



**POLITECNICO**  
**MILANO 1863**

M.Sc. Computer Science and Engineering  
Software Engineering 2 Project

# Requirements Analysis and Specification Document



Ferrara Fabiana, Formicola Stefano, Guerra Leonardo

9th December 2019

GitHub Repository: <https://github.com/ste7en/FerraraFormicolaGuerra>

Version 1.2

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Purpose . . . . .	3
1.1.1	Goals . . . . .	3
1.2	Scope . . . . .	4
1.2.1	World and Machine phenomena . . . . .	5
1.3	Definitions, Acronyms, Abbreviations . . . . .	5
1.3.1	Definitions . . . . .	5
1.3.2	Acronyms . . . . .	6
1.3.3	Abbreviations . . . . .	7
1.4	Revision History . . . . .	7
1.5	Reference Documents . . . . .	8
1.6	Document Structure . . . . .	8
<b>2</b>	<b>Overall Description</b>	<b>9</b>
2.1	Product Perspective . . . . .	9
2.1.1	SafeStreets . . . . .	9
2.2	Product Functions . . . . .	10
2.2.1	User registration . . . . .	10
2.2.2	User report acquisition . . . . .	11
2.2.3	User contribution visualization . . . . .	11
2.2.4	Public statistics visualization . . . . .	11
2.2.5	Municipality access . . . . .	11
2.2.6	User report evaluation . . . . .	11
2.2.7	Detailed statistics visualization . . . . .	11
2.2.8	Possible interventions visualization . . . . .	11
2.2.9	License plate recognition data acquisition . . . . .	11
2.2.10	Request Accidents data . . . . .	12
2.2.11	Request User report data . . . . .	12
2.3	User Characteristics . . . . .	12
2.4	Assumptions, Dependencies and Constraints . . . . .	12
2.4.1	Domain Assumptions . . . . .	12
2.4.2	Dependencies . . . . .	13
2.4.3	Constraints . . . . .	13

<b>3 Specific Requirements</b>	<b>14</b>
3.1 External Interface Requirements . . . . .	14
3.1.1 User Interfaces . . . . .	14
3.1.2 Hardware Interfaces . . . . .	22
3.1.3 Software Interfaces . . . . .	23
3.1.4 Communication Interfaces . . . . .	23
3.2 Functional Requirements . . . . .	24
3.2.1 Scenarios . . . . .	24
3.2.2 Use Case Diagrams . . . . .	25
3.2.3 Use Case Analysis . . . . .	27
3.2.4 Sequence Diagrams . . . . .	33
3.2.5 Requirements . . . . .	36
3.2.6 Satisfying Goals . . . . .	37
3.3 Performance Requirements . . . . .	40
3.4 Design Constraints . . . . .	40
3.4.1 Standard compliance . . . . .	40
3.4.2 Hardware limitations . . . . .	40
3.4.3 Any other constraint . . . . .	40
3.5 Design Constraints . . . . .	41
3.5.1 Reliability . . . . .	41
3.5.2 Availability . . . . .	41
3.5.3 Security . . . . .	41
3.5.4 Maintainability . . . . .	41
3.5.5 Portability . . . . .	41
<b>4 Formal Analysis using Alloy</b>	<b>42</b>
4.1 Alloy model . . . . .	42
4.1.1 Analysis results . . . . .	47
4.2 Graph . . . . .	47
<b>5 Effort Spent</b>	<b>49</b>
<b>6 References</b>	<b>52</b>

# Chapter 1

# Introduction

## 1.1 Purpose

This document constitutes the Requirement Analysis and Specification Document (i.e. RASD). Its purpose is to analyze the requirements that will lay the foundations of application services, to specify the application domain, the entities involved and their relationship, to clearly explain the objectives, the constraints and the features that are going to be implemented.

SafeStreets is a crowd-sourced application that intends to provide *Users* with the possibility to notify *Municipality* when *Traffic violations* occur, and in particular parking violations. The application allows *Users* to send pictures of violations, including their date, time, and position to *Municipality*.

### 1.1.1 Goals

- G<sub>1</sub> Collect *User report* in SafeStreets Database;
- G<sub>2</sub> Send *User report notification* to *Municipality* as soon as it is generated;
- G<sub>3</sub> Send *User picture* to the *Licence plate recognition service* as soon as it is received;
- G<sub>4</sub> Allow *Municipality* to visualize *Detailed statistics*;
- G<sub>5</sub> Allow *Users* to visualize *Public statistics*;
- G<sub>6</sub> Suggest *Possible interventions* to *Municipality*;
- G<sub>7</sub> Collect information about *Ticket feedback*;
- G<sub>8</sub> Allow *Users* to visualize their contribution to SafeStreets.

## 1.2 Scope

Crowd-sourced applications have become more popular nowadays thanks to the massive diffusion of smart devices and the consequent interconnection between people so that they started feeling part of a community, where everyone can contribute concretely.

SafeStreets is a crowd-sourced application whose aim is, indeed, to provide a tool to allow registered citizens to help *Municipality*, reporting *Traffic violations* that occur in their cities. In particular, they can take pictures of parking violations, specifying their type. A *License plate recognition service* recognizes the license plate from the *User picture*. SafeStreets stores the generated *User reports* and send a *User report notification* to *Municipality*. Examples of parking which represent a violation are the ones on bike lanes, sidewalks, in front of vehicle entrances or on reserved parking lots.

*Users* and *Municipalities* can also retrieve analytics about data collected by the application in order to obtain information, for example, about violations in certain areas or to identify vehicles with the highest number of violations, respectively.

Moreover, given that *Municipality* provides access to its data about *Accidents*, the system-to-be can integrate those information with its own data and finally suggest *Municipality Possible interventions* to apply.

SafeStreets can also build statistics from data sent by *Municipality* which concern issued *Traffic tickets* generated by *User reports*, for example about the most egregious offenders, or to show the effectiveness of the SafeStreets initiative.

### 1.2.1 World and Machine phenomena

Phenomena	Shared	Who controls it
The <i>User</i> signs up to SafeStreets	Y	W
The <i>Municipality</i> logs in to SafeStreets using its <i>Reference code</i>	Y	W
The <i>User</i> reports a <i>Traffic violation</i> through a <i>User report</i>	Y	W
The <i>User</i> takes a <i>User picture</i> of the <i>Traffic violation</i>	Y	W
The Machine sends received <i>User pictures</i> to the <i>License plate recognition service</i>	Y	M
The <i>License plate recognition service</i> recognizes a license plate	N	W
The <i>License plate recognition service</i> can't recognize a license plate	N	W
The Machine locates the <i>User's position</i>	N	M
The Machine adds a <i>Timestamp</i> to each <i>User report</i>	N	M
The <i>User</i> sends <i>User report</i> data	Y	W
The Machine receives <i>User report</i> data	Y	M
The <i>Municipality</i> issues <i>Traffic tickets</i>	N	W
The <i>Municipality</i> generates <i>Tickets feedback</i> on issued <i>Traffic tickets</i>	Y	W
The Machine generates statistics on received <i>User reports</i>	N	M
The <i>User</i> visualizes <i>Public statistics</i>	Y	W
The <i>Municipality</i> visualizes <i>Detailed statistics</i>	Y	W
The <i>Municipality</i> gives access to its <i>Accidents</i> data	Y	W
The Machine suggests <i>Possible interventions</i> to the <i>Municipality</i>	Y	M

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

**User:** individual registered to SafeStreets who agreed to the acquisition and processing of the data he/she sends;

**User identifier:** email provided by the *User* during the registration phase; it represents a single *User*;

**User report:** report created by the *User* including the fields *User identifier*, *User picture*, *Timestamp*, *Recognized license plate* and type of *Traffic violation*;

**User report notification** :notification sent by SafeStreets to the *Municipality* when a new *User report* is received;

**Terms and conditions:** a set of regulations which *Users* must agree to follow in order to use SafeStreets;

**Privacy statement:** describes why and how SafeStreets collects and uses personal data and provides information about *Users'* rights.

**Reference code:** unique code which represents a single *Municipality*;

**User picture:** picture of *Traffic violation* taken by the *User*;

**Traffic violation:** violation of parking laws;

**User position:** *Users'* GPS location;

**License plate recognition service:** third party service which provides the recognition of the license plate from the *User picture*;

**Recognized license plate:** text representing the license plate recognized by the *License plate recognition service*;

**Timestamp:** digital record of date and time of the day when it has been registered;

**Public statistics:** set of *User report* statistics provided by SafeStreets, available for all the registered *Users*;

**Detailed statistics:** set of *User report* statistics provided by SafeStreets, only available for the *Municipality*;

**Traffic ticket:** a notice issued to someone who violates a traffic regulation;

**Accidents:** data of *Municipality* Database, concerning previous *Traffic violations*;

**Possible interventions:** suggestions automatically generated by SafeStreets and forwarded to the *Municipality*;

**Ticket feedback:** information on whether a *User report* led to the generation of a *Traffic tickets* or not;

**Municipality:** district of a town or a city which has local government.

### 1.3.2 Acronyms

**GPS:** Global Positioning Service;

**GDPR:** General Data Protection Regulation.

### 1.3.3 Abbreviations

**G<sub>n</sub>:** n<sup>th</sup> Goal;

**D<sub>n</sub>:** n<sup>th</sup> Domain Assumption;

**R<sub>n</sub>:** n<sup>th</sup> Requirement;

**S<sub>n</sub>:** n<sup>th</sup> Scenario;

**U<sub>n</sub>:** n<sup>th</sup> Use Case.

## 1.4 Revision History

1. Version 0.1 - 20th October 2019 - Start of the RASD;
2. Version 1.0 - 10th November 2019 - First release;
3. Version 1.1 - 15th November 2019

### Major Changes

- *User reports* State Chart coherent with the specifications
- World and Machine phenomena table added
- Use Case Diagrams edited
- Alloy comments and chapter description added

### Minor Changes

- SafeStreets UML Class Diagram
- Requirements (+1) and Assumptions (-1)
- Use Case Analysis edited
- Sequence Diagrams edited
- Use Case Scenario edited

4. Version 1.2 - 9th December 2019

### Minor Changes

- Definition added (and other related changes)
- Acronyms and Abbreviations added
- Sequence Diagram and Related Use case table added

## 1.5 Reference Documents

- Rumbaugh, Jacobson, Booch. 1999. *The Unified Modeling Language Reference Manual*. Addison-Wesley.
- MIT Software Design Group. *Appendix B: Alloy Language Reference*. <http://alloytools.org/documentation.html>
- MIT Software Design Group. *Tutorial Materials, Slides*. <http://alloytools.org/tutorials/day-course/>

## 1.6 Document Structure

This document is structured as follows:

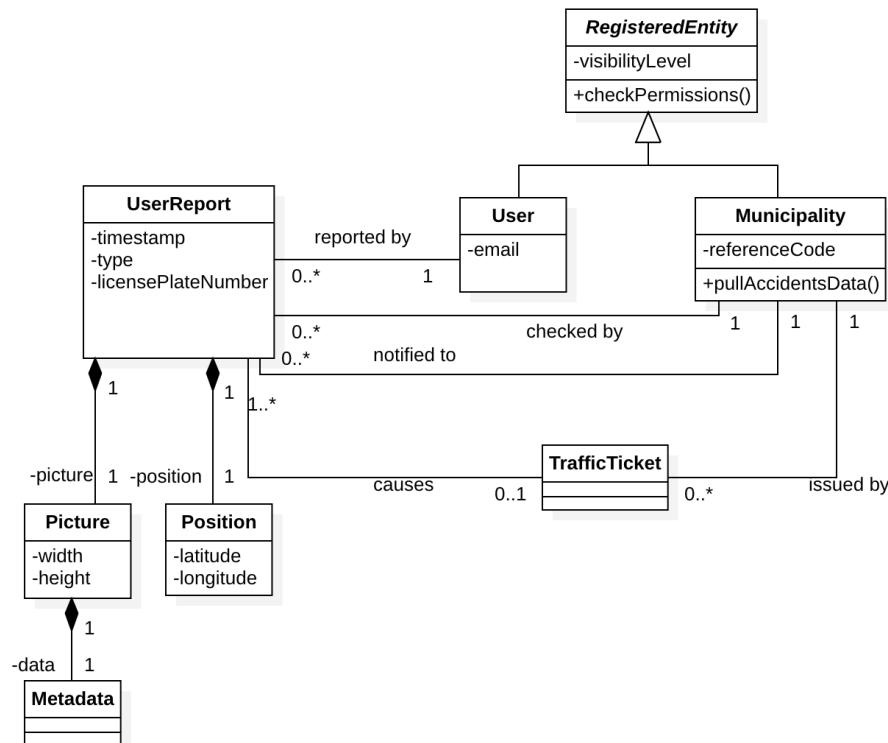
- 1. Introduction** A general introduction to the goals, the phenomena and the scope of the system-to-be. It aims giving general but exhaustive information about what this document is going to explain.
- 2. Overall Description** A general description of the product to be and its requirements. This section provides several information that are explained in detail in Section 3.
- 3. Specific Requirements** All software requirements are explained using scenarios, use-case diagram and activity diagram. Non-functional and functional requirements are also cited.
- 4. Formal Analysis using Alloy** This section includes Alloy code that describes the model and shows its soundness and correctness.
- 5. Effort spent** Effort spent by all team members shown as the list of all the activities done during the realization of this document.
- 6. References** References of documents that this project was developed upon.

# Chapter 2

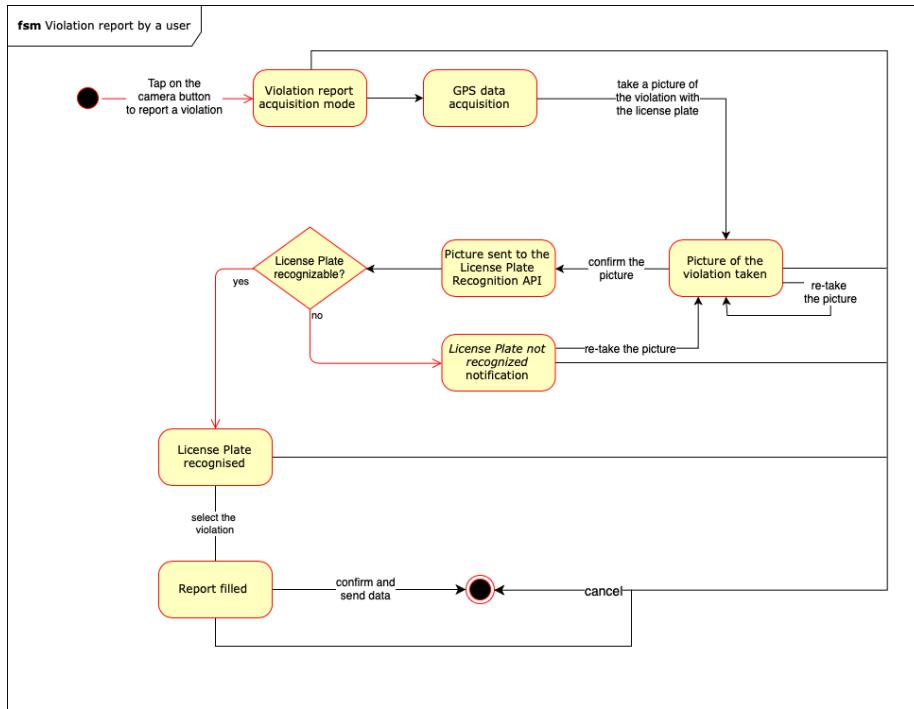
## Overall Description

### 2.1 Product Perspective

#### 2.1.1 SafeStreets



SafeStreets class diagram

SafeStreets state chart referred to *User reports* by *Users*

### Shared Phenomena

- *User* reports a *Traffic violation*.
- *Municipality* checks a *Traffic violation*.
- A *Traffic violation* causes a *Traffic ticket* issued by the *Municipality*.
- *User* and *Municipality* have different permission levels to access the system-to-be data.

## 2.2 Product Functions

### 2.2.1 User registration

SafeStreets will allow individuals to register. These will register by entering all the required information (see R<sub>1</sub>- R<sub>2</sub>). When registering to SafeStreets, they will first declare to have read the *Privacy statement* and secondly, they will have to accept the *Terms and conditions*, which specifically include their consent to the acquisition and processing of their data.

The *User* registration process will be carried out on the SafeStreets application (see section 3.1.1).

### **2.2.2 User report acquisition**

SafeStreets will collect *User report*. It will allow *User* to take a picture of the *Traffic violation* and to select the *Type of violation*. Once *License plate recognition* has recognized the license plate, SafeStreets will complete and acquire the *User report* after *User* confirmation and then will send a *User report notification* to the *Municipality*.

### **2.2.3 User contribution visualization**

SafeStreets will show to the *User* his/her contribution in dedicated section where visualize all the previous *User report* and the *Ticket feedback* of each of these, received by the *Municipality*, as well.

### **2.2.4 Public statistics visualization**

SafeStreets will show *Public statistics* required by the *User*. *User* will be able to set the parameters selecting a desired “period of time”, a “*Type of violation*” and a “specific zone”.

### **2.2.5 Municipality access**

SafeStreets will allow *Municipality* to connect to its dedicated website.

### **2.2.6 User report evaluation**

*Municipality* will visualize the list of *User reports*, select one and generate or not a *Traffic ticket*. If *Municipality* discard a *User report*, it will select a reason among those provided.

### **2.2.7 Detailed statistics visualization**

SafeStreets will show *Detailed Statistics* required by the *Municipality*. *Municipality* will be able to set the parameters selecting the “period of time”, a “*Type of violation*”, a “specific zone” and also inserting a “specific license plate”.

### **2.2.8 Possible interventions visualization**

SafeStreets will cross *Municipality Accidents* data with its own Database data. Based on the results, it could eventually suggest *Possible interventions* to *Municipality*.

### **2.2.9 License plate recognition data acquisition**

The service will be able to receive the picture taken by *User* and sent through SafeStreets. Once the license plate has been recognized, it is immediately sent back, as a text.

### 2.2.10 Request Accidents data

SafeStreets, after requesting access to *Municipality Accidents* Database, will be able to cross *Municipality* data with its own data.

### 2.2.11 Request User report data

*User*, after requesting access to SafeStreets database for his/her own *User report*, will be able to visualize all the details about his/her previous *User report*.

## 2.3 User Characteristics

**Users:** People having a smartphone with Internet connection and willing to report a *Traffic violation* which has occurred.

**Municipality:** Authorities having the possibility to receive a *User report notification* when a *User report* is received by SafeStreets, in order to evaluate it and, eventually, generate *Traffic tickets*. They can evaluate reports without being physically present and see all the details about a specific *User report*. This also allow them to motivate any possible *User report* discarded. The *Municipality* visualizes, then, all the *Possible intervention* suggested by SafeStreets (e.g. add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking . . . ).

## 2.4 Assumptions, Dependencies and Constraints

### 2.4.1 Domain Assumptions

- D<sub>1</sub> Email inserted by the *User* at sign up correspond to a valid email;
- D<sub>2</sub> *User position* collected at a certain instant corresponds to the actual position of the *User* at that exact moment;
- D<sub>3</sub> The maps in use accurately represent the world;
- D<sub>4</sub> *Users* own a working smartphone which has access to Internet connection;
- D<sub>5</sub> *Users* own a working smartphone which has a working GPS antenna;
- D<sub>6</sub> SafeStreets and *License plate recognition service* are always online;
- D<sub>7</sub> *Municipality* has institutional credentials (*Reference code* and password) which uses to access the system;
- D<sub>8</sub> *Municipality* grants access to its own *Accidents* Database.

#### 2.4.2 Dependencies

- Availability, performance and reliability of services depend respectively on the availability, performance and reliability of SafeStreets.  
SafeStreets, in order to guarantee that the data which constitutes the *User report* are not corrupted, ensures the chain of custody of information.
- Effectiveness of SafeStreets depends on *License plate recognition* response time.

#### 2.4.3 Constraints

- Smartphones must always have a working Internet connection;
- *User* willing to register to SafeStreets must accept *Terms and conditions*;
- User registered must accept the *Privacy statement*;
- SafeStreets must comply with the GDPR regulations for the protection of *Users'* personal data;
- Smartphones must have GPS activated;
- Smartphones must have enough free space for downloading and installing the SafeStreets application;
- SafeStreets must wait for *License plate recognition* response in order to allow *User* to complete the *User report*;
- *Municipality* must have its *Reference code* and password, granted by SafeStreets, to access its *Municipality* area;
- All the collected *User reports* must not be accessible by unauthorized people, but only by the *Municipality*;
- *Municipality* must grant *Accidents* Database access to SafeStreets;
- *Municipality* must have a modern browser installed on a device connected to the Internet, in order to access the *Municipality* area on its dedicated website.

