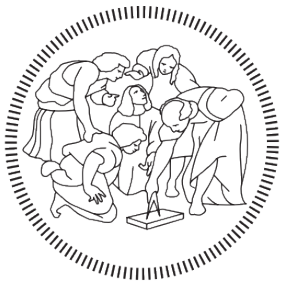


POLITECNICO
MILANO 1863

SOFTWARE ENGINEERING II PROJECT

SafeStreets

RASD – Requirements Analysis and Specifications Document

Authors:

Matteo POZZI
Sara SACCO
Andrea VENTURA

Professor:

Matteo G. ROSSI

December 9, 2019

version 2.0

Index

1. Introduction

1.1 Purpose

SafeStreets is a mobile application that relies on the help of lawful citizens to make life in the streets less stressful and more organized. The purpose of this document is to describe in depth said application in terms of functional and nonfunctional requirements, so as to help the customer and the developer be on the same page by identifying the former's needs, and documenting these in a way that makes analysis, communication, and implementation sustainable for both parties.

1.2 Scope

The given problem is to create a software system that meets the stakeholders' needs, which translate with the intent of providing people with the ability to report and notify violations, e.g. vehicles parked in the middle of bike lanes, or in spots reserved to people with disabilities, to the designated authorities.

In particular, citizens should be able to register as users by providing meaningful credentials, so as to avoid wasteful data such as fake accounts, and a way to verify them, e.g. their fiscal code. Once successfully logged in, users should be able to send pictures as proof of vehicles parked illegally and attach additional information to provide authorities with a starting point for the reviewing process, such as the date, the time, the type of violation which is to be reported and the place in which it has occurred, which can be retrieved through the geographical position of the user itself.

SafeStreets stores the information provided by its users and employs it by identifying and highlighting the zones which are found to be subject to the highest amount of violations, making them visible to both authorities and citizens.

Furthermore, SafeStreets wants to exploit its own data by combining it with information about accidents and analyzing it in order to identify zones or streets whose safety could be improved by making interventions, possibly suggesting viable solutions as well. This functionality is developed in collaboration with a third party, i.e. the municipality, meaning its usefulness will depend on the possibility of the

municipality itself to share its data and match it with the interface SafeStreets developed for the functionality.

Lastly, SafeStreets strives to assist the local police in generating traffic tickets, and possibly build various statistics of interest. To ensure the effectiveness of this service, it is necessary that the exchange of sensible data which must occur between SafeStreets and the municipality cannot be tampered with in any way, e.g. modifying the picture of the violation at hand. To avoid this scenario, SafeStreets only accepts as reliable information pictures that have been taken within the application itself.

In the following diagram (Figure 1.1), we define the boundaries of SafeStreets by identifying and distinguishing between World and Machine phenomena, with particular attention to the shared ones.

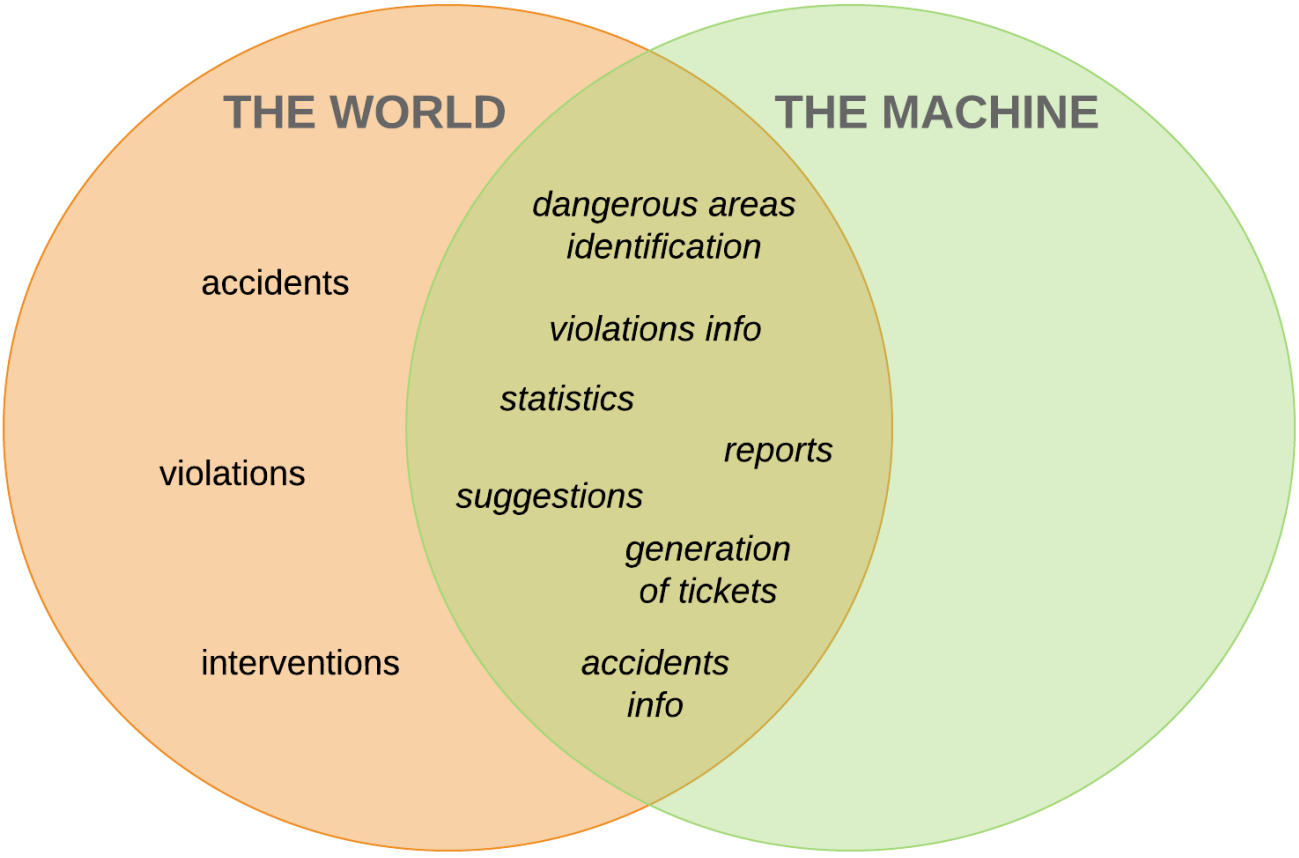


Figure 1.1: World and Machine phenomena.

1.2.1 Goals

[**G1**] Allow Citizens to report traffic violations;

[**G2**] Allow Citizens to view a history of their past reports;

[**G3**] Allow Authorities to evaluate submitted reports and Citizens to visualize accepted ones with limited information;

[**G4**] Allow Authorities to send reports about accidents;

[**G5**] Allow Users to view areas with most violations;

[**G6**] Allow Users to view areas that are marked as unsafe by a System Manager;

[**G7**] Allow the local Authority to generate traffic tickets;

[**G8**] Allow an Authority to link issued traffic tickets to relative Citizens reports;

[**G9**] Allow Users to visualize statistics about issued traffic tickets on violations in a certain area;

[**G10**] Allow System Managers to suggest possible interventions for areas that are deemed unsafe and allow Users to visualize them.

1.3 Definitions, acronyms, abbreviations

- Definitions:

User: a general actor which is registered into the application; all users can consult statistics about violations and highlight unsafe areas;

Authority: a user which receives complaints and is able to identify actual violations among them. It has the power to punish the culprits with traffic tickets;

Citizen: a user which is not an authority, he can send reports about violations;

Violation: a violation of traffic laws, in particular parking violations;

Accident: a traffic event involving two or more vehicles where people got injured or caused damages to the vehicles

Report: a notification sent by a citizen to indicate violations, containing all the meaningful information about it;

Traffic ticket: a sanction which force an offender of a violation to pay an amount of money, can be generated by authorities;

Unsafe area: an area in which many violations and accidents have been reported;

Statistics: a collection of data about issued traffic tickets for each kind of violation occurred in a certain area.

Suggested intervention: a suggestion made by a system manager to be possibly applied in order to avoid future violations of a certain type.

- Acronyms:

RASD: Requirements Analysis and Specifications Document;

GPS: Global Positioning System;

SI: International System of Units;

API: Application Programming Interface.

- Abbreviations:

[**Gn**]: n-th goal;

[**Dn**]: n-th domain property;

[**Rn**]: n-th requirement.

1.4 Revision history

- **Version 1.0 –** November 10, 2019
  - First Release.
- **Version 2.0 –** December 9, 2019
  - Fixed sequence diagrams;
  - Removed incorrect user interface mock up (Figure 3.11);
  - Fixed error in requirement R9;
  - Removed requirement R25 because equal to R17;
  - Added missing requirements;
  - Fixed error in scope;
  - Fixed class diagram;

- Fixed description of scenario for an Authority reporting an accident;
- Added missing user interfaces.

1.5 Reference documents

- Specification Document: “SafeStreets Mandatory Project Assignment”
- “Software Abstractions” – Daniel Jackson
- Alloy documentation: http://alloytools.org/documentation.html
- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

1.6 Document structure

The document at hand is composed of 4 chapters, plus an appendix:

1. Introduction: it includes the goal of the project and an analysis of the world and shared phenomena;
2. Overall description: here we provide further details on the shared phenomena, as well as user characteristics and domain assumptions;
3. Specific requirements: this section provides more details on the aspects presented in Chapter 2;
4. Formal analysis using alloy: it includes a brief presentation of the main objectives of the formal modeling activity, and a description of the model itself;
5. Appendix: an accessory part that contains a quantitative description of the effort each member put into the completion of the document.

2. Overall description

2.1 Product perspective

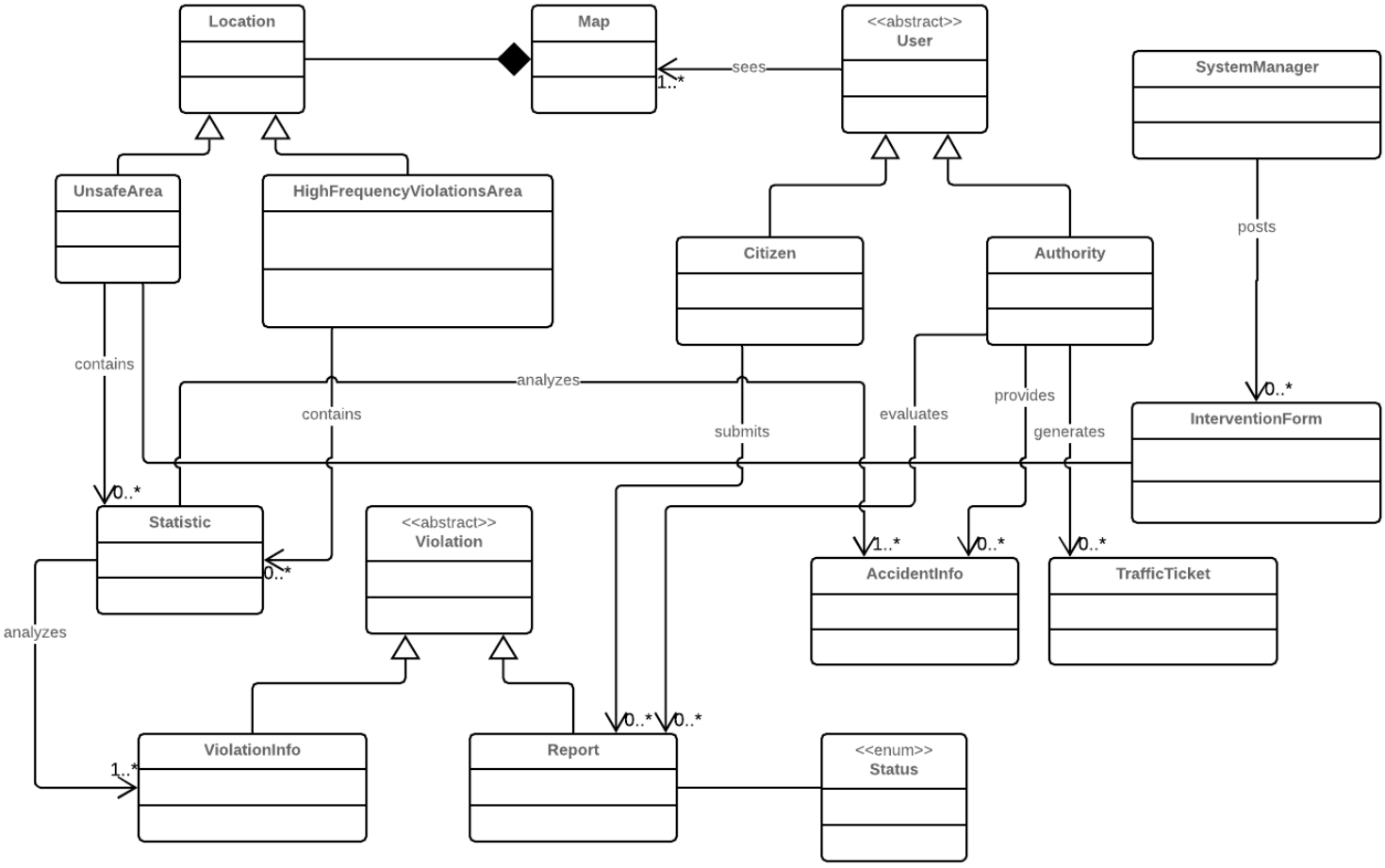


Figure 2.1: UML class diagram.

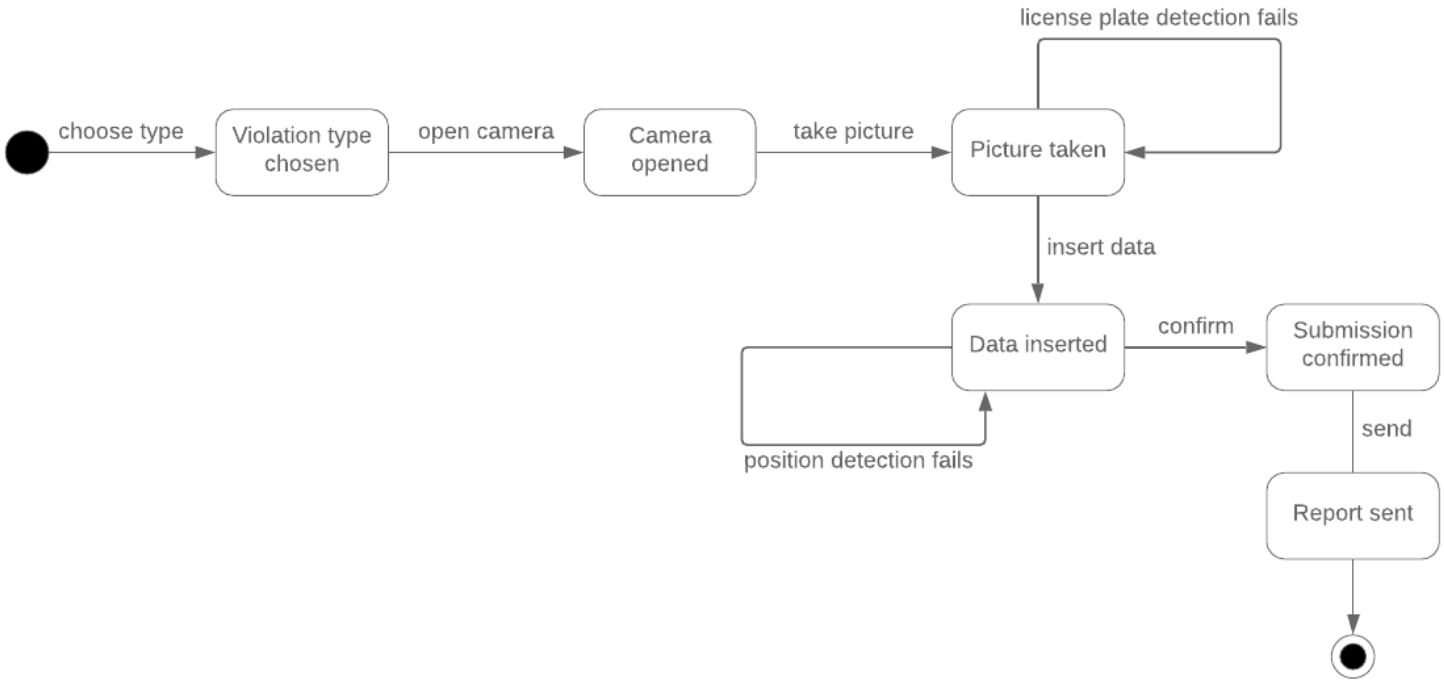


Figure 2.2: State diagram 1: Report filling.

