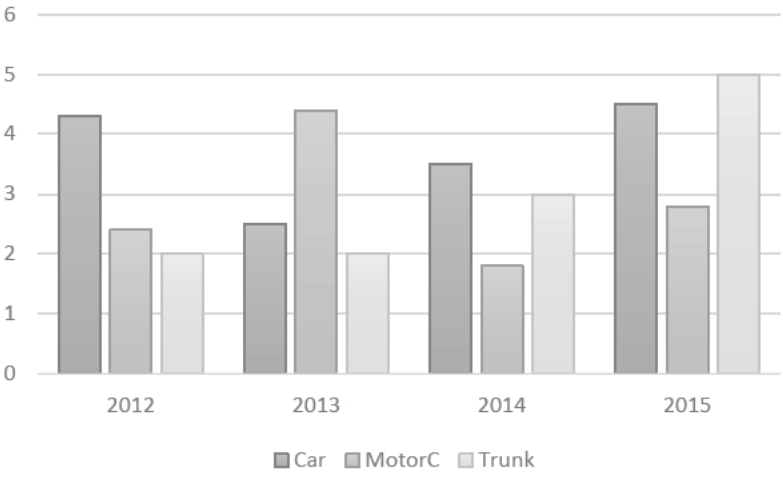
- A pie chart which show the most common types of violations.



- A line charts which show the trend of accidents and violations in a given period for a specific territory.



- A histogram which shows the most common types of vehicle that commit violations: car, motorcycle, truck, etc.



The above functionalities provide the user/authority to choose the level of granularity of the territory among city, region or state, and the granularity of the period divided in months or years.

4. Formal Analysis Using Alloy

4.1 Alloy Code

This section is dedicated to the Alloy model of the SafeStreets software, included all the functions of the application and their most important constraints. In this model, we want to prove that:

- Only the municipality can do requests referred to a single vehicle.
- All the actors can do general requests about types of violations and accidents in some city areas (safe or unsafe).
- All the valid violations are stored in the database.
- All the accidents data and issued tickets received from the municipality are stored in the database.
- When the municipality validates or rejects a report, the user is notified through his report registry, and the information will be stored or not in the database.

