



POLITECNICO DI MILANO

SOFTWARE ENGINEERING II

SAFESTREETS -
REQUIREMENTS ANALYSIS AND
SPECIFICATION DOCUMENT

Version 1.0

Authors

Iacopo MARRI

Manuel SALAMINO

Steven Alexander SALAZAR MOLINA

November 10, 2019

Contents

1	Introduction	3
1.1	Purpose	3
1.2	Scope	4
1.3	Definitions, Acronyms, Abbreviations	6
1.3.1	Definitions	6
1.3.2	Acronyms	6
1.3.3	Abbreviations	6
1.4	Reference Documents	7
1.5	Document Structure	7
2	Overall Description	8
2.1	Product perspective	8
2.1.1	Class Diagram	9
2.1.2	State Diagrams	10
2.2	Product functions	11
2.3	User characteristics	13
2.3.1	Actors	13
2.4	Assumptions, dependencies and constraints	13
2.4.1	Domain Assumptions	13
2.4.2	Dependencies	14
3	Specific Requirement	15
3.1	External Interface Requirements	15
3.1.1	User Interfaces	15
3.1.2	Hardware Interfaces	21
3.1.3	Software Interfaces	22
3.1.4	Communication Interfaces	22
3.2	Functional Requirements	23
3.2.1	Use Cases	23
3.2.2	Sequence Diagrams	35
3.2.3	Goal mapping on requirements	39
3.3	Performance Requirements	42
3.4	Design Constraints	43
3.4.1	Standards compliance	43
3.4.2	Hardware limitations	43
3.4.3	Any other constraint	43
3.5	Software System Attributes	43
3.5.1	Reliability	43
3.5.2	Availability	43
3.5.3	Security	43
3.5.4	Maintainability	44
3.5.5	Portability	44

4	Formal Analysis using Alloy	45
4.1	What we want to model?	45
4.2	Alloy Model	45
4.3	Alloy generated Worlds	49
5	Effort Spent	52
5.1	Marri Iacopo	52
5.2	Salamino Manuel	52
5.3	Salazar Molina Steven Alexander	53

1 Introduction

The Requirement Analysis and Specification Document (RASD) aims to focus on the tasks needed to develop and implement an application, taking account of the requirements of the involved stakeholders and, analyzing and documenting also the application requirements.

Then, in the second part of the document, there is a more formal definition of the requirements with the use of the Alloy language.

In general this document is meant for developers tasked with the implementation of the System and also for all the other entities involved in validation and managing of the project.

1.1 Purpose

SafeStreets is an application that was created with the intention of monitoring the compliance with traffic regulations. Its goal is to allow Users to notify traffic violations, collect those data about them, elaborate into information, and then provide them to both Users or authorities who need, for aiming different scopes.

The application provides Users a way to send data about a violation (most common are parking violations ones, e.g. vehicles parked in reserved places), like the kind of violation, pictures of the involved vehicles, and the date and the position in which the violation occurred. The User can also specify the license plate of the vehicle and the name of the street, but in case he/she doesn't, the application is equipped with algorithms capable to obtain those and other metadata by pictures and position.

SafeStreets stores all these data, received ones and computed ones, and allows Users and competent authorities to access them, in order to gather useful information about traffic and violations, e.g. areas with the higher number of violations. This could, for example, help municipality to identify areas in which they must watch over.

Moreover, the System is also able to combine its own data, with data provided by municipality authorities, in case they offer a service that provides them. Basing on the information obtained in this way, SafeStreets will give suggestions about how to improve urban mobility situation, or about how to prevent some kind of violations from being committed (e.g. putting some cameras in vulnerable areas, add some kind of barrier and so on).

For last, authorities are allowed to use information provided by the application and generate traffic tickets. This leads the application to having to implement a method for ensuring the integrity of the incriminating data,

from when the data is generated, to when it is delivered to the application. Authorities have to be sure, for example, that they're not giving a fine to a vehicle just because of a picture alteration has been carried out by a notifying User, so altered information must be discarded. SafeStreets will also use statistics on those "rotten" data with the aim to do measure on number of bad Users, or to monitor the reliability and effectiveness of the application it self.

Here below are listed out the goals we did talk about:

- [G.1] Allows the User to access the functionalities of the application from different locations and devices.
- [G.2] Allows the User to notify about traffic violations.
- [G.3] Allows the User to send pictures, type of the violation, license plate and time.
- [G.4] Must compute license plate, position and address from the received data.
- [G.5] Allows Authority to check the correctness of a report.
- [G.6] Allows Authority to generate traffic tickets from verified reports.
- [G.7] Allows both User and Authority to access information about unsafe areas.
- [G.8] Allows both User and Authority to access statistics about effectiveness of SafeStreets.
- [G.9] Allows both User and Authority to access statistics about violations.
- [G.10] Must cross its data with the municipality ones in order to provide suggestions to improve urban mobility.
- [G.11] Must ensure that corrupted information are discarded.

1.2 Scope

According to the World and Machine paradigm, introduced by M. Jackson and P. Zave. We can identify the Machine as the System to be developed and the environment in which SafeStreets will be used as the World. The separation between these two concepts allow us to classify the entire phenomena in three different types.

World phenomena, events that take place in the real world and that the machine cannot observe.

- The driver has an accident and leaves the car in an inappropriate place.
- A malicious user reports a fake traffic violation.
- A user has an old mobile phone with a low quality camera.
- Movement of a user from a position to another one before sending the picture.
- Unexpected connection losses before receiving a picture.

Machine phenomena, events that take place inside the *System* and cannot be observed by the real world.

- Encryption of sensitive data.
- All operations performed to store/retrieve collected data.
- The System retrieves information about unsafe areas from municipalities' services.
- The System manages multiple reports of the same traffic violation.

Shared phenomena:

Controlled by the world and observed by the machine.

- A guest can sign up to the application or log in if is already registered.
- The User can send report traffic violations at any time.
- The Municipality offers up-to-date information about accidents on the territory.

Controlled by the machine and observed by the World.

- The System allows the authorities to access to violation reports.
- The System allows users to view own reports.
- The System shows inferred safe/unsafe areas.
- The Municipality gets suggestions generated by the System.
- The System allows authorities to generate traffic tickets
- The System notifies authorities about adulterated pictures.
- The User can view statistics built by the System.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Guest:** a person who has not logged in or registered yet and cannot use the functionalities of the System.
- **User:** a person that uses the application to send notifications of traffic violations.
- **Authority:** a municipality worker that is able to create traffic tickets depending on the violation that a person has committed.
- **Data provided by Municipality:** all the information about traffic tickets generated in past and traffic tickets that are generated by using SafeStreets.
- **Sensitive data:** any kind of information that could be used to identify the User who reported a traffic violation.
- **Unsafe areas:** areas in which a high number of traffic violations took place.
- **Statistics:** information that allows to show particular queries to the database, for example it is possible to ask the DBMS for the offender who has committed the highest number of violations, the safest area, the number of tickets that are being generated and so on.

1.3.2 Acronyms

- DBMS: Data Base Management System
- UI: User Interface
- API: Application User Interface

1.3.3 Abbreviations

- **[G.i]** : i-th goal.
- **[D.i]** : i-th domain assumption.
- **[R.i]**: i-th functional requirement.
- **[UCi]**: i-th use case.

1.4 Reference Documents

- Alloy Documentation
- Project assignment specifications
- IEEE Standard on Requirement Engineering (ISO/IEC/IEEE 29148)

1.5 Document Structure

Section 1 introduces the problem and describes the purpose of the application SafeStreets. Furthermore, describes the scope in which the application is defined by stating the goals and a brief description of phenomena.

Section 2 presents the overall description of the project. *Product Perspective* give more details about the boundaries of the system and world, machine and shared phenomena, while in *Product Functions* are described the main functions of the system and in *User Characteristics* the main actors. At last are defined the domain assumption on which the system relies on.

Section 3 contains all the specific requirements needed to satisfy each goal. Furthermore, are highlighted the major functions and interactions between the actors and the system using use cases and sequence diagrams.

Section 4 shows the alloy model and discuss its purpose.

Section 5 explain the effort spent by each group member to accomplish this project.

2 Overall Description

2.1 Product perspective

Here we discuss in details all the shared phenomena outlined in Section 1.3 and we also provide a domain model through class and state diagrams.

Shared Phenomena, controlled by the world and observed by the machine.

- Register/login: a normal person or a municipality worker can register to the application through two different intuitive forms; SafeStreets collects all the information inserted by the User, creates an account and verify the validity of the information given through a email-confirmation message. Once the User confirms its account he/she can proceed to login.
- Report of a traffic violation: the User is able to report traffic violations at any time, he/she can open the application, select *take a picture* and then send the report including the picture, the type of violation and the name of the street where the violation occurred.
- The Municipality offers up-to-date information about accidents: in the Municipality web platform there is a service (daily updated) that indicates streets in which accidents occurred. In this way the System can observe safe/unsafe areas.

Shared Phenomena, controlled by the machine and observed by the world.

- Forwarding of traffic violations: every time a user sends a traffic violation, the System forwards it to the authorities. The message sent by the System contains the photo of the violation, the type of violation, the street in which it took place and the area in which the photo was taken (since there may be more than one street with that name).
- Visualization of own reports: the System makes it possible for the users to view the own reports by performing a query to the database and displaying them in a ListView in the application.
- Visualization safe/unsafe areas: the System through the service offered by the municipality is able to get a list of areas in which accidents took place, thus the System can show unsafe and safe areas to the User.
- Suggest possible interventions: the System, by using a list of known and effective solutions for some common violations, is able to suggest possible interventions in order to reduce the number of violations and make unsafe areas safer.

- Generate traffic tickets: the municipality, by observing the reports provided by Safe4Streets, is able to generate traffic tickets by using a functionality of the application.
- Notify about modified photos: if the User modifies a photo in order to send malicious reports, the system mark the User as a malicious User and sends a message about this event to authorities (that already received reports created by the User).
- Generate statistics: the System is able to generate graphs that illustrate statistics, like the most egregious offenders, the effectiveness of the application itself, etc.