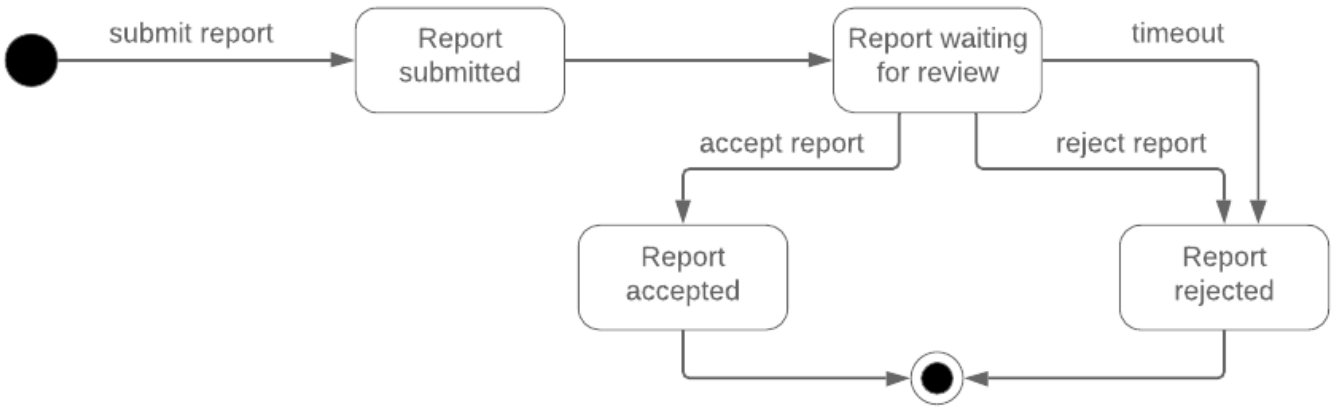


Figure 2.3: State diagram 2: Report status.

2.2 Product functions

As previously mentioned, the product offers several functions, which are described more in depth hereunder.

2.2.1 Report management

This function is the core of the system, as it provides the data needed to carry out the others listed down below. The system allows citizens to take pictures of vehicles committing traffic violations, and asks them to insert relevant information, i.e. position and type of violation, and stores the submitted reports. These can then be evaluated by authorities, who are given the option of accepting or refusing the report at hand as valid.

2.2.2 Ticket generation

As stated above, this function is built on top of the former, and it is meant for only one of the parties involved in the application, that is the authorities. As a matter of fact, the system will ask users registered as *Authority* whether or not they want to generate traffic tickets concurrently with the acceptance of a submitted report. The process of handling ticket requests is outside of the scope of the application. In particular, the municipality will be in charge of deciding whether or not the ticket actually needs to be issued, how to send it and the amount to pay.

2.2.3 Areas identification

The function at hand is meant for all end users. In this case, the system will elaborate accepted reports, so as to not consider possibly misleading information such as violations that have not actually occurred, and use them to highlight areas with a high concentration of traffic violations. Furthermore, SafeStreets will cross its data with accidents reported by authorities to identify possibly unsafe areas, which may be identified both automatically and with the input of a

system manager. Users will then be able to exploit such data by viewing the highlighted areas on the system's map.

2.3 User characteristics

The actors interacting with our system are *Citizens*, *Authorities* and *SystemManagers*.

A *Citizen* is a user who has successfully logged into the application using either their mobile phone or a web browser, and is able to access the Internet. They can submit reports via the mobile application, as well as identify areas of interest, or simply observe elaborated data, such as statistics.

An *Authority* is also a user who has successfully logged into the application using either their mobile phone or a web browser, and is able to access the Internet. As opposed to Citizens, Authorities are employees of the municipality, and as such they receive and review submitted reports by marking them as either accepted or refused. They also have the power to punish the culprits of traffic violations with traffic tickets.

A *SystemManager* is an actor who has access to the back end of the application through an internal interface and can provide additional information about possible interventions for unsafe areas.

2.4 Domain assumptions

[**D1**] Each User is unique;

[**D2**] Users that register under the Authorities category are employees of the municipality;

[**D3**] A Citizen sends a report about a violation when he notices it;

[**D4**] Information about date and time of the violation corresponds to the date and time when the report is sent;

[**D5**] Information about position is collected through GPS;

[**D6**] Picture of violations are taken at the moment and are not inserted in a second time or from already saved pictures;

[**D7**] Each Citizen reports a certain violation once;

[**D8**] Authorities have tools for assessing if a violation included in a report is an actual violation or not;

[**D9**] Authorities generate traffic tickets only for actual violations;

[**D10**] Authorities are able to find the owner of the vehicle by the license plate, which is unique to each car vehicle;

[**D11**] Authorities report accidents that really occurred;

[**D12**] Authorities report accidents occurred in their territory;

[**D13**] Suggested interventions are always meaningful and reliable and may avoid future violations;

3. Specific requirements

3.1 User interfaces

In order to represent the interactions between the system and the customers, we focused on simplicity and tried to design these interfaces ensuring that they are as intuitive as possible. This is a key aspect, as our application is thought to be used also by older people.

3.1.1 Mobile App: Citizens

The main feature of the Citizen mobile app is reporting a violation: it's important to stress that Citizens can do it only from the mobile app.
For obvious reasons, it would be a bit pointless to allow Citizens to report a violation through the web app, as that would mean they're probably in an enclosed space and therefore not in the presence of a traffic violation.

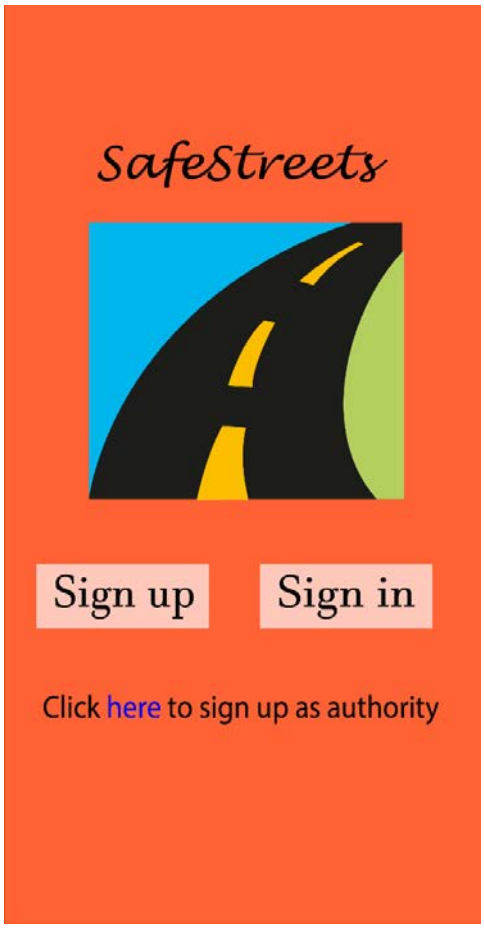


Figure 3.1.

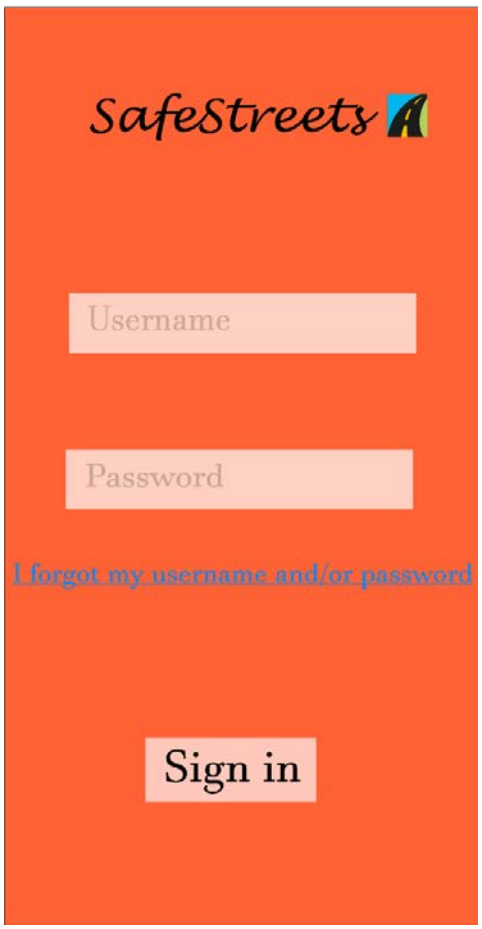


Figure 3.2.

A guest can only perform two actions: sign up, if he hasn't yet, or sign in (Figure 3.1). In this case, the guest is required to enter their own username and password (Figure 3.2).

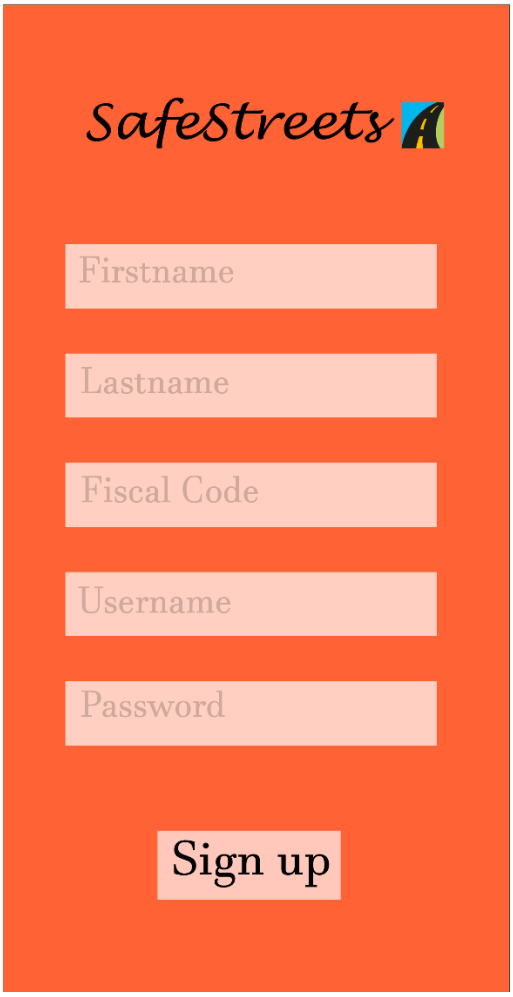


Figure 3.3.

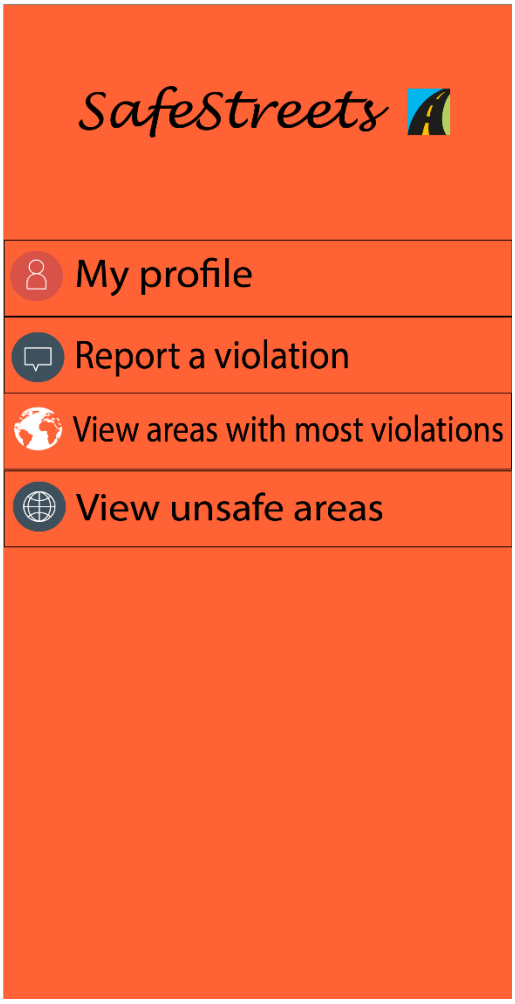


Figure 3.4.

A Citizen who wants to sign up has to insert their own data: first name, last name and fiscal code; then they have to choose an username and a password for their account (Figure 3.3). A registered user can access the menu and in particular they can visit their profile, report a violation, view areas with most violations and finally view unsafe areas (Figure 3.4).

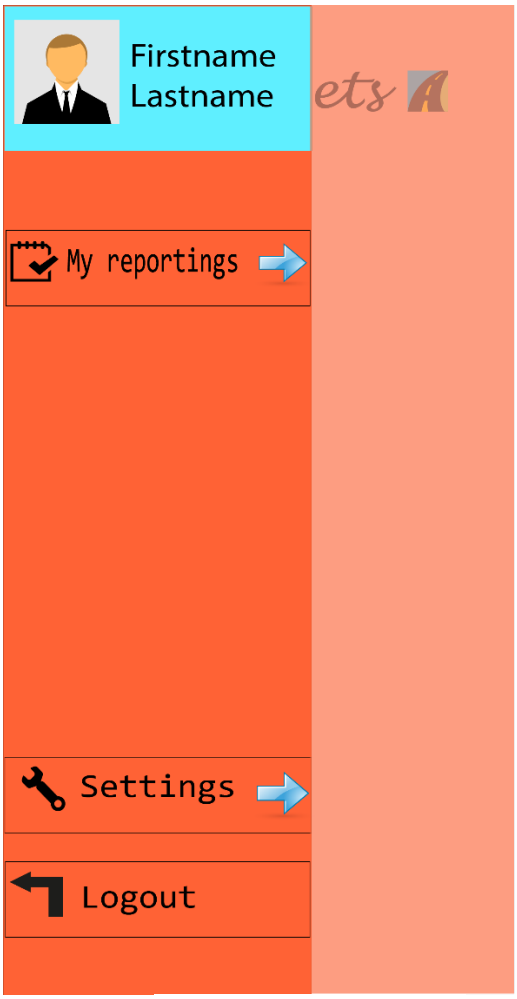


Figure 3.5.

Figure 3.6.

The first picture (Figure 3.5) shows how an user menu should look like. The user can access their own report history and check if their reports have been accepted, rejected or if they haven't been reviewed yet (Figure 3.6).

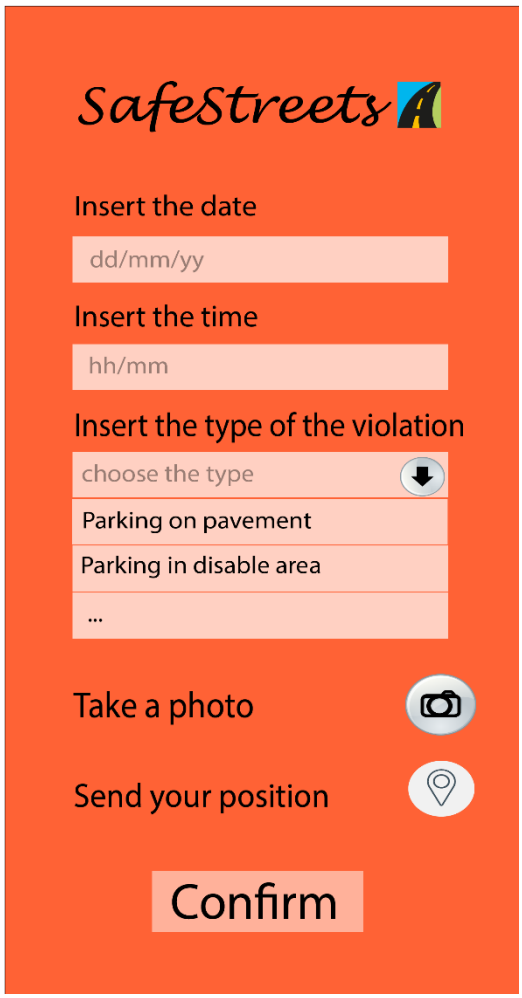


Figure 3.7.

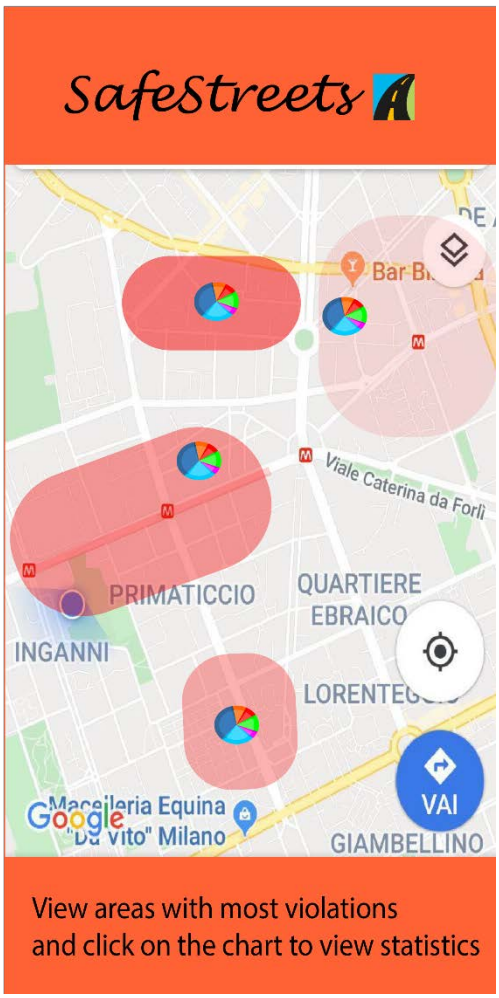


Figure 3.8.