# Chapter 3

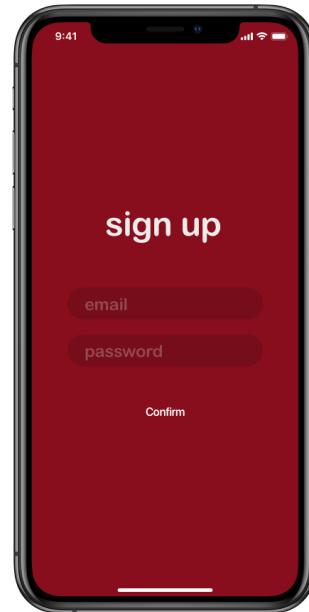
# Specific Requirements

## 3.1 External Interface Requirements

### 3.1.1 User Interfaces



SafeStreets Welcome Page



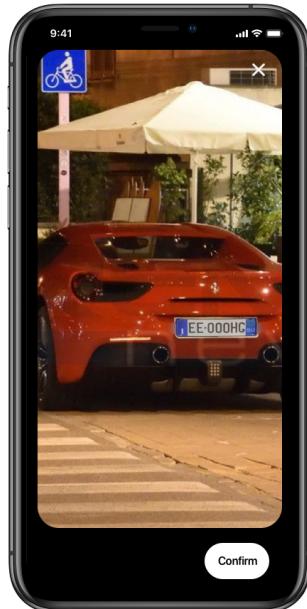
SafeStreets Sign up View



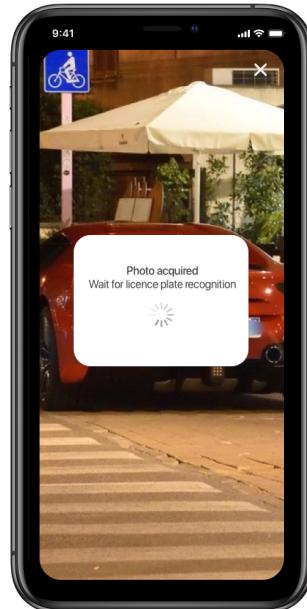
SafeStreets Login View



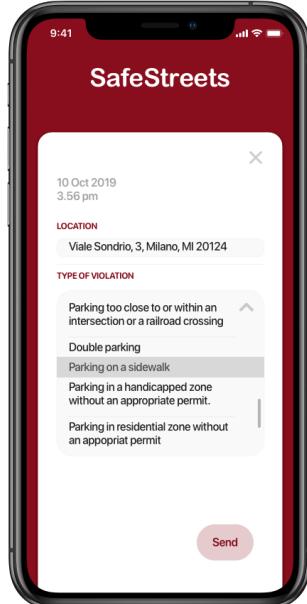
SafeStreets Camera View



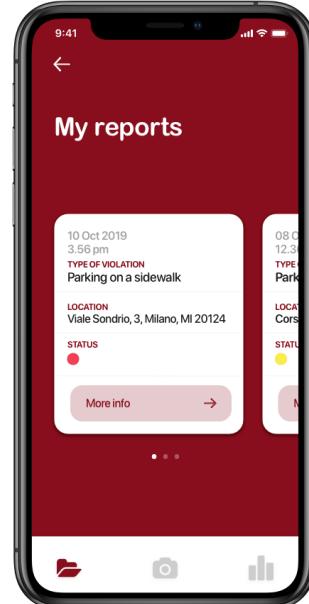
SafeStreets    *Traffic violation*  
photo



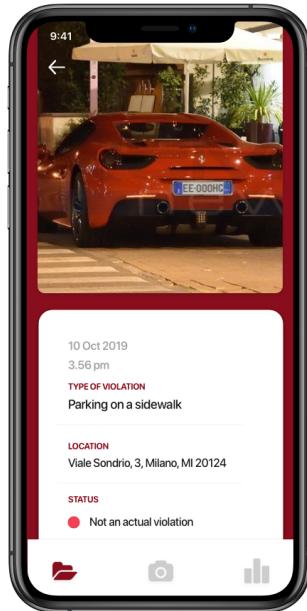
License plate recognition



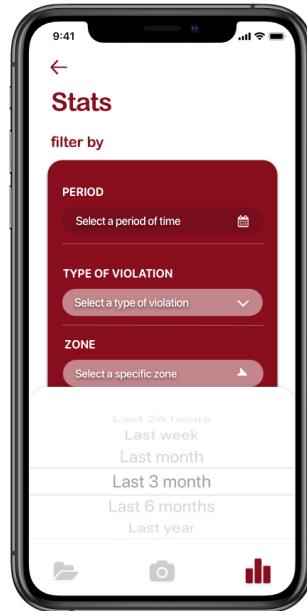
SafeStreets *User report* View



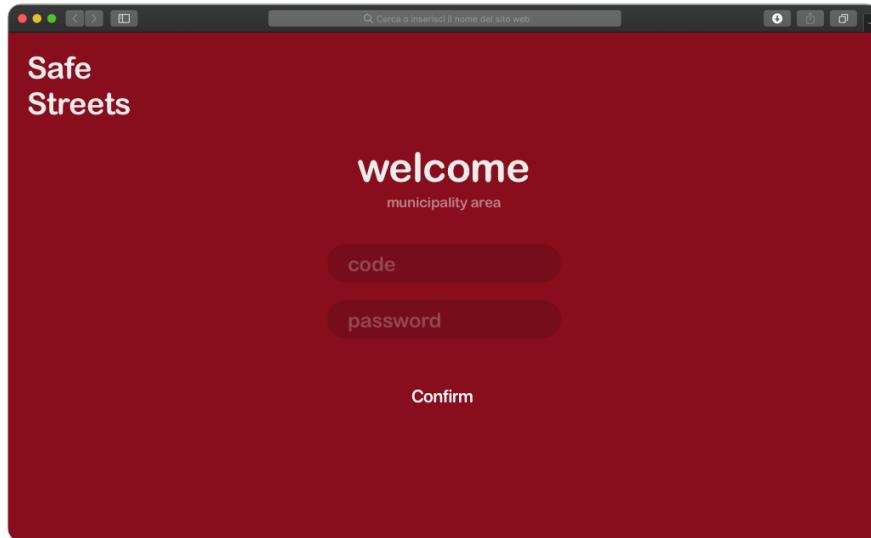
SafeStreets "My reports" View



SafeStreets detailed *User report*  
View



SafeStreets *Public Statistics*  
View

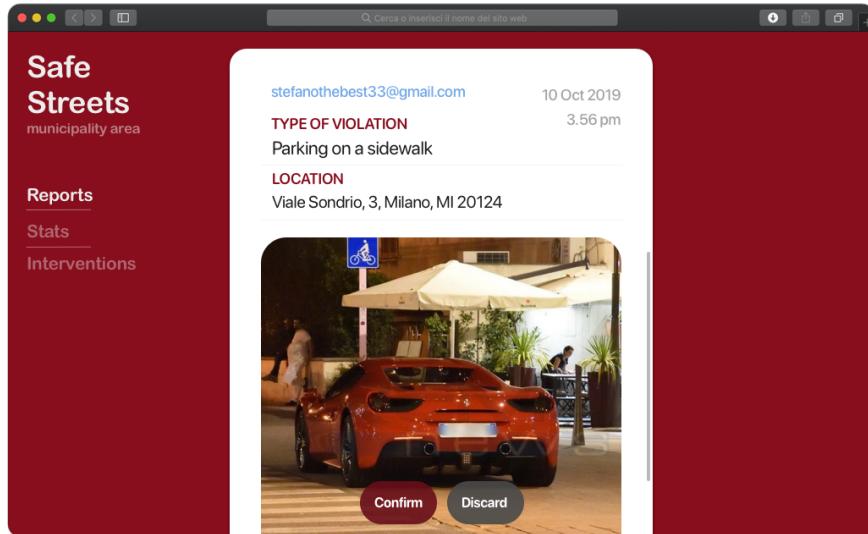


SafeStreets Welcome Page, Municipality area

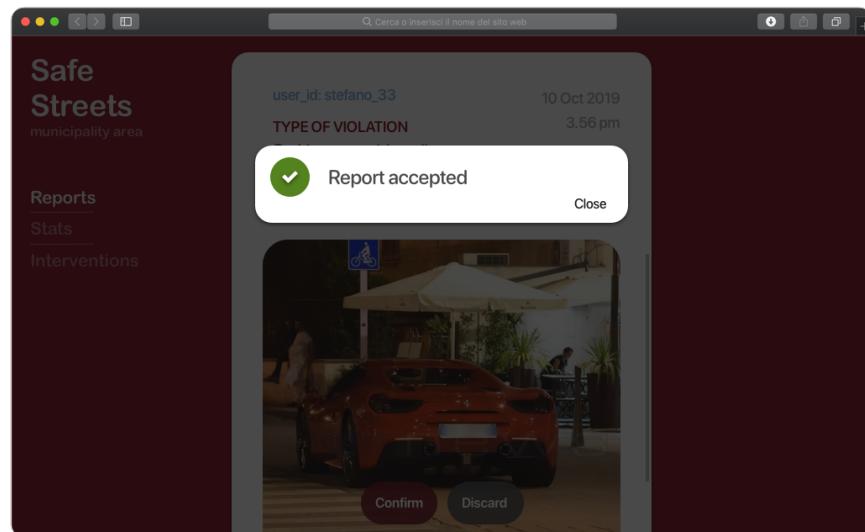
A screenshot of a web browser showing the 'Reports list' section of the Safe Streets application. The background is dark red. On the left, there is a sidebar with navigation links: 'Reports' (which is currently selected), 'Stats', and 'Interventions'. The main area has a white header with the title 'Reports list'. Below the header is a table listing five user reports. Each report includes the reporter's email, the address, the date, and the time. The table rows are as follows:

Reporter Email	Date	Time
stefanothebest33@gmail.com	10 Oct 2018	3.56 pm
Viale Sondrio, 3, Milano, MI 20124		
Parking on a sidewalk		
leowar1999@live.com	4 Nov 2018	1.45 pm
Viale Abruzzi, 12, Milano, MI 20124		
Parking too close to or within an intersection or a railroad crossing		
fab.f11@hotmail.com	22 Dec 2018	8.30 am
Via Giacinto Abbruzzesi, 9, Milano, MI 20124		
Parking in a handicapped zone without an appropriate permit		
forzacavese1919@gmail.com	10 Feb 2019	5.01 pm
Corsso Buenos Aires, 2, Milano, MI 20124		
Parking on a sidewalk		
massirario1@libero.com	9 May 2019	6.44 pm
Corsso Como, 111, Milano, MI 20124		
Parking in residential zone without an appropiat permit		