2.1.1 Class Diagram

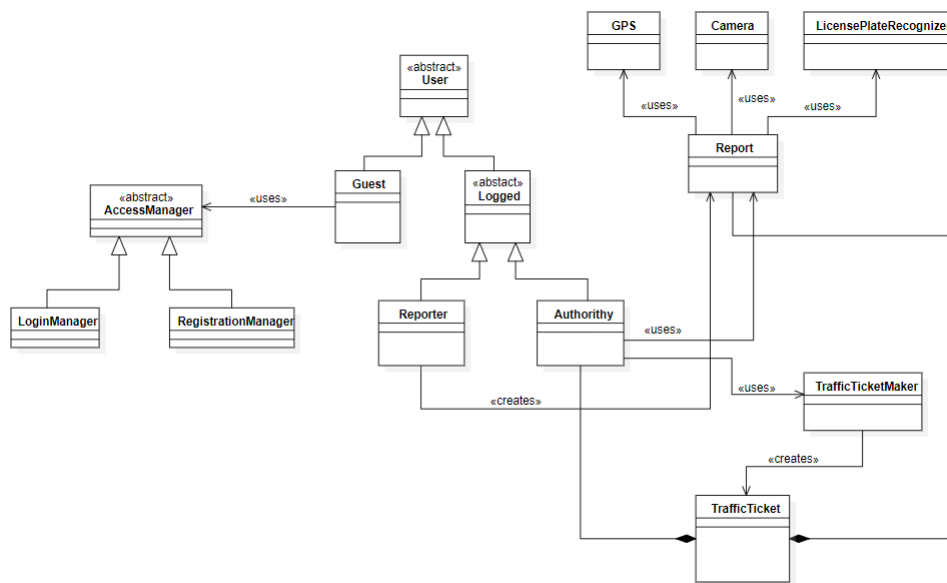


Figure 1: Class Diagram

2.1.2 State Diagrams

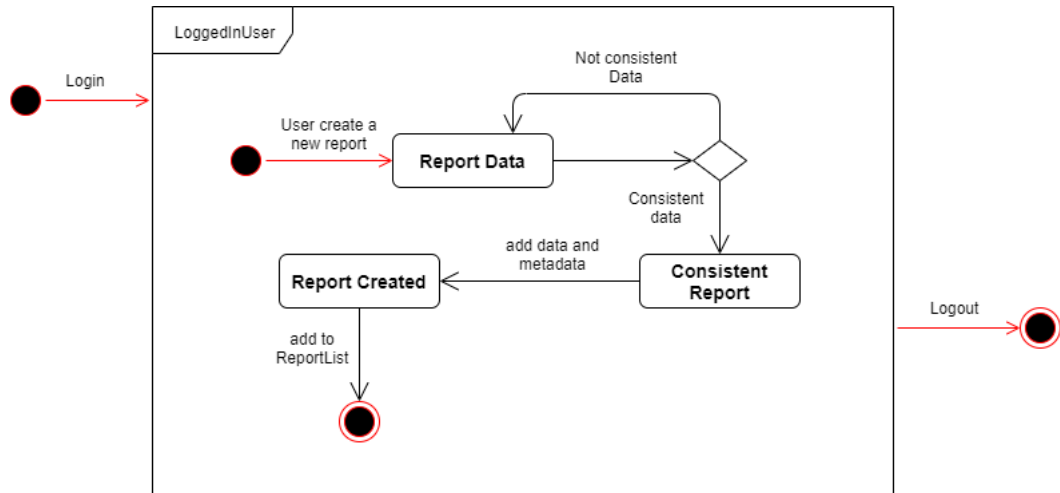


Figure 2: State Diagram of the insertion of a new Report by an User

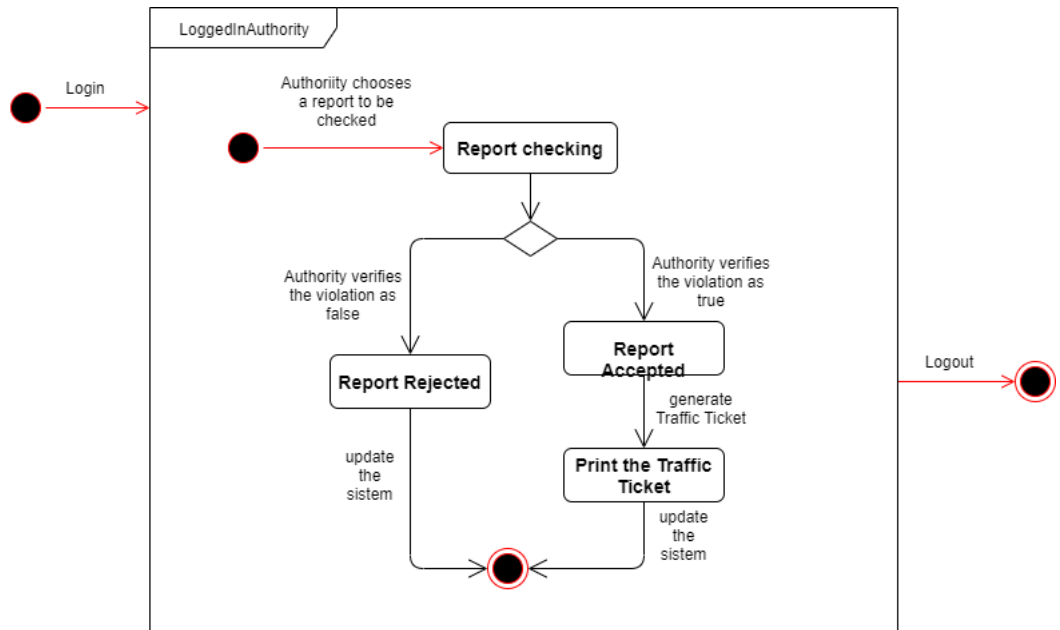


Figure 3: State Diagram of the checking of a Report by an Authority

2.2 Product functions

SafeStreets

SafeStreets was born with the intention of improving urban traffic mobility, by collecting notifies made by Users about many kinds of traffic violations. Then these information can be consulted by Users themselves, or from a Third Part, the traffic competent authorities, or the municipality, which can create tickets based on these information, on which integrity it is necessary to pay particular attention. Indeed, *SafeStreets* provides useful suggestions built upon the crossing of municipality's and *SafeStreets*'s data. Let's see more in the details the functions mentioned above:

Basic User functions

- **Notifying violations**

The Users who is witness of a rule violation, and wants to notify it through *SafeStreets*, has to be registered to the application; then he/she can fill up the notification form provided by the application, with a picture or a video of the violation occurring, the kind of violation, the kind of vehicle, the position, (or the address, which can be anyway gathered from GPS), and the license plate of the indeed vehicles.

Not all of these data are mandatory for the User to send, only a picture, the GPS position and the kind of violations are: *SafeStreets* uses an algorithm to catch the license plates from the picture, can obtain the address by the GPS position, and it implements a deep learning algorithm to recognize the vehicles by it self. *SafeStreets* is going to attach these computed metadata to the notification, in case of missing. Time is automatically taken from the notification delivery time.

- **Mining information from data**

Users can access the data stored by the application, in order to mine useful information.

SafeStreets provides an easy-use interface that allows to carry out a search for violations based upon different criteria: the *kind* of the violations, the *place* in which happen, the *time*, and the *kind of vehicle* involved. A user can then extract information from this data, for example, search for a place, and then see at which time slot of the day, more violations are committed in that place, or in which place or at which time slot, a precise type of violation occurs the most.

By the way, due to privacy purposes, license plates related to violations are not visible by a normal User, neither a User can search some information about a precise vehicle with a precise license place.

Authority Users functions

- **SafeStreets suggestion mechanism**

If the local municipality provides a way to access their own urban traffic and violations data, *SafeStreets* can cross them, to enlarge the amount of information accessible, and exploit them to formulate suggestions for the municipality, in order to increase the functionality of the urban infrastructure, reduce the violations, improve road conditions.

Authorities and municipality can ask for these suggestions for various areas, with a function button on the application interface.

Imagine a situation in which *SafeStreets* knows about huge amount of bike lane invasion violations, and municipality knows that in that zone, many people use bikes for moving, then *SafeStreets* can cross those information and provide the suggestion to build a separation line between the car road, and the bike lane, because it should be a good investment, knowing the fact that not only there are lots of violations, but also that those could be very dangerous, given the number of people using bike in that zone.

- **Searching for violations**

In addition to the mining functions provided for normal users, Authorities can also access some more "sensible data", and carry out more precise searches about violations. They can access the list of violations, with relatives license plates and data of involved people. In order to build this authorization diversification, a different type of registration will be reserved for authorities Users. (Da decidere come). Authorities can also use the application to get suggestions about which are the most unsafe urban areas, and how to get them better.

- **Traffic tickets service**

SafeStreets give road Authorities the possibility to generates traffic tickets from the violations information sent by Users. An appointee authority User can check a notified violation and all the data attached to it, confirm that it is actually a violation, and then the use function provided by *SafeStreets* (that is linked to the authorities management system) to generate a ticket.

In order to make this right, *SafeStreets* has to ensure that the chain of custody of the data, from the user to the data store, is completely reliable. To do this, security algorithms perform a validity check on the sent pictures, to be sure the picture is not been modified. In case it is, discard the notification. Discarded data are used to make statistical analysis. Another filtering level is applied by allowing Users to only send pictures taken while filling up the notification form, and not to upload previously taken ones. In this way, it's harder for a User to modify a picture before sending it.

2.3 User characteristics

SafeStreets is an application suitable for any adult person that possesses a mobile phone.

2.3.1 Actors

- **Guest:** a person who downloaded the application and still has to register, he cannot use any functionality of the application.
- **User:** once a guest has registered through the initial form of the application, he gets an account with a *username* and a *password*. Moreover, the User has accepted to give his location and access to the camera of the mobile phone.
- **Authority:** a municipality worker that is able to generate traffic tickets from the violations reported. Furthermore, he can access to other functionalities that a simple user cannot even see.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain Assumptions

- [D.1] Personal data given by Users during the registration process are assumed to be correct.
- [D.3] Pictures sent by Users are assumed to be in some precise file format.
- [D.4] The GPS is assumed to be subject to some precision error.
- [D.5] Violations for which a ticket is generated are supposed to be validated by authorities first.
- [D.6] Information obtained by authorities are supposed to be correct.
- [D.7] Is assumed that there's no bounds which suggestions provided by the S2B have to respect.

2.4.2 Dependencies

- The S2B will use the GPS service of the Users smartphone.
- The S2B will use the camera function of the Users smartphone.
- The S2B will use the internet connectivity of the Users smartphone.
- The S2B will rely on a DBMS to store all the obtained data.
- The S2B will use some external API to provide a map view service to the Users.
- The S2B will use the information provided by local municipality, to cross data and create suggestions.

3 Specific Requirement

3.1 External Interface Requirements

SafeStreets is a mobile based application, the following section will give a more detailed description, in terms of hardware, software and communication interfaces.

3.1.1 User Interfaces

Login

When a Guest downloads *SafeStreets* for the first time, the first interface will be the login one. In *Figure 4* it is also shown that if the Guest doesn't have an account it is possible to proceed by registering and creating a new account.



