POLITECNICO
MILANO 1863

Software Engineering 2 Project

Politecnico di Milano AA 2019-2020
Computer Science and Engineering

RASD - Requirements Analysis and Specifications Document

Professor:

Elisabetta Di Nitto

Group Components:

Fiozzi Davide (Matricola: ***945107***)

Frantuma Elia (Matricola: ***945729***)

Freddi Eleonora (Matricola: ***945113***)

Contents

1. Introduction

1.1 Purpose

1.1.1 General Purpose

SafeStreets is a crowd-sourced application that intends to provide multiple software-based services concerning road and traffic violations.
In detail, it allows:

1. users to have the possibility to notify the system when traffic violations occur, and in particular parking violations;
2. to cross system's data with those made available by the municipalities, like information of local accidents, to identify potentially unsafe areas, and let the system suggest possible interventions to the involved municipality;
3. local police of a municipality to search for violations in SafeStreets database and generate traffic tickets based on those reported by registered users.

SafeStreets stores the information provided by users, completing it with suitable metadata. In particular, when it receives a picture of a violation, it runs an algorithm to read the license plate. Each user's report includes the type of the violation and the name of the street where the violation occurred, retrieved from the geographical position of the infringement.

One of the main aims of SafeStreets is to collect this information so that both end users and authorities can identify areas where multiple violations occur and the vehicles that commit the most violations.

If a municipality offers a service that allows users to retrieve the information about the accidents that occur on its territory, SafeStreets can cross this information with its own data to identify potentially unsafe areas, and suggest possible interventions, like adding a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking.

The municipality and the local police could offer a service that takes the information about the violations coming from SafeStreets, and generate traffic tickets from it. In this case, entering a violation ensures that the information coming from the users is never broken, and the information is never altered, because all the provided pictures come directly from the application built-in camera. The information about issued tickets are used by SafeStreets to build statistics in order to be able to measure the effectiveness of the SafeStreets initiative.

1.1.2 Goals

SafeStreets service aims to achieve multiple goals:

- G1: Allow users to log in the system
- G2: Allow users to notify a traffic violation
- G3: Allow users to receive feedback on the notification made (it can be accepted or refused)
- G4: Allow users to see the history of their own notified violations
- G5: Allow the possibility for the municipality to generate traffic tickets from the identified violations
- G6: Build statistics starting from the information about issued tickets
- G7: Allow authorities to access to the stored information
- G8: Cross system's data with those made available by the municipalities
- G9: Allow the municipality to receive suggestions for possible interventions
- G10: Allow users to access to certain stored data (such as the most dangerous streets in a city)

1.2 Scope

The purpose of the SafeStreets' initiative is to raise citizens' awareness about the violations that are committed every day, allowing them to make reports for a more civilized city. In addition, for a municipality, a service like SafeStreets can help to determine which areas are considered unsafe and those with high rate of violations, so authorities can act on more accurate data and help to make the municipality optimally controlled. Finally, for authorities, being able to leverage on the service provided by SafeStreets can help to reduce the time needed to emit traffic tickets and to invest it in more important, resource-intensive activities.

1.2.1 Phenomena Table

In the table below some phenomena have been highlighted, both those that happen in the world and those that happen inside the machine. Some world phenomena are shared with the machine, and this is indicated in the second column. In the first column is indicated the name of the phenomenon and in the last one who controls it, the world or the machine.

Phenomena	Shared	World/Machine
Violation committed	No	W
Notification made	Yes	W
Notification stored	Yes	M
Ticket issued	Yes	W
Notification accepted	Yes	M
Notification refused	Yes	M
A user signs up	Yes	M
A user logs in	Yes	M
Authority adds an accident to his DB	Yes	W
Authority/User requests data	Yes	W
System Provides data	Yes	M
Operator builds a statistics	Yes	W
An operator makes a suggestion	No	W
A suggestion is applied	No	W
An authority operator signs up	Yes	M
An authority operator logs in	Yes	M

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Customer: a generic SafeStreets customer who must register to it (can be user or authority).
- User: a normal application customer who notifies the authorities of a traffic violation.
- Authority: customer of the application who intervenes as a result of a traffic violation. It can be distinguished into municipality or local police.
- Violation: a set of photo and metadata that describes an attitude that is not in accordance with the rules of the road.

1.3.2 Acronyms

- GPS: Global Positioning System
- RASD: Requirement Analysis and Specification Document
- OS: Operating System
- API: Application Programming Interface
- PC: Personal Computer

1.3.3 Abbreviations

- Gn: nth goal
- Dn: nth domain assumption
- Rn: nth requirement

1.4 Revision history

- Version 1.0: First release

1.5 References

- Specification document: “SafeStreets Mandatory Project Assignment”
- Document “RASD to be analyzed - A.Y. 2019-2020”
- Slides “RASD”

1.6 Document Structure

The RASD document consists of 6 chapters described below:

1. The first chapter is an introduction. It defines the purpose of the system, underlining the goals that the application has to reach and it describes the scope of the project. It also contains some basic information to better understand the other chapters of the document.

2. The second chapter is made up of a description of the project. Here are identified and described the actors, the limits of the system and the necessary assumptions. This section includes details on the shared phenomena and a domain model, with class diagrams and state diagrams.

3. The third chapter is the most important of the document and it provides a more detailed description of the aspects presented in the previous chapter. It contains all the identified requirements necessary to achieve the goals defined in the first chapter. Here interface, functional e non functional requirements are defined. This chapter additionally contains a description of some of the scenarios identified, each of them describes a particular situation where the system has to deal with. Here are also defined the use case diagrams, use cases and associated sequence diagrams, that model the system in details.

4. The fourth chapter contains the Alloy model of some critical aspects of the system. Everything is explained in detail; in addition, to show the soundness and correctness of the model, a proof of consistency and examples of the generated worlds are provided.

5. The fifth chapter contains a detailed report of all the hours spent by each member of the group working on the project.

6. The last chapter includes references to the documents, texts and resources used to write this document.

2. Overall Description

2.1 Product Perspective

The idea is to build the whole software as a unique system that provides the basic service and also the advanced functions.
Here below the Class Diagram provides a model of the application domain. It contains the essential attributes of the classes evidenced, the more important ones of the model of the system and not the whole set of classes that will be useful to define it.

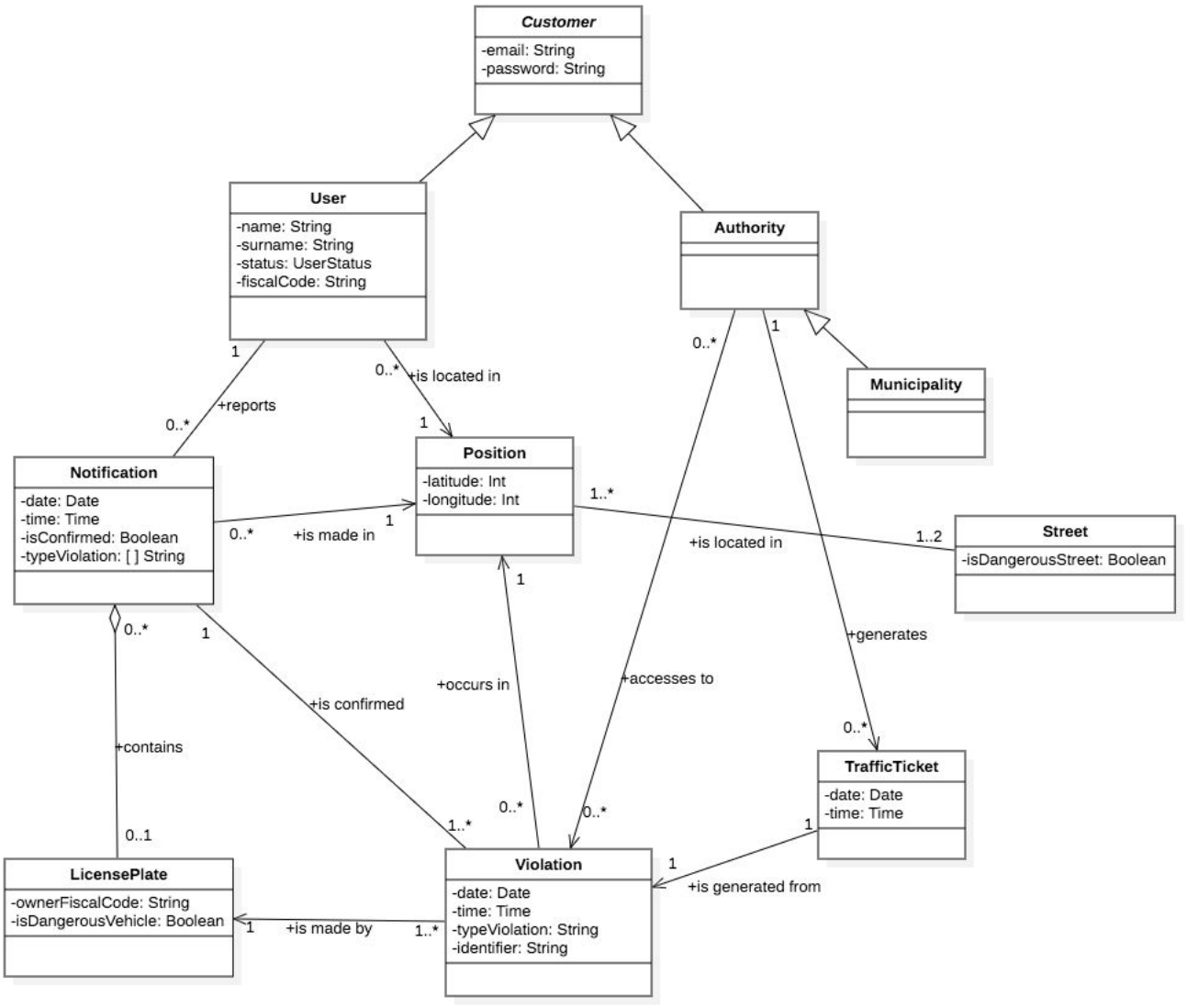

Figure 1: Class Diagram

The users and the authorities never communicate to each other directly but they exploit the SafeStreets' functions to establish an asynchronous communication in which a user can take a picture to notify a violation and send it to SafeStreet, where it is stored and next an operator will analyse to make it available to the authorities. Subsequently an authority operator will access to the stored data and in particular to the oldest notification and he will analyse it and if it is appropriate he will generate a ticket against the vehicle who committed the violation, the result of his activities is reported to SafeStreets.
The Safestreets' operators don't only act as intermediaries in the communication but they have also a more important role, they are supervisor of the whole SafeStreets project. The operators can perform this role building statistics about the effectiveness of the SafeStreets' activities looking at how many traffic tickets are generated thanks to the application's notifications or moreover they can, on their own initiative, suggest to authorities some solutions to avoid possible accident and make safer the most dangerous streets.

Now we are going to model the most important behaviors of the system and show how it evolves over time using state diagrams:

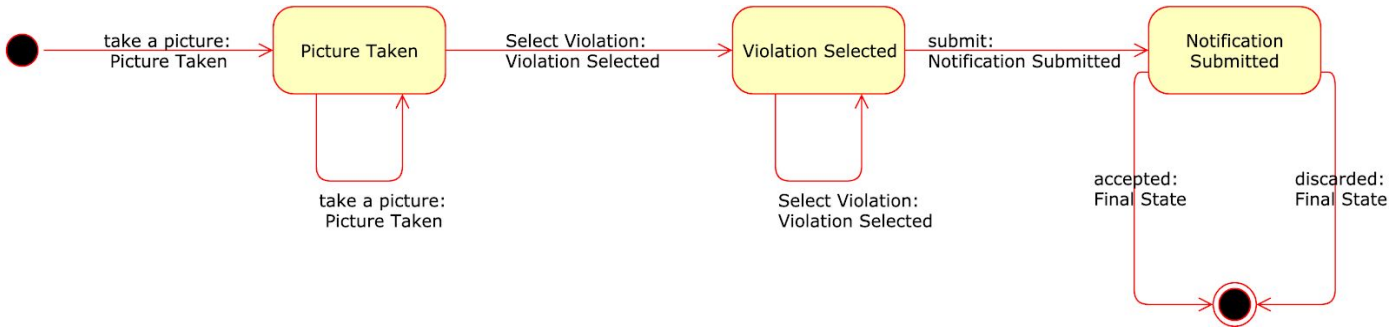

Figure 2 - State Diagram 1: Notify a violation

In this state diagram is modelled the process that describes how a user notify a violation, he must take one picture, that can be retaken, and then select one or more violation before submitting the notification, which can be accepted if a license plate is distinguishable in the image or discarded otherwise.

