



**POLITECNICO**  
**MILANO 1863**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

---

# Requirement Analysis and Specification Document (RASD)

---

SAFESTREETS

- v1.0 -

*Authors:*

**Morreale** Federico

**Maddes** Evandro

**Innocente** Federico

*Student Number:*

945258

945642

870726

November 10<sup>th</sup> , 2019

---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose . . . . .	1
1.1.1	General Purpose . . . . .	1
1.1.2	Goals . . . . .	1
1.2	Scope . . . . .	3
1.2.1	World, Machine and Shared phenomena . . . . .	3
1.3	Definitions, Acronyms, Abbreviations . . . . .	5
1.3.1	Definitions . . . . .	5
1.3.2	Acronyms . . . . .	5
1.3.3	Abbreviations . . . . .	5
1.4	Reference Documents . . . . .	6
1.5	Revision history . . . . .	6
1.6	Document structure . . . . .	6
<b>2</b>	<b>Overall description</b>	<b>7</b>
2.1	Product perspective . . . . .	7
2.2	Product functions . . . . .	10
2.2.1	Citizens functions . . . . .	10
2.2.2	Authority functions . . . . .	11
2.2.3	Municipality featuring functions . . . . .	12
2.3	User characteristics . . . . .	12
2.4	Assumptions, dependencies and constraints . . . . .	13
2.4.1	Domain assumptions . . . . .	13
2.4.2	Dependencies . . . . .	13
2.4.3	Constraints . . . . .	13
<b>3</b>	<b>Specific requirements</b>	<b>15</b>
3.1	External interface requirements . . . . .	15
3.1.1	User Interfaces . . . . .	15
3.1.2	Hardware Interfaces . . . . .	19
3.1.3	Software Interfaces . . . . .	19

3.1.4	Communication Interfaces . . . . .	19
3.2	Functional requirements . . . . .	20
3.2.1	Scenarios . . . . .	20
3.2.2	Use Case Diagrams . . . . .	22
3.2.3	Use Case Analysis . . . . .	24
3.2.4	Sequence Diagrams . . . . .	38
3.2.5	Requirements, Domain Assumptions, Goals Matrix . . . . .	42
3.3	Performance requirements . . . . .	46
3.4	Design constraints . . . . .	46
3.4.1	Standard compliance . . . . .	46
3.4.2	Hardware limitations . . . . .	46
3.5	Software system attributes . . . . .	47
3.5.1	Reliability . . . . .	47
3.5.2	Availability . . . . .	47
3.5.3	Security . . . . .	47
3.5.4	Maintainability . . . . .	47
3.5.5	Portability . . . . .	47
<b>4</b>	<b>Formal analysis using alloy</b>	<b>49</b>
4.1	Alloy Model . . . . .	49
4.2	Generated World . . . . .	53
<b>5</b>	<b>Effort spent</b>	<b>60</b>
<b>6</b>	<b>References</b>	<b>61</b>

---

# List of Tables

---

3.1	<i>Use Case 1</i>	24
3.2	<i>Use Case 2</i>	25
3.3	<i>Use Case 3</i>	26
3.4	<i>Use Case 4</i>	27
3.5	<i>Use Case 5</i>	28
3.6	<i>Use Case 6</i>	29
3.7	<i>Use Case 7</i>	30
3.8	<i>Use Case 8</i>	31
3.9	<i>Use Case 9</i>	32
3.10	<i>Use Case 10</i>	33
3.11	<i>Use Case 11</i>	34
3.12	<i>Use Case 12</i>	35
3.13	<i>Use Case 13</i>	36
3.14	<i>Use Case 14</i>	37
5.1	Time spent by all team members	60
5.2	Time spent by each team member	60

---

# List of Figures

---

2.1	Domain Model <i>Class Diagram</i> . . . . .	7
2.2	New report <i>State Chart Diagram</i> . . . . .	8
2.3	Query to database <i>State Chart Diagram</i> . . . . .	9
2.4	Build statistics <i>State Chart Diagram</i> . . . . .	10
3.1	<i>User</i> Sign Up <i>Mockup</i> . . . . .	15
3.2	<i>User</i> Login <i>Mockup</i> . . . . .	16
3.3	<i>User</i> Home <i>Mockup</i> . . . . .	17
3.4	View Report <i>Mockup</i> . . . . .	18
3.5	Report a violation <i>Mockup</i> . . . . .	19
3.6	Citizen <i>Use Case Diagram</i> . . . . .	22
3.7	Authority <i>Use Case Diagram</i> . . . . .	23
3.8	SafeStreets <i>Use Case Diagram</i> . . . . .	23
3.9	Send Report <i>Sequence Diagram</i> . . . . .	38
3.10	Get Violations <i>Sequence Diagram</i> . . . . .	39
3.11	Get Dangerous Areas <i>Sequence Diagram</i> . . . . .	40
3.12	Send Feedback <i>Sequence Diagram</i> . . . . .	41
4.1	Generated world 1, <i>Alloy</i> . . . . .	54
4.2	Generated world 2, <i>Alloy</i> . . . . .	56
4.3	Generated world 3, <i>Alloy</i> . . . . .	58

# INTRODUCTION

---

## 1.1 Purpose

### 1.1.1 General Purpose

The purpose of the Requirement Analysis and Specification Document (RASD) is to give a clear and non-ambiguous analysis of SafeStreets, an application that will be implemented, describing every aspects of it like functional/non-functional requirements, constraints, domain assumptions, providing use cases and scenerios of the external world. Moreover, a more formal analysis of some relevant functions of the system will be provided using Alloy, a declarative specification language for expressing complex structural constraints and behavior in a software system.

The application will allow every user to take pictures of a violation and send them with the date, time, position, a scan of the license plate and an optional brief description to SafeStreets, that will give access to these information to both citizens and police officers, with different levels of visibility.

If the municipality provides information about the accidents that occur in its territory, the system will be able to cross these data with its own, to define dangerous areas and suggest possible interventions. One of the objectives of SafeStreets is to help authorities to detect infringements; in order to do that, it will ensure that data will not be tempered, to prevent malicious corruptions by offenders. The authorities that will use SafeStreets to make fines will have the possibility to mark the violation on the system, to both provide a feedback to the citizens and allow SafeStreets to compile statistics.

### 1.1.2 Goals

SafeStreets is a service provided to people to notify both other people and authorities about traffic violations. There are two different classes of users to whom the software is addressed: citizens and authorities, which have two different ways to interact with the application. In order to perform correctly, the software-to-be will have to grant that some services will be guaranteed. Below is given a list of all the goal of the software-to-be:

- [G1] The application will allow users to upload pictures of the traffic violations, including pictures, date, time, classification, and optionally a textual description.

- [G2] The application will allow users to view a map of the violations registered in the last day.
- [G3] The application will allow users to see the violations uploaded by other users; however, citizens won't be able to see sensitive pictures, while authorities will be allowed.
- [G4] Users will be able to query the system to get all the reported violations that correspond to some temporal, geographical or categorical parameters.
- [G5] Users can give a feedback about violations uploaded by other users to SafeStreets.
- [G6] If SafeStreets can get the information about accidents by the municipality, it will give the possibility to merge them with its data to identify potentially unsafe areas.
- [G7] If SafeStreets can get the information about accidents by the municipality, it will give the possibility to merge them with its data to suggest possible interventions.
- [G8] The system will use the information about violations, accidents and fines that will collect by both users and authorities to build statistics.

## 1.2 Scope

### 1.2.1 World, Machine and Shared phenomena

According to *The World and the Machine* we can divide every system into two parts:

- The **machine**, which is the portion of system to be developed;
- The **world**, which is the portion of the real-world affected by the machine.

As a consequence we can classify phenomena in three different types:

- **World phenomena**: phenomena that the machine cannot observe;
- **Machine phenomena**: phenomena located entirely in the machine;
- **Shared phenomena**: phenomena that can be controlled by the world and observed by the machine or controlled by the machine and observed by the world;

Below we give an analysis of the three phenomenas described above:

- **World Phenomena**

- Users recognize the type of violation that occurred;
- People commit violations;
- People perform incidents;
- Police emits traffic tickets.

- **Machine Phenomena**

- Machine manages database queries;
- Machine stores information (users and pictures with data) on the database
- Machine manages interfaces with external software.

- **Shared Phenomena**

- Users have to take and upload the pictures of the violations;
- Users query the system about violations;
- The machine gets the information about the accidents by the local municipality;
- The machine crosses information about accidents and violations to submit possible interventions, that can be both considered or not;
- The municipality can mark reported violations as fined, to allow the system to perform statistics about effectiveness of SafeStreets.



Phenomenon	Shared	Who controls it
Users recognize the type of violation that occurred	N	W
People commit violations	N	W
People perform accidents	N	W
Police emits traffic tickets	N	W
Machine manages database queries	N	M
Machine stores information (users and pictures with data) on the database	N	M
Machine manages interfaces with external software	N	M
Users have to take and upload the pictures of the violations	Y	W
Users query the system to get violations	Y	W
The machine gets the information about the accidents from the local municipality	Y	M
Machine crosses information about accidents and violations to submit possible interventions, that can be both considered or not	Y	M
The municipality can mark reported violations as fined, to allow the system to perform statistics about the effectiveness of SafeStreets.	Y	W

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

- **User:** is a general customer that use the application. It is used to refer to both an authority and a citizen.
- **Citizen:** is the basic customer of the application. He can upload violations and query the system to get statistics on a selected area.
- **Authority:** advanced customer of the application, is a registered user that is qualified to make fines and view sensitive information. Must verify himself during the registration.
- **Municipality:** is always intended as the local municipality. Every municipality provides and receives information exactly and only about its own jurisdiction.
- **Violation:** is a general infringement reported by a user.
- **Dossier:** is the instance of a violation on the system, that includes the pictures, position, classification, textual specification and fine mark.
- **Authority database:** is the database in which are stored all data about incidents. This data are uploaded by authority while the interface to access the database is offered by the municipality. Safestreets uses this interface to retrieve data from the database.

### 1.3.2 Acronyms

- RASD: Requirement Analysis and Specification Document
- LP: License Plate
- GPS: Global Positioning System
- API: Application Programming Interface
- UML: Unified Modeling Language

### 1.3.3 Abbreviations

- $[Gn]$ : n-goal.
- $[Dn]$ : n-domain assumption.

- $[Rn]$ : n-functional requirement.
- $[UCn]$ : n-use case.

## 1.4 Reference Documents

This document follows ISO/IEC/IEEE 29148:2011 and IEEE 830:1998 standard for software product specifications. All the specifications of this project have been given by Rossi and Di Nitto for the Software Engineering 2 Mandatory Project 2019-2020.

## 1.5 Revision history

- V1.0 November 10<sup>th</sup> 2019: First release
- V1.1 December 15<sup>th</sup> 2019: Minor correction to the alloy analysis, improved the analysis of the generated world. Syntax revision and correction

## 1.6 Document structure

**Chapter 1: Introduction.** A general introduction to the goals, the phenomena and the scope of the system-to-be. It aims giving general but exhaustive information about what this document is going to explain.

**Chapter 2: Overall description.** A general description of the product to be and its requirements. This section provides several information that are detailed explained in Section 3.