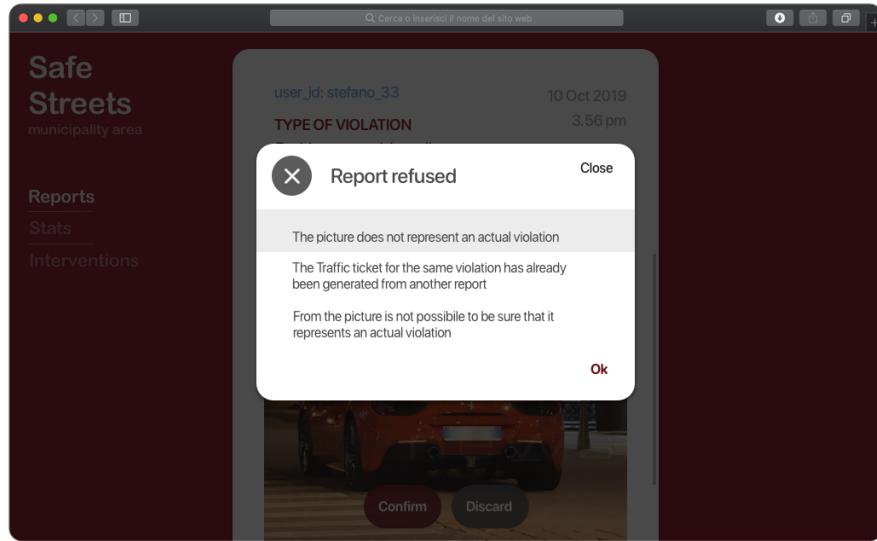
SafeStreets *User reports* list



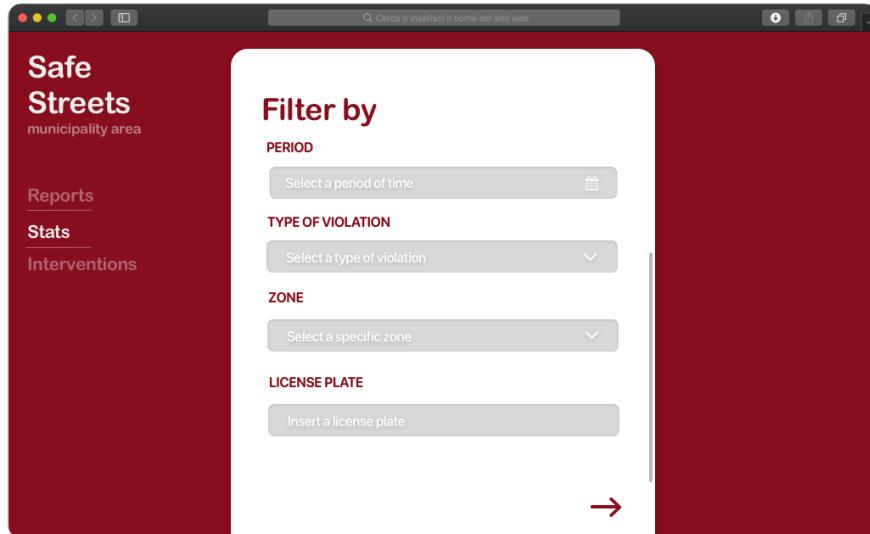
SafeStreets detailed *User report* Page



SafeStreets *User reports* refused



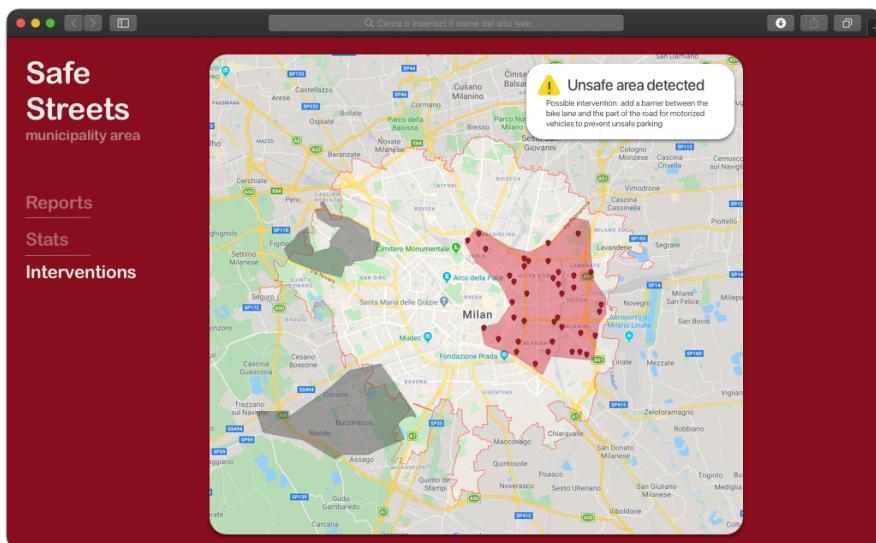
SafeStreets *User report accepted*



SafeStreets "filter by" Page

The screenshot shows a list of 10 parking violations from October 2018 to September 2019. The violations are listed in a table with columns for date, type of violation, address, license plate, and zone.

Date	Type of violation	Address	License plate	Zone
10 Oct 2018	Parking on a sidewalk	Viale Sondrio, 3, Milan, MI	AA 000 HG	
3 Nov 2018	Parking on a sidewalk	Piazza Loreto, 2, Milan, MI	CD 190 ER	
9 Dec 2018	Parking on a sidewalk	Piazza Leonardo, 3, Milan, MI	RT 234 IL	
13 Dec 2018	Parking on a sidewalk	Viale Isidoro, 12, Milan, MI	KK 333 CF	
22 Oct 2018	Parking on a sidewalk	Via G. Mazzini, 9, Milan, MI	GA 336 BO	
7 Jan 2019	Parking on a sidewalk	Corsso Umberto I, 78, Milan,	LA 788 LA	
10 May 2019	Parking on a sidewalk	Corsso Como, 111, Milan, MI	UR 690 AA	
23 June 2019	Parking on a sidewalk	Via Dei Glicini, 18, Milan, MI	FA 208 BI	
17 July 2019	Parking on a sidewalk	Viale Lombardi, 67, Milan, MI	STE 696 FA	
10 Sep 2019	Parking on a sidewalk	Via V. Veneto, 55, Milan, MI	LE 313 GU	

SafeStreets *Detailed Statistics* PageSafeStreets *Possible interventions* suggestions

### 3.1.2 Hardware Interfaces

- Web applications must be accessible by *Municipality* using a computer with characteristics specified in Section 3.4.2.

- Smartphone on which the app will work must provide to the app an Internet connection used to send *User reports* to SafeStreets servers and must have a GPS antenna built in.

### 3.1.3 Software Interfaces

- Web applications for the *Municipality* must be compatible with the most popular browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari;
- Mobile apps for standard users must be available for both iOS and Android devices;
- Application backend stores collected data in a relational DBMS;
- Web applications show data by accessing the relational DBMS;
- SafeStreets must interface with the smartphone's GPS via the operating system API in order to localize the *User position* reporting a *Traffic violation*;
- Web applications have to interface with SafeStreets in order to receive notifications and visualize data about *User reports* and eventually *Possible suggestions*;
- Web applications have to interface also with Maps in order to show the unsafe area detected;
- SafeStreets has to interface with *License plate recognition service*.

### 3.1.4 Communication Interfaces

The system to be will only make use of the usual communication protocols (TCP/IP) to guarantee the connection between the *User's* Smartphones and the backend systems.

## 3.2 Functional Requirements

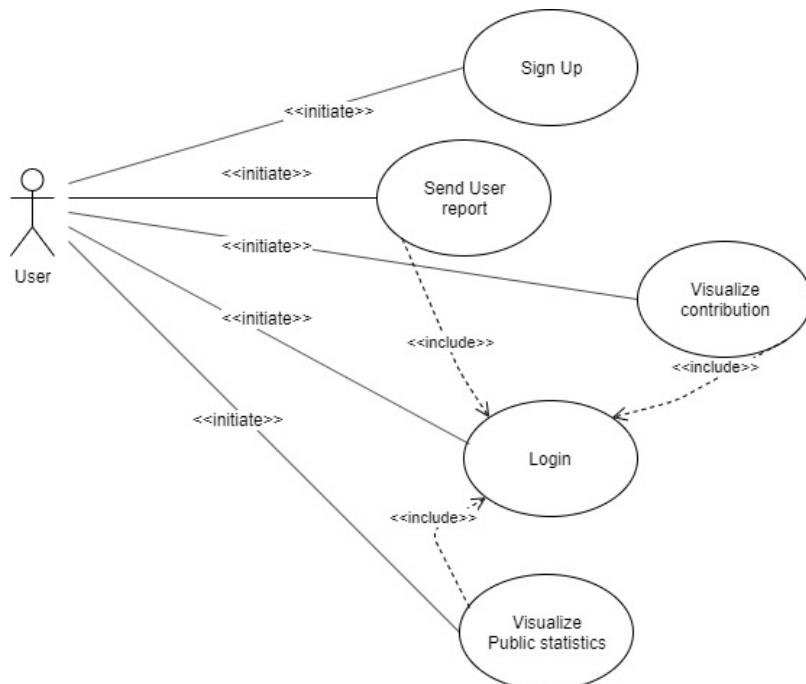
### 3.2.1 Scenarios

- S<sub>1</sub> Michael is an individual who would like to help the community and the authorities reporting parking violations occurring in his city. He downloads the SafeStreets application on his smartphone and, once opened it, he is asked to insert an email (which will correspond to his *User identifier*) and a password. After inserting them, he accepts the *Terms and conditions*, the *Privacy statement* and taps on the arrow below. Later, he tries to login using his email and password; the system accepts his credentials and Michael is now logged in;
- S<sub>2</sub> A *Municipality* officier opens SafeStreets web app. He is asked to insert *Reference code* and password in order to access the system, so he inserts his *Municipality* institutional credentials and he is now logged in and can use SafeStreets services;
- S<sub>3</sub> Michael, a SafeStreets *User*, is walking down a street and, when he's about to cross the road, he realizes he can't access the pedestrian crossing from the walk side, because of a car parked there. So he opens the app and presses the "Camera" button in the tab bar of the main view, frames the car, making sure that the violation and the license plate are clearly visible, and takes the picture. After pressing the "Confirm" button, he waits for the "License plate successfully recognized" message. Then, he is asked to select the "*Type of violation*", choosing from a drop-down list of possible violations, and presses again on the "Send" button, in order to send his *User report*;
- S<sub>4</sub> Michael, from scenario S<sub>3</sub>, opens the app and taps on "MyReports" in the tab bar of the main view, to see if the last *User report* he has sent has already been evaluated, eventually becoming a *Traffic ticket*, but he sees that the *Ticket feedback* color is red, which means that a *Traffic ticket* has not been generated from his *User report*. To know more, he presses on the "More info" button and understands that the reason why it was not generated is that it was not an actual violation;
- S<sub>5</sub> Scarlett, a SafeStreets *User*, wants to visualize *Public statistics* of her neighborhood of the last month, in order to figure out whether her neighbours have been active in reporting *Traffic violation* through SafeStreets or not, and eventually pointing it out during the monthly neighborhood meeting. For that reason, she presses the "*Public statistics*" button and selects an area, a period and a *Type of violation* from the corresponding drop-down lists and consults the statistics she is interested in;
- S<sub>6</sub> *Municipality* receives the complaint for a stolen car, so, in order to increase the possibilities to find some useful clues, looks for *User reports* received in

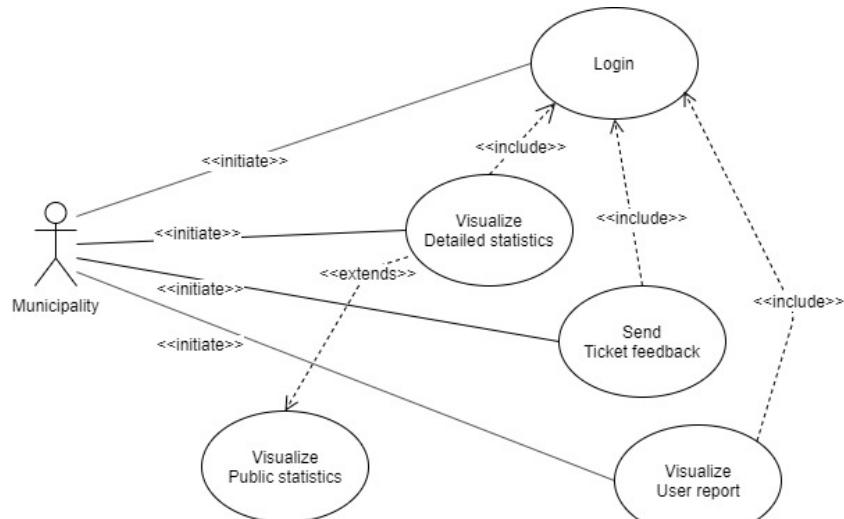
the last 24 hours and involving the stolen car license plate in the *Detailed statistics*;

- S<sub>7</sub> SafeStreets retrieves previous *Accidents* data from *Municipality* Database and, after crossing them with *User report* data in its own Database, identifies the area where the sum of the number of *Accidents* and the number of *User report*, for a particular *Type of violation*, is the highest, in order to identify and suggest a *Possible intervention* to the *Municipality*;
- S<sub>8</sub> *Municipality* wants to visualize *Ticket feedback* statistics in the *Detailed statistics* to point out the areas where the service has had the highest incidence (i.e. the areas where the highest number of *User reports* have actually become *Traffic tickets*), in order to intensify the presence of Local Police agents in those areas;
- S<sub>9</sub> *Municipality* wants to visualize who have been the most egregious offenders in a certain area of the city, so looks for them in the *Detailed statistics* focusing in particular on the *Ticket feedback* statistics.