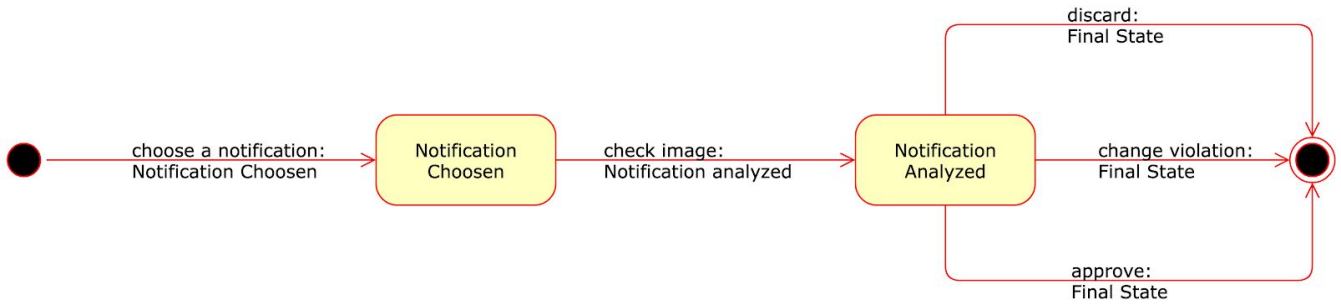

Figure 3 - State Diagram 2: Analyze a notification

Here is modelled the procedure that an operator of SafeStreets must follow for analysing a Notification. He initially chooses a notification between the ones sent by the users, by default the system provides him the oldest one, after that the operator has to analyze the image and verify if it may represent a violation, in this case he approves it, otherwise he discards it if the image is completely wrong (for instance a license plate is found but the picture was taken inside an house) or change the type(s) of violation if it recognizes that the one(s) indicated by the user does not match the type(s) actually committed.

2.2 Product Functions

In the following section the most important functions of the system are reported. It's important to make clear that the system is built as a unique application and not as different services interacting with each other.

2.2.1 Notification Management

This is one of the two most important product functions for the system. First, the system will allow the user to sign up entering an email, password and his own fiscal code, after that it will allow the user to send a notification in this way: the user must take one or more pictures that show the violation (it may be more than one) and then he has to select the violation(s) he wants to point out from a list provided by the system. Some metadata are added to the notification, for instance the name of the street where the violation occurred, which can be retrieved from the geographical position of the user at the moment of sending the violation, next the notification is sent to the server which runs an algorithm on it for identifying the license plate. If this operation is successful the notification is stored with all his own data, otherwise it is discarded by the system that also notify the user that is segnalation has been discarded.

2.2.2 Consultation Management

This is the second important function for the system and it permit to all customers to access SafeStreet's data, obviously at different customer are allowed a different level of access.
Users can only access to statistics about streets, for instance they can ask to the system which are the streets with the highest number of violations in their city. Authorities, on the other hand, are guaranteed a high level of visibility and the possibility to access the stores data with a finer granularity. This function is exploited by authorities if they accept to take SafeStreets as a trusted source of information, in this case they can use data stored in SafeStreets to automatically generate traffic tickets, obviously after they have verified that the violation is real.

2.2.3 Crossing Data Management

This important function is thought for being used from SafeStreets' operators they can check if a municipality offers a service that allows them to retrieve the information about the accidents that occur on that specific territory, and if this is true they can cross this data with the SafeStreets' and identify potentially unsafe areas. After that the operators can search if there is some possible common cause in violations happening in the same street and if these are spotted they can think of a solution and suggest it to the authority.

2.2.4 Analysis Management

This function allows SafeStreets' operators to build statistics starting from the information about tickets that are issued thanks to users' notifications this can help to retrieve more information about most egregious offenders or moreover if after a suggestion a street is actually more safe. This function is important for the ideators and managers of SafeStreets' that can ask their operator to perform these statistics so that they can evaluate the effectiveness of the SafeStreets initiative.

2.3 User Characteristics

The actors of the application are the following:

- Customer: a person using SafeStreets without being registered. The only thing a customer can do is proceed with the registration.
- User: a person who is registered to SafeStreets. The user can login to the system and, after that, use the platform's functionalities. For instance, he can send notifications about executed violations. To send a notification, he can take one or more pictures, but instantly within the application, and must have the GPS active on his device. The user can be blocked by the system if he makes three or more notifications in which there is no violation. A blocked user can no longer notify traffic violations. A user can also access some of the data stored by SafeStreets, for example he can search for the streets of a city classified as dangerous.
- Authority: is represented by a single operator who, after registering, can login to the system. An authority operator has free access to the information collected by SafeStreets. The main purpose is to collect the violations reported to SafeStreets by users, analyze them and generate the corresponding traffic tickets.
- SafeStreets operator: a SafeStreets employee who is responsible for analyzing and classifying reports made by users. He also has the opportunity to suggest to the municipality actions to optimize traffic and reduce accidents. It can also cross-reference stored data with those made available by municipalities in order to create statistics.

2.4 Assumptions, Dependencies and Constraints

2.4.1 Domain Assumptions

- D1: The information coming from the users is never broken
- D2: Each email is unique
- D3: Each fiscal code is unique
- D4: The internet connection works properly without failure on user's device
- D5: GPS works correctly on the user's device when taking the photo
- D6: The date and time on the user's device are always updated and working properly
- D7: Each license plate is unique
- D8: Each vehicle has one owner
- D9: The traffic tickets are consistent with the violations committed
- D10: The data provided by the municipalities are always correct
- D11: Municipalities are willing to receive suggestions from the system
- D12: Authorities are confirmed by the email used at the time of registration

2.4.2 Text Assumptions

- Credentials that a customer has to provide to become a registered user are: name, surname, email, fiscal code, date of birth, place of birth, and optionally his city.
- An operator of the authorities must provide his working email for the registration.
- In order to access the system, a user has to provide the email and password associated to it.
- The different levels of visibility were considered as different levels of granularity for access to the saved data. Authorities can access data with a finer granularity level by having access to all saved data, while users can only access some data (they can search for roads classified as dangerous and can access the history of their reports).
- It can happen that a vehicle commits more than one violation at the same time (for example, it is located both in front of the driveway and in the parking lot reserved for the disabled). So at the time of the report, the user can select more than one option from the list of possible types of violations that is provided by the system.
- The integrity of the pictures sent by the users is made possible by the fact that the user can take instant pictures at the time of the report, directly from the application. By doing

this, SafeStreets is guaranteed that the photos sent by users have not been altered and that the license plates involved are not manipulated.

2.4.3 Constraints

One of the most important constraints imposed by the system is that users can take photos to be reported only instantly and within the application. This avoids the possibility of receiving pictures manipulated by users. Another necessary constraint is the location, date and time updated and working on the user's device at the time of reporting. The Internet connection must also work properly on the device used by users.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Below we have reported the mockups that give an approximate idea of how the application interfaces should appear. Some of the most important screenshots of the interactions between the system and users are represented.

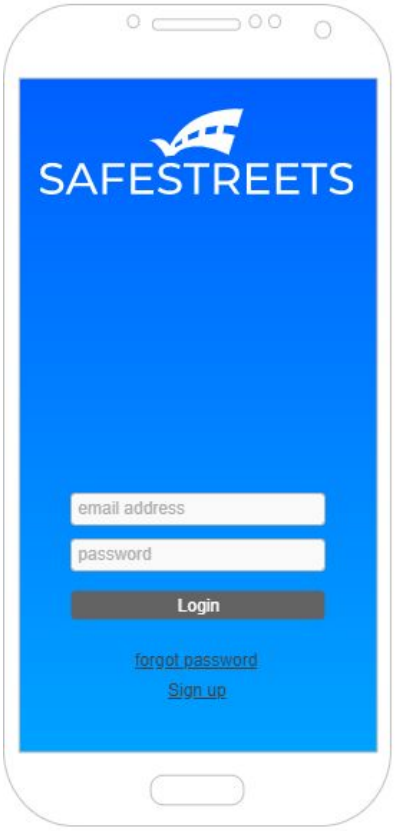

Figure 4 - Mockup: Login

Figure 5 - Mockup: Registration

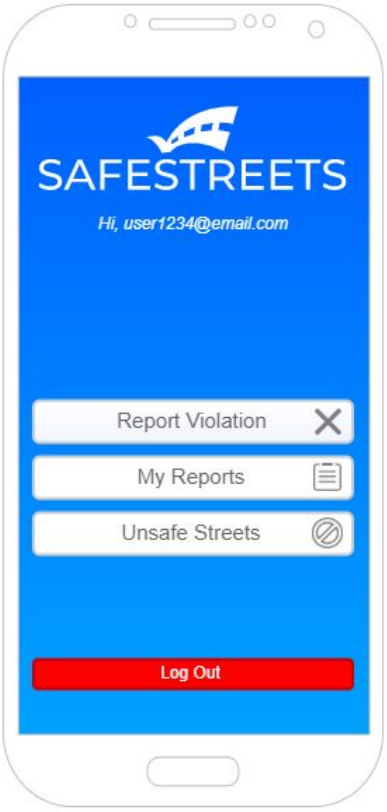

Figure 6 - Mockup: Main Menu

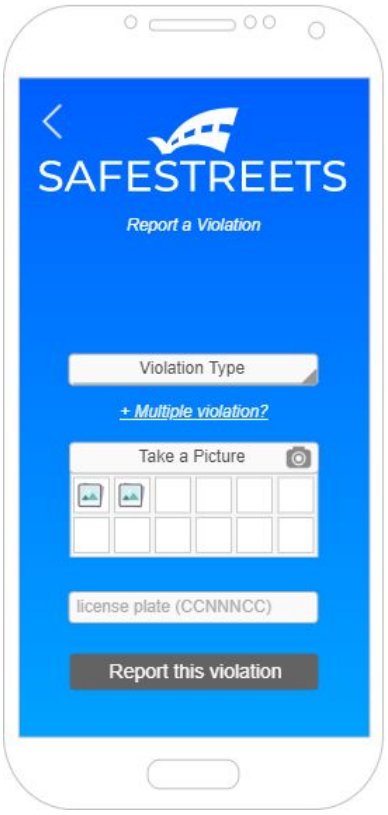

Figure 7 - Mockup: Report a Violation

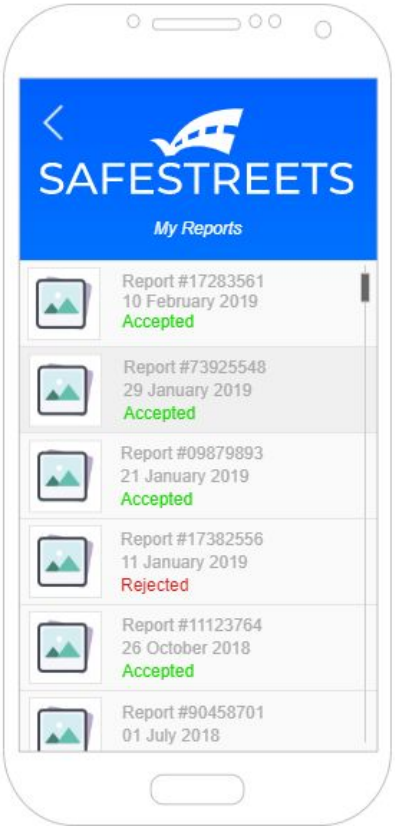

Figure 8 - Mockup: list of violations notified by the user

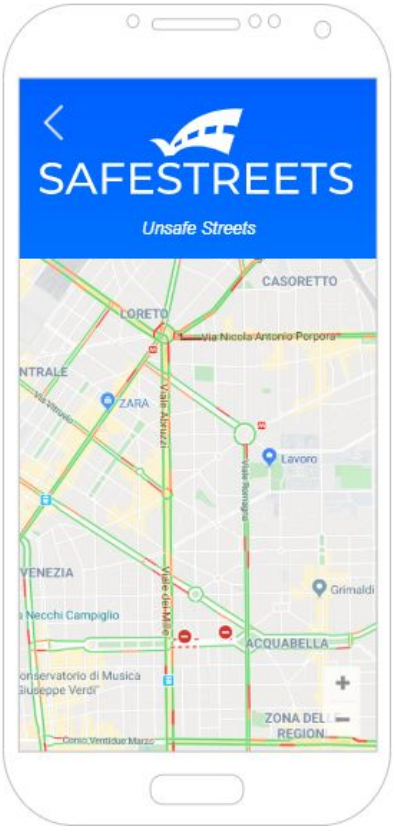

Figure 9 - Mockup: map of the most dangerous streets of the selected city

Below we report two screenshots of the interactions between the system and the authorities. The first image shows the list of violations that SafeStreets has collected and that still need to be analyzed by the authorities. The second picture shows the advanced search function that can be performed by a registered operator of the authorities. The fields to filter the data are city, type of violation, date and license plate.