If an user wants to signal a violation, they have to choose the type of violation among those proposed as well as take a photo in which the license plate is clearly visible. The date, time and current position are taken directly from the system of the device being used (Figure 3.7).
Concerning the areas with most violations, the user is able to visualize them on the map as highlighted areas, as shown in Figure 3.8.

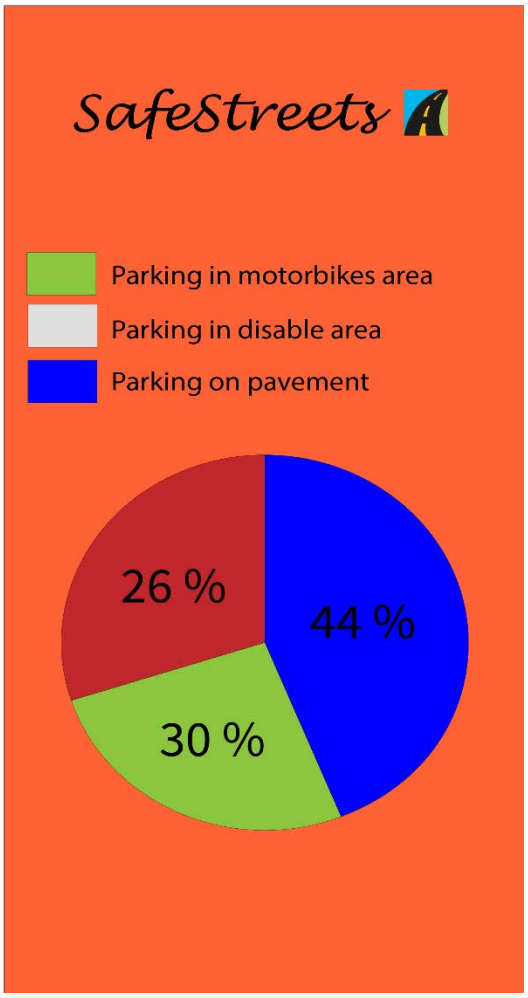


Figure 3.9.

Figure 3.10.

In the first picture we can observe a pie chart which gives information about violations related to a certain zone (Figure 3.9).
It's essential to underline that violations in the graph are based on generated tickets and not on all reports made by citizens.
We can also see how areas with most accidents are represented in the map.
As for areas with many violations, unsafe areas are highlighted as shown in Figure 3.10.

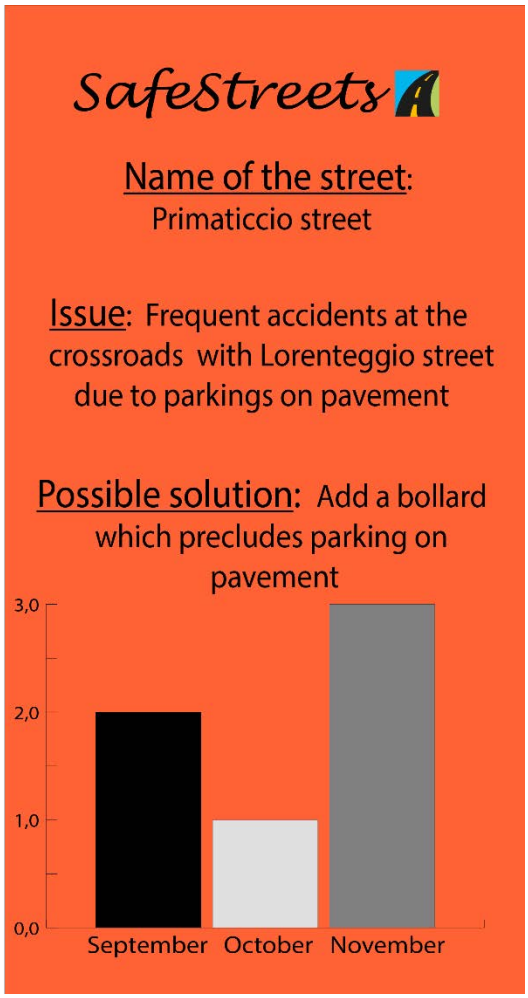


Figure 3.11.

This interface relates to the suggestions made by SafeStreets, crossing its own data with information about accidents. The info icon (i) shown in Figure 3.10 takes the user to a page containing the issue about the involved area and a possible solution. In addition, it shows an histogram with the number of accidents concerning the chosen area in the last few months (Figure 3.11).

3.1.1 Mobile app: Authorities

Municipalities who want to sign up have to do it as Authority and they have to provide information about their city and their zone; each municipality has a private account that is accessible to Authorities who work for the municipality itself. However, big cities which have different municipalities, one for each zone, can have several accounts. Furthermore, Authorities have their own personal interfaces, even if they can access the same data available for Citizens (such as areas with most violations and statistics). Through the mobile app, an Authority can see a list of incoming reports and their specific information, but it has been decided that they cannot generate tickets (or reject reports) and notify accidents: these actions are allowed only by the web app. Some interfaces for Authorities are the same as those for Citizens: we report the most relevant below.

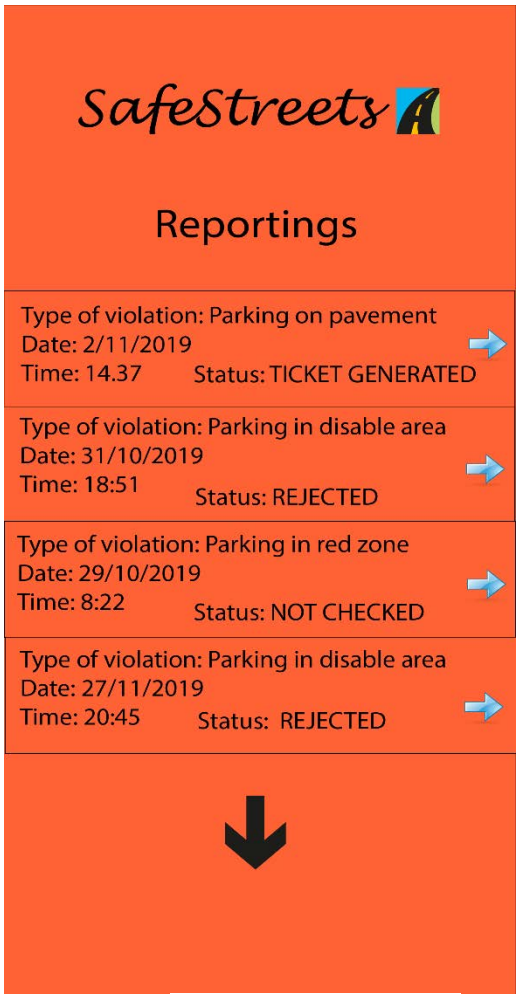


Figure 3.12.

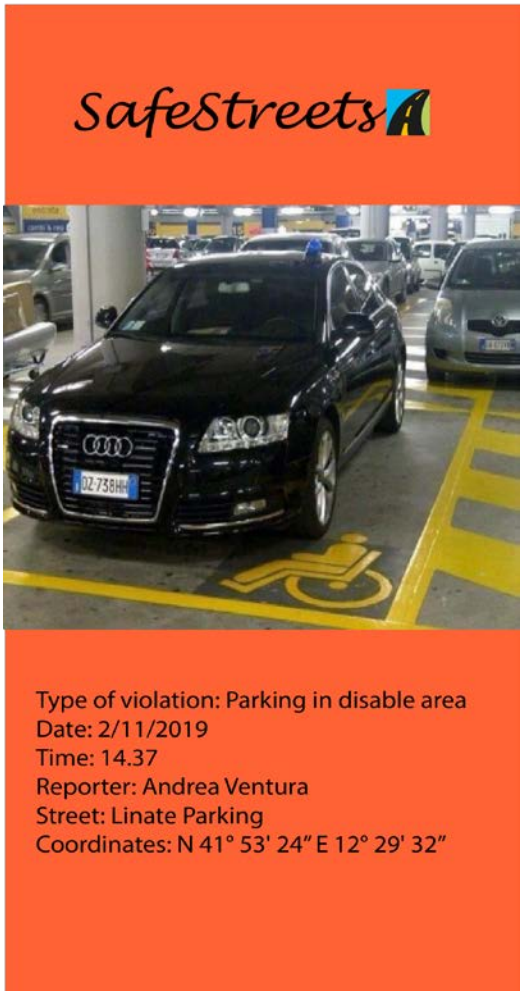


Figure 3.13.

Authorities have the list of the violations reported by Citizens concerning the area they're responsible for and they can check if a past report has been reviewed or not (Figure 3.12). Moreover, they can access sensitive information about a violation: as explained before, they aren't able to generate a ticket by using the mobile app (Figure 3.13).

3.1.2 Web app: Citizens

Although Citizens are able to exploit all the application features from the mobile app, we also report an example of web interface, because consulting charts and statistics through a web browser and therefore a possibly larger screen could be more comfortable.

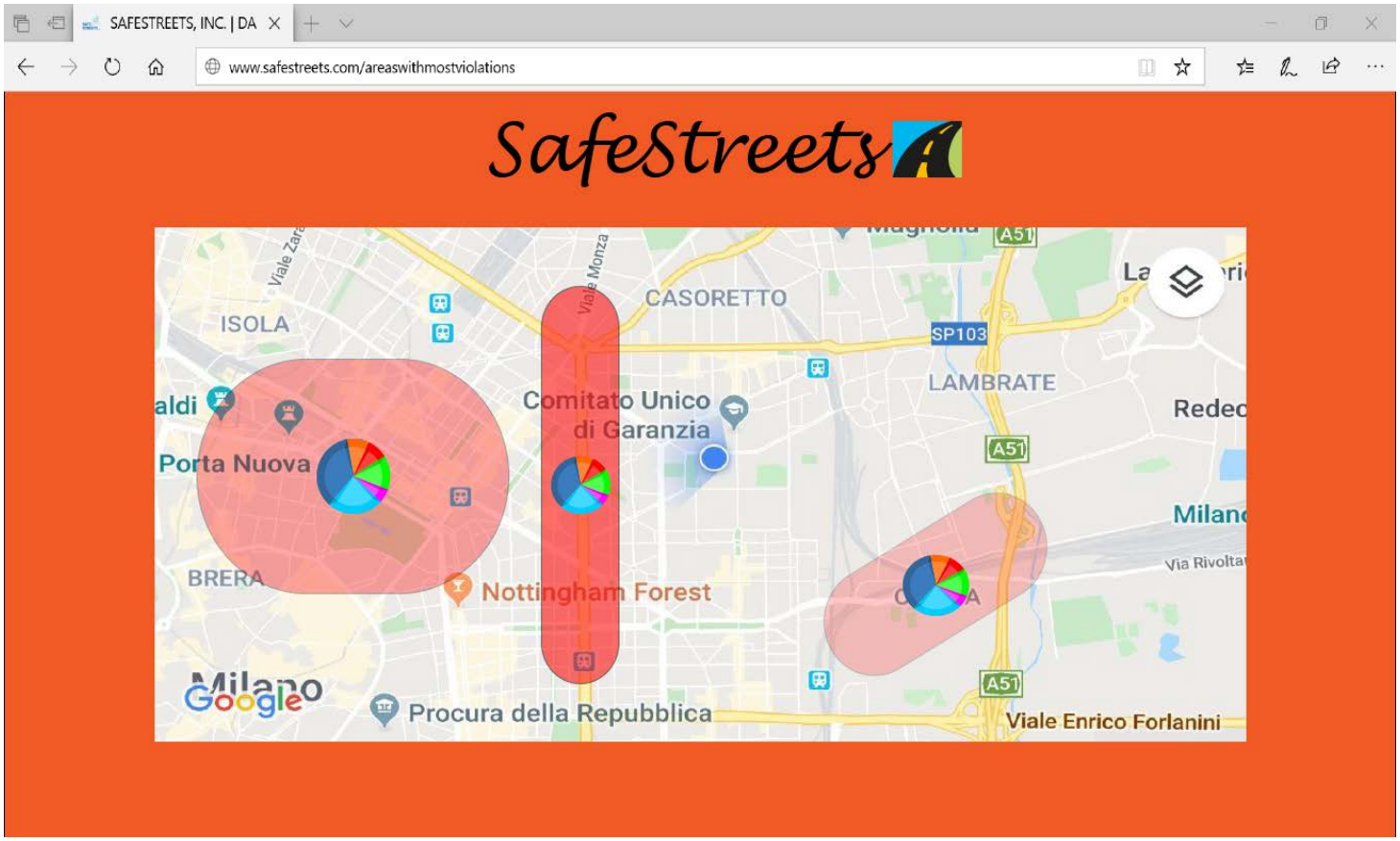


Figure 3.15.

This is an example of how our web application should show areas with most violations (Figure 3.15).

3.1.3 Web app: Authorities

The web app for Authorities provides all the functions guaranteed by the mobile app. In addition, it has two more essential features: as a matter of fact, it allows ticket generation (or reports evaluation in general) and the ability to report an accident.

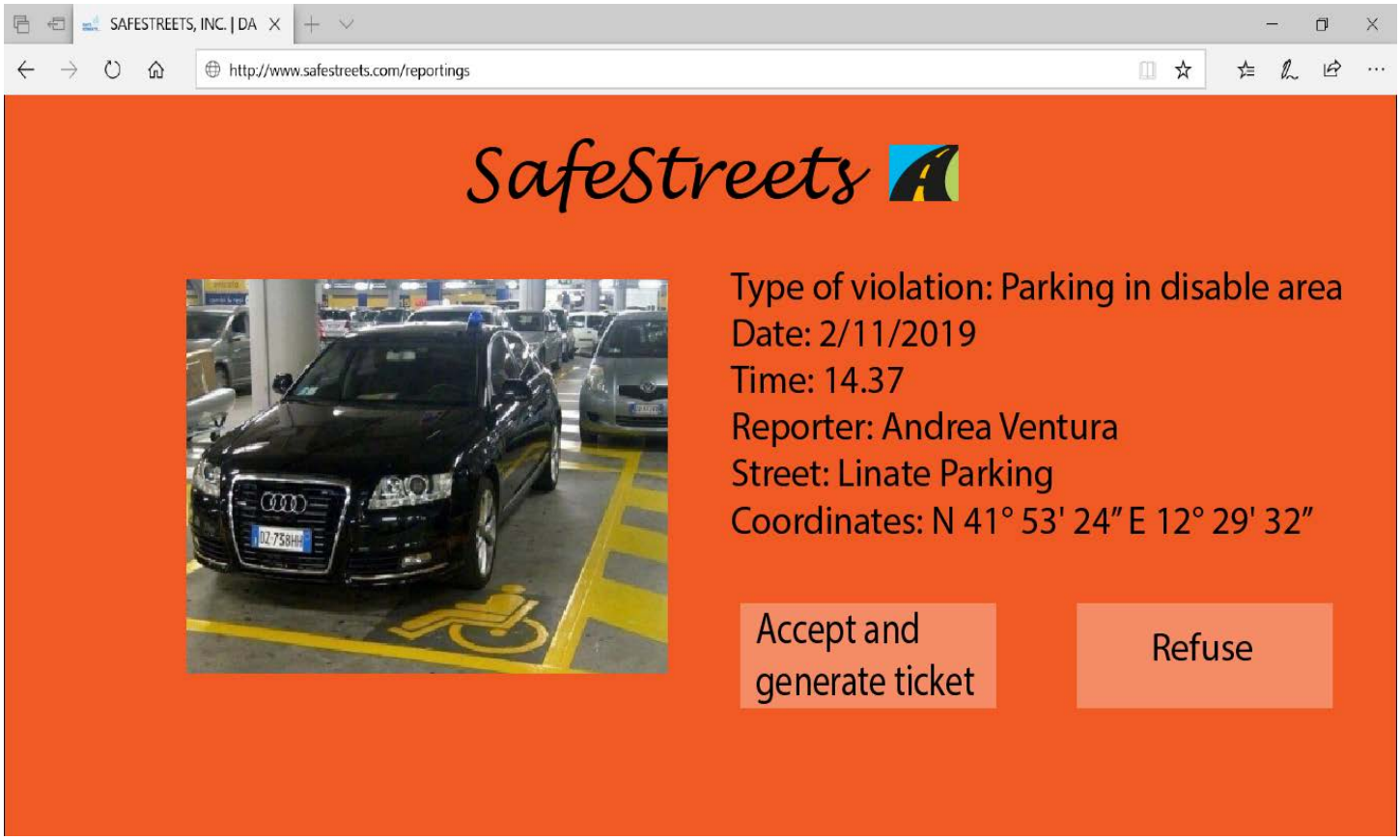


Figure 3.16.

This interface shows that an authority through the web app can accept a request and generate a ticket or he can reject a report (Figure 3.16).