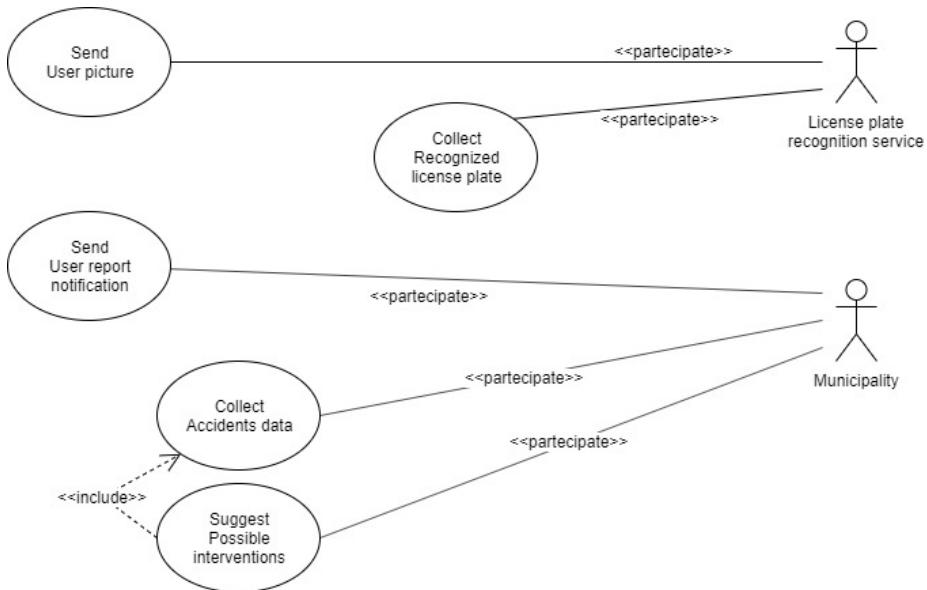
### 3.2.2 Use Case Diagrams



User Use Case Diagram



Municipality Use Case Diagram



General Use Case Diagram

### 3.2.3 Use Case Analysis

Name	<b>U1: User Sign Up</b>
Actors	<i>User</i>
Entry Conditions	<i>User</i> successfully installed SafeStreets application on the smartphone
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> taps on "Sign Up" button</li> <li>2. <i>User</i> fills the "email" and "password" fields</li> <li>3. <i>User</i> checks the "Accept <i>Terms and conditions</i>" checkbox</li> <li>4. <i>User</i> checks the "Accept <i>Privacy statement</i>" checkbox</li> <li>5. <i>User</i> taps on "Confirm" button</li> <li>6. SafeStreets saves <i>User</i> information</li> </ol>
Exit Condition	<i>User</i> successfully registered on SafeStreets
Exceptions	<ol style="list-style-type: none"> <li>1. Inserted email already registered for another <i>User</i></li> <li>2. Inserted password is not valid</li> <li>3. Not all required fields are filled in</li> <li>4. "Accept <i>Terms and conditions</i>" checkbox is not checked</li> <li>5. "Accept <i>Privacy statement</i>" checkbox is not checked</li> <li>6. <i>User</i> is already signed up</li> </ol> <p><i>User</i> is invited to try again signing up, reporting which error they have committed.</p>

U1

<b>Name</b>	<b>U2: User Login</b>
Actors	<i>User</i>
Entry Conditions	<i>User</i> successfully installed SafeStreets application on the smartphone and registered on SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> enters "email"</li> <li>2. <i>User</i> enters "password"</li> <li>3. <i>User</i> presses the "Confirm" button</li> <li>4. SafeStreets checks <i>User</i> credentials</li> </ol>
Exit Condition	<i>User</i> is successfully logged in
Exceptions	<ol style="list-style-type: none"> <li>1. Inserted "email" is not valid</li> <li>2. Inserted "password" is not valid</li> </ol> <p><i>User</i> is asked to try again to log into the system.</p>

U<sub>2</sub>

<b>Name</b>	<b>U3: Municipality Login</b>
Actors	<i>Municipality</i>
Entry Conditions	<i>Municipality</i> is connected to the <i>Municipality</i> dedicated web site
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Municipality</i> enters "Reference code"</li> <li>2. <i>Municipality</i> enters "password"</li> <li>3. <i>Municipality</i> presses the "Confirm" button</li> <li>4. SafeStreets checks <i>Municipality</i> institutional credentials</li> </ol>
Exit Condition	<i>Municipality</i> is successfully logged into the system
Exceptions	<ol style="list-style-type: none"> <li>1. Inserted "Reference code" is not valid</li> <li>2. Inserted "password" is not valid</li> </ol> <p><i>Municipality</i> is asked to try again to log into the system.</p>

U<sub>3</sub>

<b>Name</b>	<b>U4: User sends User report to SafeStreets</b>
Actors	<i>User, License plate recognition service</i>
Entry Conditions	<i>User</i> successfully logged into SafeStreets and installed SafeStreets application on the smartphone
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> takes a picture of the <i>Traffic violation</i></li> <li>2. <i>User</i> presses the “Confirm” button</li> <li>3. SafeStreets receives <i>User picture</i> and forward it to the <i>License plate recognition service</i></li> <li>4. <i>License plate recognition service</i> sends back to SafeStreets a text representing the <i>Recognized license plate</i>, if it is successfully recognized</li> <li>5. <i>User</i> selects a <i>Type of violation</i> and presses the “Send” button</li> <li>6. SafeStreets stores the newly generated <i>User report</i> in its own Database</li> <li>7. SafeStreets sends <i>Municipality</i> a <i>User report notification</i> about the newly generated <i>User report</i></li> </ol>
Exit Condition	SafeStreets correctly collected <i>User report</i>
Exceptions	<ol style="list-style-type: none"> <li>1. No <i>Type of violation</i> has been selected</li> <li>2. The <i>License plate recognition service</i> couldn't recognize a license plate inside the <i>User picture</i></li> </ol> <p>Exception 1: <i>User</i> is asked to select a choice for the missing parameter  Exception 2: <i>User</i> is asked to re-take the picture</p>

U4

<b>Name</b>	<b>U5: Municipality answers to a User report</b>
Actors	<i>Municipality</i>
Entry Conditions	<i>Municipality</i> successfully logged into SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Municipality</i> presses “Reports” button</li> <li>2. <i>Municipality</i> selects a <i>User report</i> to answer</li> <li>3. <i>Municipality</i> taps “Confirm” or “Discard” button (in the second case selects a reason as well) to generate the <i>Ticket feedback</i></li> <li>4. SafeStreets receives the <i>Ticket feedback</i></li> </ol>
Exit Condition	<i>Municipality</i> correctly answered to the <i>User report</i>
Exceptions	No Exceptions.

U<sub>5</sub>

<b>Name</b>	<b>U6: User visualizes Public statistics</b>
Actors	<i>User</i>
Entry Conditions	<i>User</i> successfully logged into SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> presses “Public statistics” button</li> <li>2. <i>User</i> selects a “period of time”, a “Type of violation” and a “specific zone”</li> <li>3. SafeStreets shows the <i>User</i> the required statistics</li> </ol>
Exit Condition	<i>User</i> correctly visualizes the required <i>Public statistics</i>
Exceptions	<ol style="list-style-type: none"> <li>1. No value was chosen for the parameters to be set <i>User</i> is asked to select at least one value between the filtering parameters.</li> </ol>

U<sub>6</sub>

<b>Name</b>	<b>U7: Municipality visualizes Detailed statistics</b>
Actors	<i>Municipality</i>
Entry Conditions	<i>Municipality</i> successfully logged into SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Municipality</i> presses the “Stats” button</li> <li>2. <i>Municipality</i> selects a “period of time”, a “Type of violation” and a “specific zone”, then, optionally, a “particular license plate” as well</li> <li>3. SafeStreets shows <i>Municipality</i> the required statistics</li> </ol>
Exit Condition	<i>Municipality</i> correctly visualizes the required <i>Detailed statistics</i>
Exceptions	<ol style="list-style-type: none"> <li>1. No value was chosen for the parameters to be set</li> </ol> <p><i>Municipality</i> is asked to select at least one value between the filtering parameters.</p>

U7

<b>Name</b>	<b>U8a: SafeStreets generates Possible interventions</b>
Actors	<i>Municipality</i>
Entry Conditions	<i>Municipality</i> grants SafeStreets access to its own <i>Accidents</i> Database
Events Flow	<ol style="list-style-type: none"> <li>1. SafeStreets requests <i>Municipality</i> previous Accidents data</li> <li>2. <i>Municipality</i> sends SafeStreets the requested Accidents data</li> <li>3. SafeStreets crosses its own <i>User report</i> data with Accidents data in order to generate <i>Possible interventions</i></li> </ol>
Exit Condition	SafeStreets correctly generates <i>Possible interventions</i>
Exceptions	No Exceptions.

U8a

Name	<b>U8b: <i>Municipality visualizes Possible interventions suggested by SafeStreets</i></b>
Actors	<i>Municipality</i>
Entry Conditions	<i>Municipality</i> successfully logged into SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>Municipality</i> presses the “Possible interventions” button</li> <li>2. <i>Municipality</i> selects a particular <i>Possible intervention</i> to visualize</li> <li>3. SafeStreets shows <i>Municipality</i> the required <i>Possible intervention</i></li> </ol>
Exit Condition	<i>Municipality</i> correctly visualizes the suggested <i>Possible intervention</i>
Exceptions	<ol style="list-style-type: none"> <li>1. No <i>Possible intervention</i> has been generated yet. <i>Municipality</i> is asked to retry later.</li> </ol>

U<sub>8b</sub>

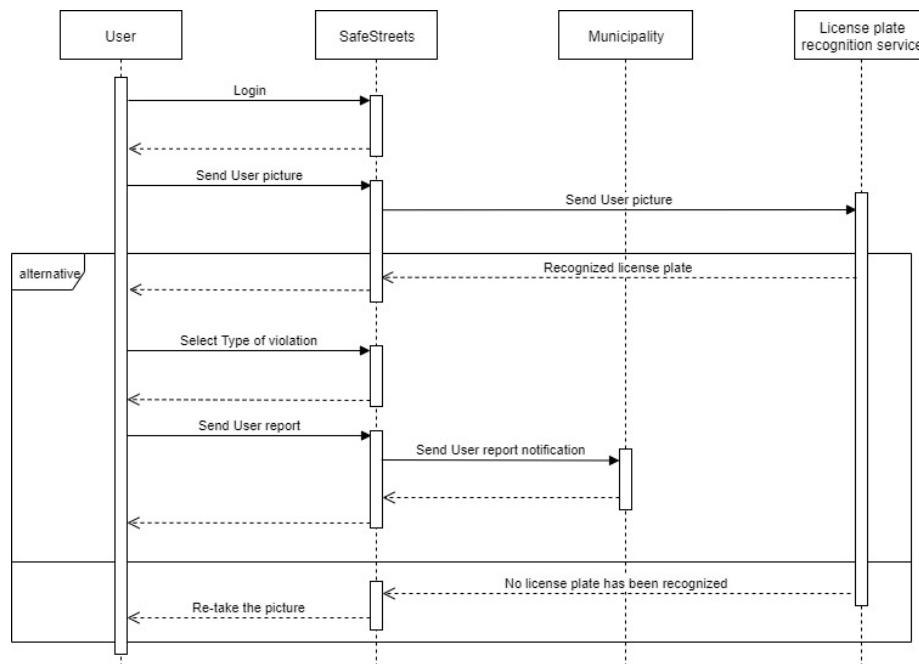
Name	<b>U9: <i>User visualizes his/her contribution to SafeStreets</i></b>
Actors	<i>User</i>
Entry Conditions	<i>User</i> successfully logged into SafeStreets
Events Flow	<ol style="list-style-type: none"> <li>1. <i>User</i> taps “My reports” button</li> <li>2. <i>User</i> chooses one of his/her previous <i>User report</i> to check</li> <li>3. <i>User</i> visualizes all the details about that particular <i>User report</i></li> </ol>
Exit Condition	<i>User</i> correctly visualizes the details about the chosen <i>User report</i>
Exceptions	<ol style="list-style-type: none"> <li>1. <i>User</i> has not reported any <i>Traffic violation</i> yet <i>User</i> is reminded he/she has not reported any <i>Traffic violation</i> yet.</li> </ol>