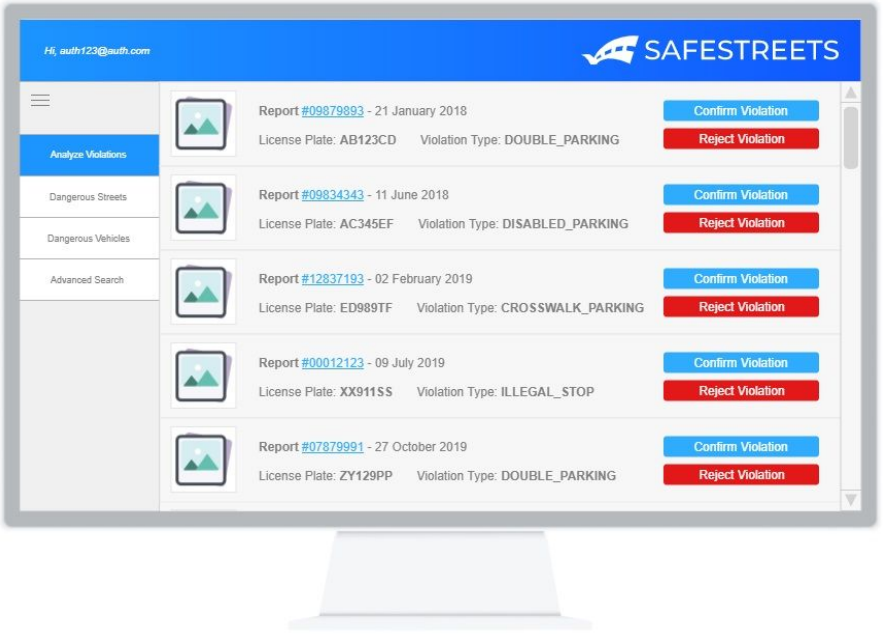

Figure 10 - Mockup: list of violations to be analysed by the authorities

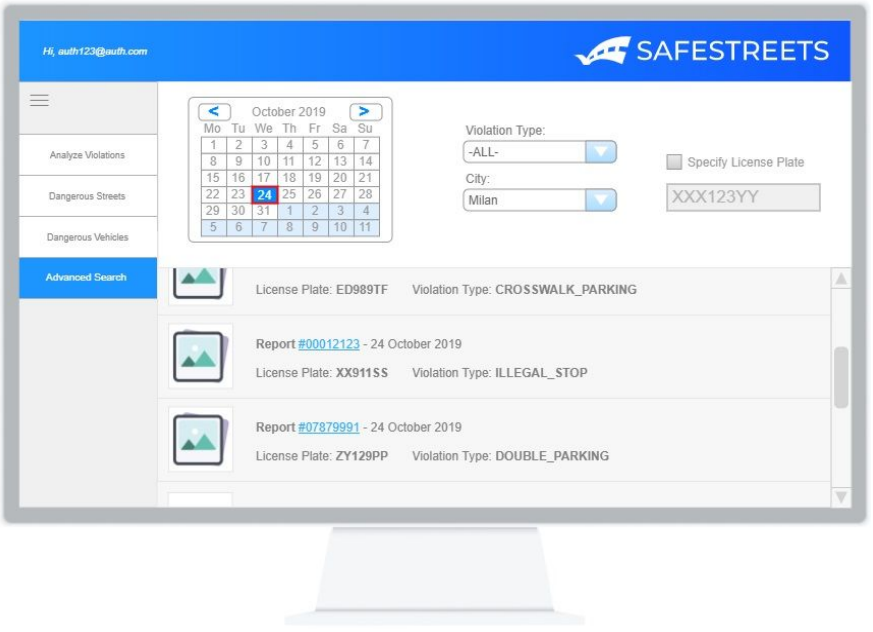

Figure 11 - Mockup: Advanced search function for authorities

3.1.2 Hardware Interfaces

The system has no hardware interface. However, it requires that the user has a smartphone that can use GPS services and mobile internet, and the authorities to have a desktop computer.

3.1.3 Software Interfaces

The system doesn't provide any API to external applications.

3.1.4 Communication Interfaces

Both Users' network-connected app and Authorities client use HTTP protocol to send and receive data. The platforms include the HttpsURLConnection client, which supports TLS, streaming uploads and downloads and configurable timeouts.

3.2 Functional Requirements

3.2.1 Scenarios

Scenario 1

Maurizio sees a car parked in front of the driveway of his apartment building. Thanks to the registration made just the day before, he decides to take a picture in which the license plate of the car and the driveway are clearly visible and sends it to SafeStreets. SafeStreets forwards the information to the local police who generates the traffic ticket and sends it to the owner of the car involved in the reported violation.

Scenario 2

SafeStreets after a year of activity decides to verify the effective working of the initiative and the interest shown by users in the service. The operator Stefano is then in charge of analyzing the stored data and generate statistics that will be studied by his bosses.

Scenario 3

Antonio to have fun with his friends sends reports where violations are non-existent. The game goes on but at the third wrong report, Antonio is blocked by the system and can no longer make reports. He can only exploit the other features, i.e. search for the most dangerous roads and see the history of his reports.

Scenario 4

Anna, a SafeStreets operator, analyzing the data, realizes that in Milan in Viale Molise many machines end up off the road, causing a lot of damage and injury. Anna then suggests to the city of Milan to add a barrier to reduce the number of accidents. Fortunately, the municipality acts immediately, listening to the advice received.

Scenario 5

Alice notes that her neighbour, since he could not find a place to park, occupies the place reserved for the disabled. She then sends a picture to SafeStreets, which is analysed. However, the notification is refused because it is not possible to identify the plate, being partly hidden by a tree. Alice is then notified of the rejection of the violation and the reason for it. Fortunately, when Alice receives the feedback, her neighbor's car is in the same place. She then decides to send SafeStreets a new picture, being careful to get the license plate of the car right.

Scenario 6

Maria wants to help her son Giacomo get his license. Since her Giacomo is still inexperienced, she wants to take him to a safe area of the city to teach him the basics and to familiarize him with the car. For this reason, Maria opens the SafeStreets app and searches for the most dangerous areas of the city so as to avoid them and to have no problems.

3.2.2 Use Case Diagrams

User Use Cases

Figure 12 - Use Case Diagram: User

Name	**Sign Up**
Actor	User
Entry conditions	The user has opened the SafeStreets' application on his device
Events Flow	1. The user selects "Sign up" between the available options 2. The user inserts his email, password, anagraphic data and fiscal code which are mandatory fields 3. The user selects "Sign up" 4. The system saves the entered data
Exit Conditions	The user is registered to the system and his data are stored
Exceptions	1. There is already an account with the entered fiscal code, that means this user has already an account. In this case the registration is aborted 2. There is already an account associated with the entered email 3. The user choose "Abort" and it's redirected to his home page 4. One or more field is empty. The system warns the user about it

Name	**Login**
Actor	User
Entry conditions	The user is signed and his information are stored
Events Flow	1. The user opens the application 2. The user fills the fields with his email address and password 3. The user presses "Login"
Exit Conditions	The user is successfully logged and is now able to visualize all the possible options
Exceptions	1. The user enters wrong information, in this case it is forced to repeat the

	procedure if he wants to log in.

Name	**Report a violation**
Actor	User
Entry conditions	The user must have successfully logged in
Events Flow	1. The user selects the “Report Violation” option 2. The user takes a picture (or more than one) with his device 3. The user selects all the violation types between the available ones 4. The User click the “Send” button
Exit Conditions	The notification is sent to the system
Exceptions	1. The system can’t read the license plate in the image so the notification is discarded.

Name	**Access his own record**
Actor	User
Entry conditions	The user must have successfully logged in
Events Flow	1. The user selects the “My Reports” option 2. The system provides a list of the reports made by the user with their result (if present) that can be “correct” or ”wrong”.
Exit Conditions	The user can see a list of his past notification
Exceptions	The user has not made a notification yet, in this case the systems alert him about this

Name	**Find dangerous streets**
Actor	User
Entry conditions	The user must have successfully logged in
Events Flow	1. The user selects the “Unsafe Streets” option 2. The user selects a city as a filter and press “Find” 3. The system provides a map of the city highlighting the dangerous streets
Exit Conditions	The user can see on the map the selected city and its streets with most violations
Exceptions	The requested city doesn’t exist or there are not enough data about it. In this case the system ask to insert another city

SafeStreets operator Use Cases

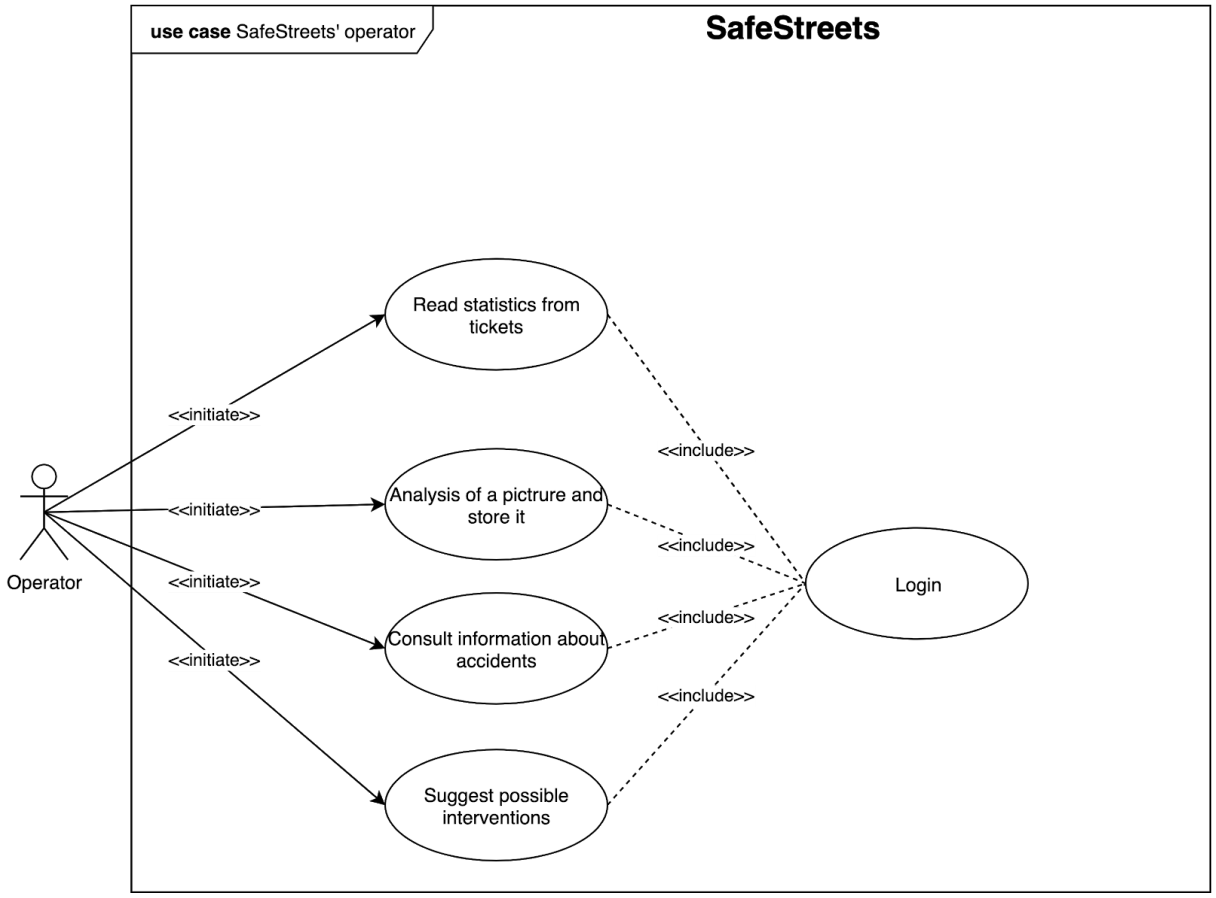

Figure 13 - Use Case Diagram: SafeStreets operator

Name	**Read statistics from tickets**
Actor	SafeStreets' operator
Entry conditions	The operator is logged in as an internal
Events Flow	1. The operator select the "statistics" option 2. The operator fill the fields for filtering data (for instance the evolution of a specific violation in the last month in a city) 3. The system provides a graph or a textual result depending on the request's type
Exit Conditions	The operator can see the result of his search
Exceptions	1. There are no data that match the request, in this case the system notify the operator