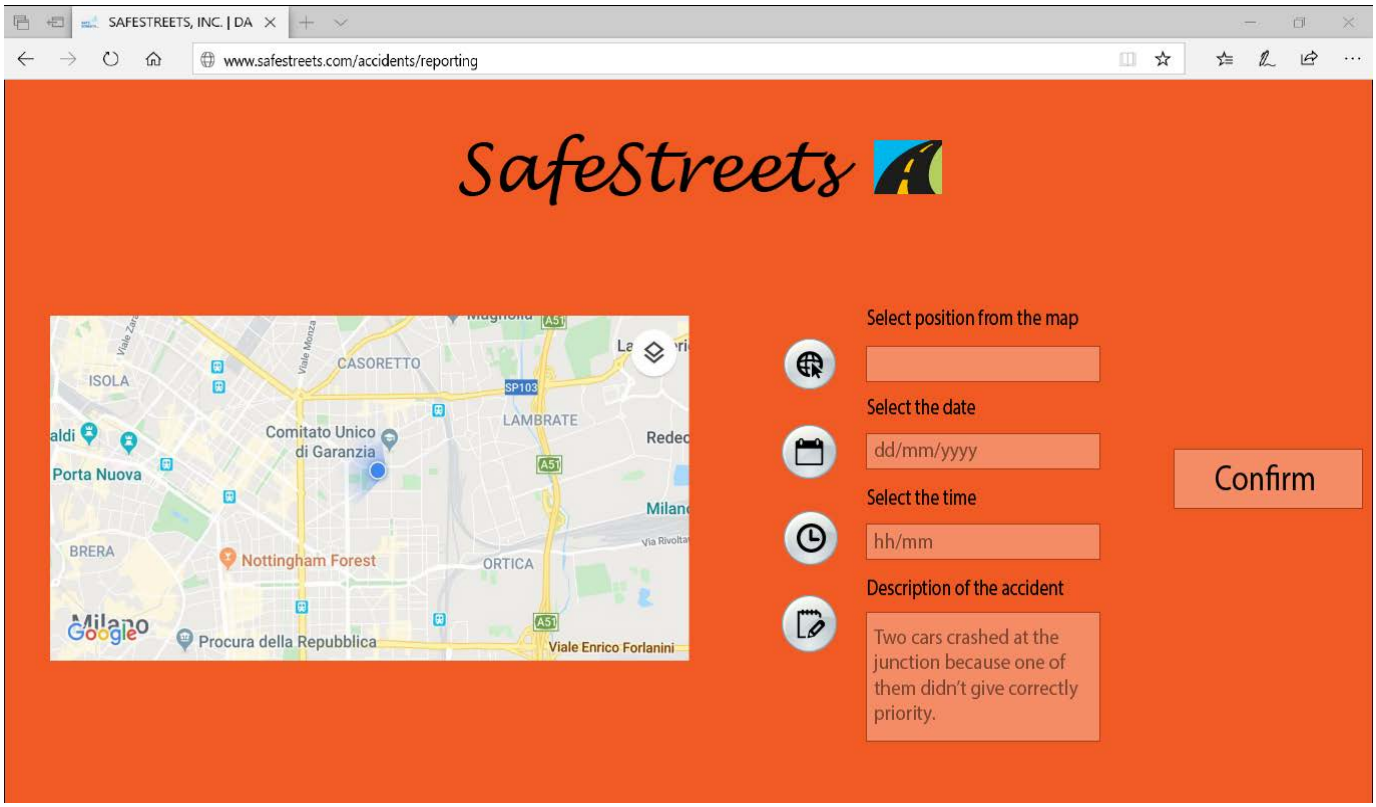


Figure 3.17.

To report an accident, an Authority has to insert the position, the date, the time and a brief description. These reports are useful to build statistics about the number of accidents in a certain area and to identify unsafe areas (Figure 3.17).

3.2 Software interfaces

The system uses map information taken from external services (e.g. Google Maps API), as well as data provided from a third party, the municipality, and in return sends information about traffic tickets generated directly through the application. The exchange must be thus enabled by an interface through which the municipality and the system can communicate in both directions, in addition to the one handling maps.

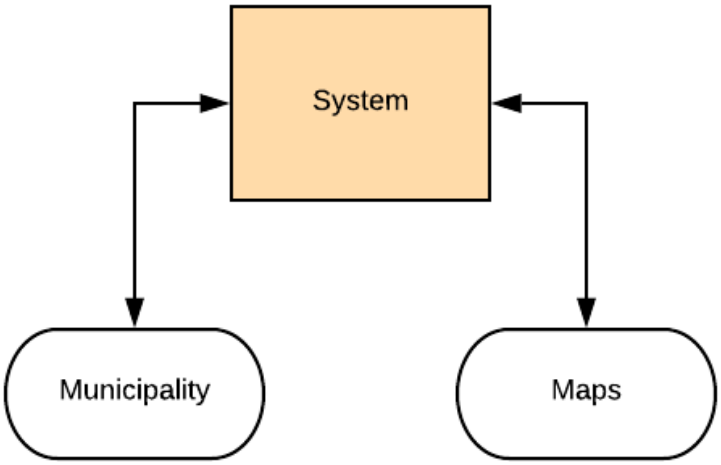


Figure 3.18: Software interfaces.

3.3 Functional requirements

[**G1**] **Allow Citizens to report traffic violations;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;

[R2] The system must be able to authenticate registered users;

[R3] The citizen must be able to insert relevant data about violations, such as type of violation, during the filling out process;

[R4] The system must be able to analyze pictures that are being submitted in a report and recognize whether the license plate is readable/present or not;

[R5] The system must inform the user whether their report has been stored successfully or not;

[R6] The system must ask users if they want to retry the submission process using the same data that failed being sent, or if they want to cancel it;

[*D1*] Each User is unique;

[*D3*] A Citizen sends a report about a violation when he notices it;

[*D4*] Information about date and time of the violation corresponds to the date and time when the report is sent;

[*D5*] Information about position is collected through GPS;

[*D6*] Picture of violations are taken at the moment and are not inserted in a second time or from already saved pictures;

[*D7*] Each Citizen reports a certain violation once;

[**G2**] **Allow Citizens to view a history of their past reports;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;

[R2] The system must be able to authenticate registered users;

[R7] The system must be able to distinguish every user unambiguously;

[R8] The system must store information with an association to the user who submitted it;

[R9] The system must be able to retrieve stored information;

[R10] The system must allow reports to have only one status at a time (accepted, rejected, to be checked);

[R11] The system must tell the user whether their report has been accepted, rejected or is still waiting to be checked;

[*D1*] Each User is unique;

[**G3**] **Allow Authorities to evaluate submitted reports and Citizens to visualize accepted ones with limited information;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R9] The system must be able to retrieve stored information;
[R12] The system must be able to distinguish between authorities and citizens;
[R13] The system must be able to distinguish between submitted reports and reports that have been reviewed and accepted by the police;
[R14] The system must anonymize data shown to regular users (citizens), that is hide information about the vehicles that were parked illegally and about who submitted a particular report; in other words, data about reports that is shown to users must only contain the type of violation, date, time and position;
[R15] The system must show the full data about a report to authorities;

[*D1*] Each User is unique;
[*D2*] Users that register under the Authorities category are employees of the municipality;
[*D8*] Authorities have tools for assessing if a violation included in a report is an actual violation or not;

[**G4**] **Allow Authorities to send reports about accidents;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R12] The system must be able to distinguish between authorities and citizens;
[R16] An authority must be able to submit reports about accidents;

[R17] An authority must be able to insert relevant information about the occurred accident, such as location and injured people;

[*D2*] Users that register under the Authorities category are employees of the municipality;
[*D11*] Authorities report accidents that really occurred;
[*D12*] Authorities report accidents occurred in their territory;

[**G5**] **Allow Users to view areas with most violations;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R9] The system must be able to retrieve stored information;
[R14] The system must be able to anonymize data shown to regular users (citizens), that is hide information about the vehicles that were parked illegally and about who submitted a particular report; in other words, data about reports that is shown to users must only contain the type of violation, date, time and position;
[R15] The system must show the full data about a report to authorities;
[R18] The system must be able to access map information;
[R19] The system must show the user their local map information;

[*D5*] Information about position is collected through GPS;

[**G6**] **Allow Users to view areas that are marked as unsafe by a System Manager;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R15] The system must be able to access map information;
[R18] The system must show the user their local map information;
[R20] The system must show the user possible solutions for unsafe areas, if there are any;
[R21] The system must allow system managers to edit the status of an area as unsafe and the other way around;

[*D5*] Information about position is collected through GPS;

[**G7**] **Allow the local Authority to generate traffic tickets from accepted reports;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R10] The system must allow reports to have only one status at a time (accepted, rejected, to be checked);
[R15] The system must show the full data about a report to authorities;
[R22] An authority must be able to generate a traffic ticket from a report;
[R23] The system must offer the possibility to generate a traffic ticket only to reports which has the status of accepted;

[*D2*] Users that register under the Authorities category are employees of the municipality;
[*D9*] Authorities generate traffic tickets only for actual violations;
[*D10*] Authorities are able to find the owner of the vehicle by the license plate, which is unique to each car vehicle;

[**G8**] **Allow an Authority to link issued traffic tickets to relative Citizens reports;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;
[R2] The system must be able to authenticate registered users;
[R7] The system must be able to distinguish every user unambiguously;
[R9] The system must be able to retrieve stored information;
[R22] An authority must be able to generate a traffic ticket from a report;
[R23] The system must offer the possibility to generate a traffic ticket only to reports which has the status of accepted;
[R24] The system must allow Authorities to know which Citizen sent each report;

[*D1*] Each User is unique;

[*D2*] Users that register under the Authorities category are employees of the municipality;

[*D7*] Each Citizen reports a certain violation once;

[**G9**] **Allow Users to visualize statistics about issued traffic tickets on violations in a certain area;**

[R1] The system must allow new users to sign up by providing personal/mandatory information;

[R2] The system must be able to authenticate registered users;

[R9] The system must be able to retrieve stored information;

[R18] The system must be able to access map information;

[R19] The system must show the user their local map information;

[R25] The system must be able to compute meaningful statistics on reports about each kind of violation in which a traffic ticket has been generated;

[R26] The system must make data about statistics visible to all users;

[*D7*] Each Citizen reports a certain violation once;

[**G10] Allow System Managers to suggest possible interventions for areas that are deemed unsafe and allow Users to visualize them.**

[R1] The system must allow new users to sign up by providing personal/mandatory information;

[R2] The system must be able to authenticate registered users;

[R9] The system must be able to retrieve stored information;

[R18] The system must be able to access map information;

[R19] The system must show the user their local map information;

[R20] The system must show the user possible solutions for unsafe areas, if there are any;

[R27] The system must allow system managers to suggest interventions for unsafe areas;

[R28] The system must make data about suggested interventions visible to all users;

[*D13*] Suggested interventions are always meaningful and reliable and may avoid future violations;

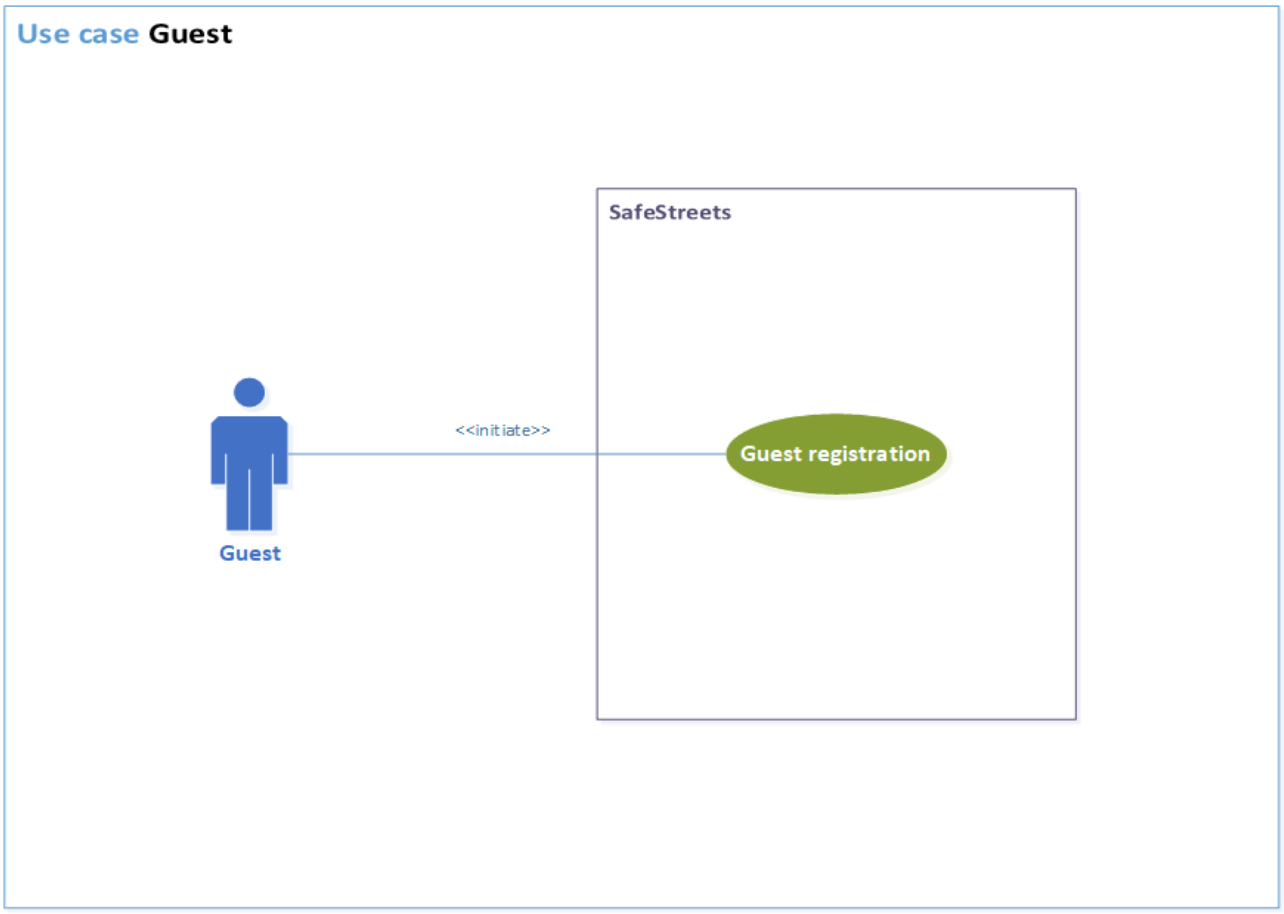
Traceability matrix

Here we present a table which links together all goals with the requirements and the use cases involved to satisfy it.
Please note that, because to perform any action on the system and satisfy the goals, the actor must be correctly registered and logged in, the use cases for registration and signing in for users and system managers are not reported, and requirements R1 and R2 are implied to stand for each use case and goal.

Goal ID	Involved Requirements	Involved Use Cases
[G1]	[R3] [R4] [R5] [R6]	Issue report
[G2]	[R7] [R8] [R9] [R10] [R11]	View past reports
[G3]	[R9] [R12] [R13] [R14] [R15]	View verified reports Evaluate submitted report
[G4]	[R12] [R15] [R17]	Report accident
[G5]	[R9] [R14] [R15] [R18] [R19]	Check areas with most violations
[G6]	[R18] [R19] [R20] [R21]	Check unsafe areas Report unsafe area
[G7]	[R10] [R15] [R22] [R23]	Generate traffic ticket
[G8]	[R7] [R9] [R22] [R23] [R24]	Issue report Generate traffic ticket
[G9]	[R9] [R18] [R19] [R25] [R26]	View statistics
[G10]	[R9] [R18] [R19] [R20] [R27] [R28]	Check unsafe areas View possible interventions Suggest intervention

3.4 UML models

3.4.1 Guest

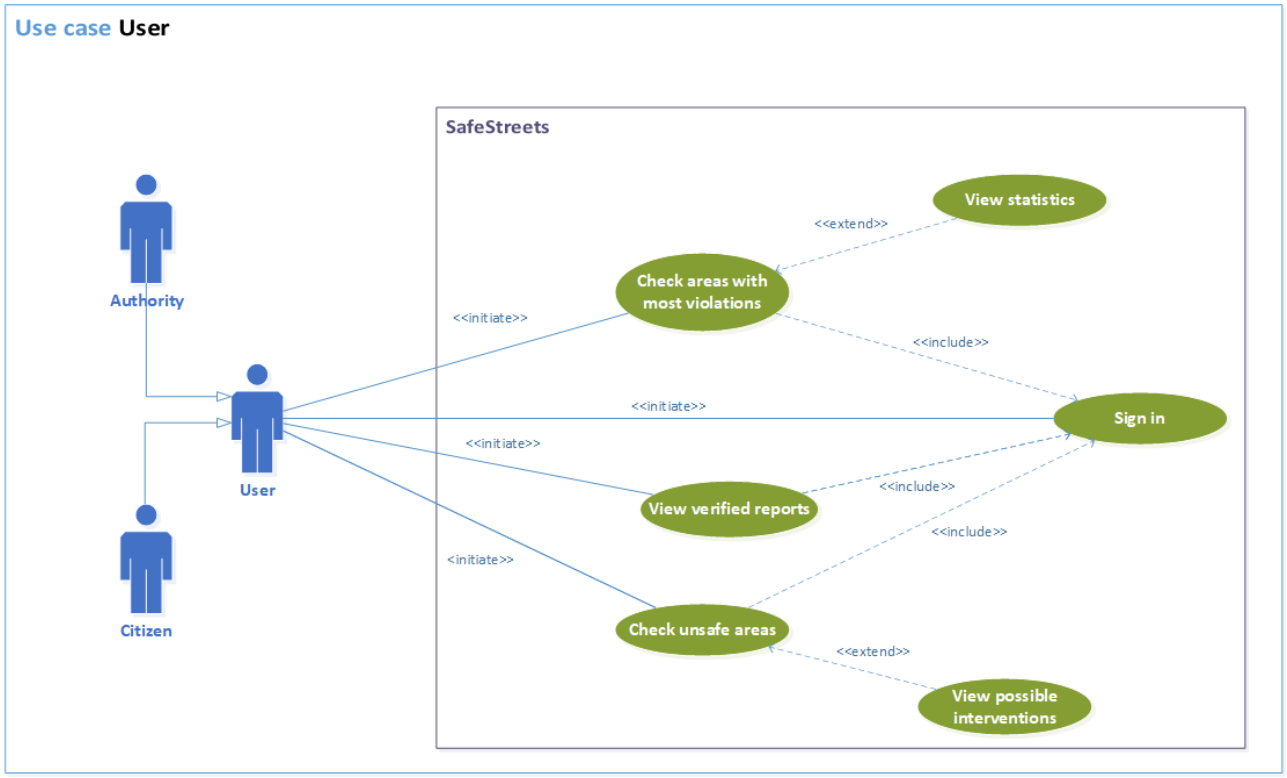


Scenario 1

Andrea works in Milan and, both because he really cares about climate issues and to avoid traffic congestions, he usually reach his workplace by bike. Indeed, finding a good spot to park his bicycle is not that easy, as there are often many cars parked very close to the bicycle parkings where Andrea is used to leave his bike, and he is getting frustrated about this. When he learns about SafeStreets, he decides to immediately download it on his smartphone and to register himself, providing his personal information and his address, with the intention to report each time when a car is parked too near a bicycle parking, hoping that he will soon be able to leave his bike safely near his workplace.