Figure 4: Login and Register interface

Register

In the register interface *Figure 5*. It will be asked to the Guest to insert his/her personal information, to upload an Identification card (since it is required that all accounts must be authenticated) and to choose an *Username* and a *Password*.

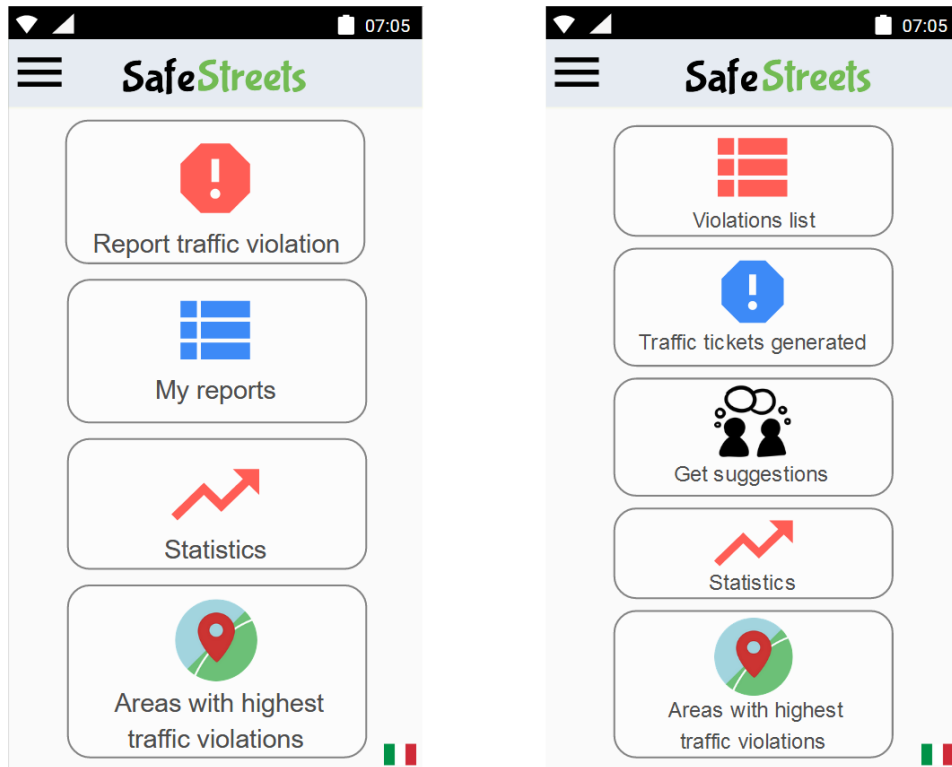


Figure 5: User and Authority Home Page interface

User Home Page

This activity contains all the possible functionalities that a simple User can use. Thus, a User can:

- Report a Traffic Violation.
- View his/her own notifications sent to Authorities and their status.
- View statistics like the effectiveness of the application, the most egregious offenders, etc.
- Open a Map that shows safe and unsafe areas (with the option to use also the data from municipality).

Authority Home Page

For Authorities the functions available are a bit different, an authority can:

- Access to the notifications sent by users.
- View the traffic tickets generated.
- Get the suggestions elaborated by the application.
- View statistics like the effectiveness of the application, the most egregious offenders, etc.
- Open a Map that shows safe and unsafe areas (with the option to use also the data from municipality).



Figure 6: Report traffic violation interface

Report a traffic violation

This form allows users to notify authorities when traffic violations occur, an user takes a photo of the violation, selects the type of the violation and inserts the location where it occurred. Moreover, the User can also include further information about the vehicle like the license plate, the model, etc.

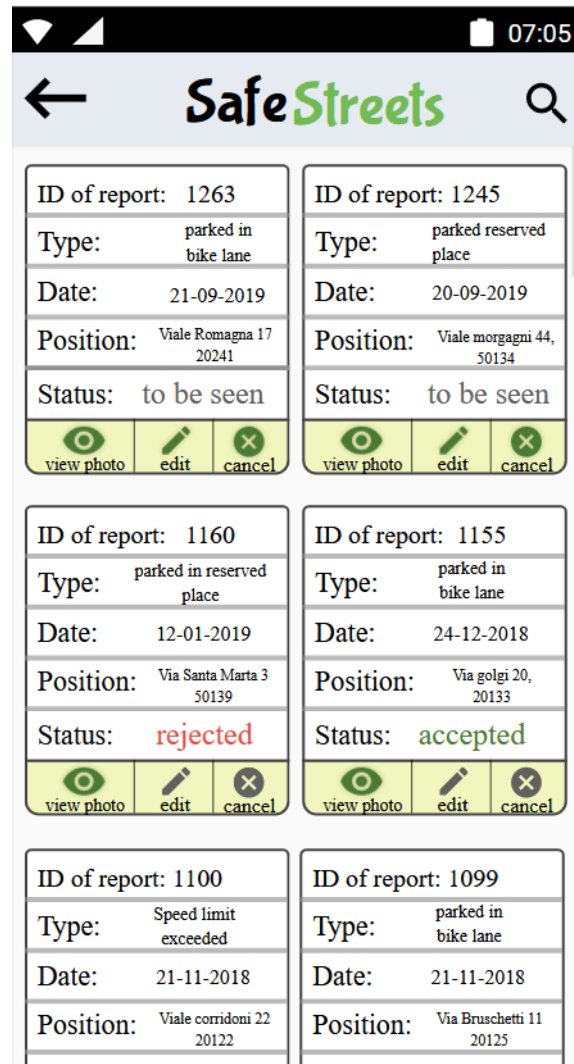


Figure 7: My reports interface

My Reports

Once a User sent a notification it is possible to see all the notifications sent to authorities, *Figure 7*. In this activity it is also possible to search them by their ID and see their status. There are three possible status: to be seen, accepted, rejected.

Violations list

An Authority is able to see all the violations sent by users, the list shows the most important information to know if an Authority is interested or not in that violation; type, date and position. It is also possible to see the trustness of the User who sent each notification.

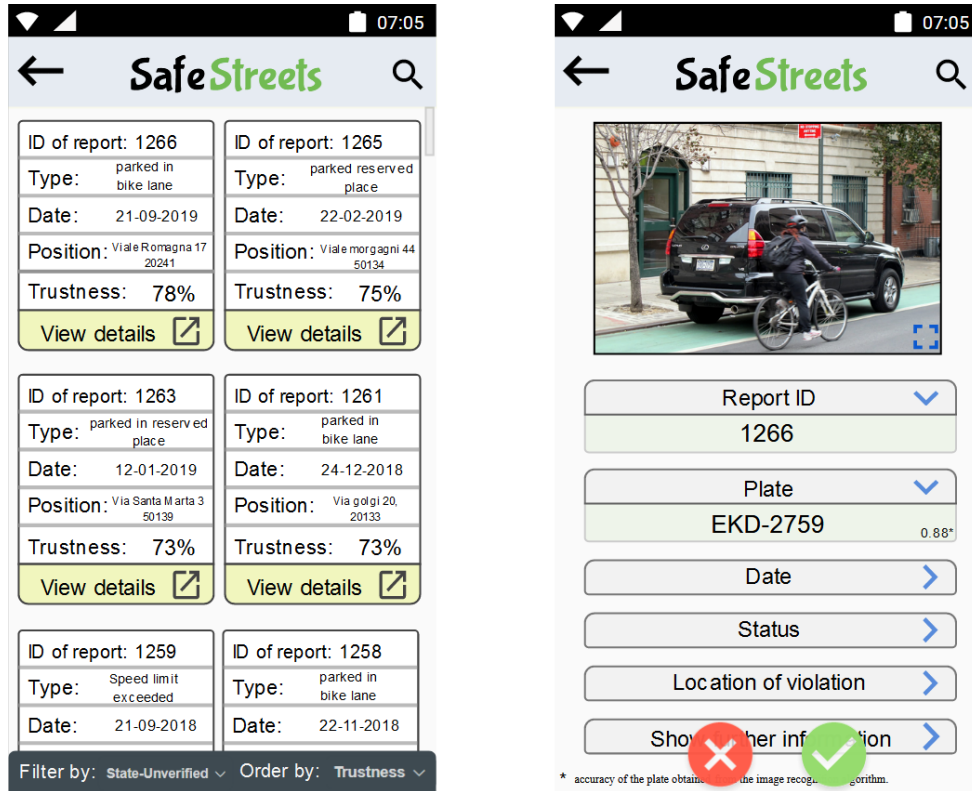


Figure 8: Violations list interface

Details about a notification

After an Authority clicks on *View details* from the Violations list, he can access to details like license plate, model of the car, position where the violation occurred, the picture sent by the user, etc. The license plate of being shown is the output obtained from a image recognition algorithm, this output comes also with a number $\in [0, 1]$ that represents the accuracy of the license plate obtained.

Traffic tickets generated

In the previous activity it was possible to see that an Authority can press on *X* button to reject a report from a user or to press on the check button to accept and generate a traffic ticket related to it. In *Figure 12* all the traffic tickets generated by the Authority logged in are shown, It is possible to download the traffic ticket as a PDF file.

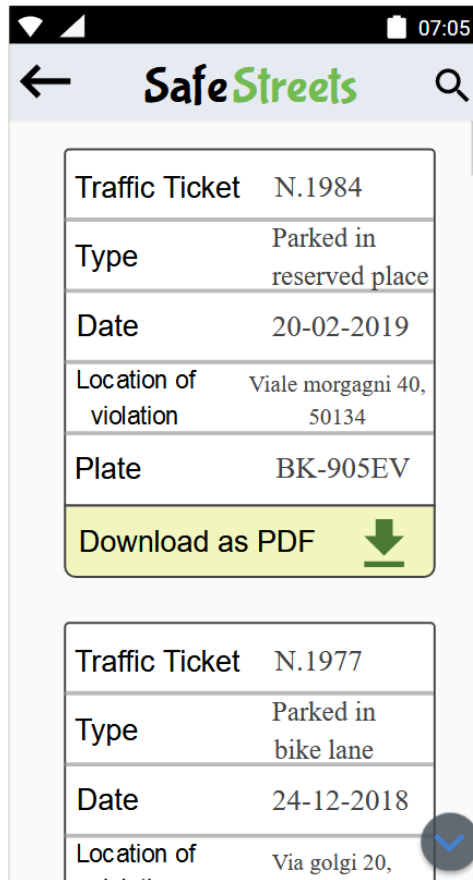


Figure 9: Traffic tickets interface

Get Suggestions

Figure 10 shows the list of possible suggestions that authorities can see, each Authority see only the possible suggestions related to the own city, so this list can be shared between different districts but in the same city. An authority can print the suggestion by downloading as a PDF file or he/she can share it to the department in charge of receiving suggestions by citizens. The suggestions are taken from an AI algorithm that is continuously updating its own dataset by modifying the effectiveness of each possible solution to each problem.