Name	**Analysis of a notification and storage of it**
Actor	SafeStreets' Operator
Entry conditions	The operator is logged in as an internal
Events Flow	1. The operator select the "Analize notification" option 2. The system provides him the oldest notification that has not been analysed 3. The operator checks if the image is actually describing a possible violation and presses "Confirm" 4. The system stores the notification
Exit Conditions	The notification is successfully stored
Exceptions	1. The image is inappropriate, in this case the operator discards it 2. The reported violation is not correct but there is a valid violation, in this case the operator changes the violation and press "Confirm"

Name	**Cross information about accidents**
Actor	SafeStreets' operator
Entry conditions	1. The operator is logged in as an internal 2. The Operator has retrieved the information from the municipality
Events Flow	1. The operator modifies the information's syntax to make them consistent with the ones stored by SafeStreets 2. The operator stores the information
Exit Conditions	The SafeStreets' database contains also the information given by the municipality

Exceptions	The information can't be modified in a way that is coherent with the database. In this case the operator stop the operation

Name	**Suggest possible interventions**
Actor	SafeStreets' operator
Entry conditions	1. The operator is logged in as an internal
Events Flow	1. The operator looks for the street with most violation 2. The operator filters the most recurrent violation 3. The operator checks a correlation between the images and a possible solution 4. The operator send his suggestion to the municipality
Exit Conditions	The municipality has obtained a suggestion on how to avoid a violation
Exceptions	The operator can't find a relationship between the violation, in this case he doesn't send anything and the notification are labeled so that they will not be considered again

Authority operator Use Cases

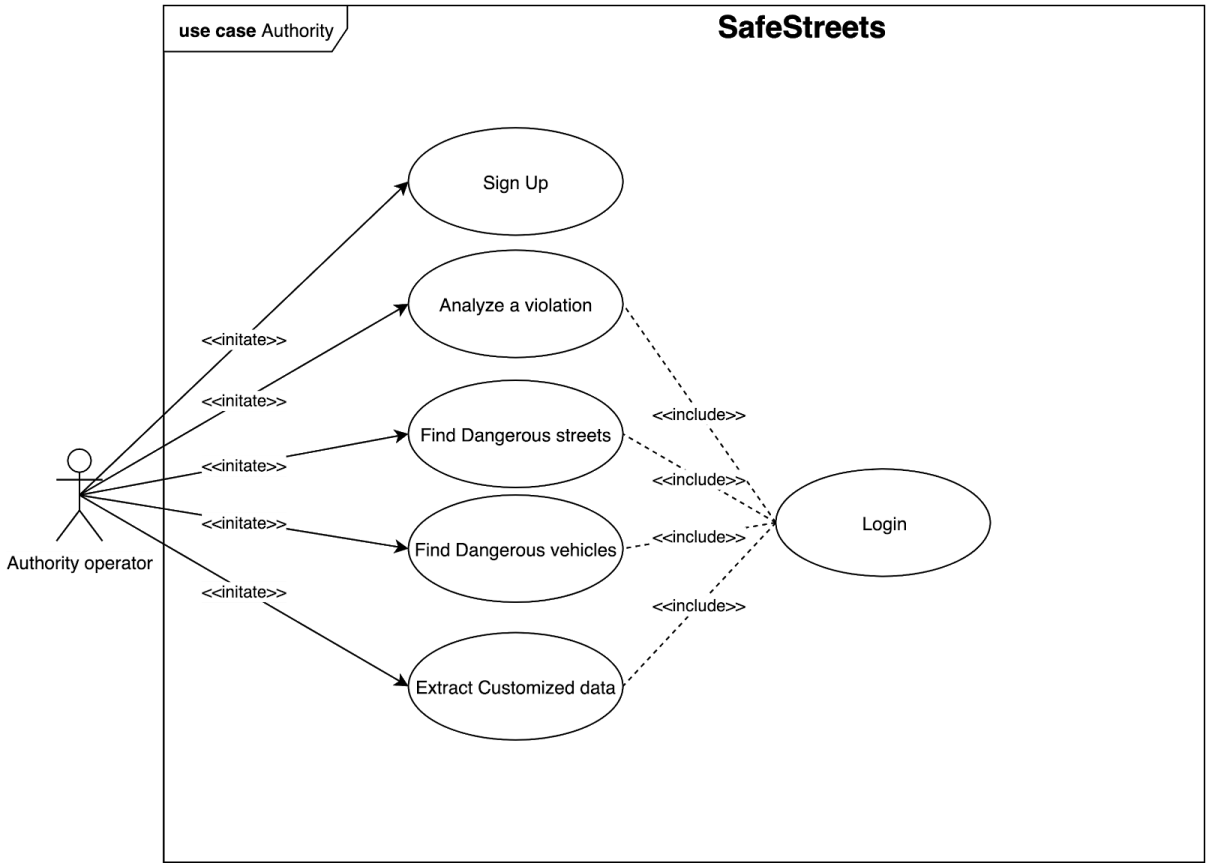

Figure 14 - Use Case Diagram: Authority operator

Name	**Sign up**
Actor	Authority operator
Entry conditions	The operator has opened the SafeStreets' application
Events Flow	1. The operator select the "Sign up" option 2. The operator inserts his work mail and his password 3. The operator clicks the "Sign up" button 4. The system recognise the mail as an authority mail and creates the operator's account
Exit Conditions	The operator has now his account
Exceptions	1. There is already an account associated with the entered email 2. The operator chooses "Abort" and it's

	redirected to his home page 3. One or more field is empty. The system warns the operator about it

Name	**Login**
Actor	Authority operator
Entry conditions	The operator is signed up and has opened the application
Events Flow	1. The operator fills the fields with his email address and password 2. The user presses “Login” 3. The system recognises the mail as an authority mail and redirects the operator to the proper page
Exit Conditions	The operator is successfully logged and is now able to visualize all the possible options
Exceptions	1. The operator enters wrong information, in this case it is forced to repeat the procedure if he wants to log in.

Name	**Analyze a violation**
Actor	Authority operator
Entry conditions	The operator has successfully logged in as authority member
Events Flow	1. The operator select the “Analyze violations” option 2. The system provides the notification that was stored the longest ago 3. The operator check if the image is consistent with the submitted violation 4. The operator generates the traffic ticket and makes it official
Exit Conditions	A traffic ticket against the vehicle is

	generated
Exceptions	1. The image is not pertinent, in this case the operator discards it and notify the system about this. 2. system provides a list of the 10 most dangerous streets and highlights it on a map

Name	**Find dangerous streets**
Actor	Authority operator
Entry conditions	The operator has successfully logged in as authority member
Events Flow	1. The operator selects the “Unsafe Streets” option 2. The operator fills all the fields he need to filter the result 3. The system provides a list of the 10 most dangerous streets and highlights it on a map
Exit Conditions	The operator can see the result of his query
Exceptions	1. There is no street matching the filters, in this case a warning is reported to the operator

Name	**Find dangerous vehicle**
Actor	Authority operator
Entry conditions	The operator has successfully logged in as authority member
Events Flow	1. The operator selects the “Find dangerous vehicle” option

	2. The operator fills all the fields he need to filter the result (for instance the city) 3. The system provides a list of the license plates of the 10 most dangerous vehicles
Exit Conditions	The operator can see the result of his query
Exceptions	1. There is no street matching the filters, in this case a warning is reported to the operator

Name	**Extract customized data**
Actor	Authority operator
Entry conditions	The operator has successfully logged in as authority member
Events Flow	1. The operator selects the “Extract customized data” option 2. The operator select from a given list what he want to extract (for instance a set of notification, of streets or of drivers) 3. The operator select all the fields he needs to filter data 4. The system provides a list of all the records matching the inputs
Exit Conditions	The operator can see the result of his query
Exceptions	1. There is no matching record, in this case a warning is reported to the operator

3.2.3 Sequence Diagrams

Visitor Sign Up

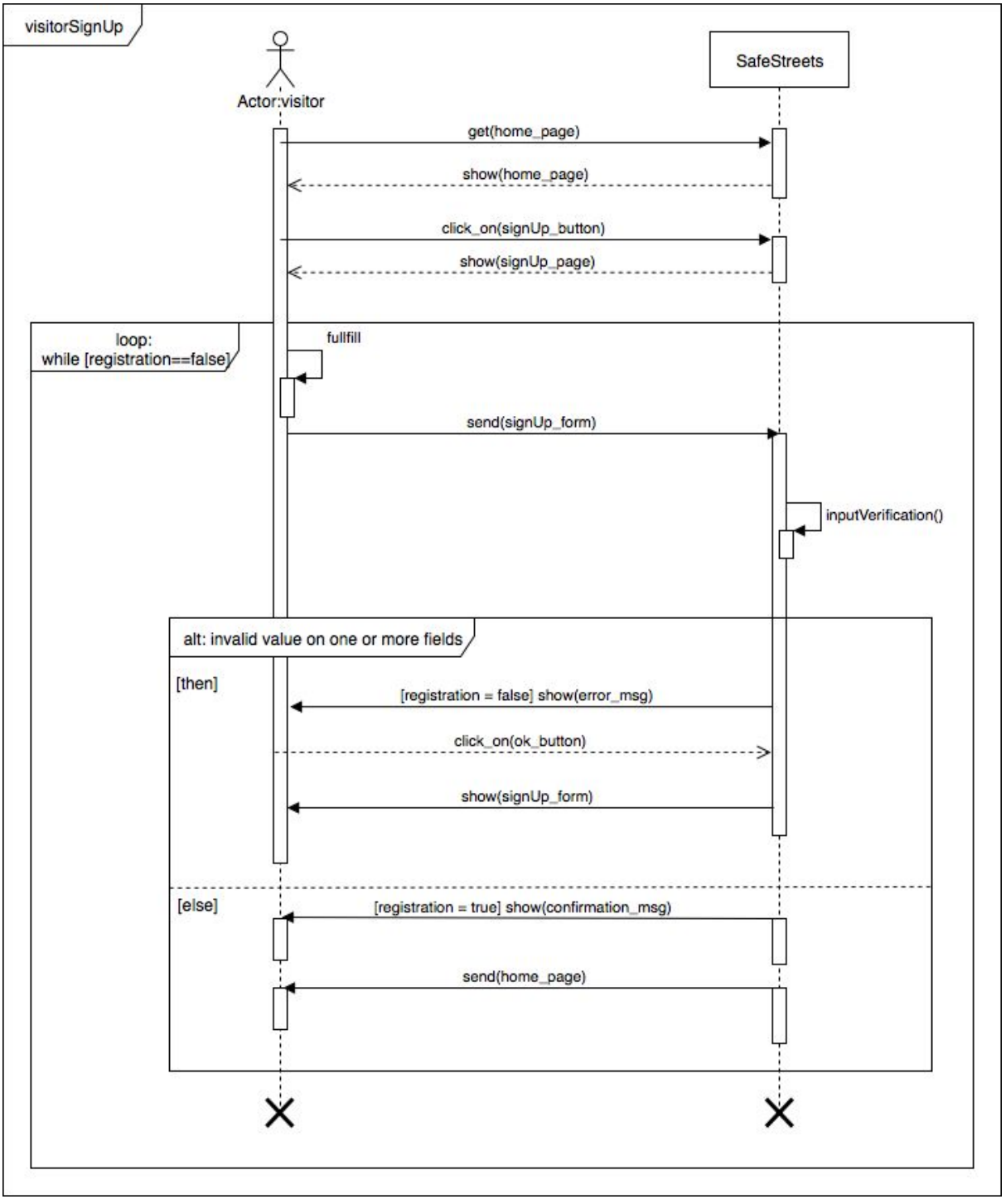

Figure 15 - Sequence Diagram: Visitor Sign Up

The sequence diagram for the sign up of an authority is the same as the one represented above, so it's omitted.