Name	Guest registration
Goals	[G1]…[G10]
Actors	Guest
Entry conditions	The guest has downloaded the application and launched it
Events flow	1. The guest chooses the “sign up” option. 2. The guest selects if he wants to register either as an Authority or a Citizen. 3. The guest fills all mandatory data concerning the chosen category. 4. The guest confirms the operation by selecting the confirmation option. 5. The system saves the data.
Exit conditions	The Guest has become an User and can now access the application function offered to the chosen category. The system has saved the data about the User
Exceptions	1. The Guest is already registered into the application. In this case the system invites him to execute the “sign in” operation. 2. One or more of the mandatory fills contain invalid input. In this case the system sends a warning to the Guest and invites him to correct them.

3.4.2 User



Name	Sign in
Goals	[G1]…[G10]
Actors	User
Entry conditions	The User is registered to the application and is on the home page.
Events flow	1. The User selects the “sign in” option. 2. The User inserts his credentials into the fields. 3. The User selects the confirmation option. 4. The system redirects the User to his personal home page.
Exit conditions	The system recognizes the User as registered and redirects him successfully.
Exceptions	1. The User inserts invalid credentials. In this case the system warns the User and invites him to re-insert them.

Scenario 2

Sara would like to join a friend of her for a snack at a well known cafè in their town lately this afternoon, and as she has been studying the whole day, she would like to reach the place on foot to stretch her legs and relax. But to reach the cafè, she has to cross a road known to be very busy, especially at that time of the day, without traffic lights but with a single crosswalk which might be dangerous as well because in the past years she sometimes noticed cars parked on it. Sara has recently registered on SafeStreets and she knows that statistics about violations are collected, so she decides to check whether in the last days someone has reported a violation of this kind in that street. Gladly, she notices that nobody has parked a car on the crosswalk for a very long period, so she will enjoy her walk and her brief moment of freedom.

Name	Check areas with most violations
Goals	[G5]
Actors	User
Entry conditions	The User is logged into the system and is on the home page.
Events flow	1. The User selects the “view areas with most violations” option. 2. The system provides the User with a map centered on the User’s current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. Eventually, the User specifies a filter about violations to look for. 5. The User consults the areas with the most specified violations.
Exit conditions	The system has provided the User with all the areas within the desired range from the selected location in which the specified violations have occurred the most.
Exceptions	1. The system couldn’t get the User’s actual position. In this case the

	system centers the map onto the last visited position and the execution gets to point 3. 2. There are no areas within the specified zone where the filtered violations (or all) occurred. In this case the system shows a message to the User and suggests him to remove some filters or to enlarge the research area.

Scenario 3

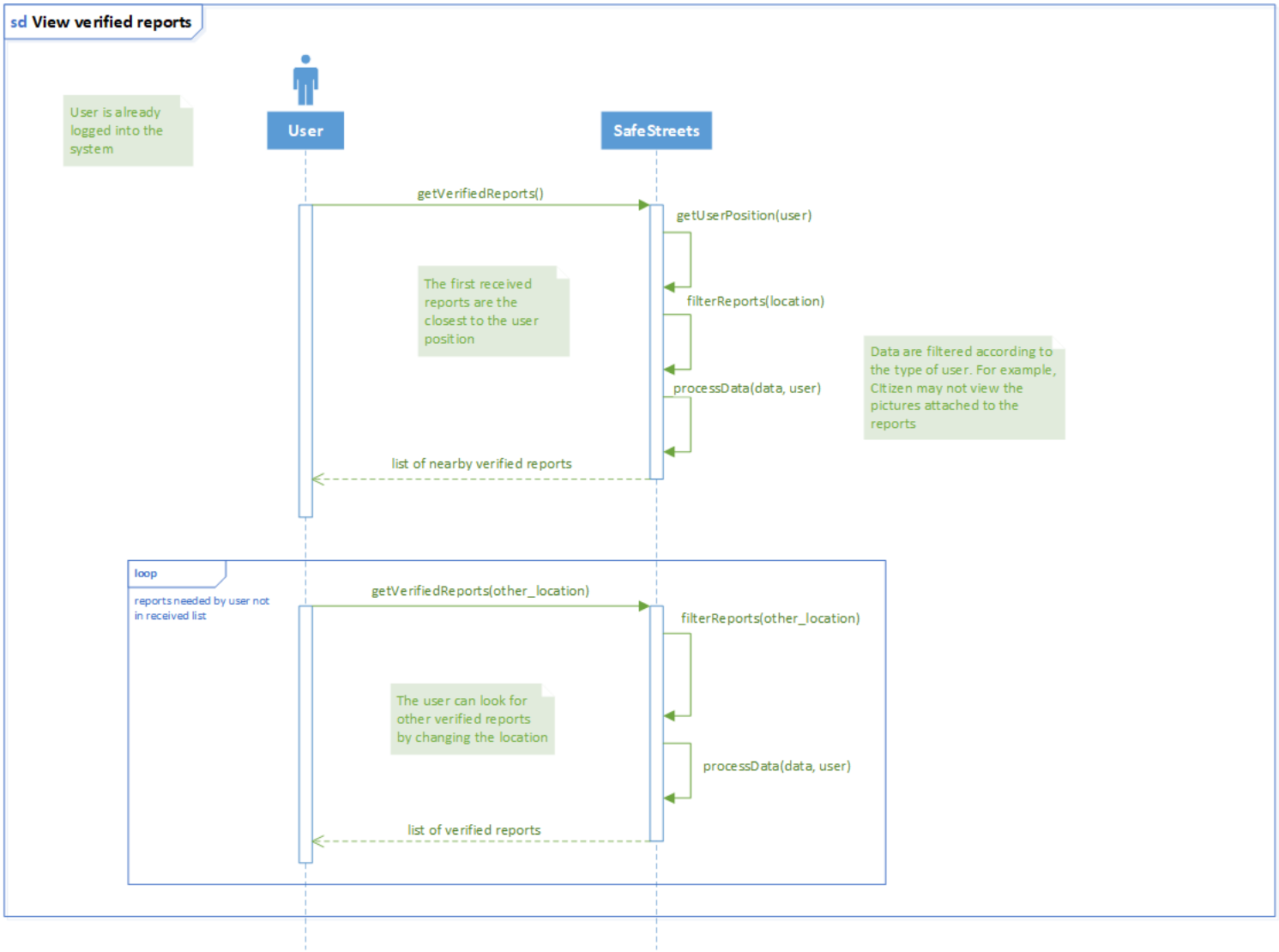
For this Saturday, Nicholas is organizing a round of golf with some friends, and he is looking for the best way to reach the place by car without worries. With SafeStreets, he notices that the fastest road he would probably have chosen is highlighted as an unsafe area; in particular, in the last months some accidents occurred in the zone, including car crashes (sometimes with injured people). Since Nicholas is a cautious driver, probably he will choose another way around to get to the golf club, a slower one but safer, and he will warn his friends about the danger too.

Name	Check unsafe areas
Goals	[G6]
Actors	User
Entry conditions	The User is logged into the system and is on the home page.
Events flow	1. The User selects the “view unsafe areas” option. 2. The system provides the User with a map centered on the User’s current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. The User consults the areas highlighted as unsafe within the specified range.

Exit conditions	The system has provided the User with all the areas highlighted as unsafe within the desired range from the selected location.
Exceptions	1. The system couldn't get the User's actual position. In this case the system centers the map onto the last visited position and the execution gets to point 3. 2. There are no unsafe areas within the specified zone. In this case the system shows a message to the User and suggests him to enlarge the research area.

Name	View verified reports
Goals	[G3]
Actors	User
Entry conditions	The User is logged into the system and is on the home page.
Events flow	1. The User selects the "view verified reports" option. 2. The system provides the User with a map centered on the User's current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. The system shows a list of all accepted reports coming from the specified area. If the User is a Citizen, the reports will be shown with limited information. 5. The User consults the list of reports.
Exit conditions	The system has provided the User with a list of verified reports which have been accepted by an Authority and have

	occurred within the specified area.
Exceptions	1. The system couldn't get the User's actual position. In this case the system centers the map onto the last visited position and the execution gets to point 3. 2. There are no verified reports within the specified zone. In this case the system shows a message to the User and suggests him to enlarge the research area.



Scenario 4

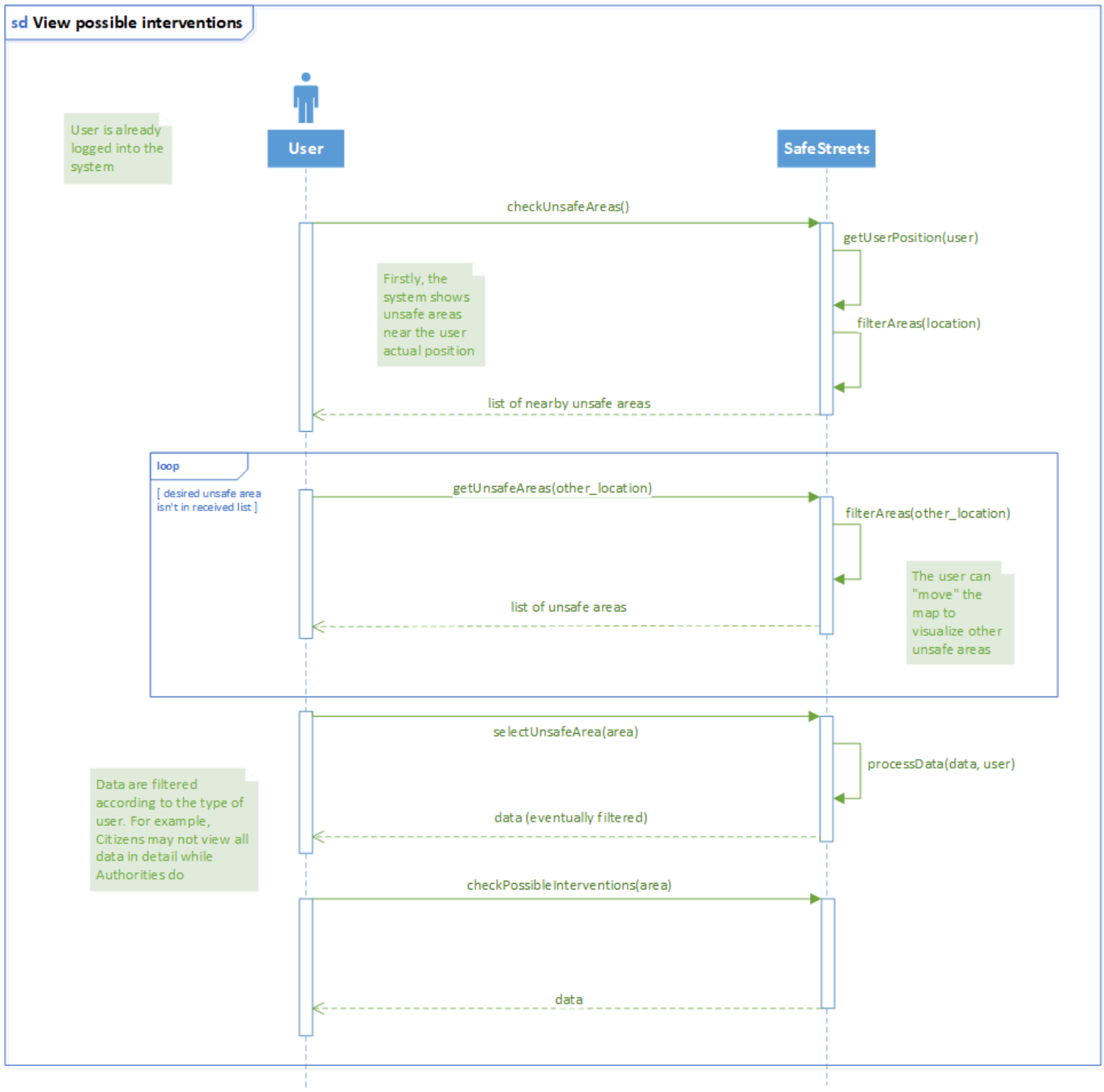
In the last few months, the citizens of Cologno Monzese have been more and more complaining about the increasing traffic violations occurring in the territory of the municipality. At the moment local authorities aren't able to displace enough resources to catch most violations, so they decide to provide citizens more responsibility and register to the SafeStreets service as Authority. With the help of the citizens and the system, the local police is able to displace patrols in the zones where most violations are reported, issuing many more traffic tickets to offenders (and managing a pretty good income for the municipality income as well). After some months, the municipality consults the statistics about traffic tickets provided by SafeStreets and notices that citizens are used to park their cars on pavements many times, so the local police decides to increase controls in the areas where this violation occur mostly, in order to make the offenders lose this bad habit.