**Chapter 3: Specific requirements.** All software requirements are explained using scenarios, use-case diagrams and sequence diagrams. Non-functional and functional requirements are also mentioned.

**Chapter 4: Alloy.** This section includes Alloy code that describes the model and checks whether it is consistent or not.

**Chapter 5: Effort Spent.** A summary of the worked time by each member of the group.

**Chapter 6: References.** Some useful documents that we followed in order to produce this document.

# OVERALL DESCRIPTION

## 2.1 Product perspective

In the previous section, the scope of the application was delimited and explained in a shallow way, but at this point it is useful to include further details on the shared phenomena and a domain model as a visual representation of the system.

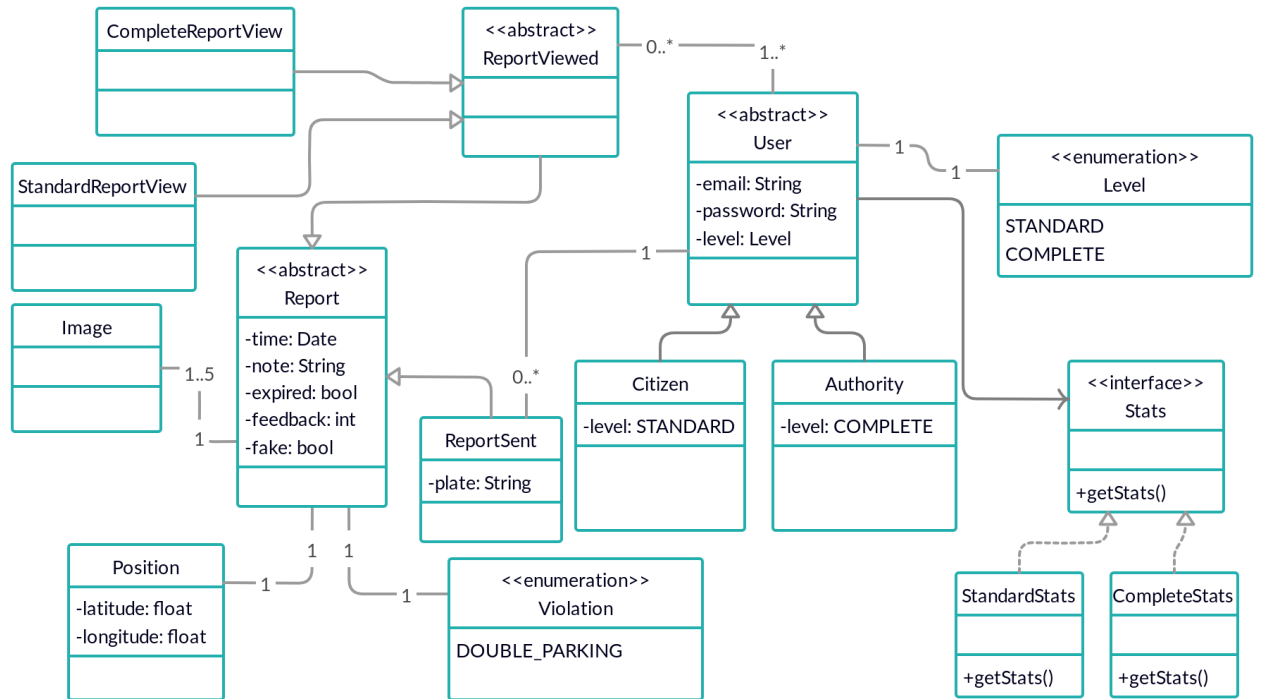


Figure 2.1: Domain Model *Class Diagram*

### Generic violation report:

To understand the main events happening in the system, it is useful to use statechart diagrams (described in the figure below). At the beginning each report (i.e. when a user uploads an image) is considered as unprocessed. While the constraints are checked (there must be at least one photo, the type of violation, the position, if it's a car violation the license plate is mandatory too as a picture), the report is on a pending state and

can become either rejected or approved. Then SafeStreets checks if the report is already present in the database but the final choice is done by the user: user can choose to add or not the new report. In the end, it's considered completed and the user is notified with the outcome. If it's been approved, the report is kept in the database.

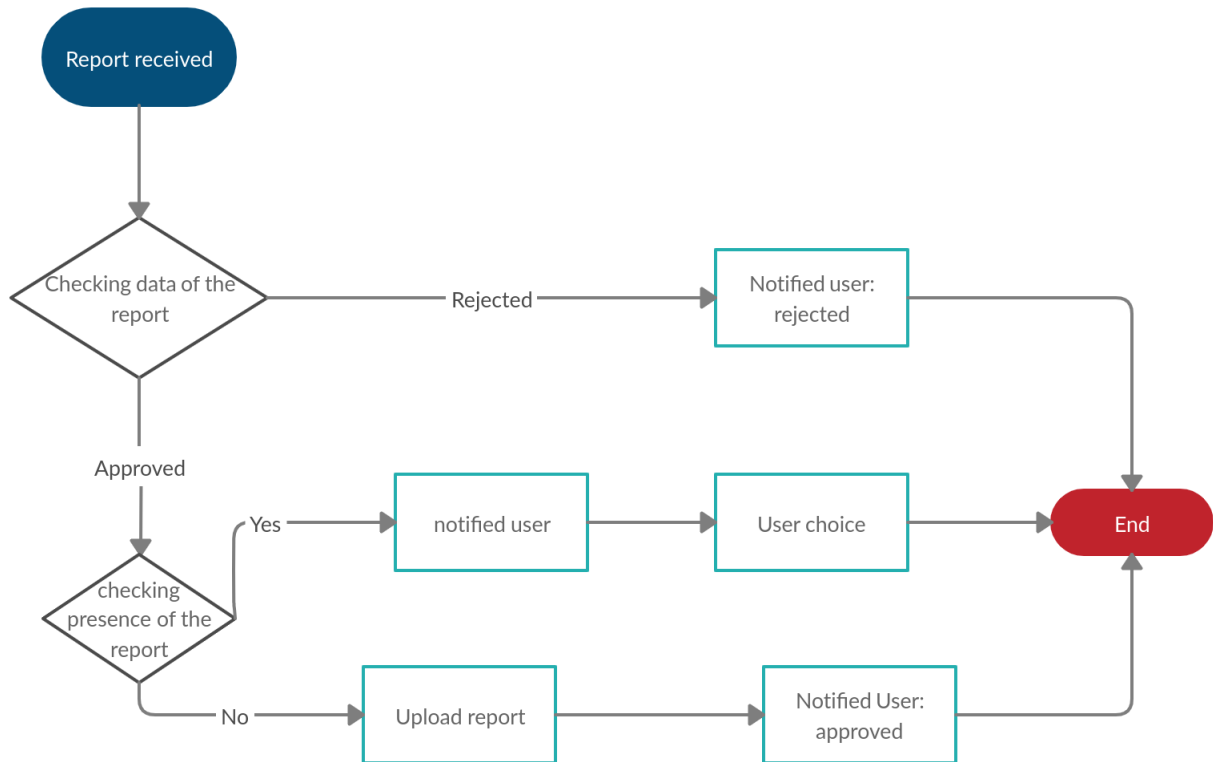


Figure 2.2: New report *State Chart Diagram*

### Query to database:

In case there is a query for get information about one selected violation, it can be made by a citizen or by an authority and this establishes different results showed by the system: in the former case SafeStreets returns information and photos about selected violation but doesn't show photos about license plates for privacy reason (system must guarantee anonymity) instead in the latter case SafeStreets returns all the photos and other information about violation so authority can use them to generate traffic tickets.

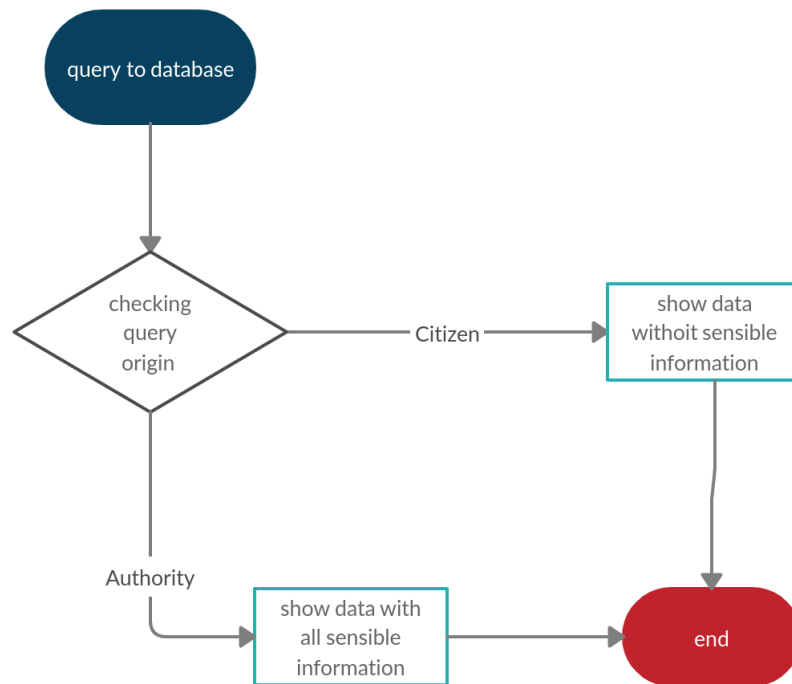
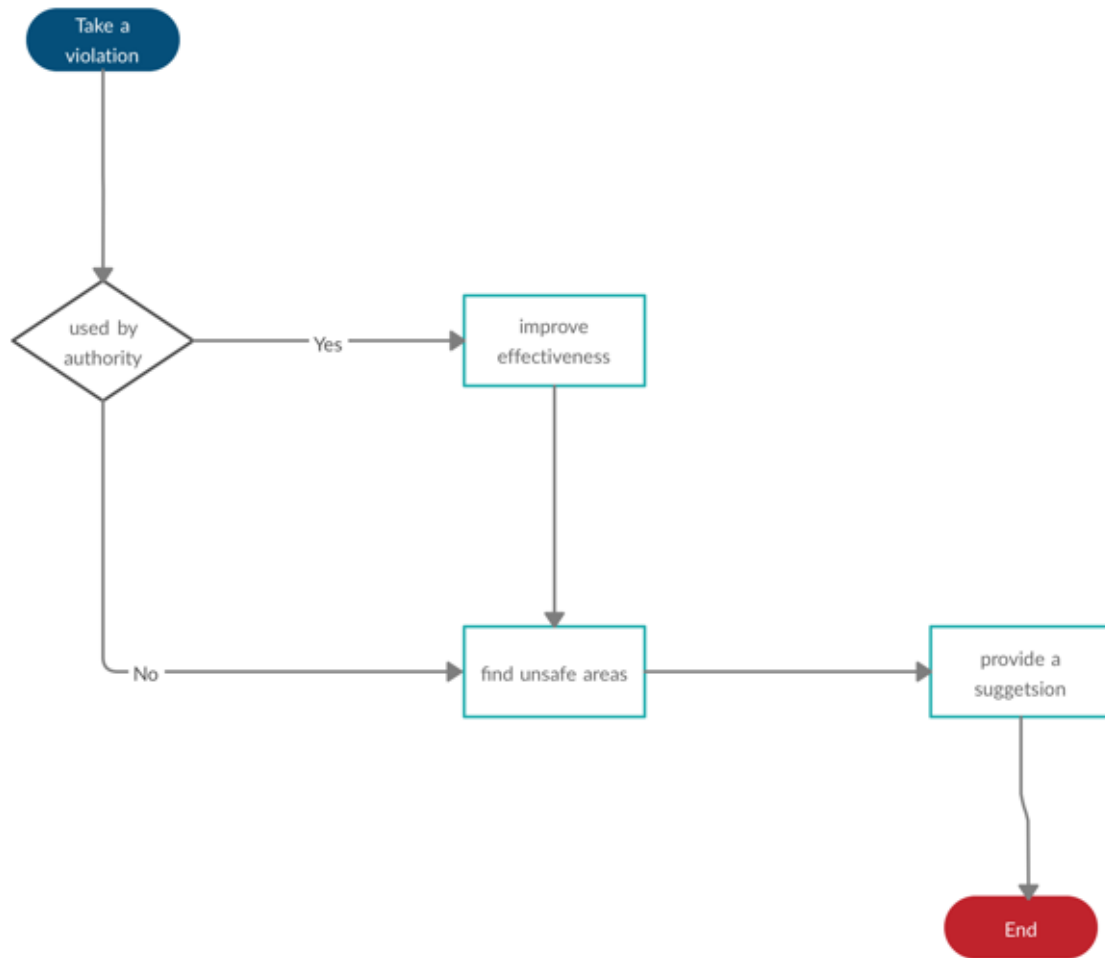