User Login

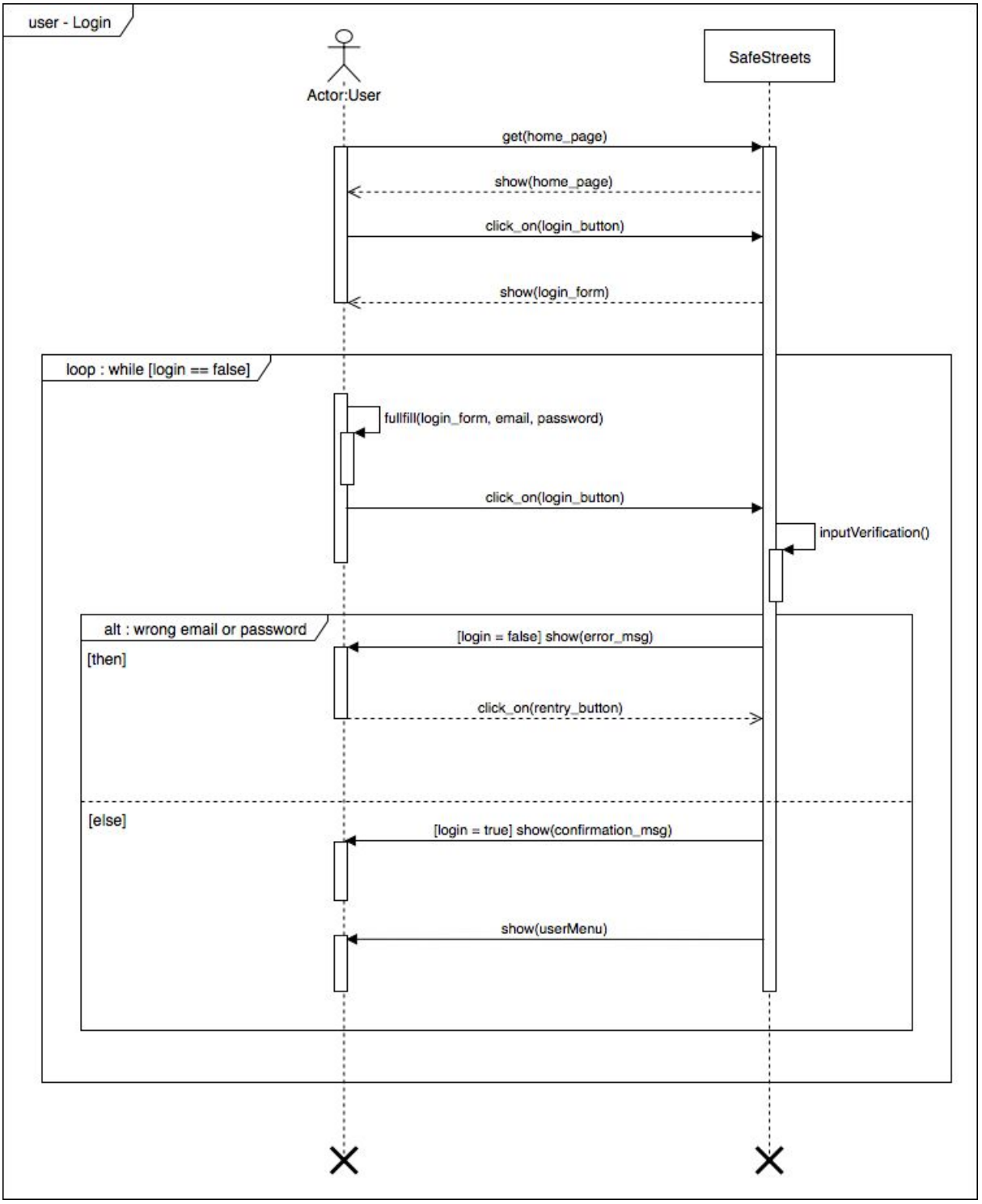

Figure 16 - Sequence Diagram : User Login

The sequence diagram for the login of an authority is the same as the one represented above, so it's omitted.

User reports a violation

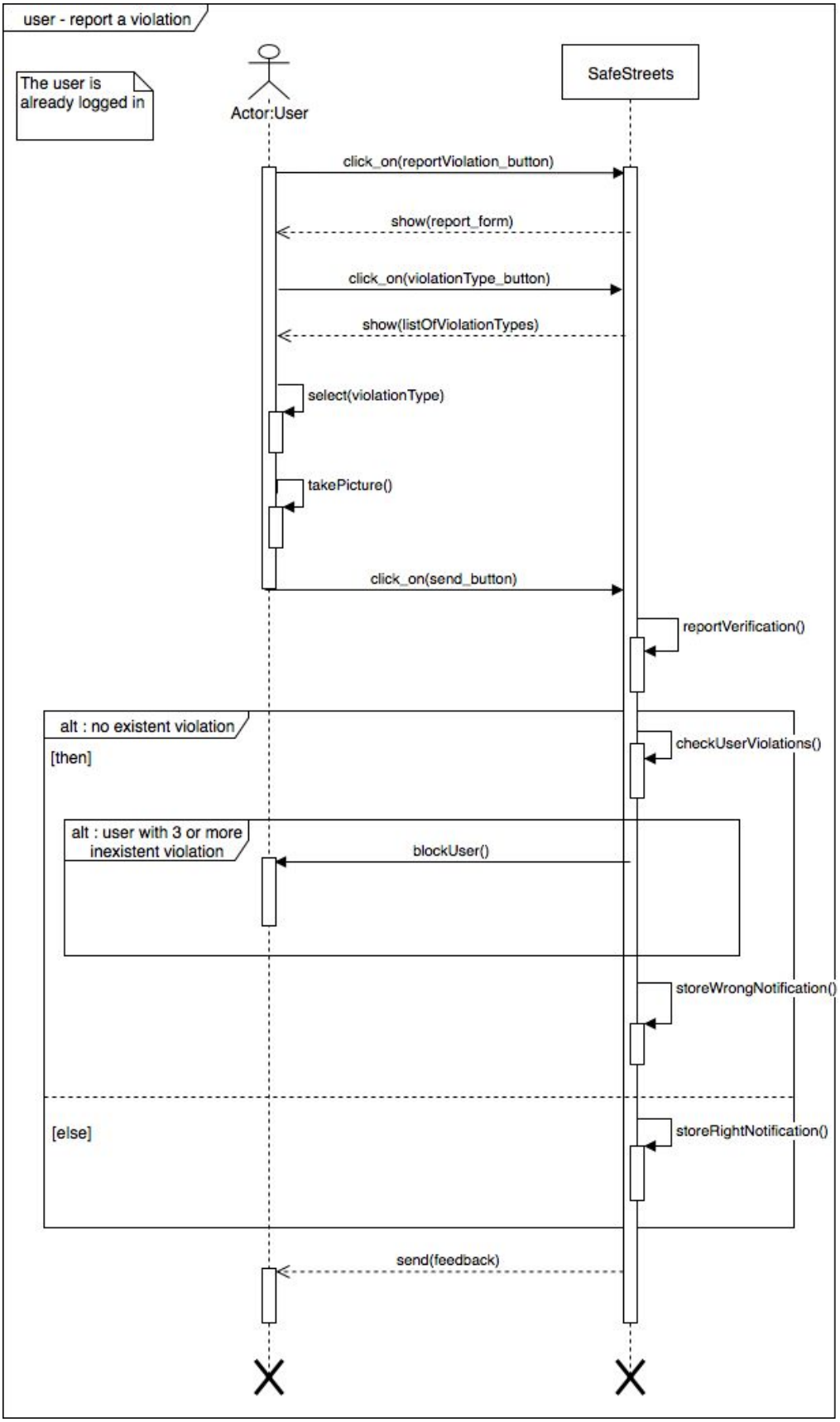

Figure 17 - Sequence Diagram : User reports a violation

User searches for dangerous streets

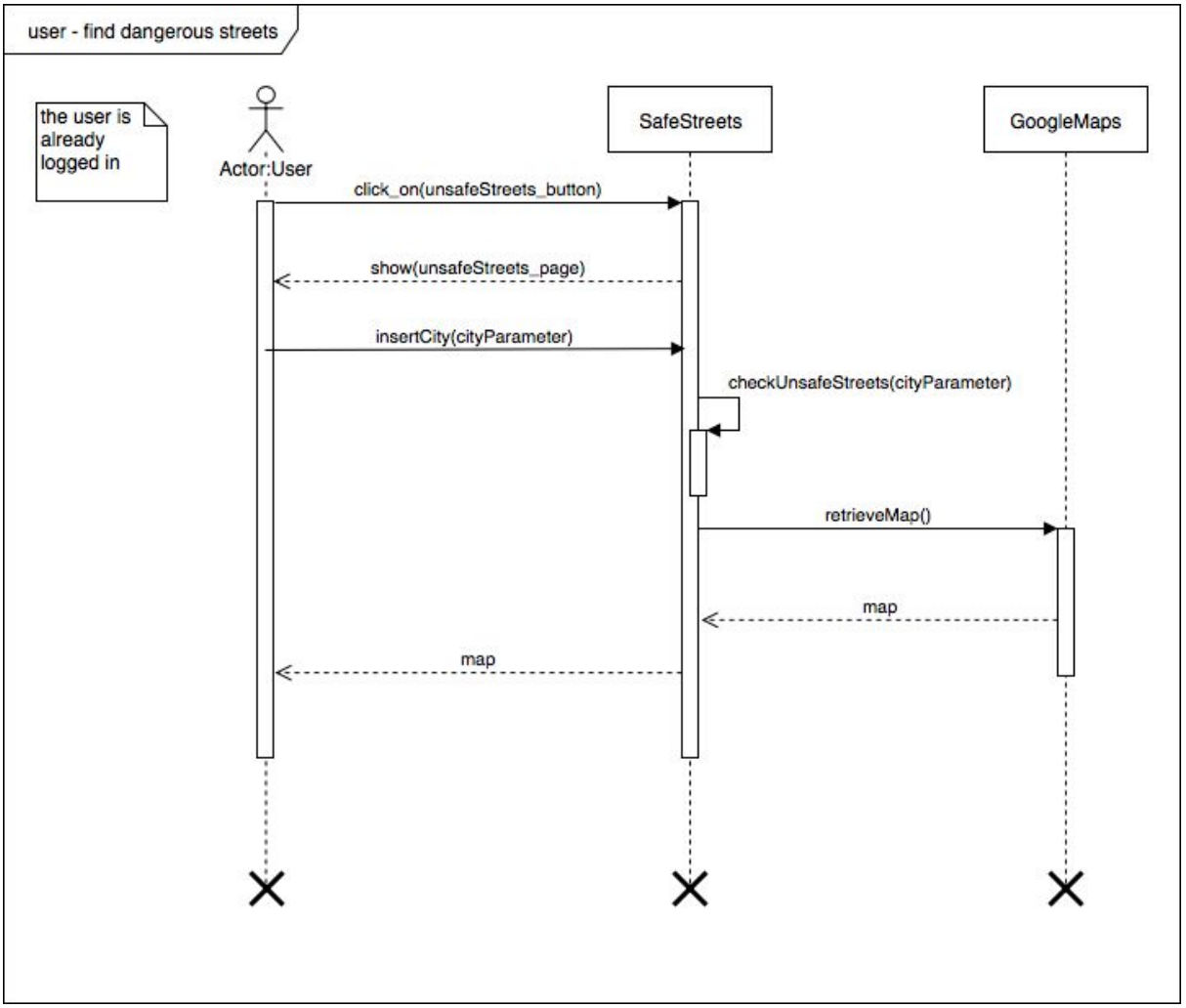

Figure 18 - Sequence Diagram: User searches for dangerous streets

The sequence diagram for the search of dangerous streets by an operator of the authorities is the same as the one represented above. It's very similar also the sequence diagram for the search of dangerous vehicles by an operator of the authorities. So these diagrams are omitted.