U<sub>9</sub>

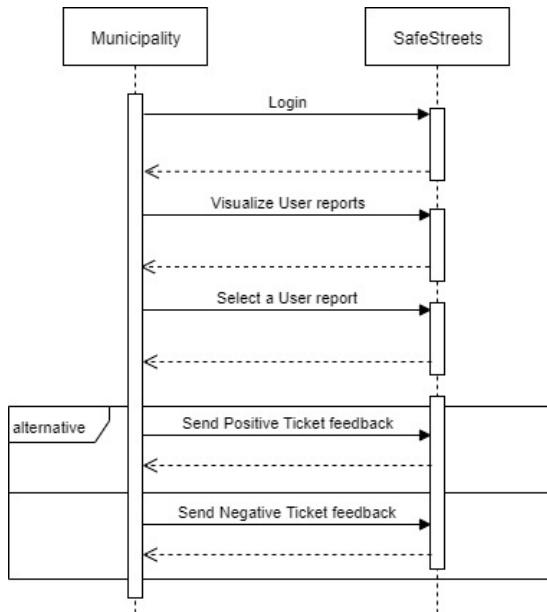
### 3.2.4 Sequence Diagrams

In the following Sequence Diagrams, most of the Use Cases described in detail in subsection 3.2.3 are shown, except for the Use Cases  $U_1$ ,  $U_2$  and  $U_3$ , representing respectively the *User* sign up, the *User* login and the *Municipality* login phases, for simplicity reasons, as they are trivial.

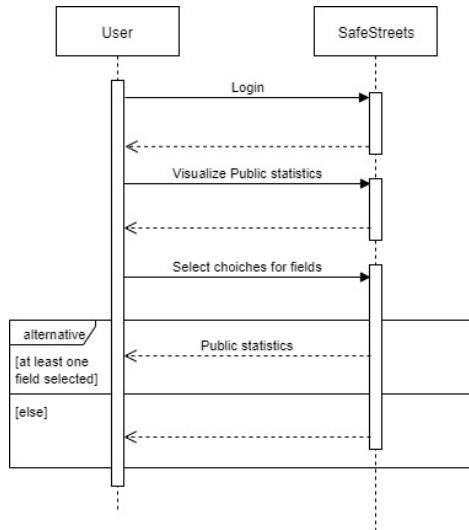
Still, the login process is here shown in all the Sequence Diagrams which refer to the Use Cases requiring it as first task, to highlight that all those Use Cases require it, but  $U_{8a}$ .



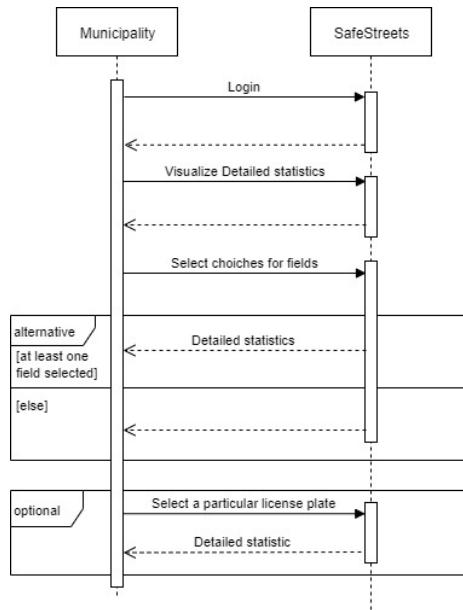
U4: User sends User report to SafeStreets Sequence Diagram



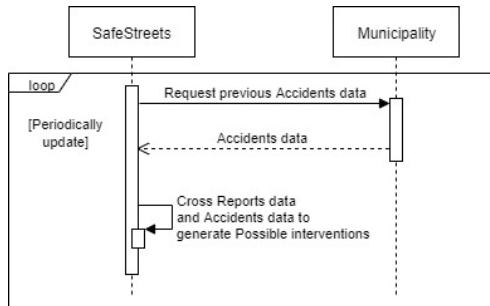
U5: Municipality answers to a User report Sequence Diagram



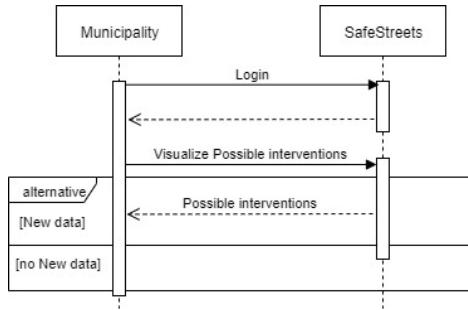
U6: User visualizes Public statistics Sequence Diagram



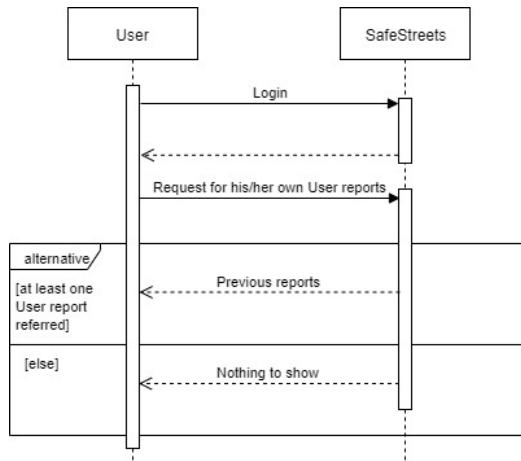
U7: Municipality visualizes Detailed statistics Sequence Diagram



U8a: SafeStreets generates Possible interventions Sequence Diagram



U8b: Municipality visualizes Possible interventions Sequence Diagram



U9: User visualizes his/her contribution to SafeStreets Sequence Diagram

### 3.2.5 Requirements

- R<sub>1</sub> At sign up, *User* must provide email and password;
- R<sub>2</sub> At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*;
- R<sub>3</sub> Identify a *User* by his/her *User identifier*;
- R<sub>4</sub> Receive *User picture*;
- R<sub>5</sub> Receive *User choice* for the type of *Traffic violation*;
- R<sub>6</sub> Receive *User position*;
- R<sub>7</sub> Send *User picture* to the *License plate recognition service*;
- R<sub>8</sub> Receive *Recognized license plate* from the *License plate recognition service*;

- R<sub>9</sub> Create *User report*;
- R<sub>10</sub> Store *User report* in SafeStreets Database;
- R<sub>11</sub> Send *User report notification* to *Municipality*;
- R<sub>12</sub> Store *Ticket feedback* in SafeStreets Database;
- R<sub>13</sub> Query SafeStreets Database for a *User* by his/her *User identifier*;
- R<sub>14</sub> Retrieve specific *User reports* by querying SafeStreets Database;
- R<sub>15</sub> Validate the data which constitutes the *User report*;
- R<sub>16</sub> Generate *Possible interventions* crossing *Municipality Accidents* data with SafeStreets Database data;
- R<sub>17</sub> Store generated *Possible interventions* for the *Municipality*;
- R<sub>18</sub> Allow *Municipality* to set specific constraints to define *Detailed statistics*;
- R<sub>19</sub> Send *Detailed statistics* to the requesting *Municipality*;
- R<sub>20</sub> Allow *Users* to request all their *User reports* and the related *Ticket feedback* stored in SafeStreets Database at any time;
- R<sub>21</sub> Send to a specific *User* his/her *User report* stored;
- R<sub>22</sub> Send to a specific *User* the *Ticket feedback* related to his/her *User report*;
- R<sub>23</sub> Allow *User* to choose among filters which define *Public statistics*;
- R<sub>24</sub> Send *Public statistics* to the requesting *User*;
- R<sub>25</sub> Receive *Ticket feedback* from *Municipality*;
- R<sub>26</sub> Allow *User* to re-take the *User picture*;
- R<sub>27</sub> Request *Accidents* data to *Municipality*;
- R<sub>28</sub> Receive *Accidents* data from *Municipality*;
- R<sub>29</sub> Identify *Municipality* by its *Reference code*.

### 3.2.6 Satisfying Goals

- G<sub>1</sub> Collect *User report* in SafeStreets Database;
  - R<sub>1</sub> At sign up, *User* must provide: first name, last name, email and password;
  - R<sub>2</sub> At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*;
  - R<sub>3</sub> Identify a *User* by his/her *User identifier*;

- R<sub>4</sub> Receive *User picture*;
  - R<sub>5</sub> Receive *User choice* for the type of *Traffic violation*;
  - R<sub>6</sub> Receive *User position*;
  - R<sub>7</sub> Send *User picture* to the *License plate recognition service*;
  - R<sub>8</sub> Receive *Recognized license plate* from the *License plate recognition service*;
  - R<sub>9</sub> Create *User report*;
  - R<sub>10</sub> Store *User report* in SafeStreets Database;
  - R<sub>15</sub> Validate the data which constitutes the *User report*;
  - R<sub>26</sub> Allow *User* to re-take the *User picture*;
- 
- D<sub>1</sub> Email inserted by the *User* at sign up correspond to a valid email;
  - D<sub>2</sub> *User position* collected at a certain instant corresponds to the actual position of the *User* at that exact moment;
  - D<sub>3</sub> The maps in use accurately represent the world;
  - D<sub>4</sub> *Users* own a working smartphone which has access to Internet connection;
  - D<sub>5</sub> *Users* own a working smartphone which has a working GPS antenna;
  - D<sub>6</sub> SafeStreets and *License plate recognition service* is always online;
- 
- G<sub>2</sub> Send *User report notification* to *Municipality* as soon as it is generated;
- 
- R<sub>11</sub> Send *User report notification* to *Municipality*;
  - R<sub>29</sub> Identify *Municipality* by its *Reference code*;
- 
- D<sub>6</sub> SafeStreets and *License plate recognition service* are always online;
- 
- G<sub>3</sub> Send *User picture* to the *License plate recognition service* as soon as it is received;
- 
- R<sub>4</sub> Receive *User picture*;
  - R<sub>7</sub> Send *User picture* to the *License plate recognition service*;
  - R<sub>26</sub> Allow *User* to re-take the *User picture*;
- 
- D<sub>6</sub> SafeStreets and *License plate recognition service* are always online;
- 
- G<sub>4</sub> Allow *Municipality* to visualize *Detailed statistics*;
- 
- R<sub>10</sub> Store *User report* in SafeStreets Database;
  - R<sub>13</sub> Query SafeStreets Database for a *User* by his/her *User identifier*;
  - R<sub>14</sub> Retrieve specific *User reports* by querying SafeStreets Database;