In this model, we assumed that all the pictures sent to the municipality are not fake.

```
abstract sig Position{}

sig City{}

sig Street{}

sig ExactPosition extends Position{
city: one City,
street: one Street,
civicNumber: one Int
}

sig GPSPosition extends Position{
latitude: one Float,
longitude: one Float
} {
latitude.beforePoint < 90 and latitude.beforePoint > -90
longitude.beforePoint < 180 and longitude.beforePoint > -180
}

sig Float{
beforePoint: one Int,
afterPoint: one Int
} {
afterPoint>0
}

abstract sig Safety{}

one sig SAFE extends Safety{}
one sig LESSSAFE extends Safety{}
one sig UNSAFE extends Safety{}
```

```
sig Date{
number: one Int,
month: one Int,
year: one Int
} {
number>0
month>0
year>0
}

abstract sig State{}
one sig VALIDATE extends State{}
one sig REJECTED extends State{}
one sig EVALUATION extends State{}

sig Time{
hours: one Int,
minutes: one Int,
seconds: one Int
} {
hours>=0
minutes>=0
seconds>=0
}

sig Report{
id: one Int,
comment: lone String,
state: one State,
date: one Date,
violation: one Violation,
time: one Time,
} { id>0 }

sig LicensePlate{
code: one Code,
violations: set Violation
} {
violations.licensePlate.code.id = code.id
violations in Data.violations
}

sig Violation{
licensePlate: one LicensePlate,
vehicleType: lone String,
violationType: set Type,
exactPosition: one ExactPosition,
pictures: set Picture,
ticket: lone Ticket
} { licensePlate in Data.licensePlates }
```

```
sig ViolationSent{
vehicleType: lone String,
violationType: set Type,
exactPosition: one ExactPosition
}

sig Picture{
date: one Date,
time: one Time,
position: one GPSPosition
}

sig Code{
id: set String
}

sig Type{
type: set String
}

sig User{
registry: set Report,
queue: set Report
}

one sig Data{
violations: set Violation,
accidents: set Accident,
licensePlates: set LicensePlate
}

sig DataSent{
violations: set ViolationSent, //violation that not contain license plate for general request
accidents: set Accident, //area request
licensePlates: set LicensePlate, //that contains the set of violations performed
} {
accidents in Data.accidents
licensePlates in Data.licensePlates
}

sig Statistics{}

sig Area{
safetyLevel: one Safety,
delimitedBy: set ExactPosition
}

sig Intervention{
description: one String
}
```

```
sig Ticket{
price: one Float
} { price.beforePoint > 0 }

sig Accident{
date: one Date,
time: one Time,
position: one ExactPosition
}

sig Municipality{
id: one Int,
cityName: set String,
dividedIn: set Area
} { id > 0 }

sig LocalPolice extends Municipality{}

//user sends a report to municipality
sig ReportSent{
sender: one User,
report: one Report,
receiver: one Municipality
}

//municipality validates or not the report received
sig ValidationReceived{
result: one State,
sender: one Municipality,
report: one Report,
ticket: lone Ticket,
receiver: one User
}

//request to mine information
abstract sig Request{}

sig IndividualRequest extends Request{
municipality: one Municipality,
subject: one LicensePlate
} { subject in Data.licensePlates }

sig GeneralRequest extends Request{
municipality: lone Municipality,
user: lone User,
type: one Type
} { (municipality = none and user != none) or (municipality != none and user = none) }
```

```
//request regarding unsafe areas
sig AreaRequest extends Request{
user: lone User,
municipality: lone Municipality,
area: set Area
} { user != none or municipality != none }

//answer a request with a subset of data
sig AnswerRequest{
request: one Request,
dataSent: one DataSent
}

//FACTS

//every license plate corresponds to at least one violation
fact allLicensePlateCorrespondsAViolation {
all l: LicensePlate | some v: Violation | l = v.licensePlate
}

fact allLicensePlateAreInData {
all l: LicensePlate | one d: Data | l in d.licensePlates
}

fact allRealViolationAreInData {
all r: Report | one d: Data | r.state = VALIDATE implies r.violation in d.violations
}

fact allRejectedViolationAreNotInData {
all r: Report | one d: Data | r.state = REJECTED implies r.violation not in d.violations
}

fact allAccidentsAreInData {
all a: Accident | one d: Data | a in d.accidents