Name	View statistics
Goals	[G9]
Actors	User
Entry conditions	The User is checking the areas with most violations.
Events flow	1. The User selects a location on the map where some violation occurred. 2. The system collects all the violations which took place on the selected location. 3. The system computes statistics about issued traffic tickets for each kind of violation in the area and summarizes them on a chart. 4. The system shows the chart to the User, with a short legend.
Exit conditions	The User can consult the chart and infer the type of violations most occurring a certain area.
Exceptions	/

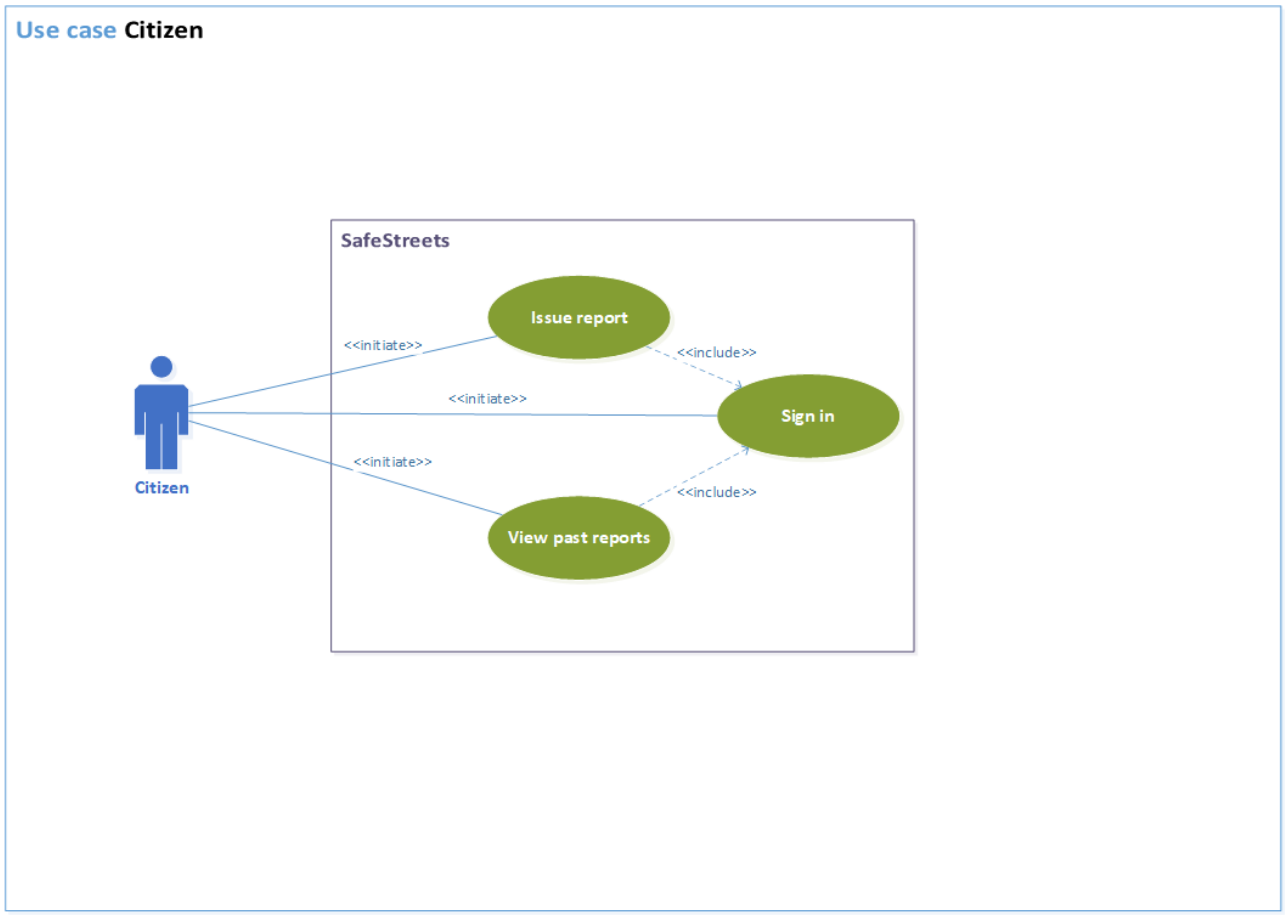
Scenario 5

Matteo is the mayor of a small city, and has approved the project of including SafeStreets in the system of the municipality, to improve safety and citizens' responsibility in his town. Recently, his collaborators brought him to attention many minor violations reported by citizens, and as he really cares about his fellow citizens' welfare, he decides to check SafeStreets statistics to get some ideas about the situation, and to take also into account possible suggestions from the system managers on how to prevent violations from occurring in future.

Name	View possible interventions
Goals	[G10]
Actors	User
Entry conditions	The User has just looked for unsafe areas within a certain zone.
Events flow	1. The User consults the list of all unsafe areas found. 2. The User selects one among the unsafe areas. 3. The User selects the “view suggested interventions” option. 4. The User consults all the possible interventions for the unsafe area.
Exit conditions	The system has provided the User with a list of all possible interventions for the selected unsafe area.
Exceptions	1. There are no suggested interventions for the selected area. In this case the system shows the User a message to explain this fact.


sd View possible interventions
User is already logged into the system
User
SafeStreets
checkUnsafeAreas()
getUserPosition(user)
Firstly, the system shows unsafe areas near the user actual position
filterAreas(location)
list of nearby unsafe areas
loop
[desired unsafe area isn't in received list]
getUnsafeAreas(other_location)
filterAreas(other_location)
The user can "move" the map to visualize other unsafe areas
list of unsafe areas
selectUnsafeArea(area)
processData(data, user)
Data are filtered according to the type of user. For example, Citizens may not view all data in detail while Authorities do
data (eventually filtered)
checkPossibleInterventions(area)
data

3.4.3 Citizen

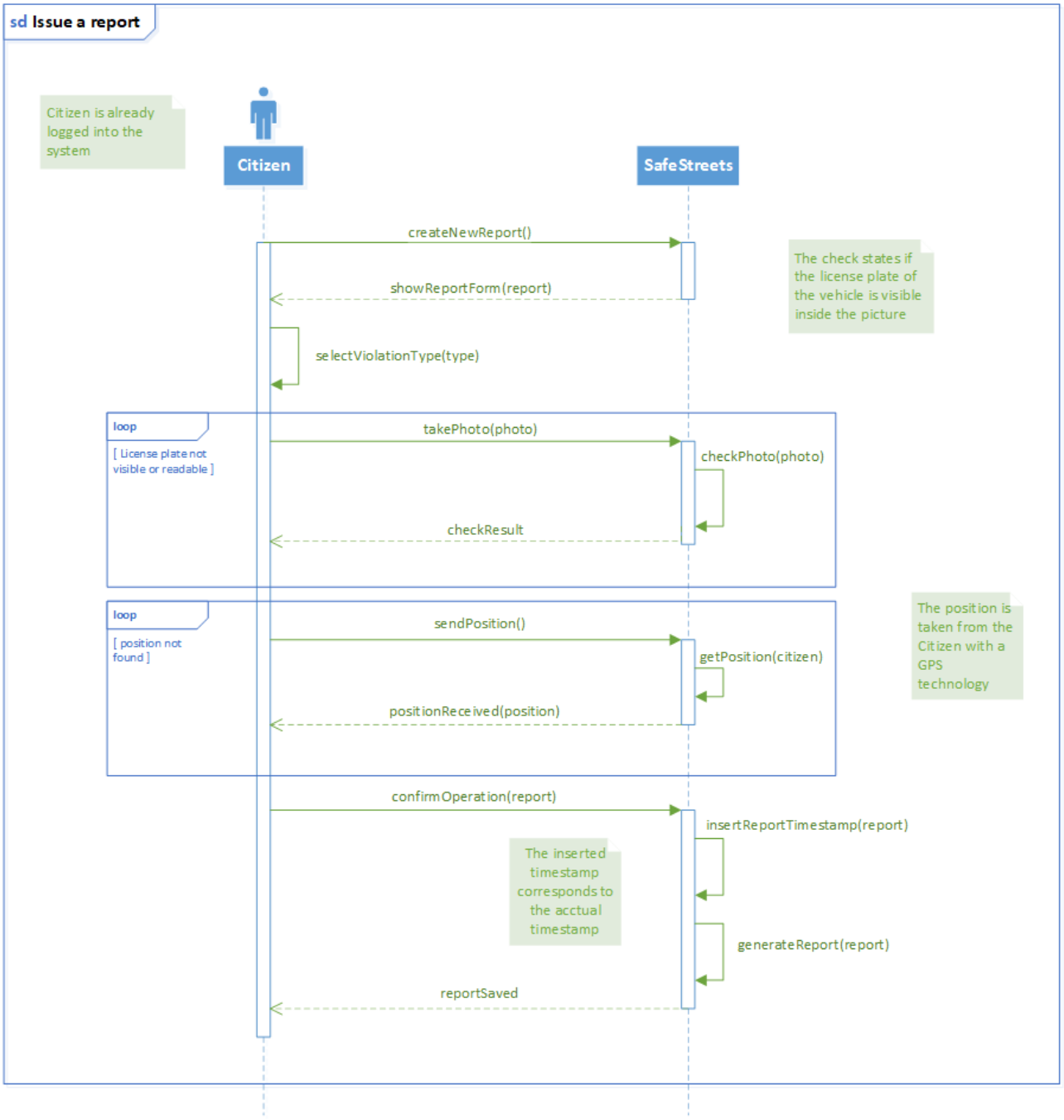


Scenario 6

Unfortunately Luca's brother recently broke his leg during a football match and he will be confined to a wheelchair for a few months. One day, when Luca is taking him to hospital for a routine visit, he notices that the disabled parking where he usually park his car is already busy, so he is forced to look for a parking far away from the hospital. When he finally reaches the hospital, he notices that the car is not showing the badge that allows it to use that spot so, as he is registered to SafeStreets service, he logs into the application, creates a new report, takes a picture of the car's license plate, includes a short description of the issue and sends his report to the system.

Name	Issue report
Goals	[G1]
Actors	Citizen
Entry conditions	The Citizen has spotted a possible violation and wants to report it.
Events flow	1. The Citizen selects the “report a violation” option. 2. The Citizen selects the type of violation from a list containing the reportable ones. 3. The Citizen selects the “take a photo” option. 4. The Citizen takes a photo of the involved vehicle making sure that the license plate is visible and readable. 5. The system analyzes the photo. 6. The system accepts the photo and takes the Citizen back to the report form. 7. The Citizen selects the “send your position” option. 8. The system collects the actual position of the Citizen. 9. The Citizen confirms the operation by selecting the relative option. 10. The system receives the data and saves them.
Exit conditions	The system has received the incoming data correctly and managed to store them.
Exceptions	4. The system couldn’t recognize or read the license plate from the photo. In this case the system warns the Citizen about the issue and asks him to take the photo again. 5. The system couldn’t retrieve the Citizen’s current position. In this case, the system warns the Citizen about the issue and asks him to

	activate his GPS. 6. The system didn't receive the data about the report. In this case the system shows the Citizen an error message and asks him to redo the whole operation.



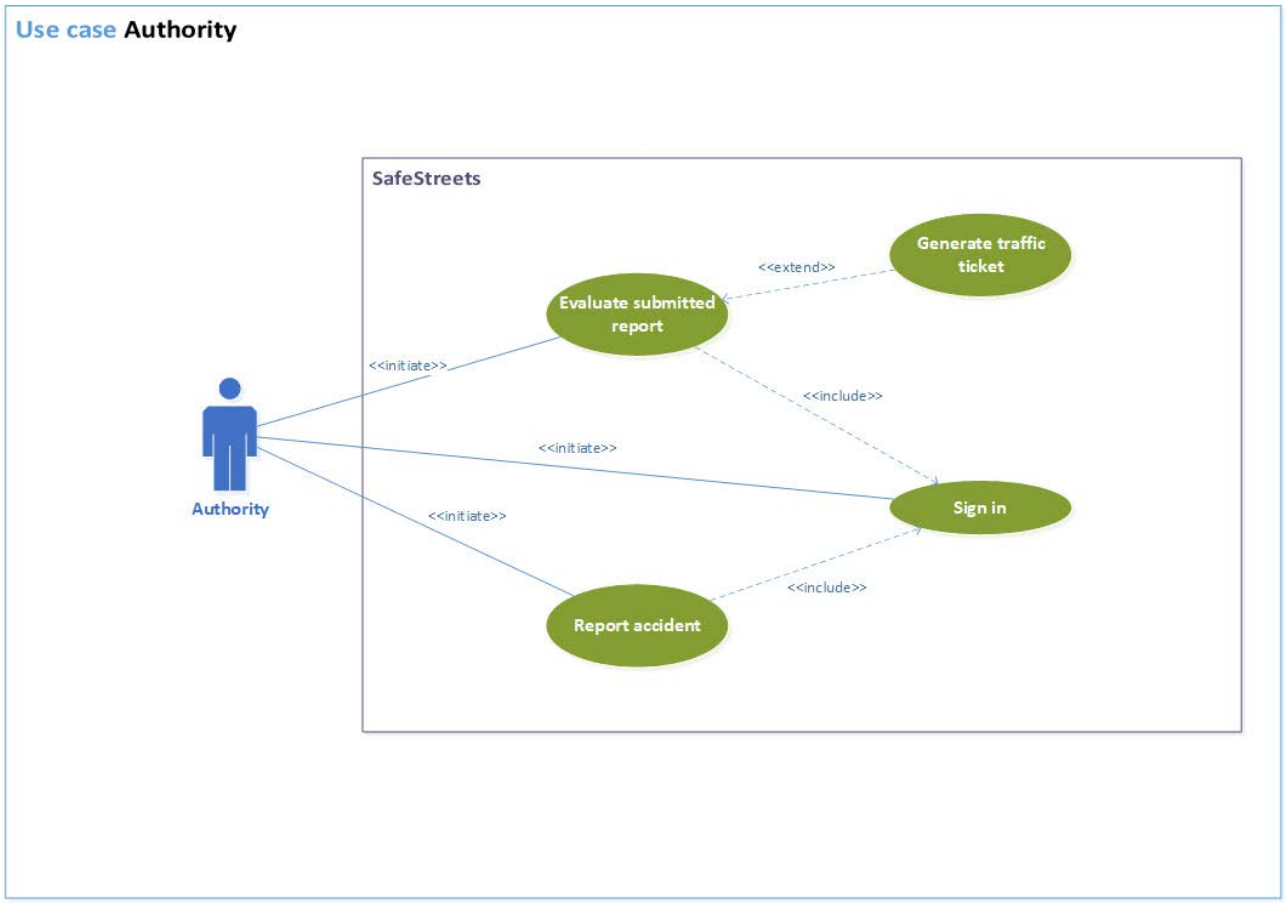
Scenario 7

Yesterday Michele was supposed to attend a job interview for an important company but he found a car parked in a double row exactly where his car was parked, and that disappointed him a lot. Since he wasn't that late yet, he had the time to send a report to SafeStreets and managed to find another solution to reach the place where the interview was scheduled, with not too much delay. Today he wants to check if his report had already been evaluated by the local police, so he navigates into the application to his past reports and he selects the interested one; with satisfaction, he notices that his report was evaluated positively, and a traffic ticket was issued to the owner of the car, so hopefully this situation won't happen again. By the way, the job interview went well.

Name	View past reports
Goals	[G2]
Actors	Citizen
Entry conditions	The Citizen is logged into the system and is on the home page.
Events flow	1. The Citizen selects the “my profile” option. 2. The system redirects the Citizen to his personal profile. 3. The Citizen selects the “my reports” option. 4. Eventually, the Citizen enters a time filter to exclude too “old” reports. 5. The Citizen consults a list of his submitted reports in the desired time period.
Exit conditions	The system has shown a list of the Citizen’s past reports submitted in a time such that the filter is satisfied, including a description of the report’s status.
Exceptions	1. The Citizen never submitted a report. 2. There are no reports that satisfy

	the inserted filter. In both cases, the system shows a message to the Citizen describing the issue.

3.4.4 Authority



Scenario 8

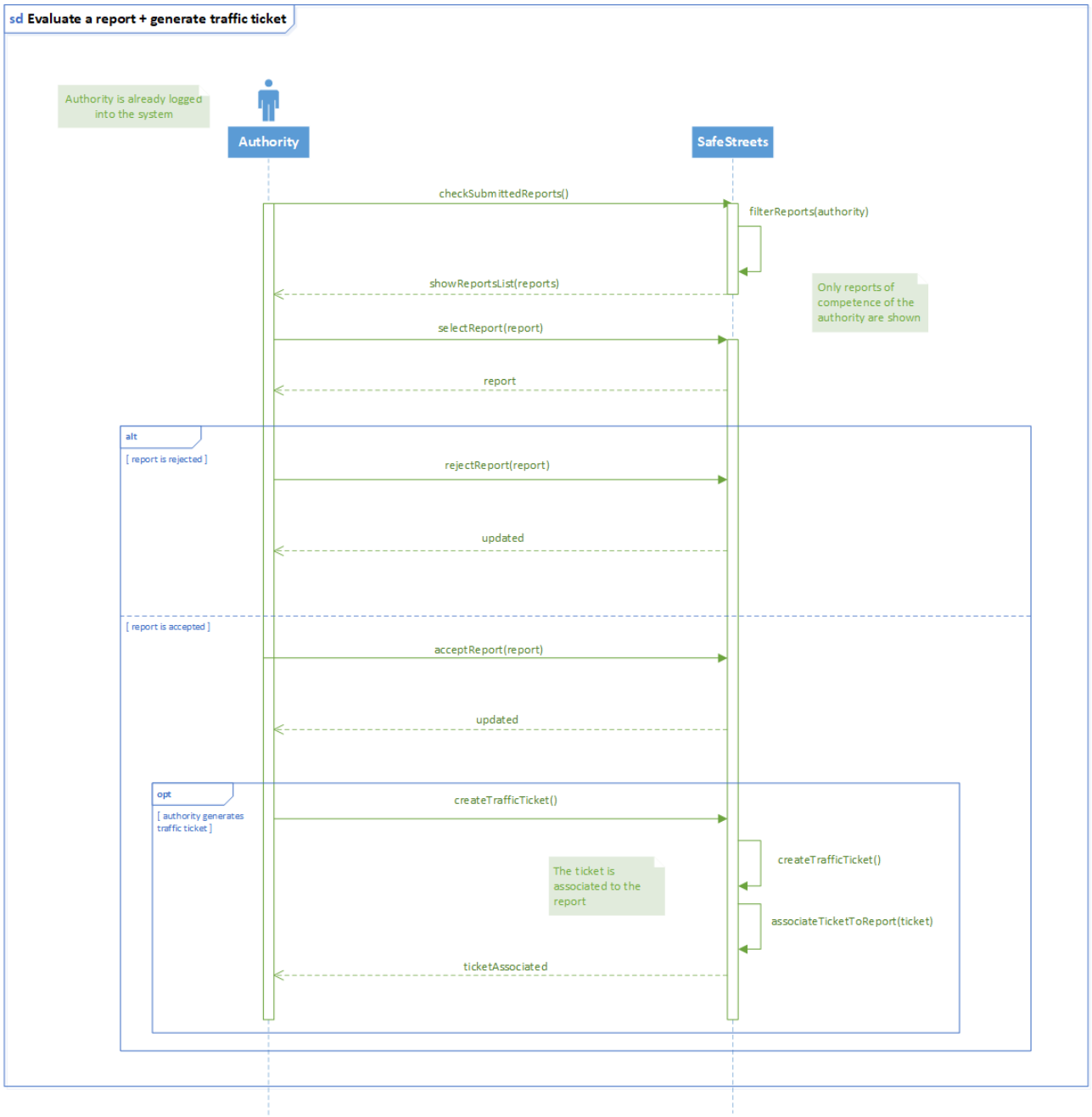
Davide is a policeman of Baranzate and he is well aware about all the traffic violations occurring in his town but, however, he knows that he can't notice and punish them all because he can't be everywhere at once, so when he learns about SafeStreets he thinks that it would give the police a great advantage as they may check incoming reports instead of just wandering around looking for violations. As his department is registered into the application, he looks for all the reports made by citizens in his town and checks them. Actually, one of them results as being a quite severe violation, so he selects the option for generating a traffic ticket and activates a police procedure to deliver it to the guilty. Then he marks the report as verified and that the traffic ticket has been dispatched.