Authority analyze a violation

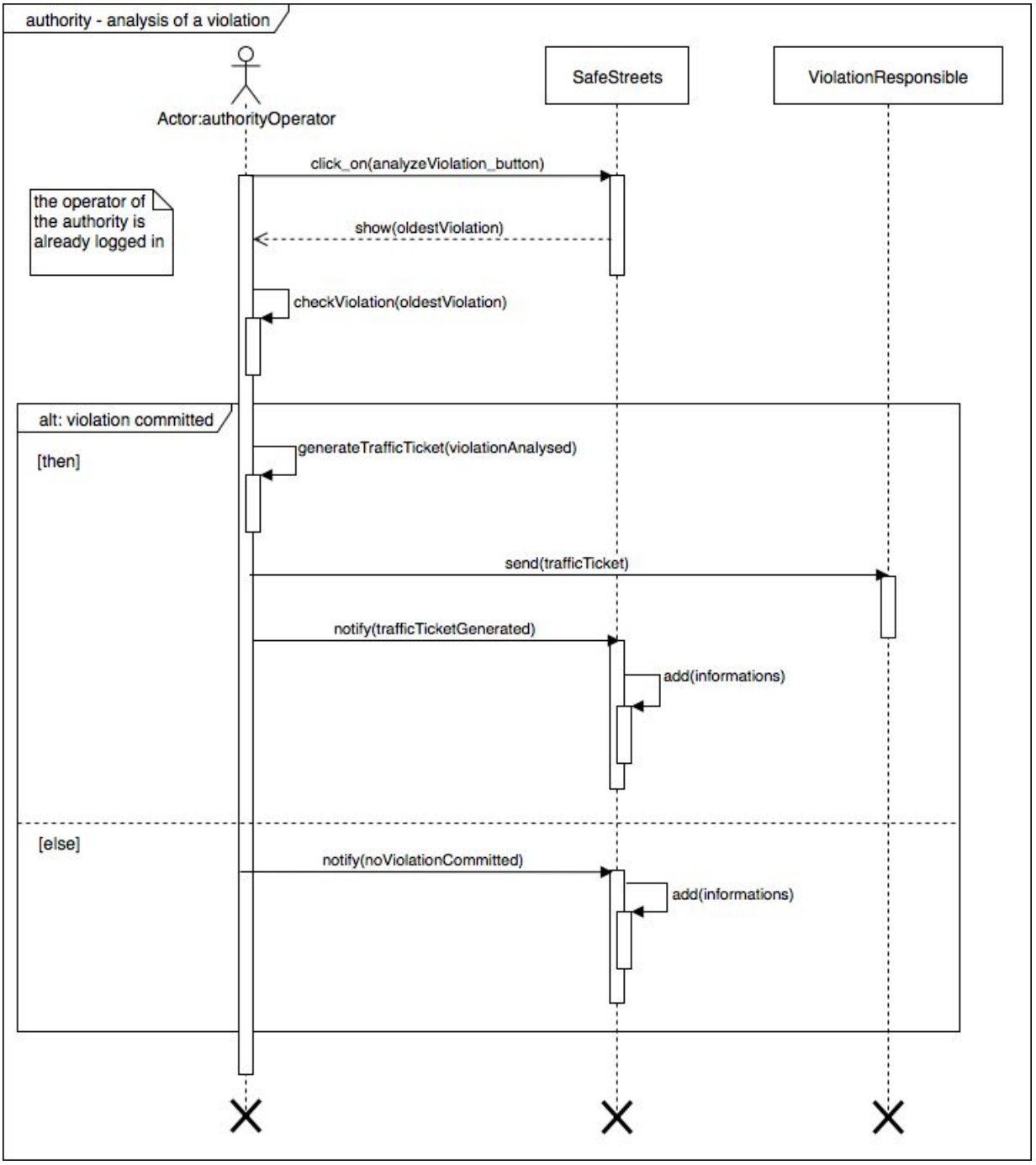

Figure 19 - Sequence Diagram: an authority operator analyzes a violation saved by SafeStreets

The message "notify(trafficTicketGenerated)" is used to notify SafeStreets when a violation analysed by the authorities has been confirmed and the corresponding fine has been generated. The message "notify(noViolationCommitted)" is used to notify SafeStreets when a violation analysed by the authorities has been rejected and therefore no fine has been generated. In both cases, a SafeStreets operator updates the corresponding data.

A SafeStreets operator suggest to municipality possible interventions

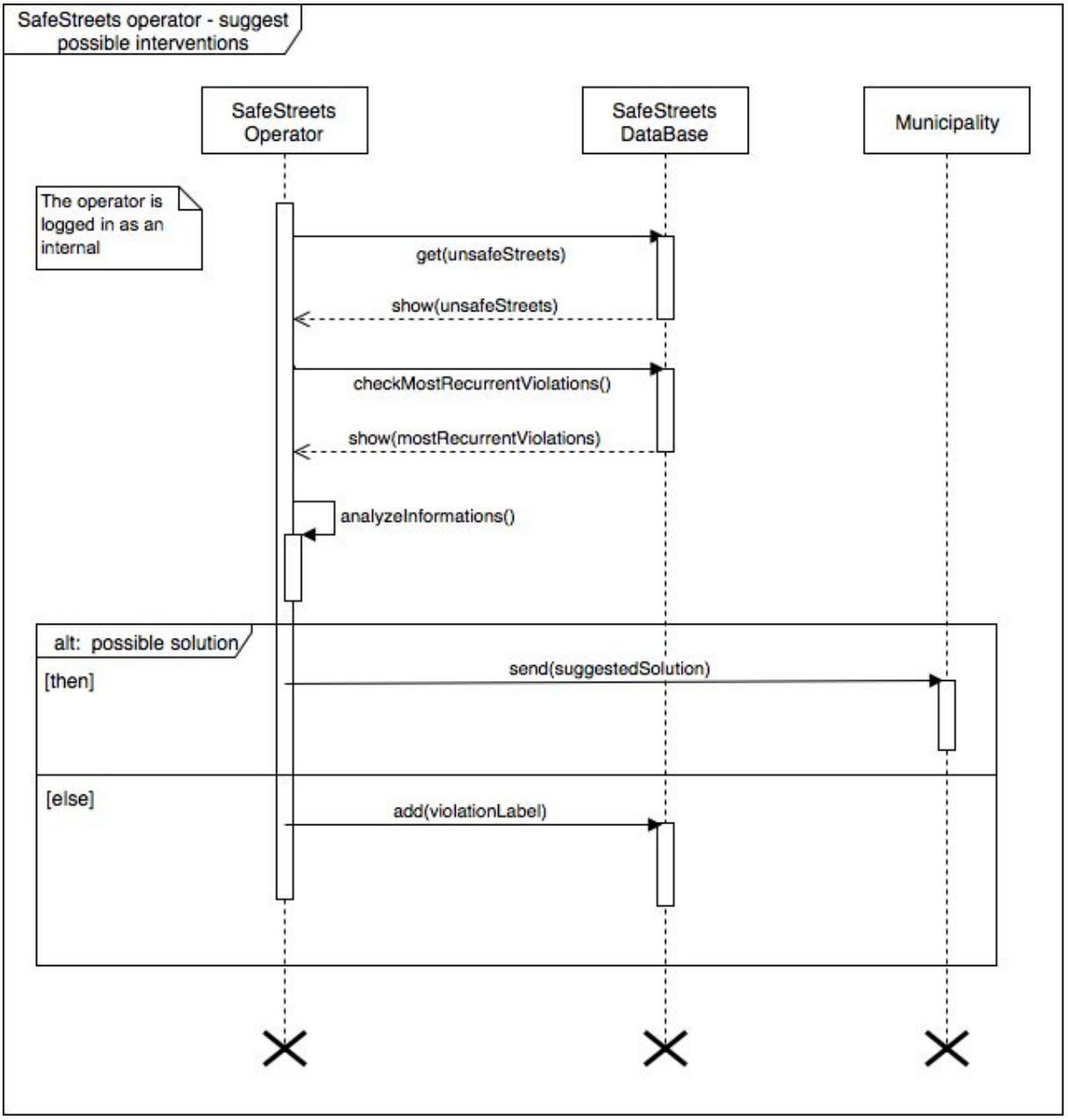

Figure 20 - Sequence Diagram: a SafeStreets operator suggest possible interventions to the municipality

If the operator can't find a relationship between the violations, he doesn't send anything and the notification are labeled so that they will not be considered again.

3.2.4 Requirements

G1: Allow users to log in the system

- R1: The system should be available 24/7
- R2: The system offers the possibility to register, with the insertion of e-mail, password and fiscal code
- R3: The email used must be different for each account
- D2: Each email is unique
- D3: Each fiscal code is unique

G2: Allow users to notify a traffic violation

- R1: The system should be available 24/7
- R4: The system should allow registered users to exploit its features
- R5: The system should allow users to take a picture instantly and send it
- R6: The system offers a list of possible types of violations to report
- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- R9: An algorithm allows to recognize license plates involved in a notified violation
- R10: A user is blocked if he makes more than 3 incorrect reports (there is no traffic violation in the report)
- D1: The information coming from the users is never broken
- D4: The internet connection works properly without failure on user's device
- D5: GPS works correctly on the user's device when taking the photo
- D6: The date and time on the user's device are always updated and working properly

G3: Allow users to receive feedback on the notification made (it can be accepted or refused)

- R8: The system must verify the correctness of the notification made
- R9: An algorithm allows to recognize license plates involved in a notified violation

- R10: A user is blocked if he makes more than 3 incorrect reports (there is no traffic violation in the report)
- D4: The internet connection works properly without failure on user's device

G4: Allow users to see the history of their own notified violations

- R1: The system should be available 24/7
- R4: The system should allow registered users to exploit its features
- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- D1: The information coming from the users is never broken
- D4: The internet connection works properly without failure on user's device
- D5: GPS works correctly on the user's device when taking the photo
- D6: The date and time on the user's device are always updated and working properly

G5: Allow the possibility for the municipality to generate traffic tickets from the identified violations

- R6: The system offers a list of possible types of violations to report
- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- R9: An algorithm allows to recognize license plates involved in a notified violation
- R14: The system offers the possibility to the employees of the authorities to register by entering the working email and a password
- D1: The information coming from the users is never broken
- D3: Each fiscal code is unique
- D7: Each license plate is unique
- D8: Each vehicle has one owner
- D9: The traffic tickets are consistent with the violations committed
- D12: Authorities are confirmed by the email used at the time of registration

G6: Build statistics starting from the information about issued tickets

- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- R9: An algorithm allows to recognize license plates involved in a notified violation
- D1: The information coming from the users is never broken
- D7: Each license plate is unique
- D8: Each vehicle has one owner
- D9: The traffic tickets are consistent with the violations committed

G7: Allow authorities to access to the stored information

- R1: The system should be available 24/7
- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- R9: An algorithm allows to recognize license plates involved in a notified violation
- R11: The system has to ask the authorities which parameters to use to filter data
- R14: The system offers the possibility to the employees of the authorities to register by entering the working email and a password.
- D1: The information coming from the users is never broken
- D12: Authorities are confirmed by the email used at the time of registration

G8: Cross system's data with those made available by the municipalities

- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- R12: The system can access the data made available by the municipalities
- R13: The data to be integrated must be made homogeneous (data saved by SafeStreets and data coming from the municipality)
- D10: The data provided by the municipalities are always correct

G9: Allow the municipality to receive suggestions for possible interventions

- R7: The system must save the information about the notified violations

- R8: The system must verify the correctness of the notification made
- D1: The information coming from the users is never broken
- D11: Municipalities are willing to receive suggestions from the system.