}

fact noSameReportInQueueAndRegistry {
all u: User | no r1, r2: Report | r1 in u.registry and r2 in u.queue and r1 = r2
}

//one answer for each request
fact oneAnswerRequestForEachRequest {
all a: AnswerRequest | one r: Request | a.request = r
}

//when the answer is individual the data sent are about license plates
fact correctAnswerIndividualRequest {
all a: AnswerRequest | (a.request = IndividualRequest) implies
(a.dataSent.violations = none and
a.dataSent.accidents = none and
a.dataSent.licensePlates != none )
}
```

```
//when the answer is general the data sent are about violations
fact correctAnswerGeneralRequest {
all a: AnswerRequest | (a.request = GeneralRequest) implies
(a.dataSent.violations != none and
a.dataSent.accidents = none and
a.dataSent.licensePlates = none)
}

//when the answer is for unsafe areas the data sent are about accidents
fact correctAnswerAreaRequest {
all a: AnswerRequest | (a.request = AreaRequest) implies
(a.dataSent.violations = none and
a.dataSent.accidents != none and
a.dataSent.licensePlates = none)
}

//every validation corresponds to a unique report
fact oneReportSentForEveryValidationReceived {
all v: ValidationReceived | one r: ReportSent | r.report = v.report
}

//when a validation is sent the state of the report appears in the registry of the user
fact correctStateInValidationReceivedAndRegistry {
all v: ValidationReceived | v.result = v.report.state and
all v: ValidationReceived | one u: User | v.receiver = u and v.report in u.registry
}

//report validate implies its data are stored
fact correctValidationReceivedInData{
all v: ValidationReceived | v.result = VALIDATE implies (v.report.violation in Data.violations and (#v.ticket=1
implies v.report.violation.ticket=v.ticket))
}

//report rejected implies its data are discard
fact rejectedValidation {
all v: ValidationReceived | v.result = REJECTED implies (v.report.violation not in Data.violations and
#v.ticket=0 and #v.report.violation.ticket=0)
}

//one Area corresponds to a unique Municipality
fact areaForAUniqueMunicipality {
all a: Area | one m: Municipality | a in m.dividedIn
}

fact uniqueIDMunicipality {
all disj m, m': Municipality | m.id != m'.id
}

fact uniqueIDReport {
all disj r, r': Report | r.id != r'.id
}
```

```
fact uniqueLicensePlate {
all disj l, l': LicensePlate | l.code != l'.code
all disj c,c': Code | c.id != c'.id
}

//different answer for each request
fact differentAnswerRequest {
all disj a, a': AnswerRequest | a.request != a'.request and a.dataSent != a'.dataSent
}

//all violation sent correspond to a violation
fact oneToOneCorrespondencebetweenViolationAndViolationSent{
all vs: ViolationSent | one v: Violation | vs.vehicleType = v.vehicleType and
vs.violationType=v.violationType and vs.exactPosition = v.exactPosition
}

//ASSERTIONS

//no report with more than 10 pictures
assert tenPictures {
no r: Report | #r.violation.pictures <= 10
}

check tenPictures

//PREDICATES

//user sends report to municipality
pred sendReport(u: User, r: Report, rs: ReportSent, m: Municipality, d: Date, v: Violation){
r.state = EVALUATION
r in u.registry
rs.sender = u
rs.report = r
rs.receiver = m
r.date = d
r.violation = v
}

run sendReport

//municipality validates the report
pred receiveValidation(vr: ValidationReceived, u: User, r: Report, m: Municipality, v: Violation, d: Data){
r.state = VALIDATE
r in u.registry
r.violation = v
vr.result = r.state
vr.sender = m
vr.report = r
vr.receiver = u
v in d.violations
}

run receiveValidation
```

```
//municipality rejects the report
pred receiveRejection(vr: ValidationReceived, u: User, r: Report, m: Municipality, v: Violation, d: Data){
r.state = REJECTED
r in u.registry
r.violation = v
vr.result = r.state
vr.sender = m
vr.report = r
vr.receiver = u
v not in d.violations
}

run receiveRejection

//user and municipality send general, individual and area requests
pred showRequests {
#User = 2
#Municipality = 1
#Report = 2
#Ticket = 2
#Accident = 3
#Area = 3
#Data = 1
#DataSent = 3
#Request = 3
#AreaRequest = 1
#IndividualRequest = 1
#AnswerRequest = 3
}

run showRequests for 3
```