Statistics

Users and Authorities can also see statistics about traffic violations and how SafeStreets impacts in streets safeness. An example is the graph *Effectiveness of SafeStreets*, in this graph it is possible to see how the rate between traffic tickets generated and notifications sent has been increasing since the last years.

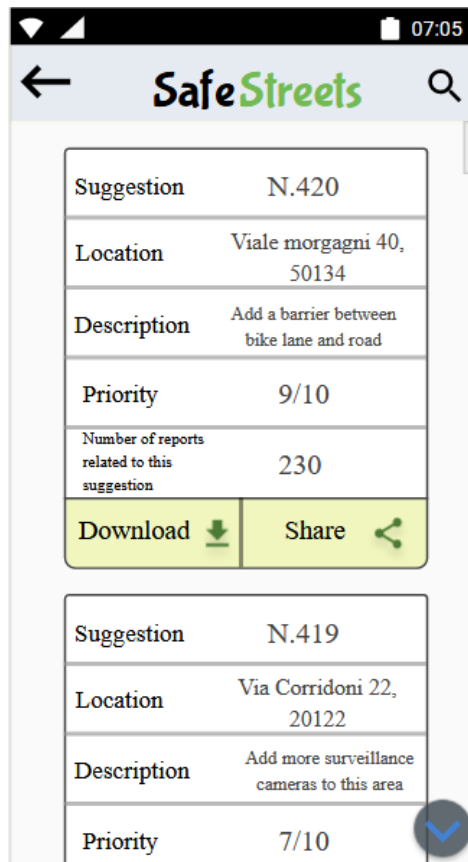


Figure 10: Get suggestions interface

Areas with highest traffic violations

This activity gives the possibility to see the tag assigned to each area in which SafeStreets has been used (or areas that are present in a service offered by the municipality). There are 4 possible labels: Safe, Unsafe, Very Safe and Dangerous. The tags inferred can be produced by merging the own data of SafeStreets with the data retrieved from a service provided by the Municipality, if the option is enabled.

3.1.2 Hardware Interfaces

SafeStreets is available for smartphones that guarantee location data acquisition, access to the camera and internet access.

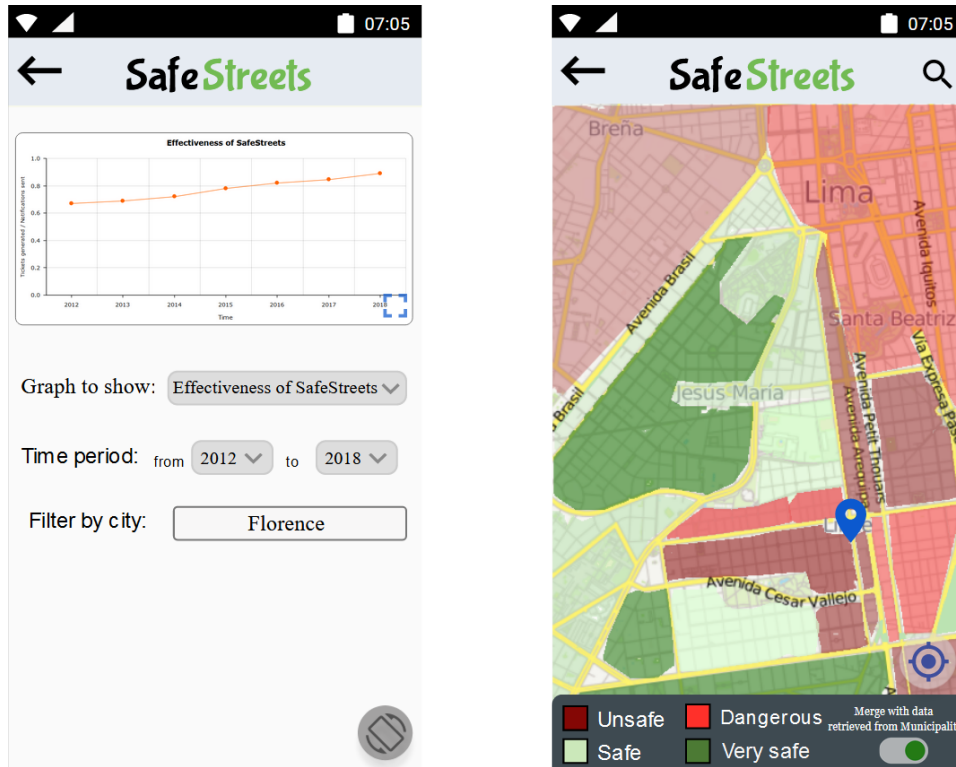


Figure 11: Statistics and Safe/Unsafe Areas interface

3.1.3 Software Interfaces

- **Maps Service:** the system to be access to a map service that allows to show safe and unsafe areas.
- **Municipality Accidents History Service:** SafeStreets can cross its own data with the information provided by the Municipality Accidents History API to suggest possible interventions.
- **Municipality Traffic Tickets Service:** Municipality offers a service that allows to generate traffic tickets from the violations coming from SafeStreets. Safestreets sends information (confirmed as traffic violations) to the department in charge of generating traffic tickets in the municipality.

3.1.4 Communication Interfaces

The system uses HTTPS protocol to transmit data over the internet and to the DBMS.

3.2 Functional Requirements

3.2.1 Use Cases



Figure 12: Use Case Diagram: User

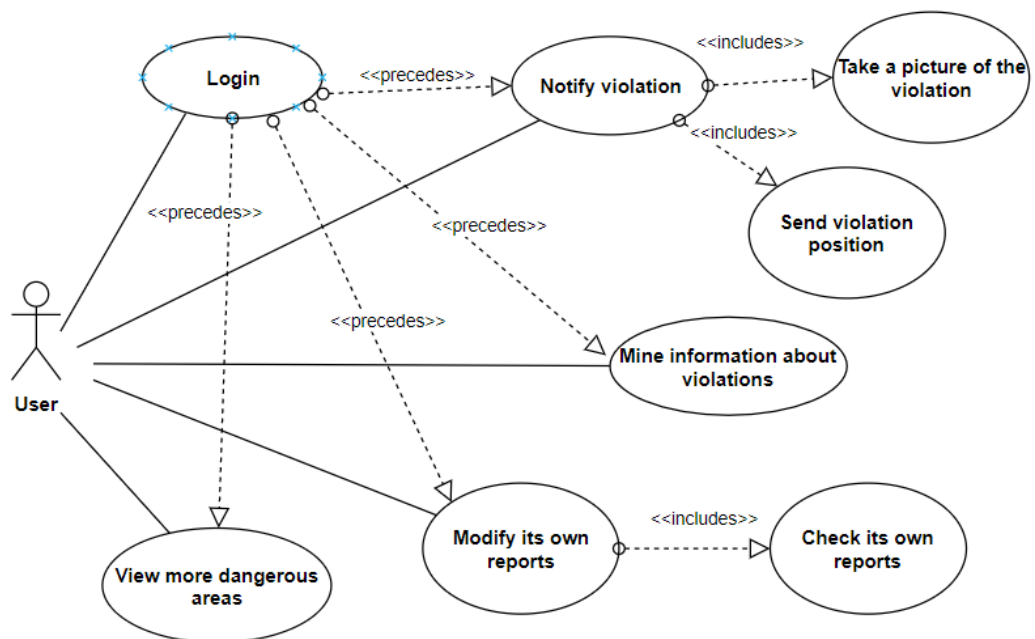


Figure 13: Use Case Diagram: User

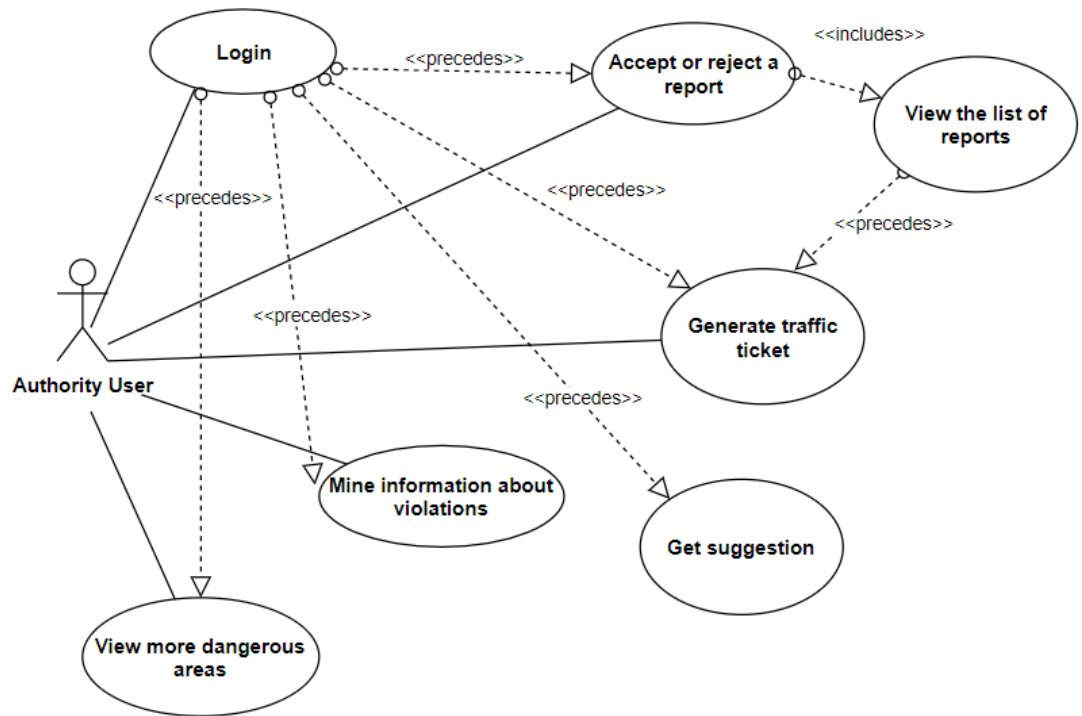


Figure 14: Use Case Diagram: Authority

Scenarios:

Scenario 1

Steven just decided he wants to stop using public services, and start going to the university by bike every morning, to save some money, and to boost his health conditions. But Milan is a tricky place where to hang around by bike, and Steven knows that he has to pay attention to the traffic, and to the large amount of cars that usually cross the bike lane, or park on it, forcing him to pass through the car road and putting him in danger.