Figure 2.3: Query to database *State Chart Diagram*

### Build statistics:

When one authority uses a violation taken from SafeStreets and then he generates a traffic ticket from it, he marks the violation as fined. The system periodically builds statistics analyzing the usage of violation by the authority (marked fined or not) to show the effectiveness of the application. Furthermore, SafeStreets can access to a database of the authorities in which they upload data on incidents so it can also analyze, querying the databases, the different type of the violation and incidents and the vehicles involved. These information are both used to build statistics and to find unsafe area(also using data about accidents), in the last case system provide a suggestion for the problem.

Figure 2.4: Build statistics *State Chart Diagram*

## 2.2 Product functions

In the following we will present the main aspects and functions of the system, to put in evidence the most important requirements that will be formalized in chapter 3.

### 2.2.1 Citizens functions

A person can register himself to the SafeStreets system by providing a valid email address and a password. No other data will be mandatorily required, but the user will be allowed to set a residential location to personalize the user interface.

Every registered user that see a traffic violation can report it through SafeStreets: to do that, he must upload at least one photo of the infringement and provide some information

about it. The pictures can be only taken by a specific tool of the application that won't allow to modify the photos; this restriction is applicate to not let people to upload pictures that could have been modified to accuse someone extraneous to the burglary. The user will have to add to the images the position (that will be possible to be taken through the device GPS) and will have to indicate the category of the violation as mandatory information, while will be able to add a text description to possibly give more information. Every user will be able to consult a map that will show all the reported violations of the last 24 hours. They will even be allowed to query the application about the violations reported by the community. To do that, they will have to define one or more constraints between a marked zone on a map, a time frame and a specific category, to get all the correspondent infringement reported, respectively, in that zone, period or category. When a user will look for a violation, he will be able to see all the information uploaded for it, except for the vehicle target plates, that will be obscured because of privacy reasons. Every user will be allowed to report a violation that don't subsist, because of the description and the pictures don't match with each other. When a violation has been reported by five different people is removed from the system and is no more available. In the same way, a citizen can report one of the photos of the violation because of an affection of the privacy of another person (for instance if a license plate is visible). After five report from different people, the picture will be deleted from the dossier if at least one other photo is available; otherwise, the whole violation will be removed by the system. The user will also be able to get statistics through the application about the most committed violations in a user-defined geographic area.

### **2.2.2 Authority functions**

To register as an authority, a person must be part of the national police force. To accede to the application, he must register by entering an email address, a password that he will use to sign in, and his freshman to certificate himself. An authority can access both all the functionality of a citizen and some advanced ones. When an authority queries the system to get the violations, SafeStreets will show him all the information about the report, including the uncensored pictures uploaded by the users. This is done in order to give to the authority the possibility to fine whoever committed the crime if they can track him down. An authority can mark the dossier of a violation that he fined to let all the users know that it has been sanctioned. This information will also be used to give more accurate statistics to the users.

Moreover, SafeStreets will also provide to the authority a functionality to draw up the ranking of the most frequent offenders, for all the cases in which the system will be able to track down the licence plate from the violation.



### 2.2.3 Municipality featuring functions

If the municipality has a database that provides information about the accidents that occurs in its territory, SafeStreets can collect them to cross them with the reported violations. If the system can find a correlation between the violations and the accident it can mark a map of the most unsafe areas, identifying them with a higher level of dangerous with the increasing number of accidents. The system will also be able to verify the typology of the violations that have been reported by the user in the area that concern every accident, to provide an ad hoc solution to cope the most frequent class of violation, if this has been registered a significant number of times.

## 2.3 User characteristics

The target users of the SafeStreets system are:

- **Citizens:**

- Can register and then login to the app;
- Can upload violations;
- Can get all the violations uploaded by the other users without sensitive information;
- Can check the presence of unsafe areas;
- Can see statistics about violations for a specific area;
- Can report inappropriate pictures and dossiers;

- **Authorities:**

- Can register and then login to the app using their IDs to acknowledge ;
- Can upload violations;
- Can get all the violations uploaded by the other users;
- Can check the presence of unsafe areas;
- Can see statistics about violations for a specific area;
- Can report inappropriate pictures and dossiers;
- Can get the list of the worst offenders in the city;
- Can mark a reported violation as fined after having generate a traffic ticket;

## 2.4 Assumptions, dependencies and constraints

### 2.4.1 Domain assumptions

- [D1] GPS position collected when the report is being uploaded is sufficiently accurate and corresponds to the user location.
- [D2] SafeStreets guarantees very strong data protection and integrity against malicious external attacks.
- [D3] The map that shows all the current violations represent the topology of the city.
- [D4] The license plates attached to the cars are the original ones that are registered with the cars.
- [D5] Municipalities provide information about accidents that corresponds to real data.

### 2.4.2 Dependencies

- Traffic tickets can be emitted only if the user upload the picture of the violation with the license plate visible and recognizable by the authorities; because of that, only that kind of violations can be marked as fined.
- Users must input the right violation according to what they are reporting, otherwise statistics may be inaccurate, and it will be possible that other users will report it.
- Current date and time of a report are taken by SafeStreets from its local clock as soon as it is being uploaded, that is synchronized according UTC.

### 2.4.3 Constraints

- Smartphones must have GPS and be activated when uploading a violation.
- Smartphones must have internet connection turned on (WIFI or 3G/4G) while using the app.
- Users must accept the privacy policy in order to use the application.
- Smartphones must have at least 100Mb of available memory in order to install the app
- Smartphones must have at least 1Gb RAM in order to run the application.
- Smartphones must have a camera in order to upload violations.

- Users must have a personal email address in order to sign up and therefore use the app.

---

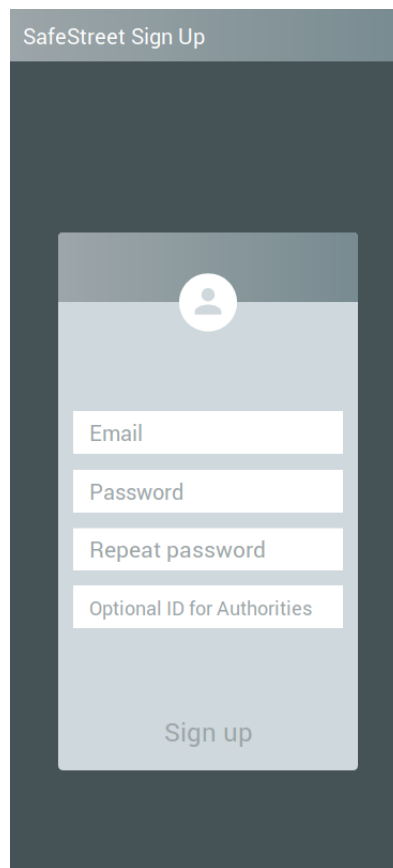
# SPECIFIC REQUIREMENTS

---

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1 External interface requirements

### 3.1.1 User Interfaces



The image shows a mobile application registration screen titled "SafeStreet Sign Up". The screen has a dark grey background. At the top, there is a header bar with the text "SafeStreet Sign Up". Below the header, there is a light grey rectangular area containing a white circular icon with a person silhouette. Underneath the icon, there are four white input fields stacked vertically, labeled "Email", "Password", "Repeat password", and "Optional ID for Authorities". At the bottom of this light grey area is a "Sign up" button.

Figure 3.1: shows an example of the registration page of SafeStreets

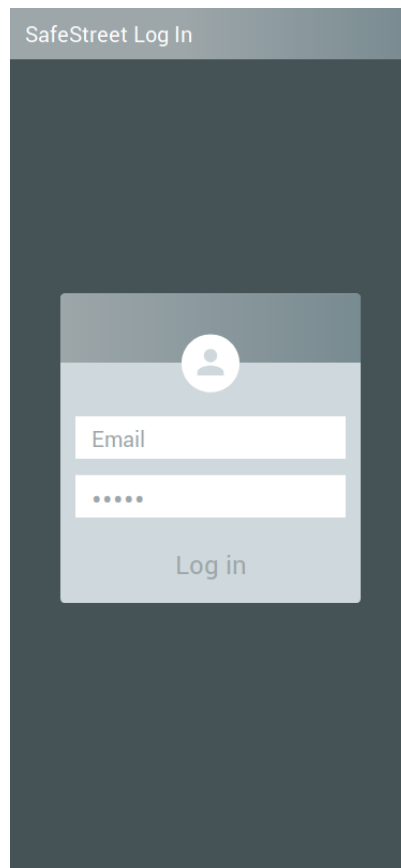


Figure 3.2: shows an example of the login page of SafeStreets

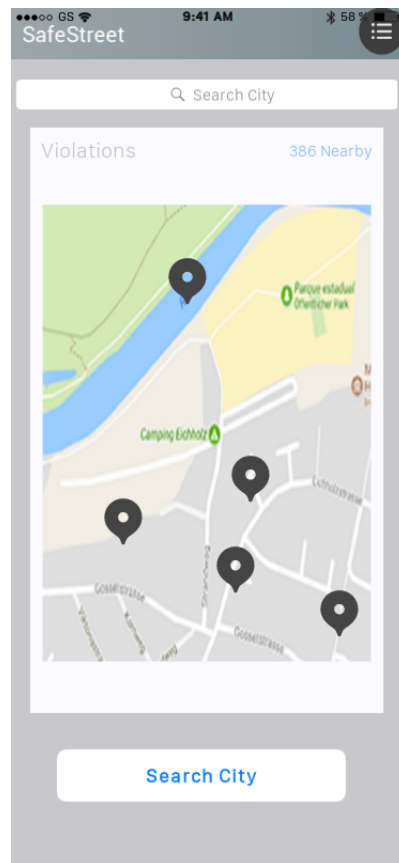


Figure 3.3: shows the homepage of the app after the login phase, displaying for example the map with all the violation reported