Name	Evaluate submitted report
Goals	[G3]
Actors	Authority
Entry conditions	The Authority is logged into the system and is on the home page.
Events flow	1. The Authority selects the “check submitted reports” option. 2. The system provides the Authority with a list of all not already evaluated reports from its areas of competence, with a brief description. 3. The Authority selects a report among the list. 4. The system expands the selected report including all its data. 5. The Authority checks the data of the selected report. 6. The Authority evaluates the report as accepted or rejected. 7. The system saves the new status of the report.
Exit conditions	The report’s new status is either

	accepted or rejected and the system saves the status according to the Authority's decision.
Exceptions	1. All the reports of competence of the Authority have already been evaluated. In this case the system shows the Authority a message explaining the issue. 2. The Authority doesn't choose between accepting or rejecting the report. In this case the system warns the Authority and asks him to choose between interrupting the process or complete it. If the process is interrupted the report maintains its precedent status, else the process resumes from where it was suspended.

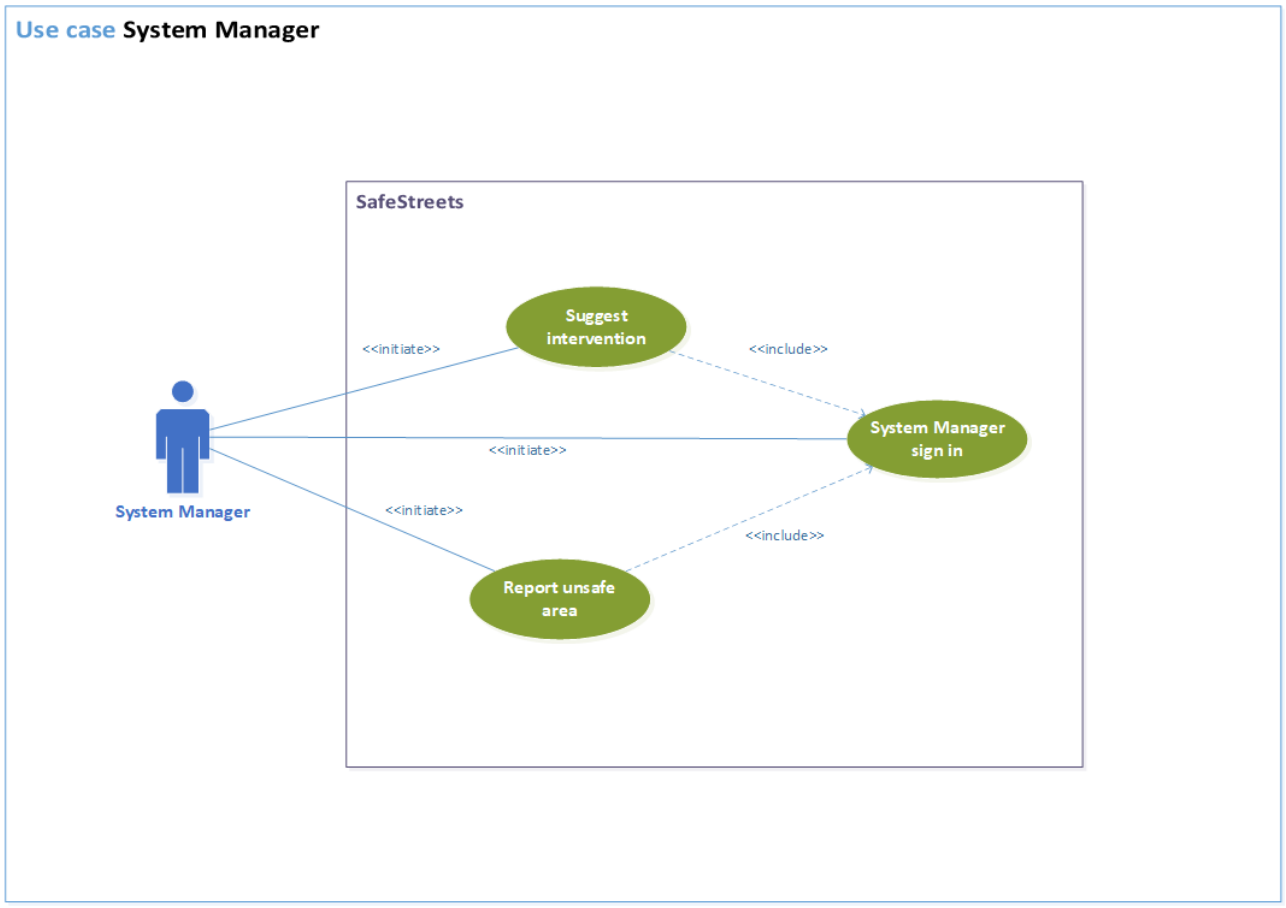
Name	Generate traffic ticket
Goals	[G7] [G8]
Actors	Authority
Entry conditions	The Authority has just evaluated a report as accepted.
Events flow	1. The system asks the Authority if he wants to generate a traffic ticket from the evaluated report. 2. The Authority takes a decision on this. If it accepts, then the process continues to step 3, else it is terminated. 3. The Authority confirms the operation. 4. The system emits the traffic ticket and it associates it to the license plate of the vehicle which committed the violation. 5. The system associates the emitted traffic ticket to the evaluated

	report.
Exit conditions	If the traffic ticket isn't emitted, the report is saved just as accepted. If the traffic ticket is emitted, it is sent to the owner of the vehicle found by license plate, the system saves the new status of the report with the issued traffic ticket.
Exceptions	/



Name	Report accident
Goals	[G4]
Actors	Authority
Entry conditions	The Authority has all the information about an accident occurred in its area of competence.
Events flow	1. The Authority selects the “report accident” option. 2. The Authority inserts the data about date and time when the accident occurred. 3. The Authority inserts the data about the location where the accident occurred. 4. The Authority inserts a brief description, including the type and how many vehicles were involved in the accident, how many people were injured in the accident, and whether or not any emergency vehicle was dispatched and if it was able to reach the place in good time. 5. The Authority confirms the operation and send the report. 6. The system receives the data about the report and saves them.
Exit conditions	The system has received the report and managed to store the data.
Exceptions	1. The Authority inserts an invalid input in any of the fields (most likely the date). In this case the system warns the Authority and asks him to correct the input.

3.4.5 System Manager



Name	System Manager sign in
Goals	[G6] [G10]
Actors	System Manager
Entry conditions	/
Events flow	1. The System Manager opens the application from his private interface. 2. The System Manager enters his credentials. 3. The System Manager selects the confirmation option. 4. The system redirects the System Manager to the maintenance area.
Exit conditions	The System Manager is logged into the system and can perform maintenance

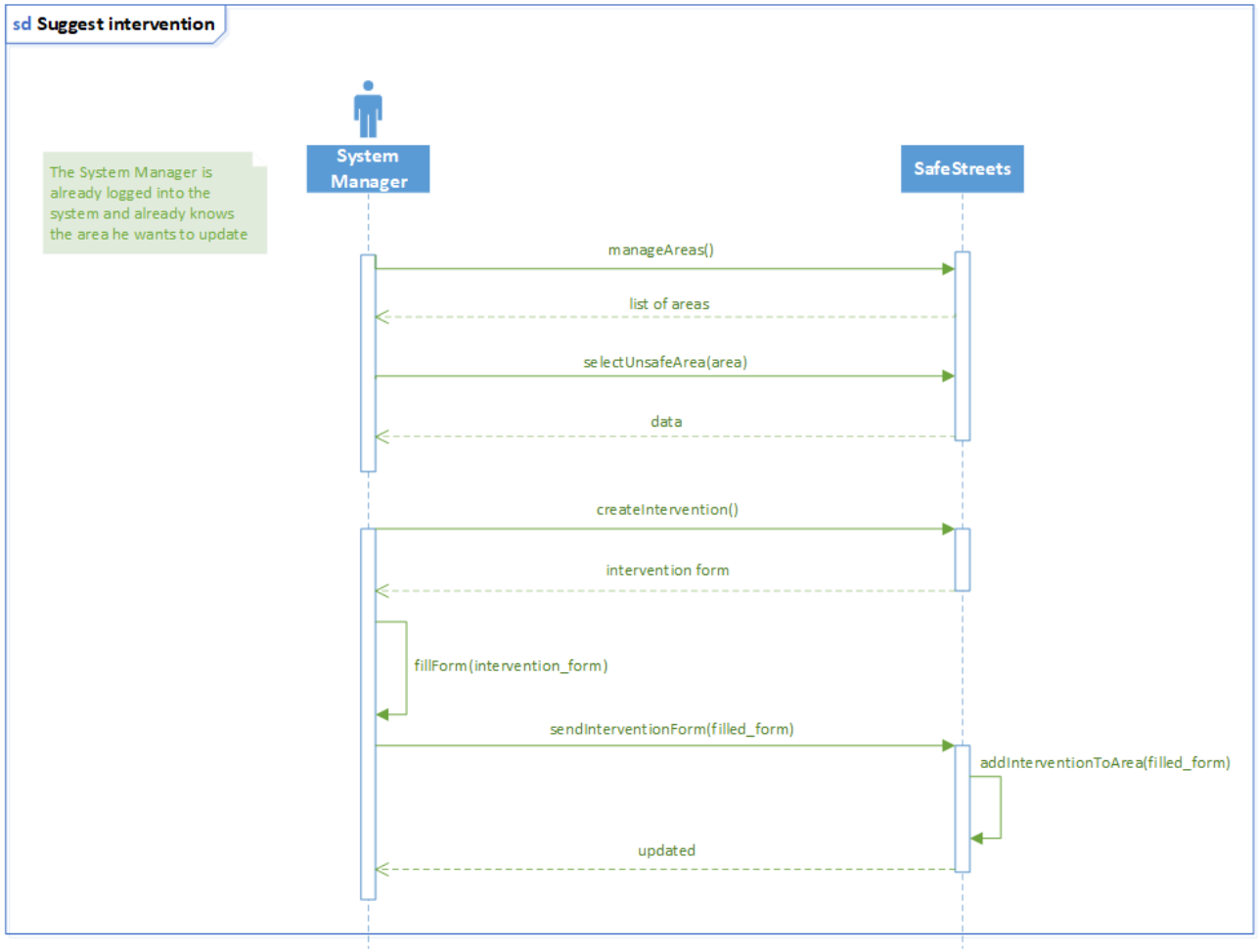
	operations.
Exceptions	1. The System Manager inserts wrong credentials. In this case the system warns the System Manager and invites him to insert the correct ones.

Name	Report unsafe area
Goals	[G6]
Actors	System Manager
Entry conditions	The System Manager has noticed a dangerous area because of many accepted violations and/or accidents and he is logged into the system.
Events flow	1. The System Manager selects the “areas management” option. 2. The System Manager searches the area he wants to highlight. 3. The System Manager selects the area. 4. The System Manager marks the area as “unsafe”. 5. The System Manager confirms the modification. 6. The system saves the new status of the area.
Exit conditions	The system has successfully saved the modification.
Exceptions	/

Scenario 9

Francesco, a system manager of SafeStreets, is responsible of collecting the incoming reports from the zone near Magenta. During the last three months, he started receiving many more reports from citizens as well as information from about accidents, sometimes with severely injured people, especially from a main road connecting small town of the zone. So, Francesco decides to highlight the road as dangerous to the local authorities, and also suggests to provide patrols or checkpoints in order to have a better supervision and prevent more accidents from occurring.

Name	Suggest intervention
Goals	[G10]
Actors	System Manager
Entry conditions	The System Manager has noticed the same violation or accident occurring several times in a certain area and he is logged into the system.
Events flow	1. The System Manager selects the “area management” option. 2. The System Manager searches the area he wants to revise. 3. The System Manager selects the area. 4. The System Manager selects the “suggest intervention” option. 5. The System Manager fills with a brief description the possible intervention to prevent other violations to occur. 6. The System Manager confirms the modification. 7. The system saves the suggestion description in the area.
Exit conditions	The system has successfully saved the modification.
Exceptions	/



3.5 Performance requirements

The system should notify the user whether their submission has been submitted properly, that is, it has been received and stored without corruption of data, within one minute after the moment it has been sent.

3.6 Design constraints

3.6.1 Standards compliance

The system adopts units of measure that are compliant with the SI. In particular, distance is expressed in terms of meters [*m*].

3.6.2 Hardware limitations

As stated previously, the SafeStreets mobile application requires that the device being used can:

- access the Internet through a secure and stable connection;
- provide its own position through GPS;
- take pictures, meaning it has to be equipped with a camera.

3.7 Software system attributes

3.7.1 Availability

The system must have an availability of 99,9%, meaning at most 8.76 hours/year of downtime.

3.7.2 Security

The system must encrypt sensitive information about users and make sure it is protected during the exchange of data; furthermore, it is necessary that personal information can only be released in case the request to see it was sent by authorities.

3.7.3 Portability

Users must be able to access the service through mobile platforms, in particular the most common ones, e.g. iOS, Android, as well as web browsers, as some functionalities are only available on specific platforms.

4. Formal analysis using Alloy

Signatures

```
--Signatures concerning users

sig Username {}

sig Password {}

sig FiscalCode {}

sig Registration {
	username: one Username,
	password: one Password
}

abstract sig User {
	registration: one Registration
}

sig Citizen extends User {
	reports: set Report,
	fiscalCode: one FiscalCode
}

sig Municipality extends User {
	reports: set Report,
	accidents: set Accident,
	area: some Position
}
```

```
--Signatures concerning reports

abstract sig Status {}

sig Accepted extends Status {}
sig Refused extends Status {}
sig Evaluating extends Status {}

sig Intervention {}

sig LicensePlate {}

sig Date {}

sig ViolationType {}

sig Photo {}

sig Time {}

sig Position {}

sig HighFrequencyViolationsArea extends Position {}

sig UnsafeArea extends Position {
	possibleSolutions: set Intervention
}

sig Accident {
	position: one Position
}

abstract sig Boolean {}

sig True extends Boolean {}
sig False extends Boolean {}

sig Report {
	idReport: one Int,
	reporter: one Citizen,
	date: one Date,
```

```
        time: one Time,
        violationType: one ViolationType,
        photo: one Photo,
        position: one Position,
        status : one Status,
        correspondence: one LicensePlate,
        ticket: one Boolean
} {
        idReport > 0
}

sig AcceptedReports {
        acceptedReports: set Report
}
```