In order to avoid those risky paths, Steven downloads *SafeStreets*, creates an account, logs in, and accesses the information about violations stored by the application, to locate the most dangerous zones with relation to his problem. This allows him to choose the safest way (even if a little bit longer maybe) to get to the university from his place.

Scenario 2

As a consequence of popular protests due to bad urban conditions, the municipality of Firenze is in a situation in which, to accomplish the citizens and to raise the city liveability, must to bring some improvement to the urban infrastructure, but they don't know where to start because of a limited budget.

By using the *SafeStreets* suggestion function, they can exploit the melting of their own information, crossed with the information that *SafeStreets* stored in its lifetime about the violations. From all these information, *SafeStreets* can suggest the municipality the best measures to implement to get the maximum urban situation improvement possible.

Scenario 3

Diane has a website in which she writes about social problems in her country, Peru. Her next article is about safeness, the solutions that each city has implemented and their effectiveness. She finds out that some municipalities decided to collaborate with a mobile application that involves citizens by allowing them to report traffic violations. During the investigation in which Diane was working on, she realizes that the cities that implemented it became safer and so she decides to use the statistics in *SafeStreets* to show how many violations are being committed in each city that collaborated with the application and how effective is *SafeStreets*.

ID	UC1
Description	A <i>Guest</i> creates a normal <i>User</i> account to use the application
Actors	<i>Guest</i>
Preconditions	<ul style="list-style-type: none"> • <i>Guest</i> has downloaded the app onto his device • <i>Guest</i> has a working internet connectivity • <i>Guest</i> hasn't an account still.
Flow of Events	<ol style="list-style-type: none"> 1. The <i>System</i> ask the <i>Guest</i> to Log in or to Register 2. The <i>Guest</i> choose the Register option 3. The <i>System</i> return the registration form 4. The <i>Guest</i> fill in the form with its personal information and click confirm button 5. The <i>System</i> checks the validity of the input 6. The <i>System</i> generates a new account with a new identifier 7. The <i>System</i> sends a confirmation e-mail to the <i>Guest</i>
Postconditions	The <i>Guest</i> becomes a <i>User</i> , and is now able to log in and use the application services
Exceptions	<ul style="list-style-type: none"> • The <i>Guest</i> inputs a non-valid e-mail address • The <i>Guest</i> inputs a password that not matches with security standards. <p>In both cases, the <i>System</i> will not let the confirm button to be pressed until all fields are correctly filled in.</p>

ID	UC2
Description	A normal <i>User</i> logs in
Actors	<i>User</i>
Preconditions	<ul style="list-style-type: none"> • <i>User</i> has downloaded the app onto his device • <i>User</i> has a working internet connectivity • <i>User</i> has an account
Flow of Events	<ol style="list-style-type: none"> 1. The <i>System</i> ask the <i>User</i> to Log in or to Register 2. The <i>User</i> choose the log in option 3. The <i>System</i> return the log in form 4. The <i>User</i> fill in the form with its log in data, and press the log in button 5. The <i>System</i> checks the validity and the matching of the inputs 6. The <i>User</i> accesses the interface with the various <i>User</i> functions
Postconditions	The <i>User</i> is now able to browse in the application and to use the application services
Exceptions	<ul style="list-style-type: none"> • The <i>User</i> inputs a non-existing username • The <i>User</i> inputs a wrong password for that username. <p>The flow of the events restarts at point 4, showing an error message.</p>

ID	UC3
Description	A normal <i>User</i> notifies a violation
Actors	<i>User, System</i>
Preconditions	<ul style="list-style-type: none"> • <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The <i>User</i> browses to the report violation section 2. The <i>System</i> return the form for reporting a violation 3. The <i>User</i> fill in the form with the mandatory data 4. The <i>User</i> press the button to take a picture of the violation 5. The <i>User</i> press the send button, and send the notification. 6. The <i>System</i> runs an algorithm to extract metadata about the notification, and to check the validity of the information. Then submits the report to the <i>Authorities</i>. 7. The <i>System</i> add the report to the <i>User</i> reports section.
Postconditions	<ul style="list-style-type: none"> • The <i>User</i> can see the his/her new report and its status in the <i>My Report</i> section. • The <i>Authorities</i> can see the notification in their report list
Exceptions	<ul style="list-style-type: none"> • The <i>User</i> inputs non valid data • The <i>User</i> doesn't input some mandatory data. <p>The send button will not be available until fields are correctly filled in.</p> <ul style="list-style-type: none"> • The <i>System</i> will find irregularities by running its algorithms. <p>In this case, the notification will be rejected.</p>

ID	UC4
Description	A <i>User</i> mines information about urban violations
Actors	Normal <i>User</i> , Authority <i>User</i>
Preconditions	<ul style="list-style-type: none"> • <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The <i>User</i> browses to the statistic section 2. The <i>System</i> return the form for searching for statistics 3. The <i>User</i> selects the information which he/she wants information about 4. The <i>User</i> press the search button 5. The <i>System</i> shows the results of the search
Postconditions	<ul style="list-style-type: none"> • The search parameters are cached in a preference list
Exceptions	<ul style="list-style-type: none"> • No results exist for the search performed by the <i>User</i>. So an advice is showed up.

ID	UC5
Description	A <i>User</i> checks and modify its own reports
Actors	Normal <i>User</i>
Preconditions	<ul style="list-style-type: none"> • <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The <i>User</i> browses to the report list section 2. The <i>System</i> return the form with the list of the reports made by the <i>User</i> 3. The <i>User</i> order the reports by some criteria, or search for some precise reports by the filter option. 4. The <i>System</i> shows the results of the search 5. The <i>User</i> taps on the reports he/she wants to check 6. The <i>System</i> shows the report details. 7. The <i>User</i> tap on modify button, and change the parameters of the report, or mark it as "canceled". 8. The <i>User</i> tap save button
Postconditions	<ul style="list-style-type: none"> • The report is now changed. • Authorities <i>Users</i> can see which modifications has been made to the report. • Authorities <i>Users</i> can still see the reports marked as "cancelled", and decide about them.
Exceptions	<ul style="list-style-type: none"> • The <i>User</i> didn't send any report still. So an empty list is showed up. • The <i>User</i> tries to modify a report that has been already accepted. The modification is not allowed, and the flow of events restart from event 6.

ID	UC6
Description	An Authority <i>User</i> validates a report
Actors	Authority <i>User</i>
Preconditions	<ul style="list-style-type: none"> • Authority <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The authority <i>User</i> browses to the reports list section 2. The <i>System</i> return the list of reports from normal <i>Users</i> 3. The authority <i>User</i> filters and orders the reports according to his/her needs 4. The authority <i>User</i> taps on the chosen report 5. The authority <i>User</i> taps the validate green button.
Postconditions	<ul style="list-style-type: none"> • The report is now marked as "validated", and it is officially a violation • The <i>User</i> who sent the report can no longer modify it
Exceptions	<ul style="list-style-type: none"> • There are no reports in a "to be seen" status

ID	UC7
Description	An Authority <i>User</i> rejects a report
Actors	Authority <i>User</i>
Preconditions	<ul style="list-style-type: none"> • Authority <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The authority <i>User</i> browses to the reports list section 2. The <i>System</i> return the list of reports from normal <i>Users</i> 3. The authority <i>User</i> filters and orders the reports according to his/her needs 4. The authority <i>User</i> taps on the chosen report 5. The authority <i>User</i> taps the reject red button.
Postconditions	<ul style="list-style-type: none"> • The report is now marked as "rejected", and so it is not a violation • The <i>User</i> who sent the report can no longer modify it
Exceptions	<ul style="list-style-type: none"> • There are no reports in a "to be seen" status

ID	UC8
Description	An Authority <i>User</i> generates a traffic ticket
Actors	Authority <i>User</i>
Preconditions	<ul style="list-style-type: none"> • Authority <i>User</i> is logged in the application
Flow of Events	<ol style="list-style-type: none"> 1. The authority <i>User</i> browses to section for generating tickets 2. The <i>System</i> return the form the fill in for generating a ticket 3. The authority <i>User</i> fills in the form, with all data and link the ticket to the relative reports (o comunque una roba del genere) 4. The authority <i>User</i> clicks the generate button
Postconditions	<ul style="list-style-type: none"> • A new ticket now exists in the authorities database, and it will be sent to the right recipient. • The status of the report in relation with this ticket will change in "processed". (o qualcosa che indica che per quel report è già stata fatta una multa)
Exceptions	<ul style="list-style-type: none"> • The authority <i>User</i> doesn't fill in some field, or fill it in a wrong way. <p>When "generate" button is pressed, the <i>System</i> will shows up an alert message, and the flow of the events will restart from event 3</p> <ul style="list-style-type: none"> • The authority <i>User</i> links the ticket with a violation whose involved license plates or vehicles are not the same included in the generated ticket. <p>The <i>System</i> will shows up an alert message, and the flow of the events will restart from event 3</p> <ul style="list-style-type: none"> • The authority <i>User</i> links the ticket with a report which is not in a "validated" state. <p>The <i>System</i> will shows up an alert message, and the flow of the events will restart from event 3</p>

ID	UC9
Description	An Authority <i>User</i> searches for a suggestion
Actors	Authority <i>User</i>
Preconditions	<ul style="list-style-type: none"> • Authority <i>User</i> is logged in the application • Involved municipality must provide an interface to share its data stored with <i>SafeStreets System</i>
Flow of Events	<ol style="list-style-type: none"> 1. The authority <i>User</i> browses to the suggestions section 2. The <i>system</i> retrieves the list of suggestions 3. The authority <i>User</i> clicks on search button 4. The authority <i>User</i> inserts the keywords he/she wants to look for 5. The <i>System</i> retrieves che list of filtered suggestions 6. The authority <i>User</i> clicks on the suggestion he wants to look at and share 7. The <i>System</i> shows all the detail of the suggestion 8. The authority <i>User</i> clicks on the share option 9. The <i>System</i> pops up a window from which to chose where to share the suggestion
Postconditions	<ul style="list-style-type: none"> • None
Exceptions	<ul style="list-style-type: none"> • No suggestions match the keywords specified. In this case, no results are showed, and a alert message show up