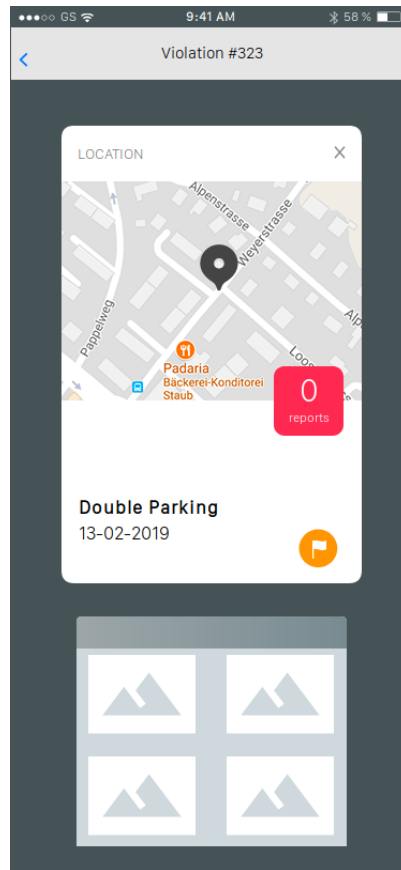


Figure 3.4: shows a violation uploaded by a user

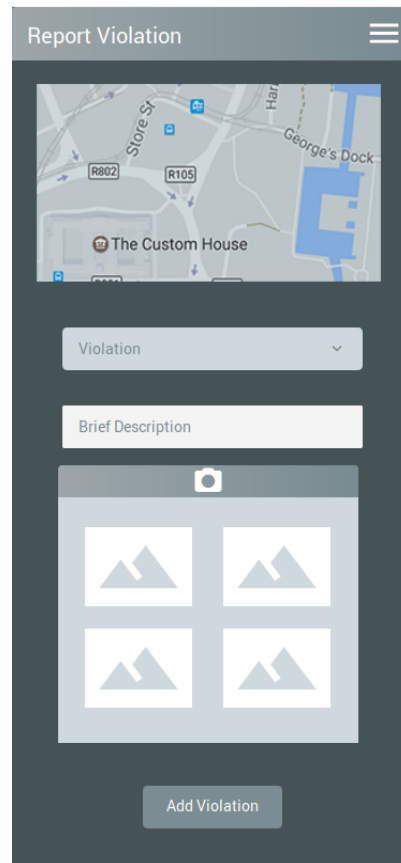
The image shows a mobile application interface for reporting violations. At the top, there is a header bar with the text "Report Violation" and a hamburger menu icon. Below the header is a map showing a street intersection with labels for "Store St", "R802", "R105", "Har", and "George's Dock". A location marker is placed on the map, and a label "The Custom House" is visible. Below the map is a dropdown menu labeled "Violation" with a downward arrow. Underneath is a text input field labeled "Brief Description". Below the text field is a camera icon and a grid of four placeholder images, each showing a mountain range. At the bottom of the form is a button labeled "Add Violation".

Figure 3.5: shows the report form, same for both citizens and authorities

### 3.1.2 Hardware Interfaces

The system has no hardware interface, because SafeStreets does not interact physically with other systems.

### 3.1.3 Software Interfaces

The system uses an API provided by the municipality that allows SafeStreets [to retrieve data from database of the authority ]to cross incidents received with its database in order to suggest possible interventions on the high-risk areas. Moreover, the system uses third party API in order to display the map of the city and decode license plates from images (in order to blur them from citizens).

### 3.1.4 Communication Interfaces

SafeStreets uses secure internet connection in order to provide a reliable exchange of messages between clients and server, like HTTPS.



## 3.2 Functional requirements

### 3.2.1 Scenarios

**Scenario 1:** Federico is a citizen who wants to report a parking violation. For this purpose, he decides to use SafeStreets. He downloads the application on his smartphone and proceeds to sign up. He is asked to insert an email and a password, and of course, being only a citizen, the authority ID is not required by the system. Federico inserts his personal email and his password, but the system tells him that the given password is shorter than 8 characters, so he tries again with a new one. Eventually he inserts a valid password, accepts the Terms and conditions and taps on “Create an Account”. He is now successfully signed up, after receiving a confirmation email by SafeStreets. He tries to login to the application by inserting the same email and password originally submitted. The system accepts the credentials and Federico is in. Now, he navigates to the report section, where he is asked to insert one or more images and select the type of violation, and optionally write a note. Federico takes and insert 2 photos, one of them also containing the license plate of the vehicle, so that the police can fine the transgressor. Once done, Federico can also see his report, through the map section.

**Scenario 2:** Evandro is a police officer whose job is to fine cars that committed violations in the city of Milan. For that purpose, he decides to use SafeStreets. He downloads the app and signs up using his work email, a strong password of 10 characters, and he inserts his ID to the system. He is now recognized as an authority and he can use all the complete functionalities of SafeStreets and has access to all the sensitive information, like the license plates uploaded by users. Once in, Evandro navigates to the map section and selects one of the violations displayed in the map, but that violation, even if it breaks the law, does not contain an image with the license plate visible, so Evandro cannot fine the transgressor.

**Scenario 3:** Erik is a police officer whose job is to fine cars that committed violations in the city of Milan. For that purpose, he decides to use SafeStreets. He downloads the app and signs up using his work email, a strong password of 10 characters, and he inserts his ID to the system. He is now recognized as an authority and can use all the complete functionalities of SafeStreets and has access to all the sensitive information, like the license plates uploaded by users. Once in, Erik navigates to the map section and selects one of the violations displayed in the map. That violation contains an image with the license plate of the transgressor. After having judged the report, he decides to fine him using a third-party system. After that, Erik marks the report as “fined”, so that

SafeStreets can then build statistics.

**Scenario 4:** Jack is a police officer that has SafeStreets installed and he is already logged in. During his work, Jack goes in the extra area for the municipality. Jack sees on the map of SafeStreets that there are some violation in the neighborhood under Jack's custody. So Jack opens one of the violations and he discovers that it's a "parking in the wrong way". Jack wants to check the gravity of the violation so he looks carefully at the three photos of the violation, he analyzes them and decides not to generate a traffic ticket because even if the report underlines a borderline situation, the car respects all the laws. So Baldo does not mark as "fined" the report.

**Scenario 5:** Tom is an old man who has downloaded SafeStreets on his smartphone. Tom wants to bike to the market, but he feels unsafe because people usually park their cars on the bicycle lane, so he decides to check on SafeStreets if any violation has been reported on the way to the market. Looking on the map, he saw a violation classified as "parking on bicycle line" and opens it to see the time of the report. When he opens the dossier, he takes a look at the picture and notices that the car is just close to the bike lane, but it does not really obstruct the passage. Tom thinks that the car is parked correctly and does not represent a violation, so he opens the feedback area and sends a negative feedback, selecting the option "false violation".

**Scenario 6:** Tom from scenario S5, during breakfast checks also another violation always on the way to his work. He opens the report on the map and analyzes the photos. The report is marked as "trash cannot be on bike line" but on the pictures there are only a car parked on zebra crossing. In this case the description doesn't match with the images and Eddy one more time goes in the report area of the violations and sends a negative feedback to SafeStreets pushing on "negative feedback" button. SafeStreets uses this feedback to build statistics.

**Scenario 7:** The municipality of Monza wants to improve its management of the traffic violations. In order to do that, they decide to be supported by SafeStreets as an immediate and low-cost solution. They offer a public database on which all the accidents in the territory are registered, to allow SafeStreets to collect data. After only two weeks SafeStreets provides the first report to Monza's municipality: the application noticed that just a few meters away from a reported accident three no parking zone violations have been reported, so it sends a notification to the municipality to suggest to add some anti-parking poles.

**Scenario 8:** Mary is a young woman who lives in Milan and has a six year-old son. She must enrol his son to the elementary school and she has already found two possible solutions, but since they are both located in an intense traffic area she is afraid that they could be in danger. The municipality of Milan provides through SafeStreets the information about all the accidents occurring on its territory, so Mary wants to use this information to take a decision. She queries the system to get all the possible unsafe areas of Milan. She finds out that her first option is located in a high-level dangerous zone: clicking on the area, she notices that three car accidents occurred at a crossroad that is located just few meters from the school building. Since her second option is in an area not marked as dangerous by SafeStreets, and where no accidents have been reported, Mary decides that she will enrol her son there.

### 3.2.2 Use Case Diagrams

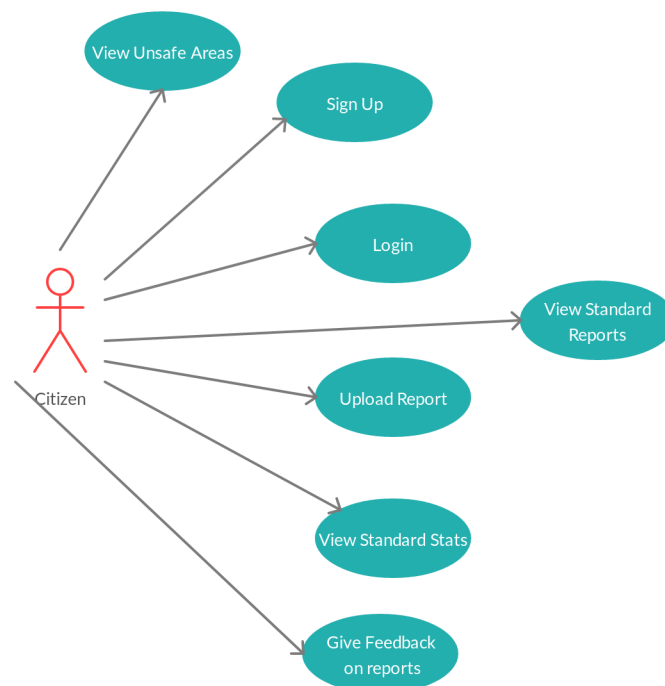
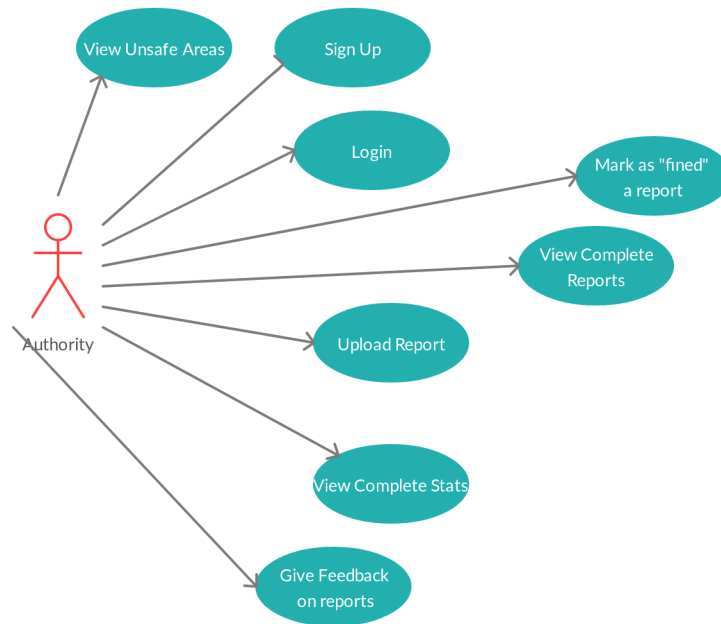
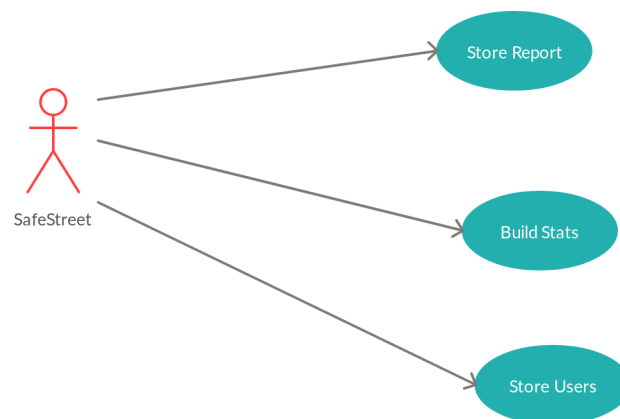