- R<sub>18</sub> Allow *Municipality* to set specific constraints to define *Detailed statistics*;
- R<sub>19</sub> Send *Detailed statistics* to the requesting *Municipality*;
- R<sub>29</sub> Identify *Municipality* by its *Reference code*;
- D<sub>7</sub> *Municipality* has institutional credentials (*Reference code* and password) which uses to access the system;
- G<sub>5</sub> Allow *Users* to visualize *Public statistics*;
- R<sub>10</sub> Store *User report* in SafeStreets Database;
- R<sub>23</sub> Allow *User* to choose among filters which define *Public statistics*;
- R<sub>24</sub> Send *Public statistics* to the requesting *User*;
- G<sub>6</sub> Suggest *Possible interventions* to *Municipality*;
- R<sub>10</sub> Store *User report* in SafeStreets Database;
- R<sub>13</sub> Query SafeStreets Database for a *User* by his/her *User identifier*;
- R<sub>27</sub> Request *Accidents* data to *Municipality*;
- R<sub>28</sub> Receive *Accidents* data from *Municipality*;
- R<sub>16</sub> Generate *Possible interventions* crossing *Municipality Accidents* data with SafeStreets Database data.
- R<sub>17</sub> Store generated *Possible interventions* for the *Municipality*;
- R<sub>29</sub> Identify *Municipality* by its *Reference code*;
- D<sub>8</sub> *Municipality* grants access to its own *Accidents* Database;
- G<sub>7</sub> Collect information about *Ticket feedback*;
- R<sub>12</sub> Store *Ticket feedback* in SafeStreets Database;
- R<sub>25</sub> Receive *Ticket feedback* from *Municipality*;
- R<sub>29</sub> Identify *Municipality* by its *Reference code*;
- G<sub>8</sub> Allow *Users* to visualize their contribution to SafeStreets;
- R<sub>3</sub> Identify a *User* by his/her *User identifier*;
- R<sub>10</sub> Store *User report* in SafeStreets Database;
- R<sub>12</sub> Store *Ticket feedback* in SafeStreets Database;
- R<sub>13</sub> Query SafeStreets Database for a *User* by his/her *User identifier*;
- R<sub>20</sub> Allow *Users* to request all their *User reports* and the related *Ticket feedback* stored in SafeStreets Database at any time;
- R<sub>21</sub> Send to a specific *User* his/her *User report* stored;
- R<sub>22</sub> Send to a specific *User* the *Ticket feedback* related to his/her *User report*;
- R<sub>25</sub> Receive *Ticket feedback* from *Municipality*.

### 3.3 Performance Requirements

The system to be does not have specific requirements on performances such as response time or algorithm complexity.

### 3.4 Design Constraints

#### 3.4.1 Standard compliance

- GDPR 2016/679 (EU) - General Data Protection Regulation
- ISO/IEC/IEEE 29148:2011 - Standard on requirement engineering

#### 3.4.2 Hardware limitations

In order to use the service, *user's* hardware should comply to these minimum requirements:

##### Mobile application

- Smartphone:
  - iOS or Android operative system;
  - UMTS/4G Internet connection with a minimum speed of 1Mb/s;
  - GPS antenna;
  - 300 Mb available memory;
  - Phone camera of at least 720p resolution;
  - Dual-core processor;
  - 1 Gb RAM.

##### Web application

- Computer:
  - Internet connection with a minimum speed of 1Mb/s;
  - Browser application;
  - 720p monitor resolution.

#### 3.4.3 Any other constraint

SafeStreets has no other constraints.

## 3.5 Design Constraints

### 3.5.1 Reliability

The reliability of the system mainly depends on that of the Internet connection when it comes to sending data to SafeStreets and to the *License plate recognition service*. Fault detection time must be kept minimum so as to promptly identify a fault and repair it as soon as possible.

### 3.5.2 Availability

The system must offer the maximum availability, granting its service every day at any time. The lack of service must be minimal.

### 3.5.3 Security

A high security level is a mandatory constraint for a system like this. This constraint was implicitly filled choosing for a hardened database technology to face with the data storing management problem. Furthermore, talking about data transmission mechanism, every communication between mobile or web interface and the server will be ciphered with AES protocols, and secured with HTTPS/SSL protocols.

The login of *Users* and especially must be very safe (using state of the art login techniques is recommended) to avoid unauthorized individuals to access private information of *Users*. Moreover, the means of communication must be encrypted to save the confidentiality of information sent to SafeStreets.

In addition, integrity about every information, coming from the *User*, related to the *User report*, must be granted by SafeStreets. Information must not be altered or corrupted.

### 3.5.4 Maintainability

A modular approach is one of the several key values which we had chosen to identify for the system.

### 3.5.5 Portability

Portability of *User* data from a device to another is possible by entering personal login data, also for devices with different operating systems. Personal data and *User reports* are stored in a database and they are downloaded when a new device is connected.

## Chapter 4

# Formal Analysis using Alloy

### 4.1 Alloy model

This section includes Alloy code that describes the model and checks whether it is consistent or not. At the end of the code the Alloy generated World is shown. The main objective, driving the formal modeling activity has been the representation of the statistics visualization through SafeStreets application. In particular, the *Detailed statistics*, belonging to *Municipality*, extend the *Public statistics*, belonging to the *User*. Indeed, the *Municipality* can select an additional parameter (license plate) than the *User*, according to the different level of visibility.

Some necessary structural constraints are expressed and properly commented in the Alloy formal notation. These bonds are verified analyzing the worlds generated running the model and checking the relevant assertion.

```
----- SIGNATURES

abstract sig RegisteredEntity {}

sig Individual {}

sig Email {}

sig User extends RegisteredEntity {
    individual: one Individual,
    email: one Email,
    pStatistic: UserReport -> PublicStatistic
}

sig Municipality extends RegisteredEntity {
    referenceCode: one Int,
    dStatistic: UserReport -> DetailedStatistic
} { referenceCode > 0 }

sig Metadata {}

sig Picture {
    width: one Int,
    height: one Int,
```

```

        data: one Metadata
    } { width > 0 and height > 0}

sig Position {
    latitude: one Int,
    longitude: one Int
} { latitude > 0 and longitude > 0 }

// Fields describing the type of violation and
// the license plate number have been, for simplicity
// reasons, modeled as integers instead of strings
sig UserReport {
    user: one User,
    timestamp: one Int,
    typeOfViolation: one Int,
    licensePlateNumber: one Int,
    picture: one Picture,
    position: one Position
} { licensePlateNumber > 0 and timestamp > 0 and
    (typeOfViolation=0 or typeOfViolation=1 or typeOfViolation=2 or
     ↪ typeOfViolation=3) --for simplicity 4 types of violation
}

one sig SafeStreets {
    registeredUsers: set User,
    registeredMunicipalities: set Municipality,
    pStatistic: UserReport -> PublicStatistic,
    dStatistic: UserReport -> DetailedStatistic
}

abstract sig Filter {
    typeOfViolation: lone Int,
    timestamp: set Int,
    position: set Position
} { (#typeOfViolation>0 or #timestamp>0 or #position>0) }

sig PublicFilter extends Filter {}

sig DetailedFilter extends Filter {
    licensePlateNumber: lone Int,
} { licensePlateNumber > 0}

// Public Statistics are a subset of Detailed Statistics.

sig PublicStatistic {
    user: one User,
    pFilter: one PublicFilter
}

sig DetailedStatistic {
    municipality: one Municipality,
    dFilter: one DetailedFilter
}

----- FUNCTIONS

fun getReportsOfDetailedStatistic [ss:SafeStreets, ds: DetailedStatistic]: set
    ↪ set UserReport {
    (ss.dStatistic).ds
}

fun getReportsOfPublicStatistic [ss: SafeStreets, ps: PublicStatistic]: set
    ↪ UserReport {
    (ss.pStatistic).ps
}

```

```

fun retrieveReportsToDetailedStatistic [ss: SafeStreets, ds:
    ↪ DetailedStatistic]: UserReport -> DetailedStatistic {
    ss.dStatistic :> ds
}

fun retrieveReportsToPublicStatistic [ss:SafeStreets, ps: PublicStatistic]:
    ↪ UserReport -> PublicStatistic {
    ss.pStatistic :> ps
}

----- FACTS

//All Emails are associated to a unique User
fact uniqueEmailForUsers {
    all e: Email | one u: User | u.email = e
}

// All Municipalities have a unique Reference Code
fact uniqueRefCodeForMunicipalities {
    all m1, m2: Municipality | (m1 ≠ m2 iff m1.referenceCode ≠ m2.
        ↪ referenceCode)
}

// All Users belonging to SafeStreets are registered
fact allUsersBelongToSafeStreets {
    all u: User | u in SafeStreets.registeredUsers
}

// All Municipalities belonging to SafeStreets are registered
fact allMunicipalitiesBelongToSafeStreets {
    all m: Municipality | m in SafeStreets.registeredMunicipalities
}

// All Individuals are associated to a unique User
fact userIndividualIsUnique {
    all i: Individual | one u: User | u.individual = i
}

// All Pictures belong to one User report
fact pictureBelongsToOneReport {
    all p: Picture | one r: UserReport | r.picture = p
}

// All Positions belong to one User report
fact positionBelongsToOneReport {
    all p: Position | one r: UserReport | r.position = p
}

// All Metadata belong to one picture
fact metadataBelongsToOnePicture {
    all m: Metadata | one p: Picture | p.data = m
}

// All User reports belong to one User
fact UserReportBelongsToOneUser {
    all r: UserReport | one u: User | u in r.user
}

// All Public Filters belong to one Public Statistic
fact publicFilterCorrespondsToOnePublicStatistic {
    all pf: PublicFilter | one stat: PublicStatistic | stat.pFilter = pf
}

// All Public Statistics are associated to a unique User
fact publicStatisticUserIsUnique{
    all ps: PublicStatistic | one u: User | ps.user = u
}