3.2.2 Sequence Diagrams

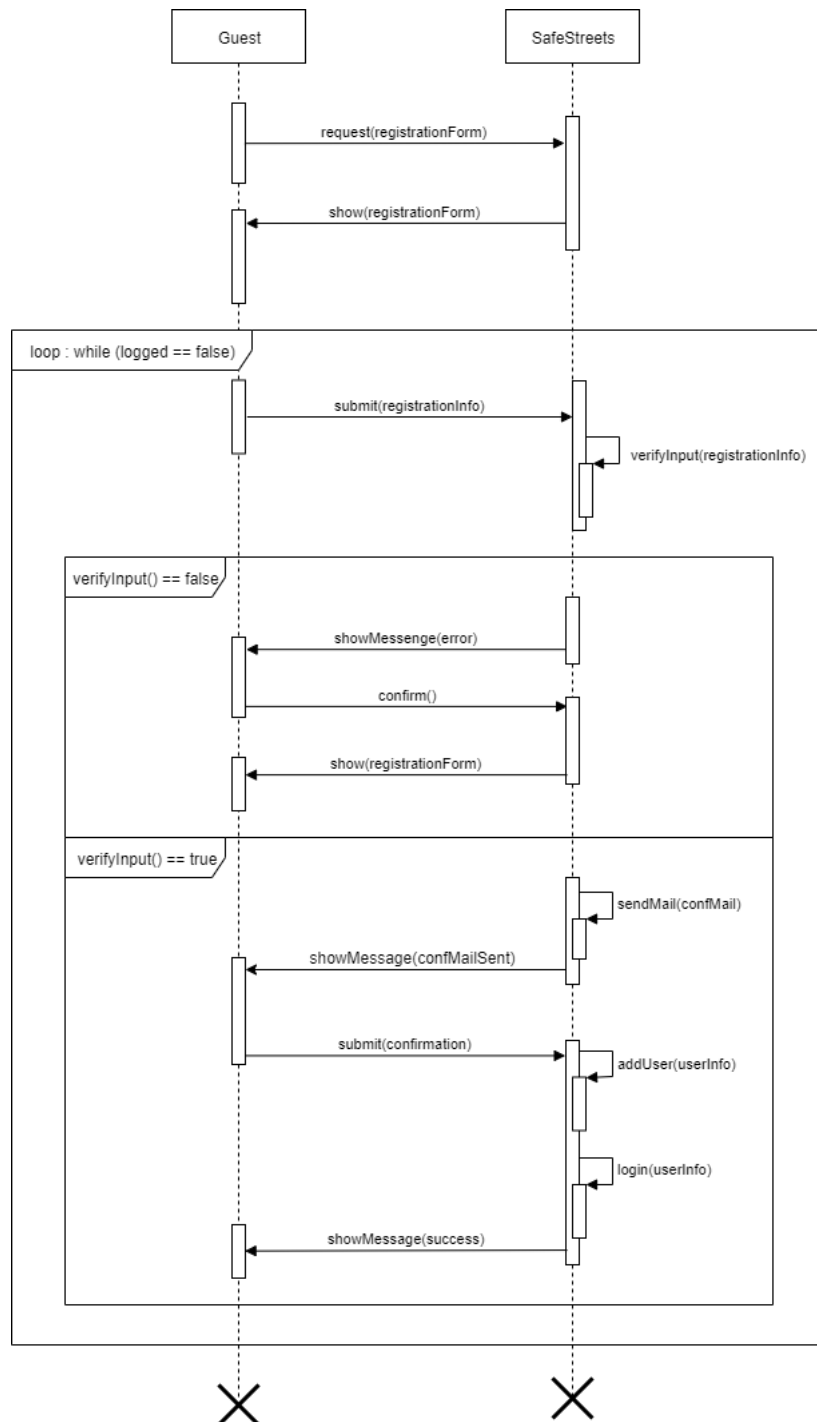


Figure 15: Sequence Diagram of the registration of a User

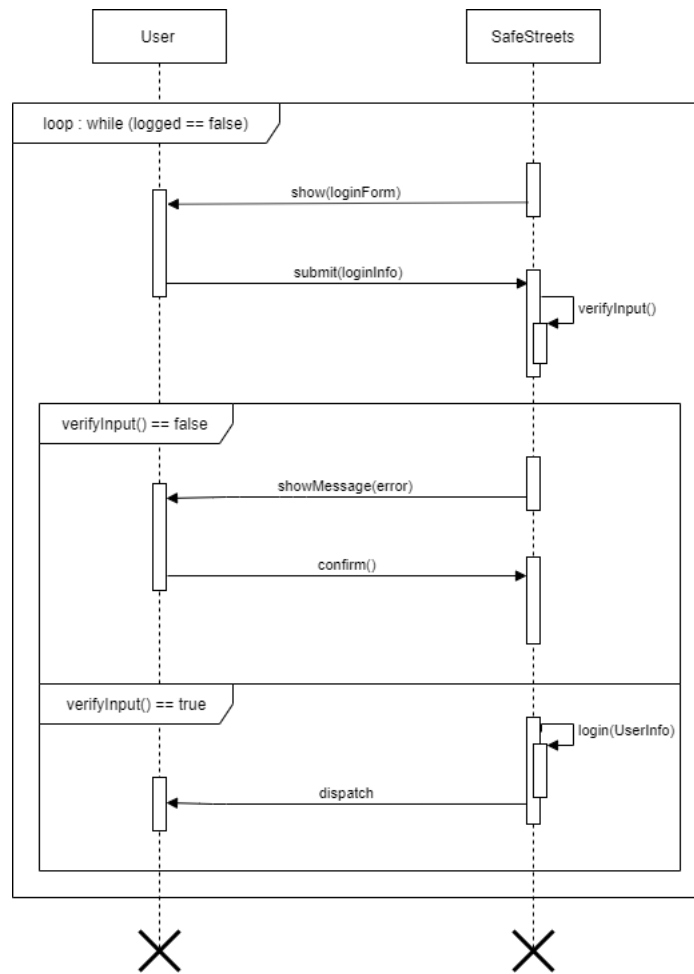


Figure 16: Sequence Diagram of the login of a User or of an Authority

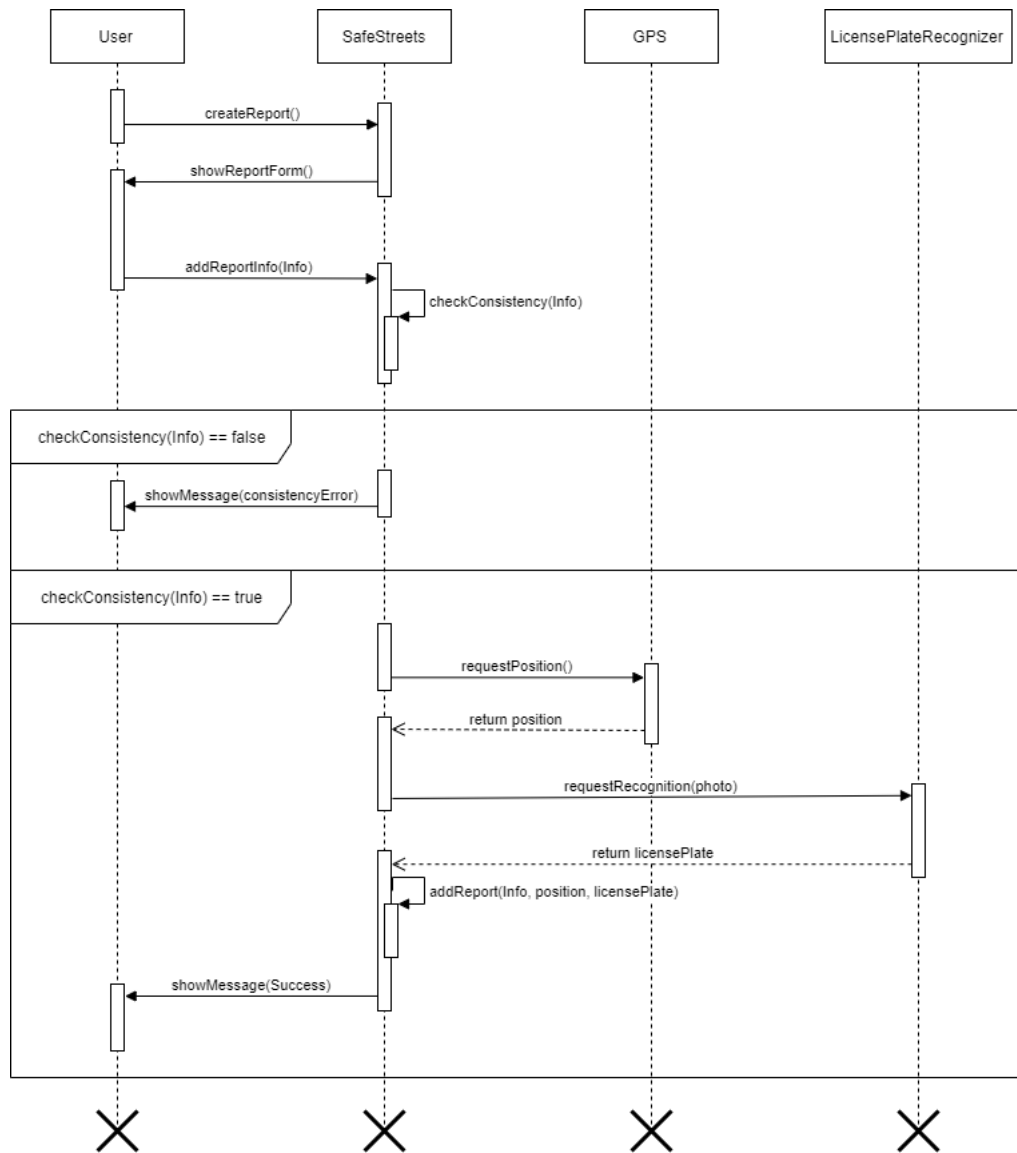


Figure 17: Sequence Diagram of the insertion of a Report by a User

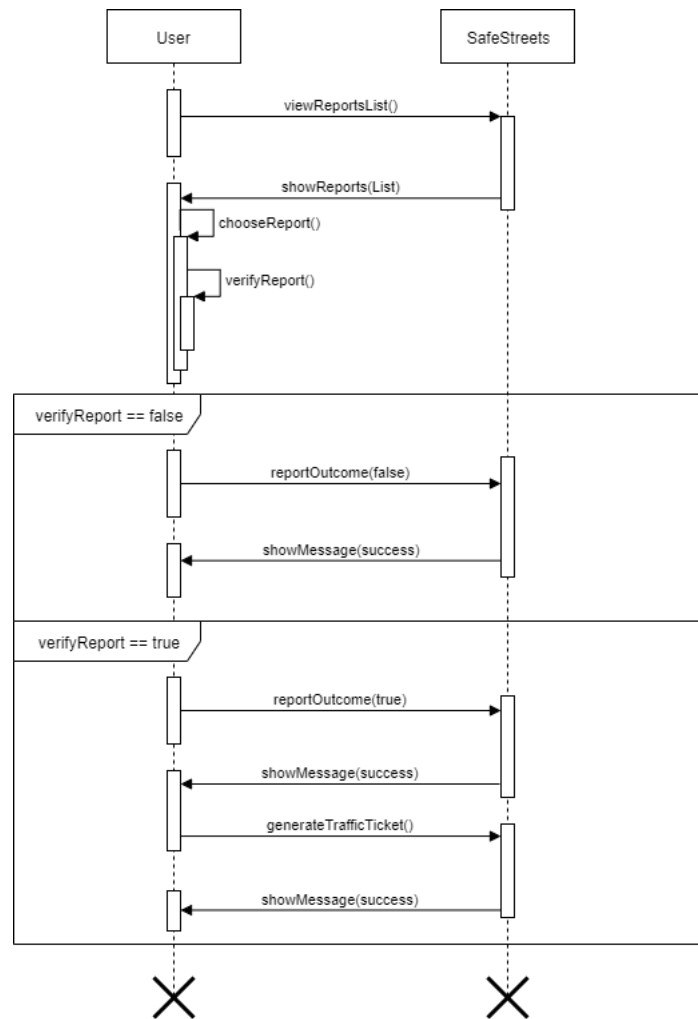


Figure 18: Sequence Diagram of the checking of a Report and, eventually, the generation of the corresponding Traffic Ticket