4.2 Metamodel

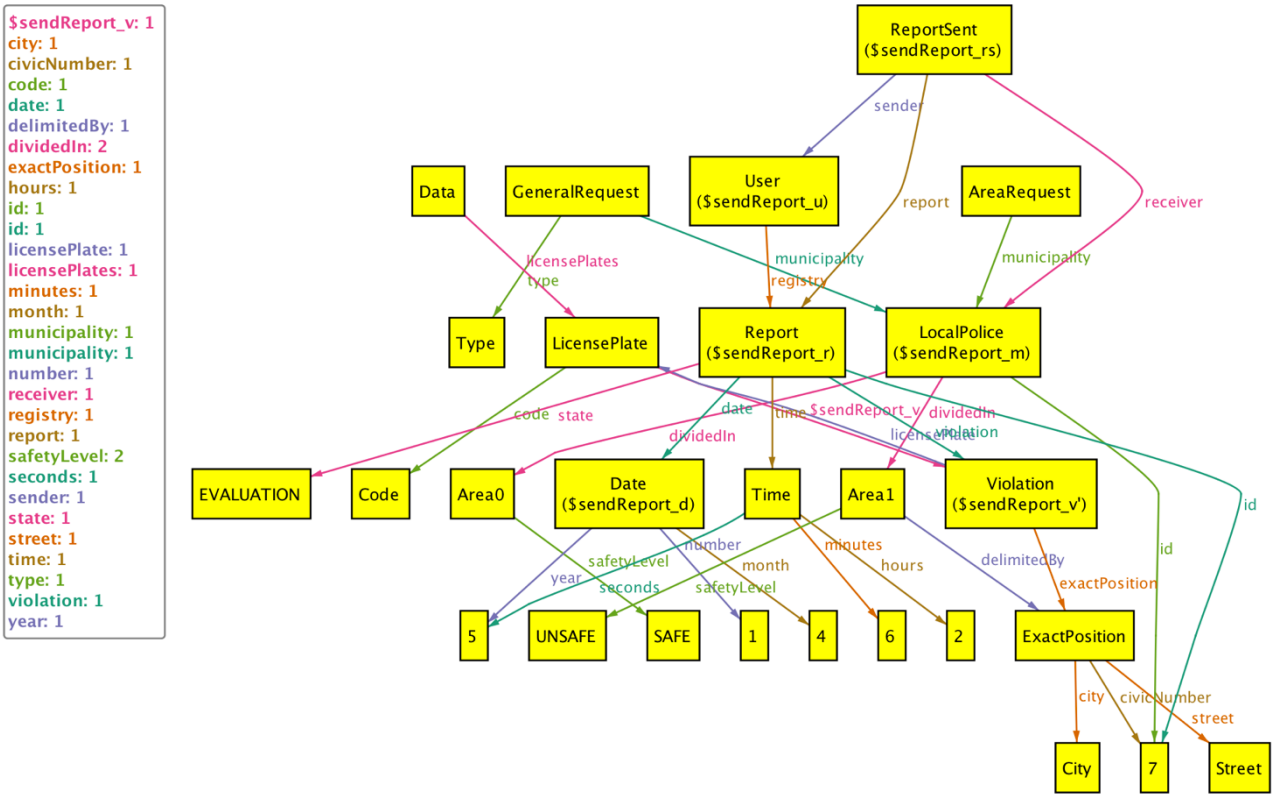


Figure 1 – sendReport

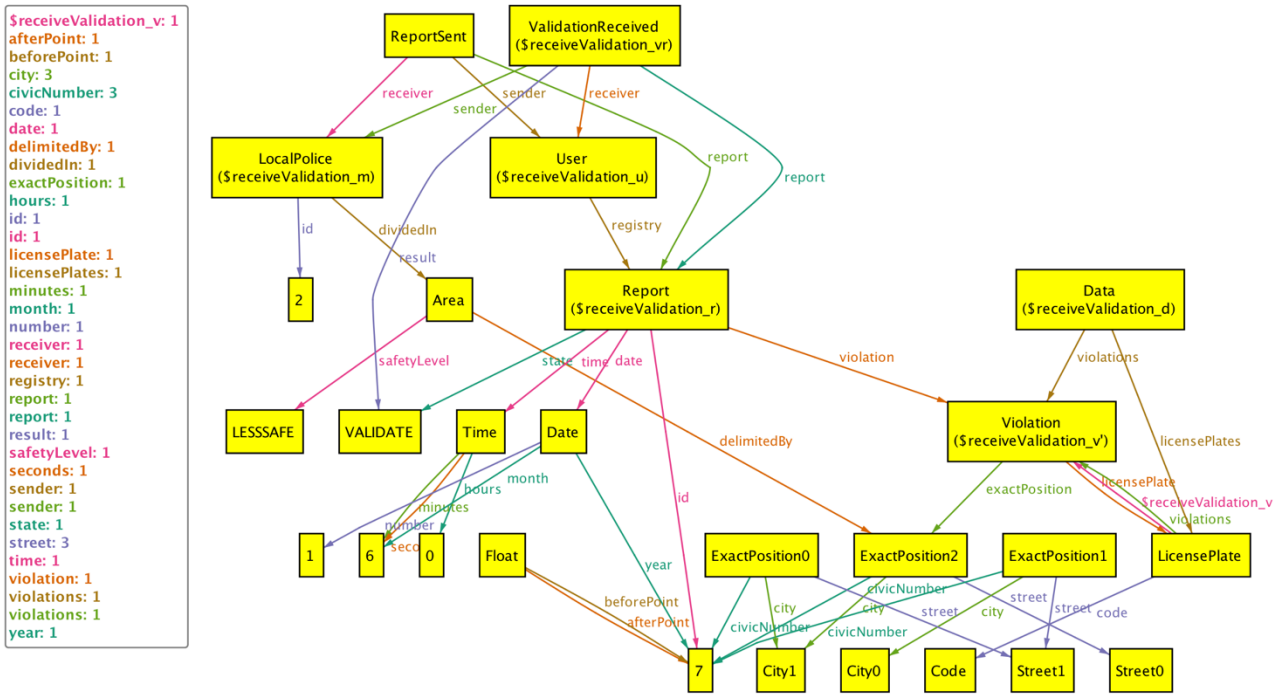


Figure 2 – receiveValidation

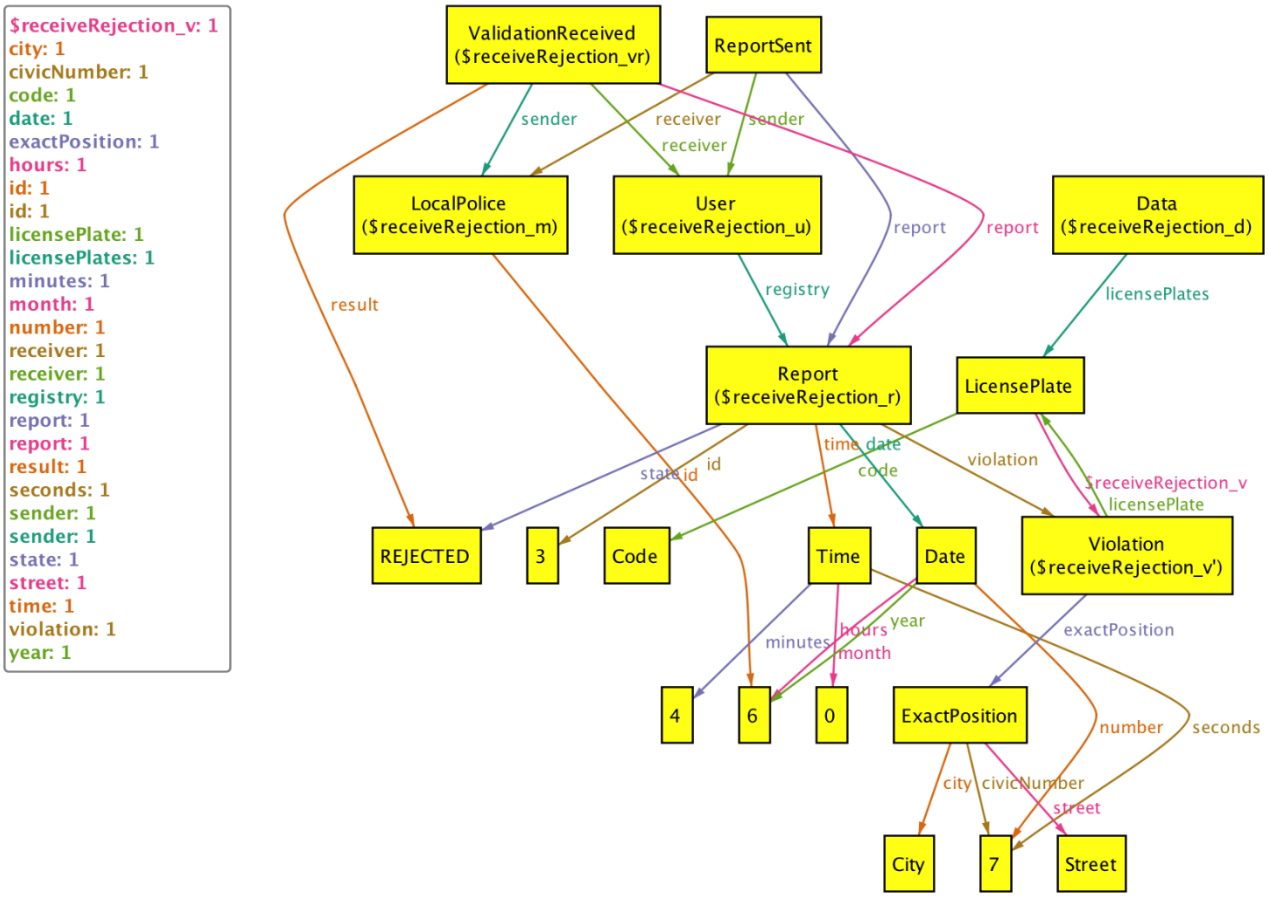


Figure 3 – receiveRejection

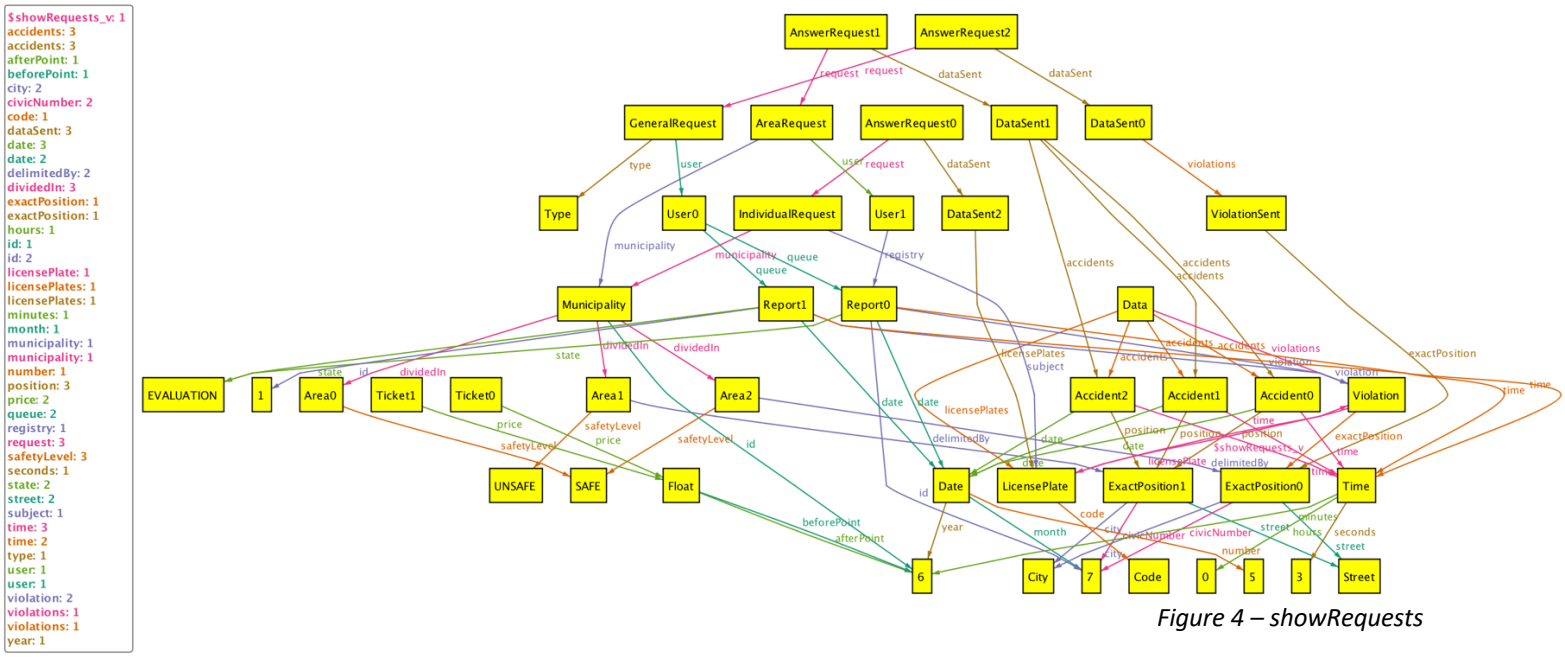


Figure 4 – showRequests

4.3 Result of Assertions

```
Executing "Check tenPictures"
  Solver=minisatprover(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  0 vars. 0 primary vars. 0 clauses. 381ms.
  No counterexample found. Assertion may be valid. 77ms.
```

4.4 Result of Predicates

```
Executing "Run sendReport"
  Solver=minisatprover(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  14209 vars. 1101 primary vars. 27785 clauses. 93ms.