```

```

}

// All Detailed Statistics are associated to a unique Municipality
fact detailedStatisticMunicipalityIsUnique{
    all ds: DetailedStatistic | one m: Municipality | ds.municipality = m
}

// All Detailed Filters belong to one Detailed Statistic
fact detailedFilterCorrespondsToOneDetailedStatistic {
    all df: DetailedFilter | one stat: DetailedStatistic | stat.dFilter =
        ↪ df
}

// All User reports belonging to Municipality are included in SafeStreets and
    ↪ viceversa
fact userReportInMunicipalityImpliesUserReportInSafeStreetsAndViceversa {
    all r: UserReport, ds: DetailedStatistic, ss: SafeStreets, m:
        ↪ Municipality |
    r -> ds in m.dStatistic iff r -> ds in ss.dStatistic
}

// All User reports belonging to User are included in SafeStreets and
    ↪ viceversa
fact userReportInUserImpliesUserReportInSafeStreetsAndViceversa {
    all r: UserReport, ps: PublicStatistic, ss: SafeStreets, u: User |
        ↪
    r -> ps in u.pStatistic iff r -> ps in ss.pStatistic
}

// The following fact ensures that all the Detailed Statistics, made by the
    ↪ Municipality,
// must include all the User reports respecting every selected filter.
// It's not mandatory to select all the filters, but at least one has to be
    ↪ selected.

fact detailedStatisticMadeOfReportsRespectingDetailedFilter {
    all ds: DetailedStatistic, ss: SafeStreets |
    ( (#(ds.dFilter.timestamp)>0 and ((getReportsOfDetailedStatistic[ss,
        ↪ , ds]).timestamp in ds.dFilter.timestamp)) or
        (#(ds.dFilter.timestamp)=0) )
    and
    ( (#(ds.dFilter.typeOfViolation)>0 and (
        ↪ getReportsOfDetailedStatistic[ss, ds]).typeOfViolation = ds.
        ↪ dFilter.typeOfViolation) or
        (#(ds.dFilter.typeOfViolation)=0) )
    and
    ( (#(ds.dFilter.position)>0 and ((getReportsOfDetailedStatistic[ss,
        ↪ ds]).position in ds.dFilter.position)) or
        (#(ds.dFilter.position)=0) )
    and
    ( (#(ds.dFilter.licensePlateNumber)>0 and (
        ↪ getReportsOfDetailedStatistic[ss, ds]).licensePlateNumber =
        ↪ ds.dFilter.licensePlateNumber) or
        (#(ds.dFilter.licensePlateNumber)=0) )
}

// The following fact ensures that all the Public Statistics, made by the
    ↪ User,
// must include all the User reports respecting every selected filter.
// It's not mandatory to select all the filters, but at least one has to be
    ↪ selected.

fact publicStatisticMadeOfReportsRespectingPublicFilter {
    all ps: PublicStatistic, ss: SafeStreets |
    ( (#(ps.pFilter.timestamp)>0 and ((getReportsOfPublicStatistic[ss,
        ↪ ps]).timestamp in ps.pFilter.timestamp)) or
        (#(ps.pFilter.timestamp)=0) )
    and
}

```

```

( (#(ps.pFilter.typeOfViolation)>0 and ((getReportsOfPublicStatistic[
    ↪ ss, ps]).typeOfViolation = ps.pFilter.typeOfViolation)) or
    (#(ps.pFilter.typeOfViolation)=0) )
and
( (#(ps.pFilter.position)>0 and ((getReportsOfPublicStatistic[ss, ps
    ↪ ]).position in ps.pFilter.position)) or
    (#(ps.pFilter.position)=0) )
}

----- PREDICATES

pred sendDetailedStatistic[f: DetailedFilter, m: Municipality, m'':
    ↪ Municipality, ss: SafeStreets, ds: DetailedStatistic] {
    ds in UserReport.(ss.dStatistic)
    f in ds.dFilter
    m'.referenceCode = m.referenceCode
    m'.dStatistic = m.dStatistic + retrieveReportsToDetailedStatistic[ss,
        ↪ ds]
}

pred sendPublicStatistic[f: PublicFilter, u: User, u'': User, ss: SafeStreets,
    ↪ ps: PublicStatistic] {
    ps in UserReport.(ss.pStatistic)
    f in ps.pFilter
    u'.individual = u.individual
    u'.email = u.email
    u'.pStatistic = u.pStatistic + retrieveReportsToPublicStatistic[ss,
        ↪ ps]
}

pred show{}


----- ASSERTIONS

// The User reports selected by the Detailed filter, are correctly sent to
    ↪ the Municipality,
// which requested the corresponding Detailed statistic.
// The previous condition holds by defining this assertion.
assert sendDetailedStatisticOK {
    all m,m'': Municipality, f: DetailedFilter, ss: SafeStreets, ds:
        ↪ DetailedStatistic |
    sendDetailedStatistic[f,m,m'',ss,ds] implies
    (
        ds in UserReport.(m'.dStatistic) and
        getReportsOfDetailedStatistic[ss,ds] in m'.dStatistic.
            ↪ DetailedStatistic
    )
}

// The User reports selected by the Public filter, are correctly sent to the
    ↪ User,
// who requested the corresponding Public statistic.
// The previous condition holds by defining this assertion.
assert sendPublicStatisticOK {
    all u,u'': User, f: PublicFilter, ss: SafeStreets, ps: PublicStatistic
        ↪ |
    sendPublicStatistic[f,u,u'',ss,ps] implies
    (
        ps in UserReport.(u'.pStatistic) and
        getReportsOfPublicStatistic[ss,ps] in u'.pStatistic.
            ↪ PublicStatistic
    )
}

```

```

----- CHECKS

check sendDetailedStatisticOK for 10

check sendPublicStatisticOK for 10

----- RUN

run show for 10 but exactly 2 User, exactly 1 Municipality, exactly 4
    ↳ UserReport, exactly 2 DetailedStatistic, exactly 2 PublicStatistic

```

#### 4.1.1 Analysis results

In this subsection are shown the results of the checks performed on the two assertions.

```

Executing "Check sendDetailedStatisticOK for 10"
Solver=sat4|Bitwidth=4|MaxSeq=7|SkolemDepth=1|Symmetry=20
82344 vars. 5241 primary vars. 235603 clauses. 802ms.
No counterexample found. Assertion may be valid. 161ms.

Executing "Check sendPublicStatisticOK for 10"
Solver=sat4|Bitwidth=4|MaxSeq=7|SkolemDepth=1|Symmetry=20
82444 vars. 5241 primary vars. 235951 clauses. 596ms.
No counterexample found. Assertion may be valid. 115ms.

Executing "Run show for 10 but exactly 2 User, exactly 1 Municipality, exactly 4 UserReport, exactly 2 DetailedStatistic, exactly 2 PublicStatistic"
Solver=sat4|Bitwidth=4|MaxSeq=7|SkolemDepth=1|Symmetry=20
25789 vars. 1815 primary vars. 77108 clauses. 125ms.
Instance found. Predicate is consistent. 376ms.

3 commands were executed. The results are:
#1: No counterexample found. sendDetailedStatisticOK may be valid.
#2: No counterexample found. sendPublicStatisticOK may be valid.
#3: Instance found. show is consistent.

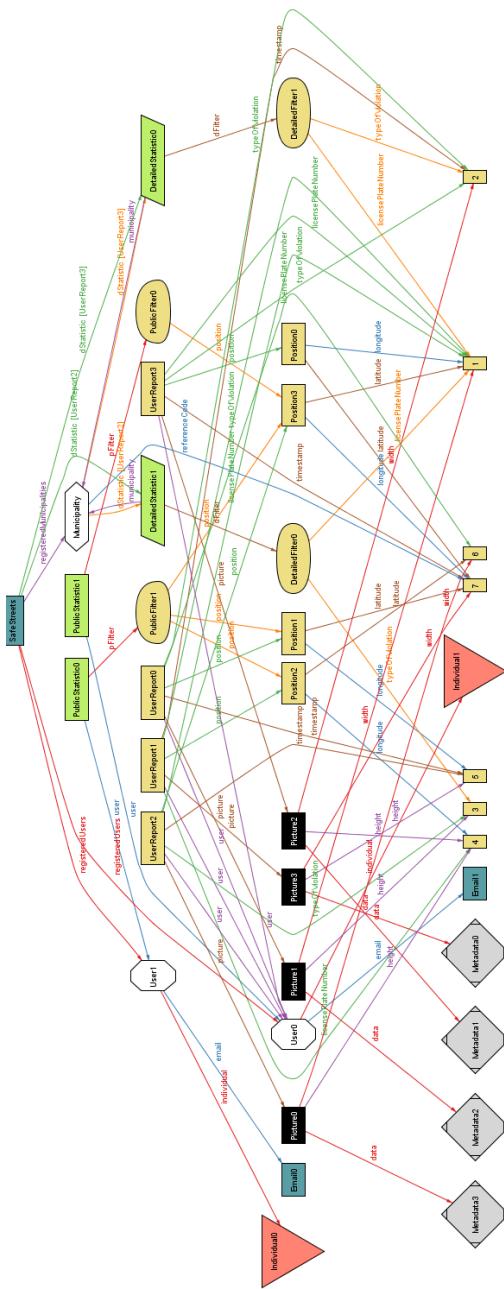
```

Alloy Analyzer summary

## 4.2 Graph

From the World in the following figure, generated by the run, it can be noticed that all the *Users* correspond to a unique Individual and have a unique Email (i.e. *User identifier*). Moreover, every *User report* has a unique Picture (i.e. *User picture*, which in turn has associated a unique Metadata), a Position (i.e. *User position*), a *Type of violation*, a *License Plate* and a Timestamp. It also shows that the two *Detailed statistics* are associated to the only *Municipality* generated in this case, while the two *Public statistics* generated, are associated to two different *Users*.

All the statistics generated are based on a unique filter and, in this case, all the filter parameters correspond to some *User report* parameters. In general, filter parameters can correspond to no or more than one *User report*.



Alloy World generated

# Chapter 5

# Effort Spent

**Ferrara Fabiana** Total hours of work: 55 h

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 4h RASD Review homework
- 4h Purpose, Scope, Goals
- 3h Assumptions, Constraints
- 2h Product perspective
- 2h Product functions
- 2h User characteristics
- 4h Goal revision, Domain Assumptions
- 10h Mockup
- 2h Design constraints
- 3h Specific Requirements
- 4h Functional Requirements, Performance Requirements
- 2h Alloy revision
- 6h General revision
- 5h Second revision

**Formicola Stefano** Total hours of work: 53 h

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 4h RASD Review homework
- 4h Purpose, Scope, Goals
- 3h Assumptions, Constraints
- 4h Product Perspective, Class Diagram and State Chart Diagram
- 1.5h Product Functions revision
- 0.5h Mockup revision
- 2h Specific Requirements
- 2h Functional Requirements revision
- 2h Requirements and Satisfying goals revision
- 5h Introduction to the Alloy specification language and its tool
- 8h Alloy modelling of the S2B
- 3h Alloy revision
- 6h General revision
- 5h Second revision

**Guerra Leonardo** Total hours of work: 60 h

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 3h RASD Review homework
- 2h Scope
- 2h Goals
- 3h Analysis of Shared phenomena
- 1h Purpose
- 1h Domain Assumptions
- 6h Definitions, Goals, Domain assumptions, Requirements, Satisfying goals
- 2h Purpose, Scope
- 3h Scenarios
- 1h Revision of Definition, Domain assumptions, Scenarios, Satisfying goals

- 1h Use Case Diagrams
- 5h Use Case Analysis
- 4h Use Case Diagrams
- 5h Sequence Diagrams
- 2h Alloy
- 1h General revision
- 8h Alloy
- 6h Alloy
- 1h General revision
- 5h Second revision

# Chapter 6

# References

- 1 E. Di Nitto. *Lecture Slides*. Politecnico di Milano.
- 2 E. Di Nitto. *Mandatory Project Assignment AY 2019-2020*. Politecnico di Milano.
- 3 ISO/IEC/IEEE 29148:2011. *Standard on requirement engineering*.  
<https://standards.ieee.org/standard/29148-2011.html>.