G10: Allow users to access to certain stored data (such as the most dangerous roads in a city)

- R1: The system should be available 24/7
- R2: The system offers the possibility to register, with the insertion of e-mail, password and fiscal code
- R4: The system should allow registered users to exploit its features
- R7: The system must save the information about the notified violations
- R8: The system must verify the correctness of the notification made
- D1: The information coming from the users is never broken
- D4: The internet connection works properly without failure on user's device
- D10: The data provided by the municipalities are always correct

Traceability Table

In the following table we indicate for each use case the requirements related to it.

ACTOR	USE CASE	REQUIREMENTS
User	Sign up	R1, R2, R3
User	Login	R1, R4
User	Report a violation	R1, R4, R5, R6, R7, R8, R9, R10
User	Access his own record	R1, R4, R7, R8
User	Find dangerous streets	R1, R4, R7, R8
SafeStreets operator	Read statistics from tickets	R7, R8, R9
SafeStreets operator	Analysis of a notification and storage of it	R6, R7, R8, R9
SafeStreets operator	Cross information about accidents	R7, R8, R12, R13

SafeStreets operator	Suggest possible interventions	R7, R8
Authority operator	Sign up	R1, R3, R14
Authority operator	Login	R1, R4
Authority operator	Analyze a violation	R1, R7, R8, R9, R14
Authority operator	Find dangerous vehicles	R1, R7, R8, R9, R14
Authority operator	Find dangerous streets	R1, R7, R8, R11, R14

3.3 Performance Requirements

First of all, the system must ensure that reported violations are analysed quickly and correctly, so that the authorities can take action as soon as possible. It also must be able to manage authorities and several users at the same time, responding promptly to their requests and updating them on the outcome of the violations reported.

3.4 Design Constraints

3.4.1 Standards Compliance

The system uses sensitive data such as location respecting the laws of privacy. The system must in fact require users permission to obtain their location when they take photos to notify a traffic violation. In addition, the email used by customers at the time of registration will not be used for commercial purposes, respecting the law on the protection and privacy of citizens' data, the General Data Protection Regulation.

3.4.2 Hardware Limitations

For users every device with an internet connection is fine in order to sign up or log in but this is not enough to report a violation. The following are all the hardware requirements needed:

- Connection to Internet (Wi-Fi/4G/3G)
- GPS
- Camera on the used device

While for operators, both SafeStreets and authorities is enough an internet connected device for using all the possible functionalities, but a PC is recommended.

3.4.3 Any other constraints

The system must respect the privacy of the users, therefore it must prevent the divulgation of their data and everything that concerns them directly.
It is important to note that users cannot attach previously taken photos to the report, but can only take them instantly within the application at the time of reporting. In this way SafeStreets avoids the possibility of receiving pictures manipulated by users.

3.5 Software System Attributes

3.5.1 Reliability

The system must be able to run continuously without any interruptions, ensuring a 24/7 service. With a single server the desired reliability would not be achievable. For this reason the system must be replicated using multiple servers in parallel. Due to current technological limitations, the algorithm used for reading and recognizing license plates cannot be 100% accurate. For this reason, if the algorithm shows uncertainties about recognition, a SafeStreets operator will have to check the violation and the related photo.

3.5.2 Availability

As mentioned before, a fault tolerant architecture is needed. All of the three functionalities are expected to be available 99,9%, because the services themselves don't have critical aspects that involves for instance health of the costumer.

3.5.3 Security

Customers credentials have to be safely stored. Security of the data and of the communications customer-system is a primary concern. The central database, on which the data will reside, must be protected by all the necessary measures to avoid any external and internal attack. For this reason SafeStreets wants to rely on an industry leader to manage its servers. To send data, encryption technique must be used in order to guarantee privacy and confidentiality.

3.5.4 Maintainability

The development of the application must be done so that in the future, if modifies or fixes are needed, they could be done in a very easy way and with little impact on the full working components, also these operations must be as cheap as possible. Appropriate design patterns will be used, as it will be better explained in a further document.

3.5.5 Portability

The application's services provided to the users must be compatible with many portable devices as possible (smartphones and tablets) and technologies as possible. Instead, for the operators, both Authorities and SafeStreets, the services must be compatible with many PC's OS as possible.

4. Formal Analysis

This section includes the Alloy model of some critical aspects of the system. It is used to formally describe the application domain, its properties and its constraints. Everything is explained in detail by comments.
We focused on the aspect of user-reported notifications, their analysis and the storage of information, especially those of traffic violations.
Moreover, to show the soundness and correctness of the model, proofs of consistency and examples of the generated worlds are provided.

4.1 Alloy Model

4.1.1 Signatures

```
sig FiscalCode{}

sig Email{}

sig Password{}

abstract sig Customer{
	email: one Email,
	password: one Password
}

sig User extends Customer{
	fiscalCode: one FiscalCode,
	reportedNotifications: set Notification
}

sig Authority extends Customer{}

sig Municipality extends Authority{}

abstract sig UserStatus{
	user: set User
}

sig BlockedUser extends UserStatus{}

sig Position{
	latitude: one Int,
	longitude: one Int
}
{longitude >= -180 and longitude <= 180 and latitude >= -90 and latitude <=90}

sig Date{}

sig Time{}
```

```
-- nOfViolations indicates the number of violations committed by the vehicle with this license plate

sig LicensePlate{
	nOfViolations: one Int,
	fiscalCode: one FiscalCode
}

sig TypeViolation{}

-- The attribute typeViolation is a set as it can contain a variable number of elements
-- (a vehicle can be involved in more than one violation at the same time)

sig Notification{
	authorOfNotification: one User,
	typeViolation: set TypeViolation,
	licensePlate: lone LicensePlate,
	date: one Date,
	time: one Time,
	position: one Position
}

abstract sig NotificationResult{
	notification: set Notification
}


-- This signature includes the notifications that have been accepted,
-- i.e. those that report a confirmed violation.
-- When a notification is accepted, then there is a Violation

sig NotificationRight extends NotificationResult{}


-- This signature includes notifications that have been refused,
-- i.e. those that do not report a real violation

sig NotificationWrong extends NotificationResult{}

-- Here there is only one element as typeViolation
-- When a notification is accepted, then there is a violation

sig Violation{
	licensePlate: one LicensePlate,
	date: one Date,
	time: one Time,
	position: one Position,
	typeViolation: one TypeViolation
}


-- The attribute "authority" indicates the authority that generated the traffic ticket
-- When a new violation is made, then there is a Traffic Ticket

sig TrafficTicket{
	violation: one Violation,
	fiscalCode: one FiscalCode,
	authority : one Authority
}
```

```
-- The attribute "nOfViolations" indicates the number of violations committed on the corresponding street

sig Street{
	position: one Position,
	nOfViolations: one Int
}


-- This signature represents the roads that are considered dangerous
-- because of the number of accidents that occurred there

sig DangerousStreet{
	street: set Street,
}


-- This signature represents the vehicle that are considered dangerous
-- because of the number of accidents they have committed

sig DangerousVehicle{
	vehicle: set LicensePlate
}
```

4.1.2 Facts

```
--All Fiscal Codes have to be associated to a User
fact FiscalCodeUserAssociation{
	all fC: FiscalCode | some user: User | fC = user.fiscalCode
}


--All emails have to be associated to a Customer
fact EmailAssociation{
	all e: Email | some customer: Customer | e in customer.email
}


--All passwords have to be associated to a Customer
fact PasswordAssociation{
	all p: Password | some customer: Customer | p in customer.password
}


--Every Customer has a unique email
fact NoSameEmail{
	no disj c1, c2: Customer | c1.email = c2.email
}


-- Every User has a unique FiscalCode
fact NoSameFiscalCode{
	no disj u1, u2: User | u1.fiscalCode = u2.fiscalCode
}
```

```
-- Every notification made by a user (User.reportedNotifications) must have as author the same user
fact UserNotifications{
	all u: User, n: Notification | (n in u.reportedNotifications) implies (n.authorOfNotification = u)
}


--Every License Plate is unique
fact NoSameLicensePlate{
	no disj l1, l2: LicensePlate | l1 = l2
}


--All license plates have to be associated to a violation
fact LicensePlateAssociation{
	all lP: LicensePlate | one violation: Violation | lP in violation.licensePlate
}


-- All violations have to be associated to a traffic Ticket
fact ViolationAssociation{
	all v: Violation | one trafficTicket: TrafficTicket | v in trafficTicket.violation
}


-- A street with more than 50 violations is dangerous
fact NewDangerousStreet{
	all s: Street | (#s.nOfViolations >= 50 and s not in DangerousStreet.street)
	implies (s in DangerousStreet.street)
}

-- A vehicle with more than 15 violations is dangerous
fact NewDangerousStreet{
	all lP: LicensePlate | (#lP.nOfViolations >=15 and lP not in DangerousVehicle.vehicle)
	implies (lP in DangerousVehicle.vehicle)
}


-- A user who does 3 or more notifications is blocked. A blocked user won't be able to do other notifications
fact UserBlocked{
	all n: Notification, u: User | (n.authorOfNotification = u and (n in NotificationWrong.notification)
	and #n >= 3 and (u not in BlockedUser.user)) implies (u in BlockedUser.user)
}


-- If a notification is classified as incorrect, no violation is saved
fact NotificationWrongAssociation{
	all n: Notification | no v: Violation | (n in NotificationWrong.notification) implies
	(v.licensePlate = n.licensePlate and v.date = n.date and v.time = n.time and v.position = n.position
	and (v.typeViolation in n.typeViolation))
}


-- For each type of violation insered by the user in a notification exists a Violation
fact fact1{
	all n: Notification, x: n.typeViolation | one v: Violation | v.typeViolation = x
}
```

4.1.3 Predicates

```
--If a user is blocked, he has more or 3 wrong notifications
pred pred1 [u: User]{
	all n: Notification | ((n.authorOfNotification = u) and (n in NotificationWrong.notification) and
	(u in BlockedUser.user)) implies (#n >= 3)
}

-- If a street is dangerous, it means that there have been more than 50 accidents.
-- If a vehicle is dangerous, it means that it has committed more than 15 accidents.
pred pred2{
	all s: Street, lp: LicensePlate | ((s in DangerousStreet.street) implies s.nOfViolations >= 50) and
	((lp in DangerousVehicle.vehicle) implies lp.nOfViolations >= 15)
}

-- All right notifications have to be associated to a Violation
pred newViolation{
	all n: Notification | one v: Violation | (n in NotificationRight.notification) implies
	(v.licensePlate = n.licensePlate and v.date = n.date and v.time = n.time and v.position = n.position
	and (v.typeViolation in n.typeViolation))
}

-- All violations have to be associated to a TrafficTicket
pred newTrafficTicket{
	all v: Violation | one t: TrafficTicket | one a: Authority | (v.licensePlate.fiscalCode = t.fiscalCode)
	and (t.authority = a) and (t.violation = v)
}
```

4.1.4 Assertions

```
-- The number of notifications should be greater than or equal to the number of violations
assert assert1 {
	all n: Notification, v: Violation | #v <= #n
}

-- The number of notification validated should be equal to the number of violations
assert assert2 {
	all n: NotificationRight, v: Violation | #n = #v
}

-- The number of notifications accepted and the number of notificatiosn refused should be both
--less than or equal to the number of notifications
assert assert3 {
	all n1: NotificationRight, n2: NotificationWrong, n: Notification | #n1 <= #n and #n2 <= #n
}
```