Figure 3.6: Citizen *Use Case Diagram*

Figure 3.7: Authority *Use Case Diagram*Figure 3.8: SafeStreet *Use Case Diagram*

### 3.2.3 Use Case Analysis

<b>Name</b>	<b>Citizen Signs Up</b>
Actors	<i>Citizen</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Citizen</i> taps on sign up button.</li> <li>2. <i>Citizen</i> enters his email.</li> <li>3. <i>Citizen</i> enters a new password.</li> <li>4. <i>Citizen</i> checks the “Accept terms &amp; conditions” checkbox</li> <li>5. <i>Citizen</i> taps on “create account” button.</li> <li>6. SafeStreets stores the submitted values.</li> </ol>
Exit Condition	<i>Citizen</i> successfully signs up.
Exceptions	<ol style="list-style-type: none"> <li>1. Email already exists.</li> <li>2. Invalid password.</li> <li>3. Email not inserted.</li> <li>4. "Accept terms &amp; conditions" not checked.</li> <li>5. "I am an authority" checked without inputting a valid ID.</li> </ol> <p><i>Citizen</i> is invited to try again signing up, reporting wh error(s) they have committed.</p>

Table 3.1: *UC1*

Name	Authority Signs Up
Actors	<i>Authority</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Authority</i> taps on sign up button.</li> <li>2. <i>Authority</i> enters his email.</li> <li>3. <i>Authority</i> enters a new password.</li> <li>4. <i>Authority</i> checks the “I am an authority” checkbox.</li> <li>5. <i>Authority</i> inserts his ID.</li> <li>6. <i>Authority</i> checks the “Accept terms &amp; conditions” checkbox</li> <li>7. <i>Authority</i> taps on “create account” button.</li> <li>8. SafeStreets stores the submitted values.</li> </ol>
Exit Condition	<i>Authority</i> successfully signs up.
Exceptions	<ol style="list-style-type: none"> <li>1. Email already exists.</li> <li>2. Invalid password.</li> <li>3. Email not inserted.</li> <li>4. "Accept terms &amp; conditions" not checked.</li> <li>5. "I am an authority" checked without inputting a valid ID.</li> </ol> <p><i>Authority</i> is invited to try again signing up, reporting wh error(s) they have committed.</p>

Table 3.2: *UC2*

Name	User Signs In
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed.
Events Flow	<ol style="list-style-type: none"><li>1. <i>User</i> enters his email.</li><li>2. <i>User</i> enters a new password.</li><li>3. <i>User</i> taps on “Log In” button.</li><li>4. SafeStreets checks <i>User</i> credentials.</li></ol>
Exit Condition	<i>User</i> successfully logged in.
Exceptions	<ol style="list-style-type: none"><li>1. Invalid Email.</li><li>2. Invalid password.</li></ol> <p><i>User</i> is invited to try again to log in, reporting wh error(s) they have committed.</p>

Table 3.3: *UC3*

Name	User reports a violation
Actors	<i>Citizen, SafeStreets</i>
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> clicks on “new report” button.</li> <li>2. <i>User</i> takes at least one photo of the violation.</li> <li>3. <i>User</i> selects the type of the violation and fills the optional note.</li> <li>4. <i>SafeStreets</i> checks if the violation is already in the database.</li> <li>5. <i>User</i> clicks on "confirm &amp; send" button.</li> <li>6. SafeStreets stores the newly collected violation in its database.</li> </ol>
Exit Condition	<i>SafeStreets</i> stores the new report.
Exceptions	<ol style="list-style-type: none"> <li>1. No photo uploaded.</li> <li>2. Type of violation is missing.</li> </ol>

Table 3.4: *UC4*



Name	Feedback by a user
Actors	<i>Citizen, SafeStreets</i>
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"><li>1. <i>User</i> navigates to the map.</li><li>2. <i>User</i> selects one violation.</li><li>3. <i>User</i> clicks on “thumb down” button.</li><li>4. <i>User</i> clicks on “confirm &amp; send” button.</li></ol>
Exit Condition	<i>SafeStreets</i> stores the new feedback.
Exceptions	<ol style="list-style-type: none"><li>1. none.</li></ol>

Table 3.5: *UC5*

Name	Mark a report as “fined”
Actors	<i>Authority</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and Authority signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> signs in as authority.</li> <li>2. <i>User</i> opens the violation map.</li> <li>3. <i>User</i> clicks on a violation.</li> <li>4. <i>User</i> clicks on “mark as fined” button.</li> <li>5. SafeStreets shows a fined flag on the violation’s dossier.</li> </ol>
Exit Condition	<i>SafeStreets</i> marks the violation as fined.
Exceptions	<ol style="list-style-type: none"> <li>1. The violation was already marked as “fined”.</li> </ol>

Table 3.6: *UC6*

Name	Violation query by a user
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the query violation page.</li> <li>2. <i>User</i> inserts a city that want to query .</li> <li>3. <i>User</i> selects a starting date from a calendar.</li> <li>4. <i>User</i> selects some class of violation from a list.</li> <li>5. <i>User</i> runs the query.</li> </ol>
Exit Condition	<i>SafeStreets</i> show all the violation within the parameters, sorted by date.
Exceptions	<ol style="list-style-type: none"> <li>1. The city chosen doesn't exist.</li> <li>2. No violation found for the chosen parameters.</li> </ol>

Table 3.7: *UC7*

Name	Display violation on the map
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the map page.</li> <li>2. The application retrieves the user position .</li> <li>3. The application retrieves all the violations uploaded and still valid.</li> <li>4. The application displays the map with all the violations, marking the position of the user.</li> <li>5. <i>User</i> taps on the violation he wants to check.</li> <li>6. The application will show another page with the images and the type of violation reported</li> </ol>
Exit Condition	<i>SafeStreets</i> shows the map with all violations.
Exceptions	<ol style="list-style-type: none"> <li>1. The city chosen doesn't exist.</li> <li>2. No violation found for the chosen parameters.</li> </ol>

Table 3.8: *UC8*

Name	View unsafe areas
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the map page.</li> <li>2. The application retrieves the user position.</li> <li>3. <i>User</i> taps on “view unsafe areas” checkbox.</li> <li>4. SafeStreets query the database and sends back all unsafe areas.</li> <li>5. The map displays with a colour the most dangerous areas near the user.</li> </ol>
Exit Condition	<i>SafeStreets</i> displays on the map the unsafe areas.
Exceptions	<ol style="list-style-type: none"> <li>1. No unsafe areas found.</li> </ol>

Table 3.9: *UC9*

Name	View statistics on crime
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the map page.</li> <li>2. <i>User</i> pushes button crimes.</li> <li>3. <i>User</i> selects search filters: area of the map or type of crime.</li> <li>4. SafeStreets shows the most committed violations in selected area or the “unsafe area” for selected crime.</li> </ol>
Exit Condition	<i>SafeStreets</i> displays searching results.
Exceptions	<ol style="list-style-type: none"> <li>1. No violations found for chosen one.</li> <li>2. Selected area of the map doesn’t exist.</li> </ol>

Table 3.10: *UC10*

Name	View statistics on SafeStreets
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the statistic page.</li> <li>2. <i>User</i> selects effectiveness button of SafeStreets.</li> <li>3. Display shows the number of violations and the number of violations marked as “fined”.</li> </ol>
Exit Condition	<i>SafeStreets</i> displays searched data.
Exceptions	<ol style="list-style-type: none"> <li>1. No violations found for chosen one.</li> <li>2. Selected area of the map doesn’t exist.</li> </ol>

Table 3.11: *UC11*

<b>Name</b>	<b>User edits a report</b>
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the report history page.</li> <li>2. <i>User</i> selects one report.</li> <li>3. Display shows the results.</li> <li>4. <i>User</i> can change fields of report or adds/removes images.</li> <li>5. <i>user</i> presses on confirm button to confirm the changes.</li> </ol>
Exit Condition	<i>SafeStreets</i> stores the new edit report.
Exceptions	<ol style="list-style-type: none"> <li>1. User doesn't select a type of violations.</li> <li>2. User remove all the photos and adds none.</li> <li>3. User doesn't fill all the mandatory fields.</li> </ol>

Table 3.12: *UC12*



<b>Name</b>	<b>User removes a report</b>
Actors	<i>User</i> , SafeStreets
Entry Conditions	SafeStreets successfully installed and User signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the report history page.</li> <li>2. <i>User</i> selects one report.</li> <li>3. Display shows the results.</li> <li>4. <i>User</i> can change fields of report or adds/removes images.</li> <li>5. <i>user</i> presses on remove button to confirm the changes.</li> </ol>
Exit Condition	<i>SafeStreets</i> deletes the selected report from its database.
Exceptions	<ol style="list-style-type: none"> <li>1. none.</li> </ol>

Table 3.13: *UC13*

<b>Name</b>	<b>User reports something that is already stored</b>
Actors	<i>User</i> , <i>SafeStreets</i>
Entry Conditions	<i>SafeStreets</i> successfully installed and <i>User</i> signed in.
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> clicks on “new report” button.</li> <li>2. <i>User</i> takes at least one photo of the violation.</li> <li>3. <i>User</i> selects the type of the violation and fills the optional note.</li> <li>4. <i>User</i> clicks on “confirm &amp; send” button.</li> <li>5. <i>SafeStreets</i> checks if the violation is already in the database.</li> <li>6. <i>SafeStreets</i> shows the violation to the user.</li> <li>7. <i>User</i> can choose to push either “same” button or “other” button referred to the violation.</li> <li>8. <i>User</i> selects “same” button.</li> <li>9. <i>User</i> is notified through a notification by <i>SafeStreets</i>.</li> </ol>
Exit Condition	<i>SafeStreets</i> doesn’t store the report on its database.
Exceptions	<ol style="list-style-type: none"> <li>1. No photo uploaded.</li> <li>2. Type of violation is missing.</li> </ol>