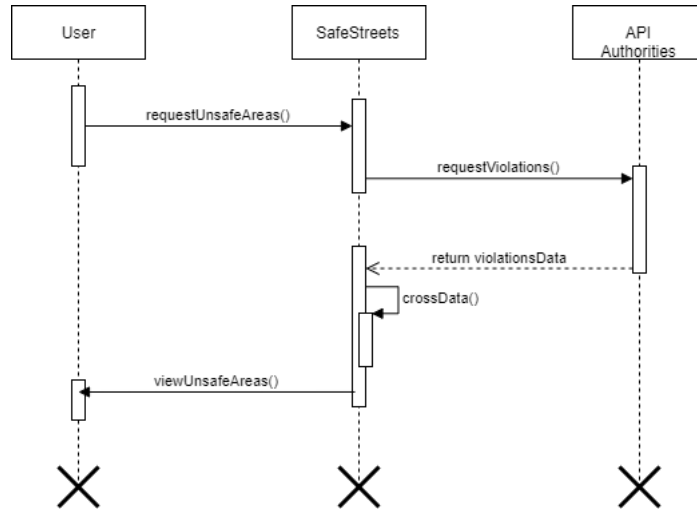


Figure 19: Sequence Diagram of the request to know which are the unsafe areas

3.2.3 Goal mapping on requirements

In this section, it will be showed the functional requirements and the domain assumption related to each goal.

- **[G.1] Allows the User to access the functionalities of the application from different locations and devices.**
 - [R.1] A *guest* must be able to register and become a *User*.
 - [R.2] The S2B must provide an already created account to the *Authorities*.
 - [R.3] The S2B must check that credentials are correct, then send a confirmation e-mail.
 - [R.4] The S2B must check the validity of the document sent by a *User* to give his/her the access to all the functionalities.
 - [R.5] The S2B must allow *User* to log in with his/her registration credentials.
 - [D.1] Personal data given by Users during the registration process are assumed to be correct.
- **[G.2] Allows the User to notify about traffic violations.**
 - [R.5] The S2B must allow *User* to log in with his/her registration credentials.
 - [R.6] The S2B must provide an algorithm for compute the trustness of reports.

- [R.7] *User* must be allowed to see the list of his/her reports.
- [R.8] *User* must be allowed to edit or cancel his/her reports.
- [R.9] The S2B must allow the *User* to send the report only within 5 minutes from when he/she starts to fill it in.
- [R.10] The S2B must submit the received reports to an *Authority* for checking its validity.
- [R.11] The S2B must recognize if a notification that is about to be made, may involve a violations already notified, and alert the *User*
- [R.12] The S2B must recognize if more done reports refer to the same violation.
- [D.3] Pictures sent by Users are assumed to be in some precise file format.
- [D.4] The GPS is assumed to be subject to some precision error.
- [G.3] **Allows the User to send pictures, type of the violation, license plate and time.**
 - [R.13] The S2B must allow the *User* to only send pictures taken while filling in the notify form, accessing the camera by the *SafeStreets* application, and not to upload some by the gallery.
 - [R.14] The S2B must allows the *User* to chose from a predefined list of "type of violation".
 - [R.15] The S2B must allow the *User* to only send time automatically taken from the operative system timer.
 - [D.3] Pictures sent by Users are assumed to be in some precise file format.
- [G.4] **Must compute license plate, position and address from the received data.**
 - [R.9] The S2B must allow the *User* to send the report only within 5 minutes from when he/she starts to fill it in.
 - [R.16] The S2B must use some out coming map service.
 - [D.3] Pictures sent by Users are assumed to be in some precise file format.
 - [D.4] The GPS is assumed to be subject to some precision error.
- [G.5] **Allows Authority to check the correctness of a report.**
 - [R.2] The S2B must provide an already created account to the *Authorities*.

- [R.5] The S2B must allow *User* to log in with his/her registration credentials.
- [R.17] The S2B must allow only 1 *Authority* per time to change the status of a report.
- [R.18] The S2B must submit to the *Authority* only the reports with a computed trust value higher than 60
- [G.6] **Allows Authority to generate traffic tickets from verified reports.**
 - [R.2] The S2B must provide an already created account to the *Authorities*.
 - [R.19] The S2B must not allow *Authority* to generate more tickets for the same violation
 - [R.20] The S2B must allow generated tickets information to be moved into the *Authorities* data store.
 - [D.5] Violations for which a ticket is generated are supposed to be validated by authorities first.
- [G.7] **Allows both User and Authority to access information about unsafe areas.**
 - [R.2] The S2B must provide an already created account to the *Authorities*.
 - [R.5] The S2B must allow *User* to log in with his/her registration credentials.
 - [R.21] The S2B must update the information about statistics and safe areas with the new validate reports within 10 seconds from the new information delivery.
- [G.8] **Allows both User and Authority to access statistics about effectiveness of SafeStreets.**
 - [R.2] The S2B must provide an already created account to the *Authorities*.
 - [R.5] The S2B must allow *User* to log in with his/her registration credentials.
 - [R.21] The S2B must update the information about statistics and safe areas with the new validate reports within 10 seconds from the new information delivery.
- [G.9] **Allows both User and Authority to access statistics about violations.**

- [R.2] The S2B must provide an already created account to the *Authorities*.
- [R.5] The S2B must allow *User* to log in with his/her registration credentials.
- [R.21] The S2B must update the information about statistics and safe areas with the new validate reports within 10 seconds from the new information delivery.
- [G.10] **Must cross its data with the municipality ones in order to provide suggestions to improve urban mobility.**
 - [R.22] SB2 must have been provided with the access authorization to the municipality data.
 - [R.23] The S2B must check the physical feasibility of the suggestions provided, with relation to the map.
 - [R.24] The S2B must generate suggestions based on the actual information and the actual urban situation.
 - [D.6] Information obtained by authorities are supposed to be correct.
 - [D.7] Is assumed that there's no bounds which suggestions provided by the S2B have to respect.
- [G.11] **Must ensure that corrupted information are discarded.**
 - [R.25] The S2B must use an algorithm to recognize picture that have been physically or digitally modified.
 - [R.26] The S2B must compare received data with the ones computed with algorithms in order to find discrepancies.
 - [R.27] The S2B must not submit to *Authorities* the reports that have been valuated with a low trust level.

3.3 Performance Requirements

This section contains some numerical requirements of the system, relative to the interaction with *Users* and to the performances. When a *User* send a report, this must be taken in account (accepted or rejected) within one month by the competent authorities, and than another month to eventually generate a ticket. Due to DBMS storage issues, it's not possible to permanently store all the reports that does not lead to a "conclusion".

When a report is accepted by authorities, its existence must be take into account by the statistics functionalities in less than 30 seconds, in order to offer information and statistics always up to the most recent date.

A suggestion must be provided by *SafeStreets* within the time span in which

this suggestion can be useful and significant.

A report by a *User* must be sended to *SafeStreets* within 5 minutes from when the report has been created and the *User* started to fill it. This comes with the intention of not allow *User* to send wrong position by moving away from the violation location while compiling the form, or to send wrong time information.

3.4 Design Constraints

3.4.1 Standards compliance

The system doesnt need to be directly compliant to any particular standard, it only needs to consider the position acquired from the User as *Latitude* and *Longitude*. In the design analysis it will be decided if it is necessary to add any further compliance.

3.4.2 Hardware limitations

SafeStreets is a software application, the device with *SafeStreets* installed must guarantee access to internet, a camera with at least 8 Megapixel and flash in order to take photos during night.

3.4.3 Any other constraint

Each User must upload a photo of his/her ID document in order to have their account verified.

3.5 Software System Attributes

3.5.1 Reliability

SafeStreets should be available 24/7 in order to allow Users to report traffic violations at any given time. However it can be accepted to have periods of 2-4 hours in which the server is down because of maintenance, this doesn't represent a problem since the mobile application can use a local Database and store the reports being sent by the User when it is down.

3.5.2 Availability

Since *SafeStreets* does not have a critical nature, 90% of availability is sufficient.

3.5.3 Security

In order to guarantee a secure system, *SafeStreets* uses HTTPS for a secure communication between Users and the Server, HTTPS allows to avoid pos-

sible Man in the Middle attacks. However any other protocol or encryption algorithm will be discussed in the Design Document.

3.5.4 Maintainability

The System will follow good software engineering practices to allow maintainability, for example it is possible to use a local Database in order to allow the maintenance of the global database server. However, this will be discussed in the Design Document.

3.5.5 Portability

SafeStreets is developed as a Web Application, it is sufficient to have a mobile phone that allows the use of web browsers. Thus, *SafeStreets* has a maximum grade of portability.