Facts

```
--All Users have different usernames

fact DifferentUsernames {
        all disj u1, u2: User | u1.registration.username != u2.registration.username
}

--All Citizens have different fiscal codes

fact DifferentFiscalCodes {
        all disj c1,c2: Citizen | c1.fiscalCode != c2.fiscalCode
}

--All Usernames match one Registration

fact UsernamesMatchOneRegistration {
        all u: Username | one r: Registration | u in r.username
}

--All Reports made by Citizens are in the list of Reports of the corresponding Municipality

fact ReportsToMunicipalityList {
        all r: Report| one m: Municipality | one c: Citizen | r in m.reports && r in c.reports
```

```
}

--All Reports are made by a Citizen registered in the system

fact ReportsHaveRegisteredCitizen {
        all r: Report | one r2: Registration | r.reporter.registration = r2
}

-- All Citizens have all their Reports in their list of Reports

fact CitizensHaveTheirReports {
        all c: Citizen | all r: Report | (r.reporter = c) <=> (r in c.reports)
}

--All Reports which are accepted have a ticket

fact AcceptedReportsHaveTicketTrue {
        all r: Report | ( r.status = Accepted)  <=>  (r.ticket = True)
}

--All Reports which are refused don't have a ticket

fact RefusedReportsHaveTicketFalse {
        all r: Report | (r.status = Refused)  => (r.ticket = False)
}

--All Reports which are being evaluated don't have a ticket

 fact EvaluatingReportsHaveTicketFalse {
        all r: Report | (r.status = Evaluating)  => (r.ticket = False)
}

--All Reports have a different ID

fact DifferentReportId {
        all disj r1, r2: Report | r1.idReport != r2.idReport
}
```

```
--All Reports with the same Reporter, Position, Time, Date and LicensePlate have the same ID

fact SameReportId {
        all r1, r2: Report | (r1.reporter = r2.reporter &&
        r1.correspondence = r2.correspondence && r1.position = r2.position &&
        r1.time = r2.time && r1.date = r2.date) <=> r1.idReport = r2.idReport
}

--All accepted Reports are in the set of AcceptedReports

fact AcceptedReportsInSet {
        all r: Report | (r.ticket = True) <=> r in AcceptedReports.acceptedReports
}

--All Report positions belong to the area of one Municipality

fact PositionBelongsToOneArea {
        all r: Report | all m: Municipality | (r.position in m.area) <=> (r in m.reports)
}

--Different Municipalities have different areas with no positions in common

fact DisjAreas {
        all disj m1, m2: Municipality | m1.area & m2.area = none
}

--All Reports are in the list of only one Municipality

fact ReportBelongsToOneMunicipality {
        all r: Report | one m: Municipality | r in m.reports
}

--All Reports with different IDs have different photos

fact DifferentIdDifferentPhoto {
        all r1,r2: Report | (r1.idReport != r2.idReport) <=> (r1.photo != r2.photo)
}
```

```
--All Accidents are in the list of the Municipality of their area

fact AccidentsBelongToMunicipalitiesOfSameArea {
        all a: Accident | all m: Municipality | (a in m.accidents) <=> (a.position in
m.area)
}

--All positions with more than one accident are marked as unsafe (NB this is a
dummy characteristic inserted to show meaningful worlds but in real situations
different unsafe areas may have different numbers of accidents)

fact AreasWithManyAccidentsAreUnsafe {
        all p: Position | (#{a: Accident | a.position = p} >= 2) <=> (one u: UnsafeArea |
p = u)
}

--All positions with more than 4 violations are marked as high frequency violations
areas (NB this is a dummy characteristic inserted to show meaningful worlds but in
real situations different high frequency violations areas may have different numbers
of violations)

fact AreasWithManyViolationsAreHFA {
        all p: Position | (#{r: Report | r.position = p} >= 5) <=> (one h:
HighFrequencyViolationsArea | p = h)
}
```

Assertions

```
assert CheckNoDifferentMunicipalitiesHaveSameReports {
        all disj m1,m2: Municipality | no r: Report | r in m1.reports && r in
m2.reports
}

assert CheckAcceptedReports {
        all r: Report | (r.ticket = True) <=> ( r.status = Accepted
        && r in AcceptedReports.acceptedReports)
}

assert CheckUnsafeAreas {
        no a: UnsafeArea | #{ac: Accident | ac.position = a} < 2
}
```

```
assert CheckHFVAreas {
	no h: HighFrequencyViolationsArea | #{r: Report | r.position = h} < 5
}
```

Predicates

```
pred show {
	#Report = 1
	#Citizen = 1
	#Municipality = 1
	#AcceptedReports = 1
}

pred worldOne {
	#Citizen = 2
	#Municipality = 1
	#Report = 2
	#AcceptedReports = 1
	#AcceptedReports.acceptedReports = 1
	(some disj c1, c2: Citizen | one m: Municipality | some disj r1,r2: Report |
r1.reporter = c1 && r2.reporter = c2 && r1.correspondence = r2.correspondence
&& r1.position != r2.position && r1.date = r2.date && r1.time != r2.time && r1 in
m.reports && r2 in m.reports && r1 in AcceptedReports.acceptedReports && r2
not in AcceptedReports.acceptedReports)

}

pred worldTwo {
	#Citizen = 1
	#Municipality = 2
	#Municipality.area = 3
	#Report = 2
	#AcceptedReports = 1
	(one c: Citizen | some disj m1, m2: Municipality | some disj r1, r2: Report |
r1.reporter = c && r2.reporter = c && r1.position in m1.area && r2.position in
m2.area && r1.status = Accepted && r2.ticket = False && r1.correspondence !=
r2.correspondence)
}
```

Commands and results: assertions

check CheckNoDifferentMunicipalitiesHaveSameReports

```
Executing "Check CheckNoDifferentMunicipalitiesHaveSameReports"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3787 vars. 306 primary vars. 8219 clauses. 28ms.
  No counterexample found. Assertion may be valid. 4ms.
```

check CheckAcceptedReports

```
Executing "Check CheckAcceptedReports"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3755 vars. 300 primary vars. 8246 clauses. 19ms.
  No counterexample found. Assertion may be valid. 18ms.
```

check CheckUnsafeAreas

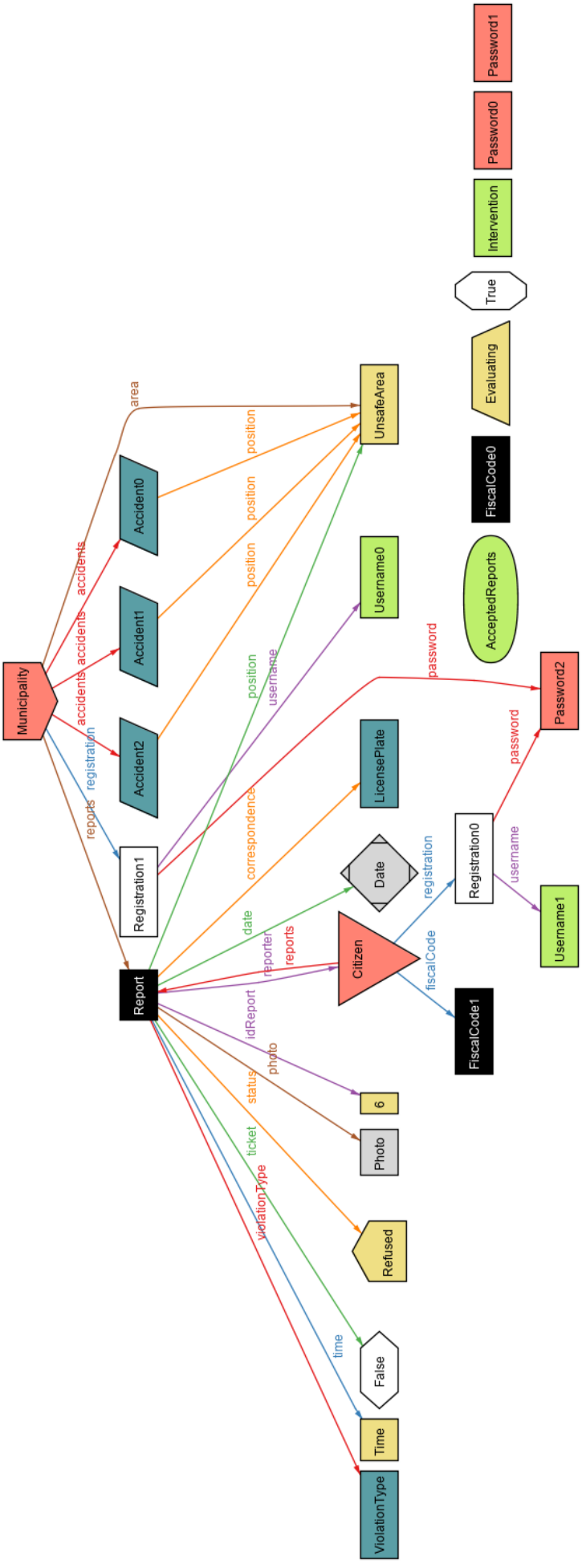
```
Executing "Check CheckUnsafeAreas"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3749 vars. 300 primary vars. 8218 clauses. 21ms.
  No counterexample found. Assertion may be valid. 9ms.
```

check CheckHFVAreas

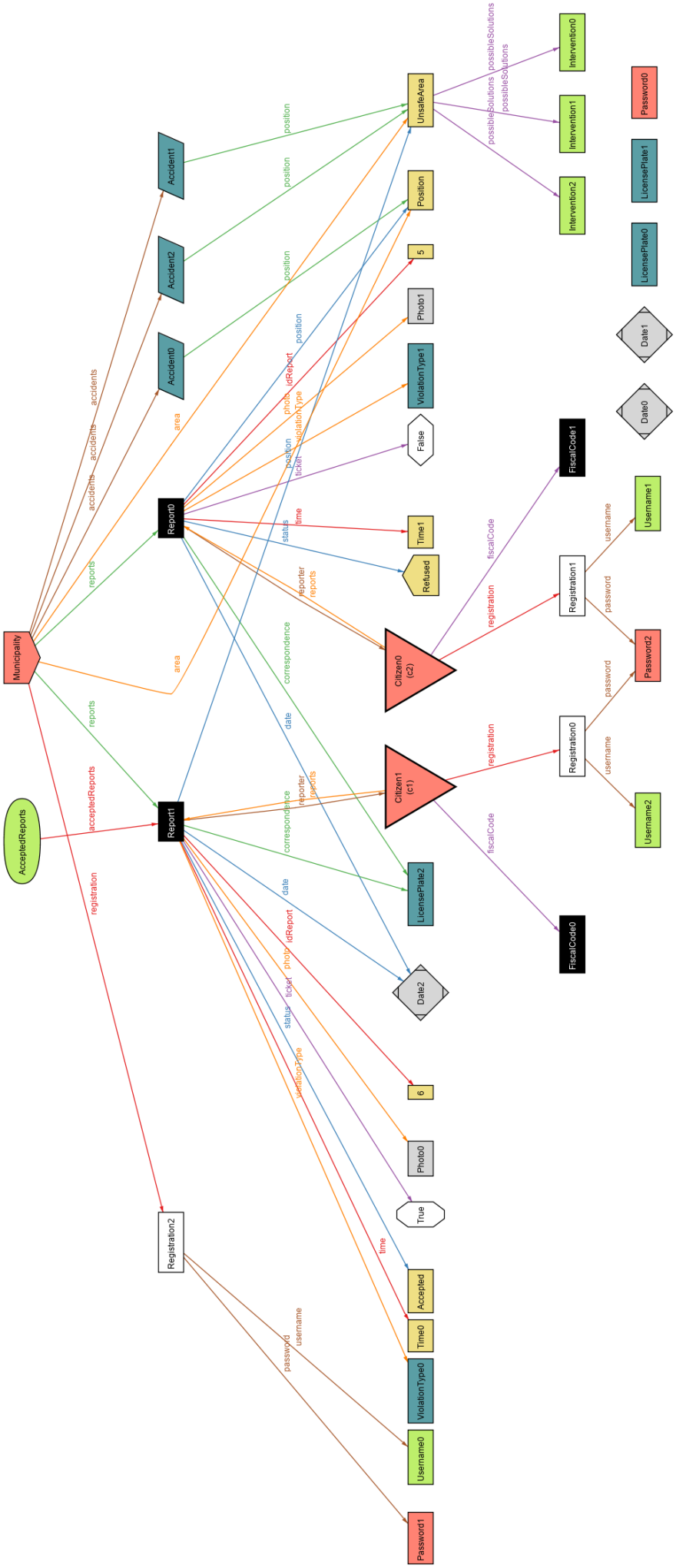
```
Executing "Check CheckHFVAreas"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3748 vars. 300 primary vars. 8213 clauses. 20ms.
  No counterexample found. Assertion may be valid. 2ms.
```

Command and results: predicates

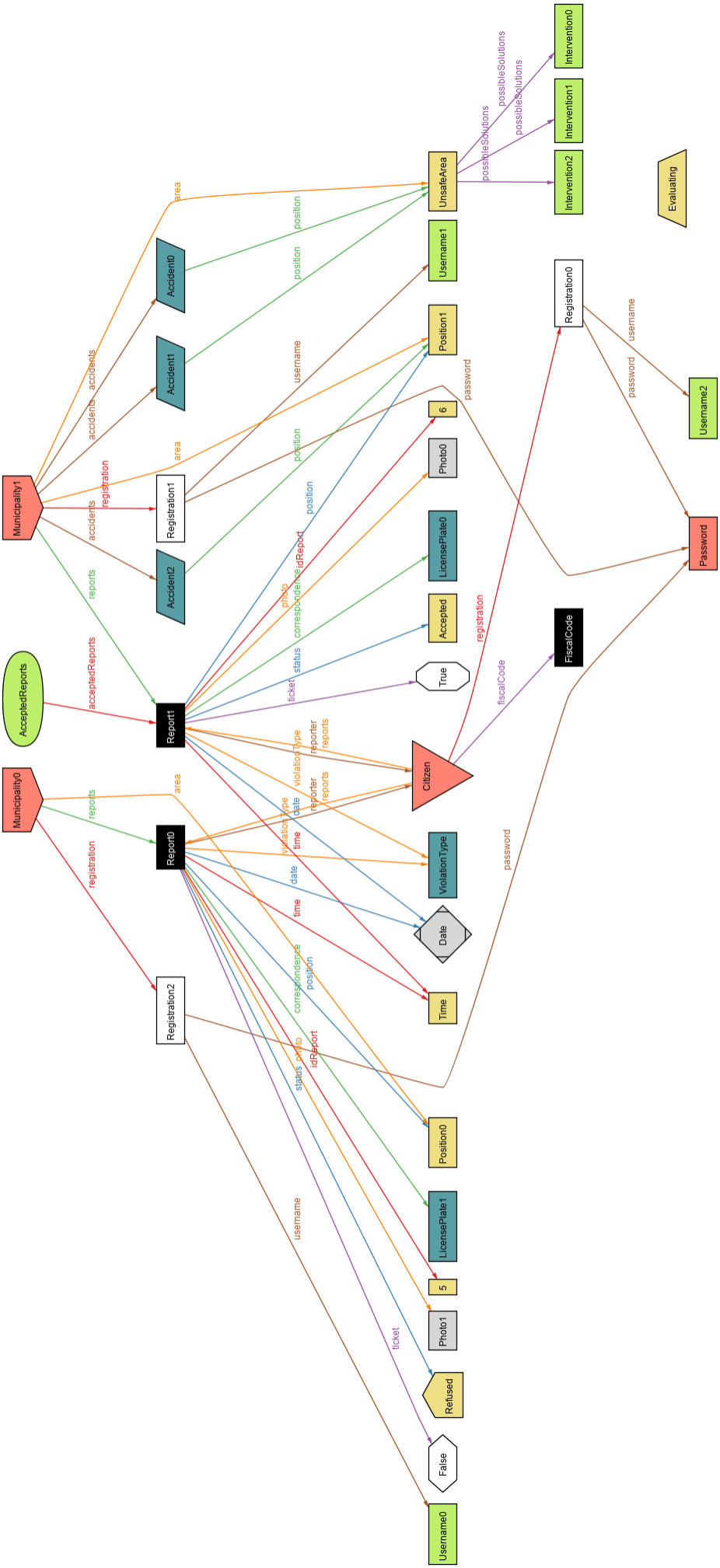
run show **for** 3



run worldOne **for** 3



run worldTwo **for** 3



5. Appendix

5.1 Effort spent

5.1.1 Pozzi Matteo

Date	*Topic*	*Effort [hrs]*
18/10/2019	Goal and domain assumptions	3
19/10/2019	Goal and domain assumptions	1
23/10/2019	Scenarios	4
25/10/2019	Scenarios	3
27/10/2019	Definitions + Use cases	3
29/10/2019	Use cases	4
01/11/2019	Use case diagrams + sequence diagrams	4
03/11/2019	Sequence diagrams	2,5
04/11/2019	Various	1,5
06/11/2019	Requirements + RASD	3
08/11/2019	RASD	1
09/11/2019	RASD	2
10/11/2019	RASD	2
	Total effort	**34**

5.1.2 Ventura Andrea

Date	Topic	Effort [hrs]
16/10/2019	Introduction	2
17/10/2019	Goal and domain assumptions	2
19/10/2019	Actors and definitions	1
24/10/2019	User Interfaces	3
25/10/2019	User Interfaces	2
26/10/2019	User Interfaces	2
30/10/2019	User Interfaces	3,5
31/10/2019	Alloy	1,5
01/11/2019	Alloy	1,5
04/11/2019	General aspects + Alloy	1,5
05/11/2019	Alloy	2
06/11/2019	Alloy	3
07/11/2019	Alloy + Review	2
08/11/2019	Alloy + Review	3
10/11/2019	Alloy + Review	3
	Total effort	**33**

5.1.3 Sacco Sara

Date	Topic	Effort [hrs]
16/10/2019	Introduction	3
20/10/2019	Scope	2
22/10/2019	Class diagram	3
24/10/2019	World and Machine	1
25/10/2019	Goals + Use cases	2
29/10/2019	Document structure + Requirements	5
30/10/2019	Domain assumptions + Software System Attributes	3
01/11/2019	State diagrams + User characteristics	3
02/11/2019	Design constraints	0,5
03/11/2019	Software interfaces	0,5
04/11/2019	Product functions	1
05/11/2019	Alloy	2
06/11/2019	Alloy + RASD fix	3
07/11/2019	Alloy	3
09/11/2019	Alloy	3
	Total effort	**35**