Table 3.14: *UC14*

### 3.2.4 Sequence Diagrams

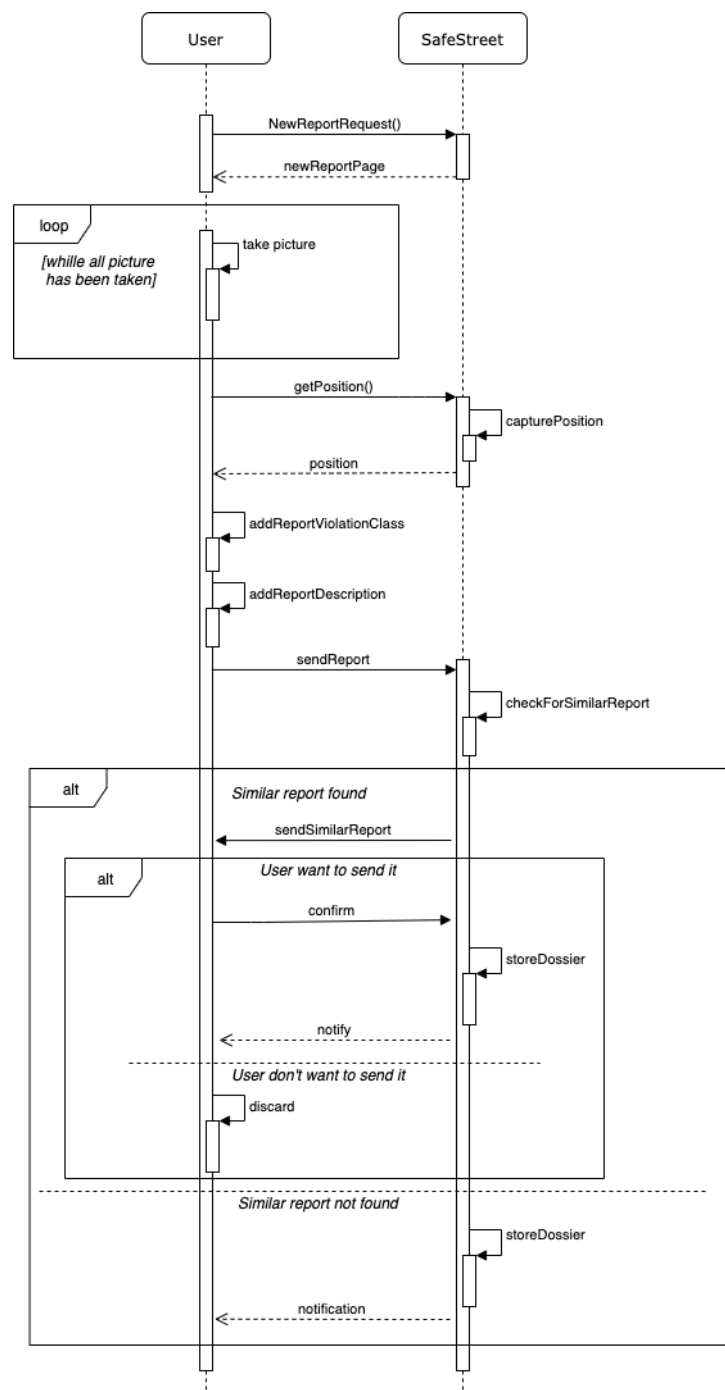
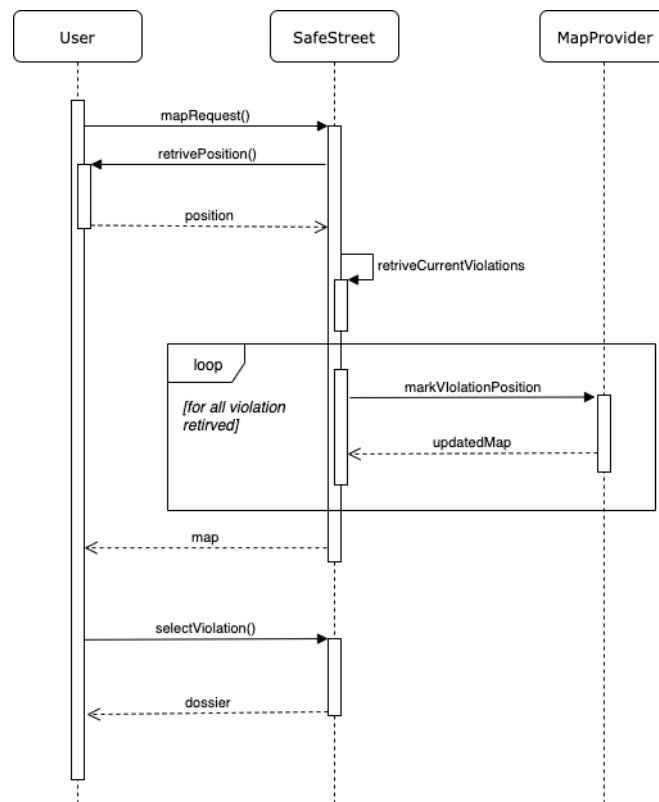
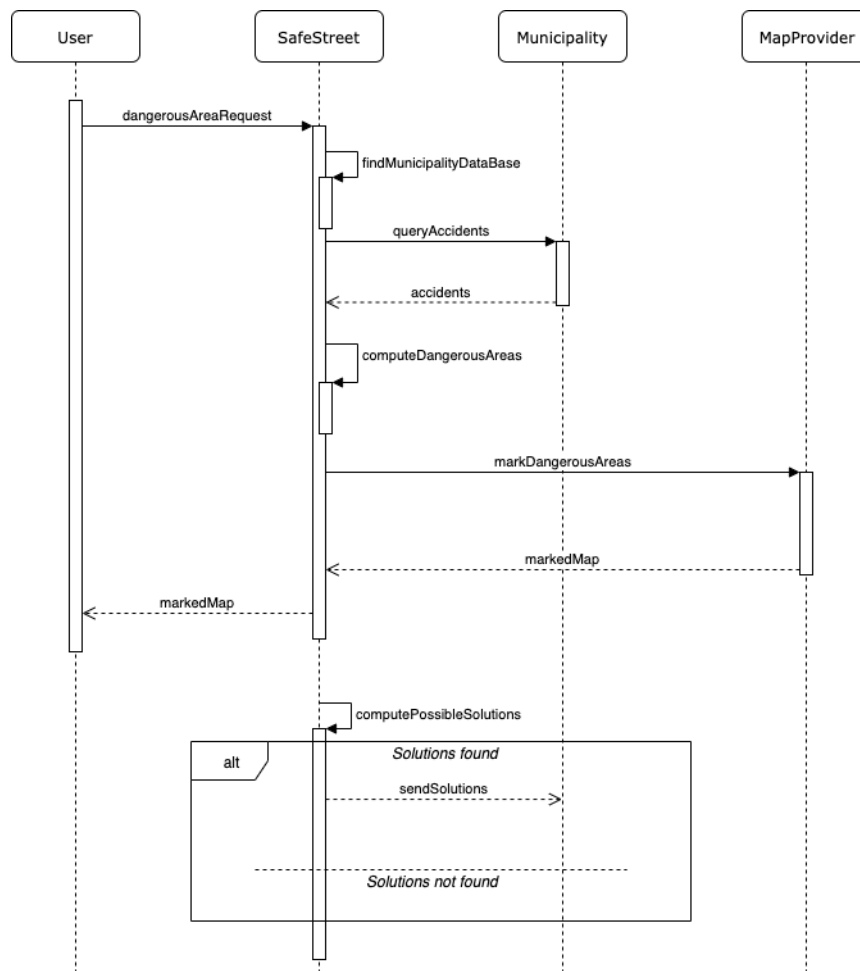
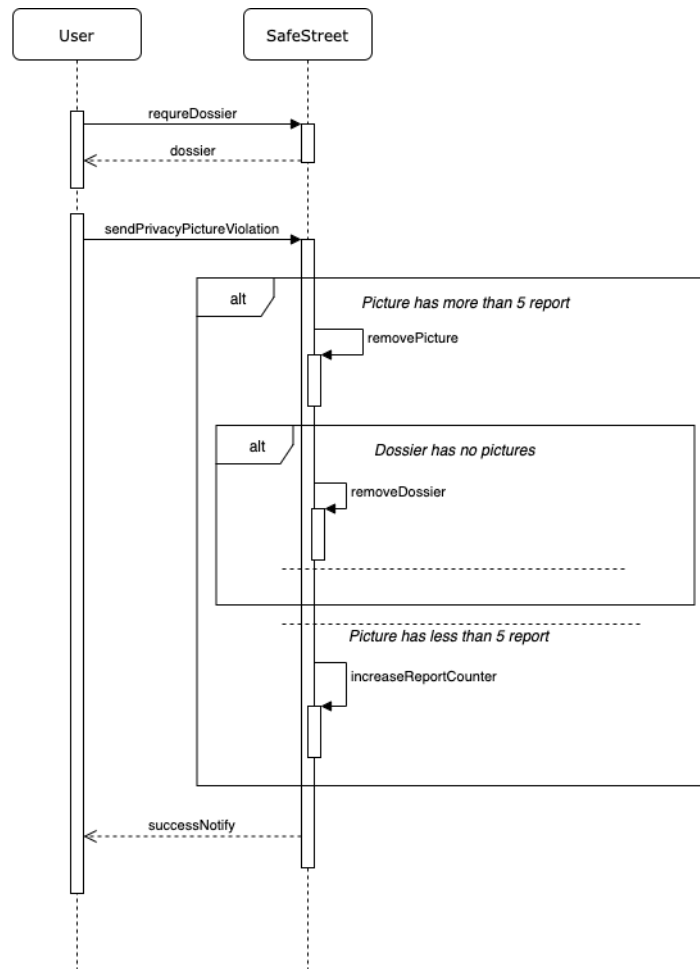


Figure 3.9: Send Report *Sequence Diagram*

Figure 3.10: Get Violations *Sequence Diagram*

Figure 3.11: Get Dangerous Areas *Sequence Diagram*

Figure 3.12: Send Feedback *Sequence Diagram*

### 3.2.5 Requirements, Domain Assumptions, Goals Matrix

The following is a list of requirements for SafeStreets.

- [R1] Unregistered user cannot use SafeStreets.
- [R2] In signing up, users must provide an email and a password.
- [R3] In signing up, an authority must also provide an ID.
- [R4] In signing up, users must accept “terms & conditions”.
- [R5] SafeStreets identifies a user by its email.
- [R6] SafeStreets collects user data in its database.
- [R7] SafeStreets stores all reports in its database.
- [R8] SafeStreets retrieves reports from its database.
- [R9] SafeStreets authenticates a user when tries to log in.
- [R10] Users can view the reports in the map.
- [R11] Authorities can see all the sensitive information contained in a report.
- [R12] Citizens cannot see sensitive information in a report.
- [R13] Users can view unsafe areas in the map.
- [R14] Users can delete reports they submitted within a day.
- [R15] Users can edit reports they submitted within a day.
- [R16] Users can upload valid reports.
- [R17] A report must contain at least one image.
- [R18] A report must indicate the type of violation.
- [R19] A report is valid if and only if R17 and R18 are satisfied.
- [R20] Images containing sensitive information (like license plates) must be blurred by the system.
- [R21] The system must notify the user if the report submitted is not valid.
- [R22] For each type of statistics there must be at least user from which SafeStreets takes data.