4 Formal Analysis using Alloy

4.1 What we want to model?

Using Alloy we want to show possible scenarios in which the system can be and we also want to test its correctness. In particular we are interested in show the following things:

- User send Reports
- Authority access the Report by the ReportManager
- Authority can generate TrafficTickets
- TrafficTicket refer to a Report
- Report is about a violation made in a Location of a Municipality
- Authority can generate TrafficTickets only for violations made in the same Municipality they belong to.
- SuggestionsManager take data form ReportManager and form Municipality to cross them and create suggestions. Both User and Authority can access SuggestiosManager to read suggestions about possible improvements

4.2 Alloy Model

```
open util/integer

-- MODEL SIGNATURES --

sig Report{
  reportID: one Int,
    location: one Location
}

abstract sig Guest {
  sm: one SuggestionsManager
}

sig User extends Guest{
  userID: one Int,
  reports: set Report
```

```

}

sig Authority extends Guest{
    authID: one Int,
    rm: one ReportManager,
    trafficTickets: set TrafficTicket,
    municipality: one Municipality
}

one sig ReportManager{
    reports: set Report
}

sig TrafficTicket{
    ticketID: one Int,
    report: one Report,
    municipality: one Municipality
}

sig Location{
    municipality: one Municipality
}

sig Municipality{}

one sig SuggestionsManager{
    rm: one ReportManager,
    municipalities: set Municipality
}

-- FACTS THAT DEFINE THE MODEL --

-- define unique key for Authority --
fact uniqueAuthorityID{
    no disj a1, a2: Authority | a1.authID = a2.authID
}

-- define unique key for Report --
fact uniqueReportID{
    no disj r1, r2: Report | r1.reportID = r2.reportID
}

```

```

-- define unique key for User --
fact uniqueUserID{
    no disj u1, u2: User | u1.userID = u2.userID
}

-- define unique key for TrafficTicket --
fact uniqueTrafficTicketID{
    no disj tt1, tt2: TrafficTicket | tt1.ticketID = tt2.ticketID
}

fact ownReport{
    -- all report are generated by only one user --
    all r: Report |
        no disj u1, u2: User |
            r in u1.reports and r in u2.reports

    -- all report are generated by someone --
    all r: Report |
        one u: User |
            r in u.reports
}

fact ownTrafficTicket{
    -- all traffic tickets are generated by only one authority --
    all tt: TrafficTicket |
        no disj a1, a2: Authority |
            tt in a1.trafficTickets and tt in a2.trafficTickets

    -- all traffic tickets are generated by someone --
    all tt: TrafficTicket |
        one a: Authority |
            tt in a.trafficTickets
}

-- Report can't stay into different TrafficTicket --
fact trafficTicketReport{
    all r: Report |
        no disj tt1, tt2: TrafficTicket | r = tt1.report and
            r = tt2.report
}

```



```

-- all the Reports are managed by the ReportManager
fact manageOnce{
    all r: Report |
        one rm: ReportManager |
            r in rm.reports
}

/* Authority can generate TrafficTicket only for violations made
   into the same municipality in which it is registered */
fact checkAuthorityZone{
    all tt: TrafficTicket |
        one r: Report |
            r = tt.report and tt.municipality =
                r.location.municipality

    all tt: TrafficTicket |
        one a: Authority |
            tt in a.trafficTickets and a.municipality =
                tt.municipality
}

-- SuggetionsManager must see all the Municipalities in order to collect data --
fact SugManagerSeeAllMunicipalities {
    all m: Municipality |
        one sm: SuggestionsManager |
            m in sm.municipalities
}

```

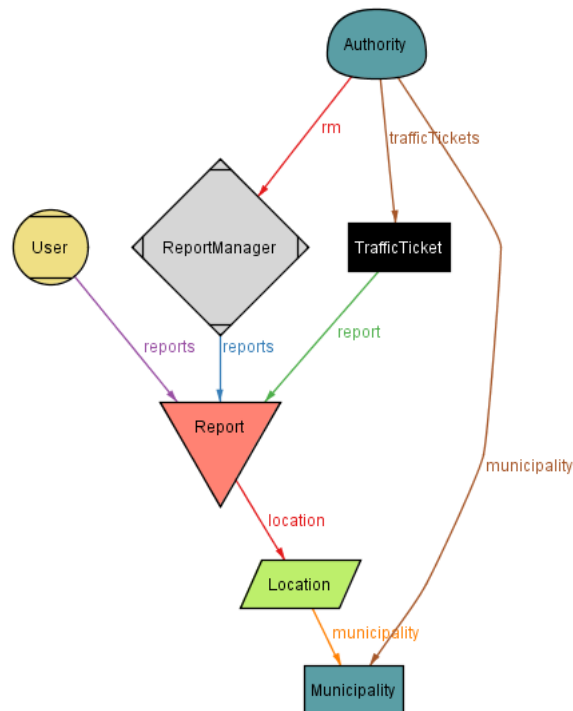
4.3 Alloy generated Worlds

- *Simple case*: Here we can see the case in which a User has sent a Report, an Authority has verified it and a TrafficTicket has been generated.

To obtain this world representation the line of Alloy code about the SuggestionsManager are not executed, while the code for the execution is the following:

```
pred show{
  #User = 1
  #Authority = 1
  #TrafficTicket = 1
  #Report = 1
}

run show for 3
```

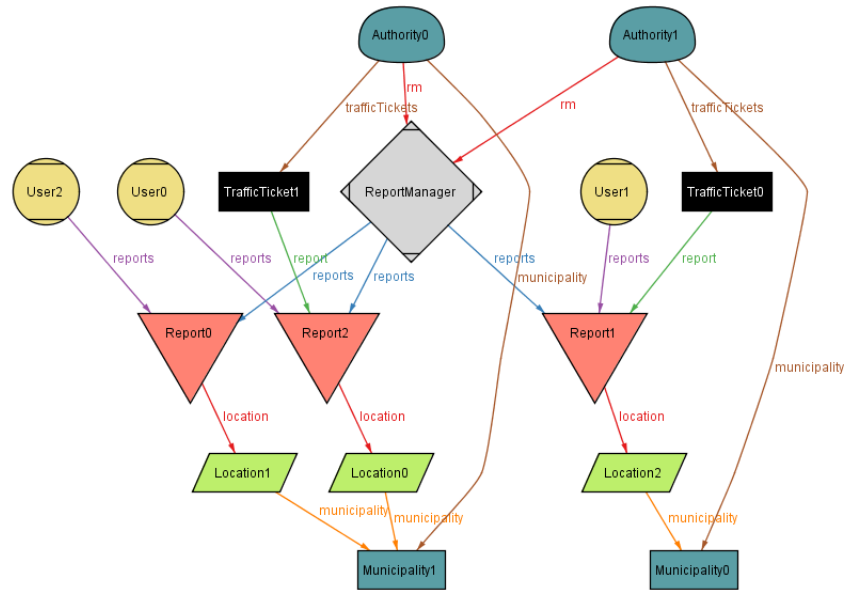


- **More Entities:** In this case we can see the same things as before but in a more complex situation because there are more Entities. Here we can also notice that the Authority can generate a TrafficTicket only to Reports made in the Municipality he belongs to. There also is a Report (Report0) in which there is no association we a TrafficTicket. This means that it can be not checked yet or it is verified but the TrafficTicket will be generated after.

Here again the code about SuggestionsManager is not executed and the execution code is:

```
pred show{
  #User = 3
  #Authority = 2
  #TrafficTicket = 2
}

run show for 3
```



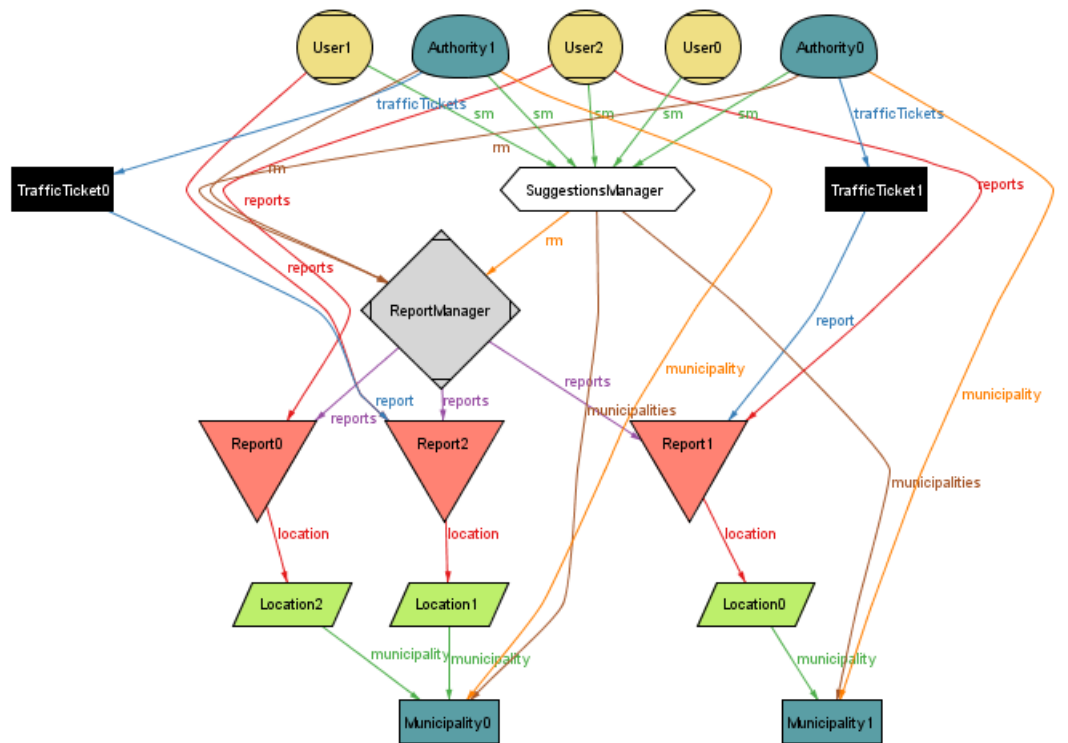
- **Allows Suggestions:** This World represent the complete scenario. There is what we have discuss before and the insertion of a SuggestionsManager which collect data from Municipalities and Reports (via ReportManager) and cross them to create Suggestion on how to improve urban mobility. Both Users and Authorities can access it to read

them. Even if a User never send a Report can look at the Suggestions.

In this last case the code about SuggestionsManager is executed and to get the model representation the code is:

```
pred show{
  #User = 3
  #Authority = 2
  #TrafficTicket = 2
}

run show for 3
```



5 Effort Spent

The following tables summarize the effort spent by each member of the team to create the RASD document.

5.1 Marri Iacopo

Description of the task	Hours
First meeting + Github setup	4
Purpose, Scope and Definition	5
Product Perspective	5
Product Functions	3
User Characteristics and Assumptions, dependencies and constrains	2.5
Specific Requirement	8
Alloy	4

5.2 Salamino Manuel

Description of the task	Hours
First meeting + Github setup	4
Purpose, Scope and Definition	6
Product Perspective	8
Product Functions	2
User Characteristics and Assumptions, dependencies and constrains	2
Specific Requirement	4
Alloy	10

5.3 Salazar Molina Steven Alexander

Description of the task	Hours
First meeting + Github setup	4
Purpose, Scope and Definition	5.5
Product Perspective	5
Product Functions	4
User Characteristics and Assumptions, dependencies and constrains	4.5
Specific Requirement	6
Alloy	5