  Instance found. Predicate is consistent. 53ms.

Executing "Run receiveValidation"
  Solver=minisatprover(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  14198 vars. 1099 primary vars. 27791 clauses. 90ms.
  Instance found. Predicate is consistent. 55ms.

Executing "Run receiveRejection"
  Solver=minisatprover(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  14198 vars. 1099 primary vars. 27789 clauses. 101ms.
  Instance found. Predicate is consistent. 62ms.

Executing "Run showRequests for 3"
  Solver=minisatprover(jni) Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  14022 vars. 1083 primary vars. 27488 clauses. 110ms.
  Instance found. Predicate is consistent. 62ms.
```

5. Effort Spent

Topic	Hours
Discussion on first part	3h
Purpose & Scope	2h
World & Shared Phenomena & Goals	3h
External Interface Requirements	3h
Performance Requirements & Design Constraint	2h
UML & statecharts	2h
Discussion on third part	3h
Writing requirements	2h
Use cases	4h
Discussion on third part	3h
Mapping	3h
Alloy code	10h
Document composition	3h

Topic	Hours
Discussion on first and second part	3h
Domain assumption	2h
Product functions	3h
User characteristics	1h
UML & state diagrams	2h
Discussion on topic 3	3h
Added some specifications in part 2	1h
Build scenarios	3h
Revision of requirements	1h
Use cases	4h
Sequence diagram	3h
Revision on part 3+mapping goal, domain assumption and requirement	4h
Software System Attributes	2h
Alloy Code	6h

Topic	Hours
Discussion on first and second part	3h
Domain assumption	2h
Product functions	3h
UML diagram	3h
State diagram	1h
Discussion on third chapter	3h
UML description	1h
Added some specifications in part 2	1h
Build scenarios	3h
Revision of requirements	1h
Use cases	3h
Sequence diagram	3h
Revision on part 3+mapping goal, domain assumption and requirement	4h
Software System Attributes	2h
Alloy Code	6h

6. References

- All the diagrams have been made with *https://www.draw.io*
- Alloy code development has been supported by *http://alloy.mit.edu/*
- Wikipedia *https://www.wikipedia.org/*