- [R23] A user can access to feedback area for each report.
- [R24] A user can select negative feedback for each report.
- [R25] Authorities can see suggestions for possible interventions.
- [R26] During adding a new report, if there is a report of same type, in the same position and at the same date, SafeStreets shows the report stored to the user if the violation is the same.
- [R27] Authorities can mark a report as “fined”.
- [R28] User can query SafeStreets to achieve a specific report.
- [R29] SafeStreets retrieves incidents from municipality database.

The following is an analysis, for each goal of the system to be, of the requirements  $R_n$  and the domain assumptions  $D_n$  that satisfy the goal itself and of these cases relative to it.

- [G1] The application will allow users to upload pictures of the traffic violations, including pictures, date, time, classification, and optionally a textual description.

[R] 1,2,3,4,5,6,9,14,15,16,17,18,19,20,21,26

[D] 1,2,4

- Requirements are about the correctness of the reports while the domain assumptions are about the gps position and the license plate.

- [G2] The application will allow users to view a map of the violations registered in the last day.

[R] 1,,2,3,4,5,6,7,8,10,11,12,17,18,28

[D] 1,3

- Requirements are about the correctness of the reports while the domain assumptions are about correctness of the map.

- [G3] The application will allow users to see the violations uploaded by other users; however, citizens won't be able to see sensitive pictures, while authorities will be allowed.

[R] 1,2,3,4,5,6,7,8,11,12,17,18,19,20

[D] 2,3,4



- Requirements are about the correctness of the reports and the different level of visibility of citizen and authority while the domain assumptions are about the correctness of the map.

[G4] Users will be able to query the system to get all the reported violations that correspond to some temporal, geographical or categorical parameters.

[R] 5,6,7,8,12,17,18,19,22,29

[D] 2,4

- Requirements are about the correctness of the reports and the query on the database while the domain assumptions are about the integrity of the data stored.

[G5] Users can give a feedback about violations uploaded by other users to SafeStreets.

[R] 1,2,3,4,5,6,7,8,23,22,24

[D] 2,4

- Requirements are about the correctness of the reports and feedback area of the user while the domain assumptions are about the the integrity of data stored.

[G6] If SafeStreets can get the information about accidents by the municipality, it will give the possibility to merge them with its data to identify potentially unsafe areas.

[R] 7,8,13,16,17,18,19,22,29

[D] 3,5

- Requirements are about the correctness of the reports and the capability of SafeStreets to find unsafe area while the domain assumptions are about the consistency of data on accidents.

[G7] If SafeStreets can get the information about accidents by the municipality, it will give the possibility to merge them with its data to suggest possible interventions.

[R] 5,6,7,8,17,18,19,20,22,25,29

[D] 3,5

- Requirements are about the correctness of the reports and capability of SafeStreets to build statistics while the domain assumptions are about the consistency of data on accidents.

[G8] The system will use the information about violations, accidents and fines that will collect by both users and authorities to build statistics.

[R] 5,6,7,8,17,18,19,23,24,22,27,29

[D] 2,5

- Requirements are about the correctness of the reports and capability of SafeStreets to build statistics while the domain assumptions are about consistency of data on accidents.

#### Traceability matrix:

Use case	Requirements
[1] Citizen signs up	1,2,4,5,6
[2] Authority signs up	1,2,3,4,5,6
[3] User signs in	1,5,9
[4] User reports a violation	7,16,17,18,19,20,21,22
[5] Feedback by a User	7,10,11,12,20,23,24
[6] Mark a report as “fined”	7,11,12,17,18,19,20,27
[7] Violation query by a user	7,8,17,18,19,20,28
[8] Display violation on the map	7,8,10,11,12,17,18,19
[9] View unsafe areas	7,8,11,12,13,29
[10] View statistics on crime	7,8,11,12,20,22
[11] View statistics on SafeStreets	7,8,11,12,20,22
[12] User edits a report	7,8,15,16,17,18,19,21
[13] User removes a report	7,8,14,16,17,18,19,21
[14] User reports something that is already stored	7,16,17,18,19,20,21,26

### 3.3 Performance requirements

The system must be able to serve a great number of users at the same time. It must guarantee that the query of the system by a user will provide the history of the violation in a short time, even with a big amount of report. For every image uploaded for a report, the system must allow a file dimension of at list 5 Mb. In the map representation of the violation, the position of a report must be no more than 10 meters away from the actual position indicated.

### 3.4 Design constraints

#### 3.4.1 Standard compliance

Since the system will show to the user pictures taken by other people, the application refers to the European General Data Protection Regulation (GDPR) as regards the privacy protection. That means that no licence plate will be shown to the users if they have not an authority level account.

#### 3.4.2 Hardware limitations

The application works independently of the hardware on which is running. However, the availability of some functionality will be dependent on the device used and its components. Internet connection will be required to use the application, since all the functionality are based on it.

It will be possible to upload a report only if the device is provided by a camera, since the violation's photos must be taken within the context of the application, and will not be possible to upload a saved picture. The GPS will either set the position of a report to upload or the reports violation map automatically.

## 3.5 Software system attributes

### 3.5.1 Reliability

The system must run continuously without interruption. Since the application require the user to take and send the pictures in real time, the operation of uploading a report is the most critical and is required to work without roll back, because it could not be possible for the user to retake them. For this situation, the system must also give the possibility to re-upload the same picture taken if something goes wrong while sending them to the server.

The system has to guarantee the data integrity as much as possible, for example by duplicating the database used to store violations. If the availability of date couldn't be provided anyway (for example, because of missing internet connection or server connection error) the application must notify the user by throwing an alert.

### 3.5.2 Availability

Since the application work in real time for the upload of a violation, the system must reduce the unavailability as much as possible. The error management system described before must be protected to expect SafeStreets to be available 99.5 % of the time.

### 3.5.3 Security

The data provided by the users to join the application (email, password, and eventually an ID) are sensitive information and must be protected. SafeStreets must never provide that information in any way to any person. To prevent information stealing, an encryption technique must be used while exchanging data with a user. Moreover, all necessary measures to avoid external or internal attacks to the databases must be taken.

### 3.5.4 Maintainability

The developing of the application must allow future modification or fixing. The system could be expanded with new features in the future, so the software must be also suitable to new functionalities. To reach those goals, the software must be well commented, readable and use appropriate design patterns.

### 3.5.5 Portability

The application main functionalities work in real time and outdoors, so the application will be required to work on smartphones. Moreover, since it must be used by many

different people, it will must be suitable to all the most important mobile operative systems. To be able to reach more people, SafeStreets will have to be compatible to as many devices and technologies as possible.

# FORMAL ANALYSIS USING ALLOY

This section presents a possible Alloy formalization of the proposed system. This Alloy model illustrates the key concepts and entities that make up the system and the relationships between them, and is to be seen as an attempt of capturing the systems essential features. This model has the purpose of verifying if the properties defined for the system are possible to satisfy and if there are no constraints being violated. Although this model is a simplified version of the real world, it is enough to show that the model stands in the scope of the project.

## 4.1 Alloy Model

```

open util/integer
open util/boolean

sig string {}

sig ViolationType{ }

// Every image must have exactly one ID and one feedback counter
sig Image {
    imageID: one Int,
    negativeFeedback: one Int
}{imageID > 0}

// Every report must have exactly one ID, one type, one date, one time,
//one fine mark and one feedback counter, at least one image and at most one description
sig Report {
    numberID: one Int,
    type: one ViolationType,
    description: lone string,
    images: some Image,
    position: one Position,
    date: one Date,
    time: one Time,
    fineMark: one Bool,
    negativeFeedback: one Int
}{numberID > 0}

// Every user must have exactly one email, one password, one level and one position
abstract sig User {
    email: one string,
    password: one string,
    reportsUploaded: set Report,

```

```

    level: one Level,
    position: one Position
}

sig Citizen extends User { }

// Every authority must have exactly one ID
sig Authority extends User {
    id: one Int
} {id > 0}

sig Date { }

sig Time { }

// Every position must have exactly one latitude and one longitude
sig Position {
    latitude: one Int,
    longitude: one Int
}{latitude > 0 longitude > 0}

// Every daily map must have exactly one date
sig DailyMap{
    dailyViolation: set Report,
    day: one Date
}

enum Level {
    Standard,
    Complete
}

// Two users can not be registered with the same email
fact noUserSameEmail {
    no disj u1, u2 : User | u1.email = u2.email
}

// Two reports can not have the same ID
fact noReportSameIdOrSamePosAndDate {
    no disj r1, r2 : Report | r1.numberID = r2.numberID
}

// Two images can not have the same ID
fact noImageSameId {
    no disj i1, i2 : Image | i1.imageID = i2.imageID
}

// Two authority can not have the same Authority ID
fact noAuthoritySameId {
    no disj a1, a2 : Authority | a1.id = a2.id
}

// Every date must be associated with exactly one daily map
fact allDateWithinOneDailyMap {
    all d1 : Date | one dm1: DailyMap | d1 in dm1.day
}

```

```

// Every time must be associated with at least one report
fact timeOnlyWithinReport {
    all t1 : Time | some r1 : Report | t1 in r1.time
}

// Every type of violation can exist only within a report
fact violationTypeOnlyWithinReport {
    all t1 : ViolationType | some r1 : Report | t1 in r1.type
}

// Every report must be associated with exactly one user
fact reportOnlyWithinReporter {
    all dossier : Report | one user : User | dossier in user.reportsUploaded
}

// Every image must be associated to exactly one report
fact ImageOnlyWithinReport {
    all i1 : Image | one r1 : Report | i1 in r1.images
}

// Every position can exist only within a user or a report
fact positionOnlyWithinUserOrReport {
    all p1 : Position | some r1 : Report, u1 : User | p1 in r1.position or p1 in u1.position
}

// Different locations have a different latitude or longitude
fact differentLocationsHasDifferentParameters {
    no disj p1, p2 : Position | (p1.latitude = p2.latitude and p1.longitude = p2.longitude)
}

// Every authority is associated with a complete authorization level
fact giveRightLevelsToUsers {
    all a1 : Authority | a1.level = Complete
}

// Every citizen is associated with a standard authorization level
fact giveRightLevelsToUsers2 {
    all c1 : Citizen | c1.level = Standard
}

// User's email can not be his password
fact emailDifferentFromPassword {
    all u1 : User | u1.email ≠ u1.password
}

// An image must have more at least 0 and at most 4 negative feedbacks
fact noImageWithMoreThanFiveOrNegativeFeedback{
    all image: Image | image.negativeFeedback < 5 and image.negativeFeedback ≥ 0
}

// A report must have at least 0 and at most 4 negative feedback
fact noReportWithM0reThanFiveOrNegativeFeedback{
    all dossier: Report | dossier.negativeFeedback < 5 and dossier.negativeFeedback ≥ 0
}