4.2 Alloy Results

```
run pred1 for 10
run pred2 for 4
run newViolation for 5
run newTrafficTicket for 6
check assert1
check assert2
check assert3
```

```
7 commands were executed. The results are:
  #1: Instance found. pred1 is consistent.
  #2: Instance found. pred2 is consistent.
  #3: Instance found. newViolation is consistent.
  #4: Instance found. newTrafficTicket is consistent.
  #5: No counterexample found. assert1 may be valid.
  #6: No counterexample found. assert2 may be valid.
  #7: No counterexample found. assert3 may be valid.
```

4.3 Worlds generated

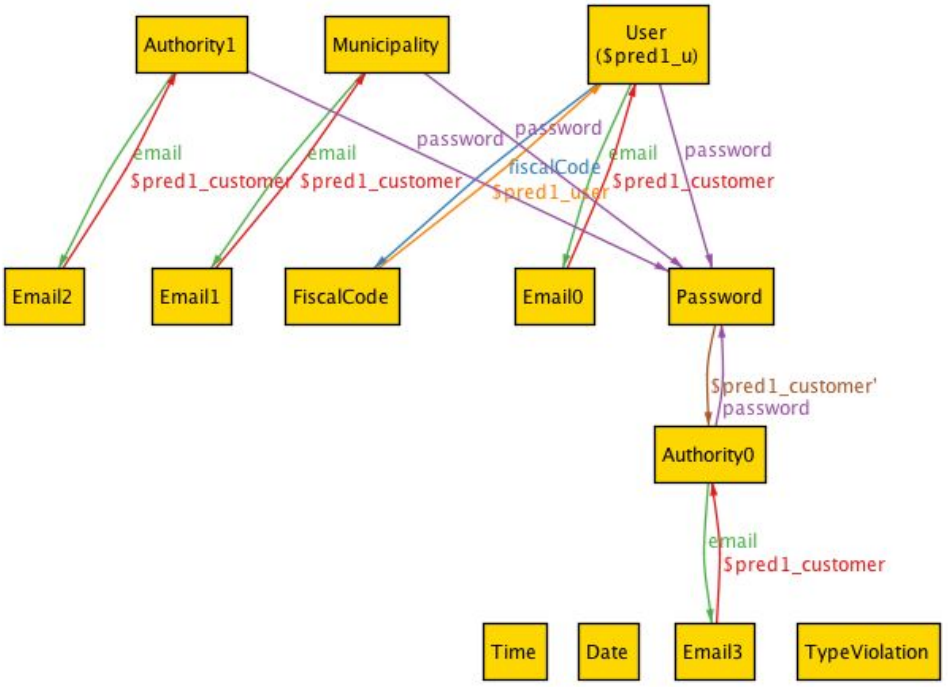

Figure 21 - Example 1 of world generated

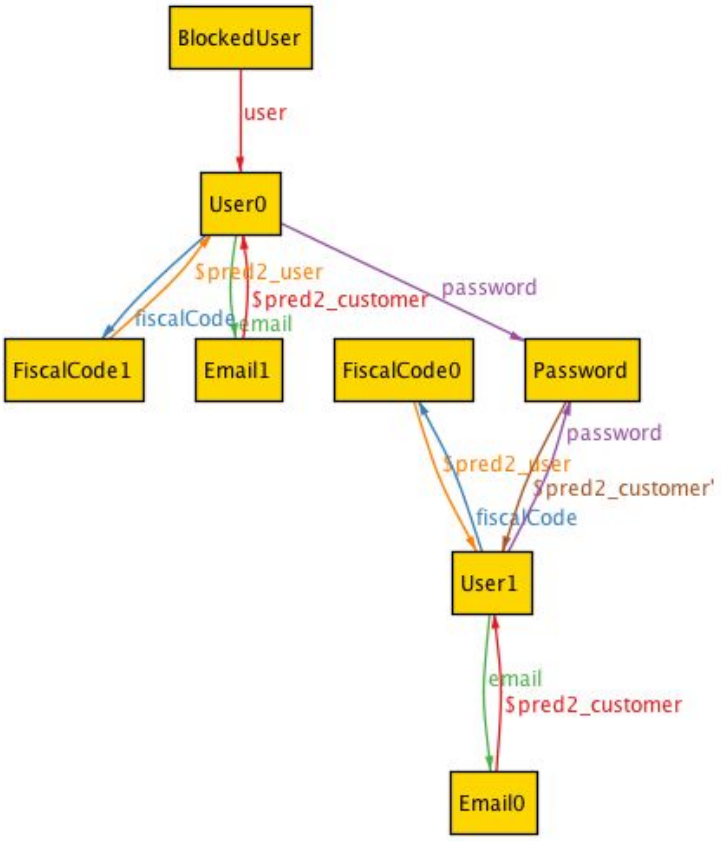

Figure 22 - Example 2 of world generated

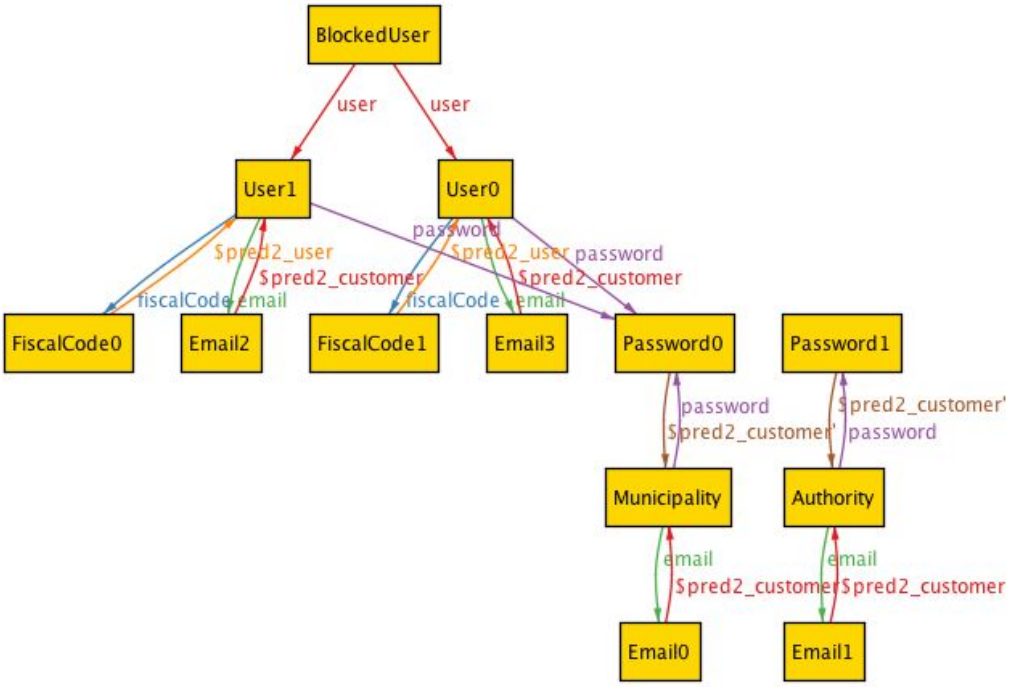

Figure 23 - Example 3 of world generated

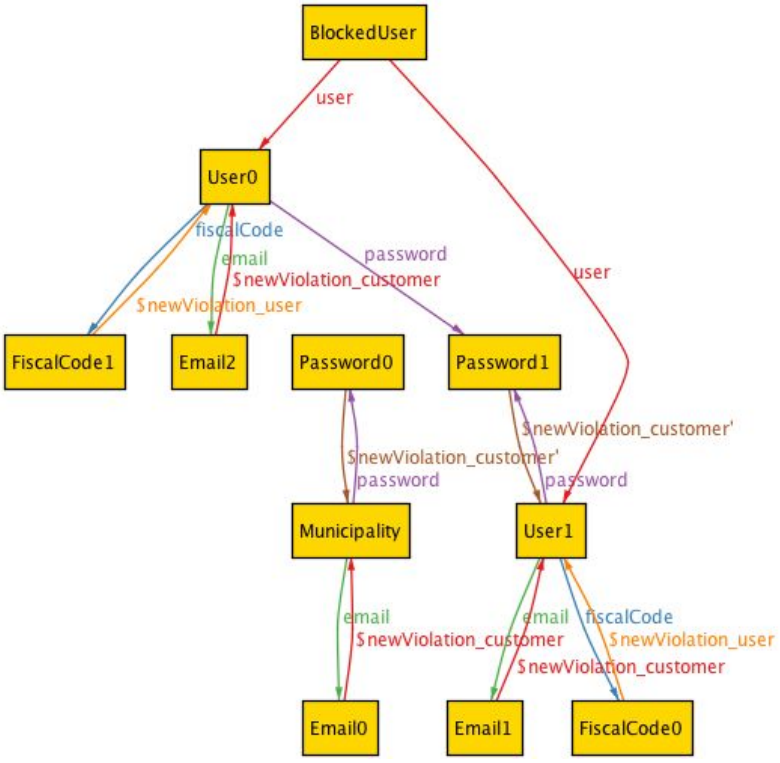

Figure 24 - Example 4 of world generated

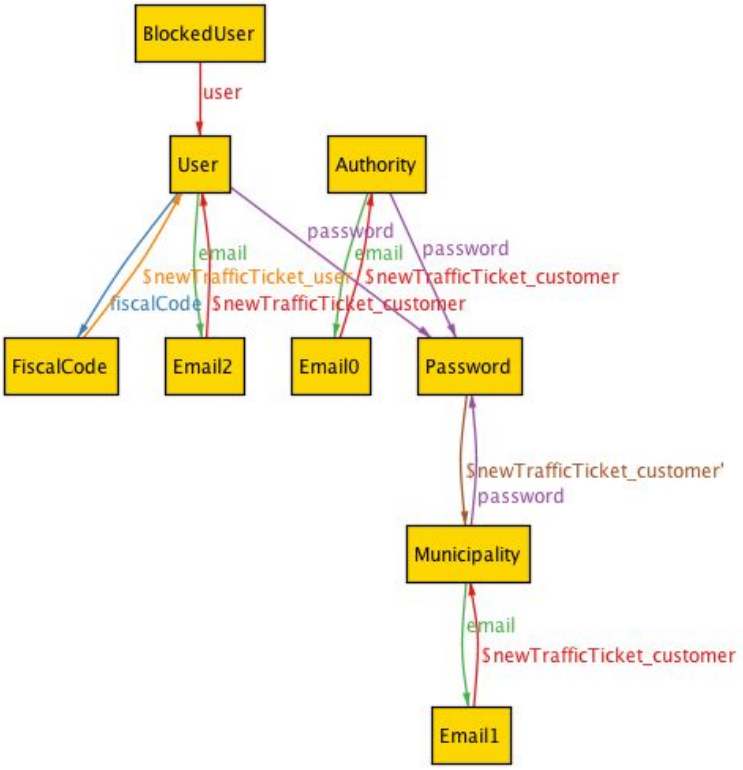

Figure 25 - Example 5 of world generated

5. Effort Spent

5.1 Fiozzi Davide

TASK	HOURS
Introduction	3
Requirements and Goals	2,5
Revision	2
Mockups	6
Document layout	2
Total	15,5

5.2 Frantuma Elia

TASK	HOURS
Introduction	3
Requirements and Goals	2,5
Use cases	8
State Diagrams	2
Product perspective	2
Product Functions	6
Software System Attributes	1,5
Scope	1,5
Hardware constraints	0,5
Total	27

5.3 Freddi Eleonora

TASK	HOURS
Introduction	6
Requirements and goals	4,5
Alloy	11
Class Diagram	1,5
Sequence Diagram	3
Scenarios	0,5
Revision	7
Total	33,5

6. References

- Slides “RASD”
- Slides “Alloy”
- Slides “Requirements engineering”
- Slides “Use of Alloy in RE”
- Slides “UML for RE”
- Specification document: “SafeStreets Mandatory Project Assignment”
- Document “RASD to be analyzed - A.Y. 2019-2020”
- IEEE Recommended Practice for Software Requirements Specifications
- Alloy doc: http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html

Used Tools

- Alloy Analyzer 5.1.0: to prove the consistency of our model
- StarUML 3.1.0: for class diagram
- www.draw.io: for sequence diagrams, use case diagrams, state diagrams and mockups