```



```

// A daily map contains only report with its same date
fact dailyMapHasOnlyTodayReport{
  all map : DailyMap, dossier : Report | dossier in map.dailyViolation
  iff dossier.date = map.day
}

// Every report is reported by exactly one user
fact dossierReportedByExactlyOneUser{
  no disj u1, u2 : User | some violation : Report | (violation in u1.reportsUploaded) and
  (violation in u2.reportsUploaded)
}

// Every daily map is associated to a different date
fact noTwoMapWithSameDay{
  no disj m1, m2 : DailyMap | m1.day = m2.day
}

// Every report must be associated to a daily map
fact everyReportWithinADailyMap{
  all r1: Report | one dm1: DailyMap | r1.date in dm1.day
}

// Every sting can exist only within a description, an email or a password
fact stringOnlyWithinContext{
  all s1: string | some r1: Report, u1: User | s1 in r1.description or s1 in u1.email
  or s1 in u1.password
}

pred world1 {
  #Report = 2
  #Image = 4
  #Citizen = 1
  #Authority = 1
  (some disj i1, i2: Image | some disj r1, r2: Report | i1 in r1.images and i2 in r2.images
  and r1 in Citizen.reportsUploaded and r2 in Authority.reportsUploaded)
}

run world1 for 4 but 1 DailyMap

pred world2 {
  #DailyMap = 2
  #Report = 3
  (some disj r1, r2: Report | some disj m1, m2: DailyMap | r1.date = m1.day and r2.date = m2.day)
}

run world2 for 4

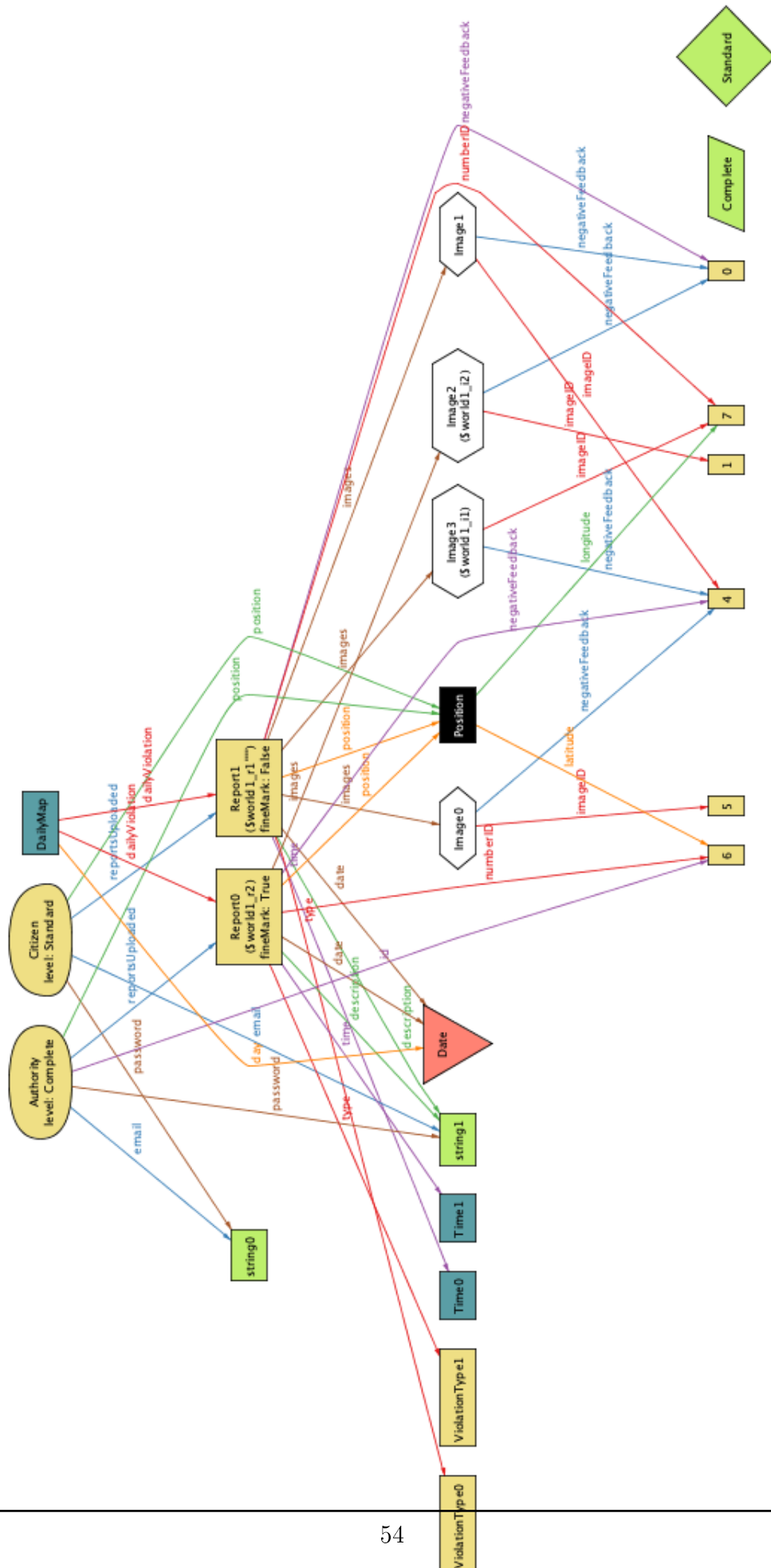
pred world3 { }

run world3 for 5

```

## 4.2 Generated World

After running the model described above, different worlds were generated. Because of simplicity, strings and integers have not been distinguished for different contexts. Even if this could seem an excessive simplification, it is actually a possible situation, since some data could be the same for different contexts.

Figure 4.1: Generated world 1. *Alloy*

From the first world generated we can see that every report must belong to a user. Every report must contains all the mandatory information: violation type, location, date, time, user itself and the images are correctly generated. Moreover, we can see the each report can contains more than one picture.

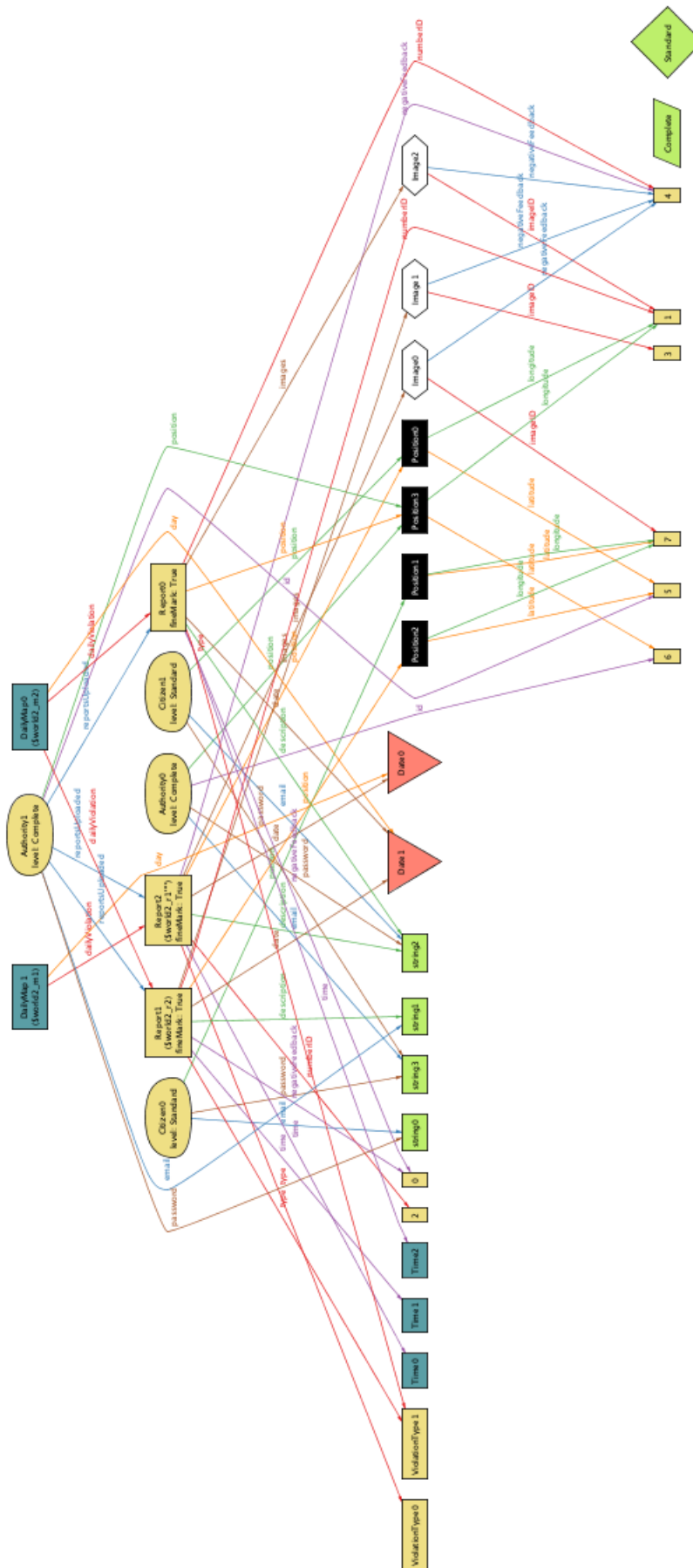
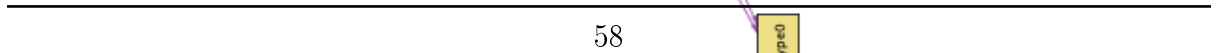


Figure 4.2: Generated world 2, *Alloy*

In this second world is showed the important correspondence between a daily violation map and its report: it is possible to notice that all the report are assigned to the daily map that correspond to their upload date (it may be useful to remember that the upload date is the actually date of the violation, since the report must be uploaded in real time)



In this last picture, we can see a generic representation of the system. It has been generated with no constrains, and it is possible to notice that it is consistent with all the assumption done in this document.



# EFFORT SPENT

Team Work	
Task	Hours
Understanding the problem	3
Brainstorming	1
World, Machine and Shared phenomena	3
Definitions, acronyms, abbreviations	0.5
Software system attributes	2
Alloy coding	3
Checking document	3
<b>Total</b>	<b>20</b>

Table 5.1: Time spent by all team members

Individual Work					
Morreale Federico		Maddes Evandro		Innocente Federico	
Task	Hours	Task	Hours	Task	Hours
World phenomena	2	Goals	3	Goal	1
Introduction	2	Machine phenomena	2	Shared phenomena	0.5
Use Case Diagrams	1	Product perspective	3	Product functions	3
Class Diagram	2	State chart diagrams	1	Scenarios	3
User Characteristics	2	Scenarios	2	UC description	1
UC description	4	UC description	5	Sequence Diagrams	4
Domain Assumptions	2	Requirements	1	Design Constrains	1
Interfaces	3	Traceability matrix	2	Software system attributes	1.5
Constraints	2	Layout	5	Alloy Model	4
Alloy Model	7	Alloy description	0.5		
<b>Total</b>	<b>24</b>	<b>Total</b>	<b>24.5</b>	<b>Total</b>	<b>24.5</b>

Table 5.2: Time spent by each team member

# REFERENCES

---

- SafeStreets Mandatory Project Assignment.pdf, Version 1.0 - AA 2019/2020
- 5.b Structure of RASD.pdf by professors Rossi and Di Nitto, Politecnico di Milano