

*SafeStreets*  
**RASD document**

Dario Miceli Pranio  
Pierriccardo Olivieri

Academic year: 2019 - 2020



**POLITECNICO**  
**MILANO 1863**

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Purpose . . . . .	3
1.2	Scope . . . . .	4
1.2.1	World Phenomena . . . . .	4
1.2.2	Shared Phenomena . . . . .	4
1.2.3	Machine Phenomena . . . . .	4
1.3	Definitions, acronyms, abbreviations . . . . .	5
1.3.1	Definitions . . . . .	5
1.3.2	Acronyms . . . . .	5
1.3.3	Abbreviations . . . . .	5
1.4	Revision History . . . . .	5
1.5	Reference documents . . . . .	5
1.6	Document Structure . . . . .	6
<b>2</b>	<b>Overall Description</b>	<b>6</b>
2.1	Product perspective . . . . .	6
2.1.1	World Phenomena . . . . .	6
2.1.2	Machine Phenomena . . . . .	6
2.1.3	Shared Phenomena . . . . .	7
2.1.4	Class Diagram . . . . .	7
2.1.5	State Charts . . . . .	9
2.2	Product functions . . . . .	9
2.3	User characteristics . . . . .	10
2.4	Assumption and Dependencies Constraints . . . . .	10
2.4.1	Domain Assumption . . . . .	10
2.4.2	Dependencies . . . . .	11
2.4.3	Constraints . . . . .	11
<b>3</b>	<b>Specific Requirements</b>	<b>12</b>
3.1	External Interface Requirements . . . . .	12
3.1.1	User Interfaces . . . . .	12
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	Use Case Diagrams . . . . .	20
3.2.2	Scenarios . . . . .	21
3.2.3	Use Cases . . . . .	22
3.2.4	Sequence Diagrams . . . . .	29
3.2.5	Goal Mapping on Requirements . . . . .	32
3.3	Performance Requirements . . . . .	32
3.4	Design Constraints . . . . .	33
3.4.1	Standards compliance . . . . .	33
3.4.2	Hardware limitations . . . . .	33
3.5	Software System Attributes . . . . .	33
3.5.1	Reliability . . . . .	33
3.5.2	Availability . . . . .	33
3.5.3	Security . . . . .	33
3.5.4	Maintainability . . . . .	34
3.5.5	Portability . . . . .	34

<b>4</b>	<b>Formal Analysis with Alloy</b>	<b>34</b>
<b>5</b>	<b>Efforts</b>	<b>34</b>

# 1 Introduction

This is the RASD document for *SafeStreets*, that provides a general view about key aspects of the project. The purpose of this document is to formalize a description of the system's requirements both functional and non-functional. In the following pages will be covered goals of the application with respect to phenomena. This document is addressed to developers as a guideline to implement the requirements that follows and as an overview for stakeholders.

## 1.1 Purpose

*SafeStreets* is a service that aims to provide *Users* with the possibility to notify authorities when traffic violations occur, and in particular parking violations. The application's goal is achieved by allowing users to share photo, position, date, time and type of violation and by enabling *Authorities* to request them.

*Safestreets* requires the *Users* to create an account to access its services, the functionalities unlocked after registration depend on the type of account created.

If a user creates an account as *Citizen*, he/she must provide a fiscal code in order to prove that he/she is a real person. Furthermore, he must provide an email with which he will be uniquely identified and a password. Once the account has been activated, user can finally start to report parking violations and can also see a statistics of the streets with the highest frequency of violations.

On the other hand, an officer will create an account as *Authority* and he will need to provide his name, surname, work's Matricola, a password and as for *Citizen*, will be uniquely identified by an email. Once the Matricola has been verified and the account has been activated, the officer can retrieve the potential parking violations sent by *Citizen* that have not been taken into account yet by other officers, analyze them and, if it is the right case, generates traffic tickets. *Authorities*, can see the same statistics of the *Citizen* and can also see statistics about vehicles' license plate that commit the most violations.

From this brief description of the functionalities we may extract the following goals for *SafeStreets*:

- [G1]: Allow *Users* to be identified as a *Citizen* or as *Authority*;
- [G2]: Allow *Citizens* to report parking violations;
- [G3]: *Citizen* has to be able to input information about the violation that he has reported for *Users*;
- [G4]: Must provide a visualization of the areas with high frequency of violations to *Users*;
- [G5]: Must provide a visualization of vehicles that commit the most violations to *Authorities*;

*Safestreets* offers also some advanced functions in addition to the basic version.

- [G6]: Must ensure the chain of custody of the information sent by *Citizens*;
- [G7]: *Authorities* can retrieve traffic violations' in order to generate traffic tickets;

- [G8]: *System* must build statistics with the informations about issued tickets;

## 1.2 Scope

Here we will describe all the relevant phenomena that may occur.

### 1.2.1 World Phenomena

Those are the events that may occur in the real word and are not affected by the Machine.

We identify:

- *Citizen* sees a parking violation and wants to report it;
- *Users* want to know about some violations that have been occurred;
- A *parking violation* occurs;

### 1.2.2 Shared Phenomena

Shared phenomena are the events based on the link beetween World Phenomena and Machine Phenomena. We can distinguish them in two types:

Controlled by the world observed by the machine:

- A *Citizen* reports a violation;
- *Users* can enter data for registration/login;
- *Users* can request data;

Controlled by the machine observed by the world:

- Track position of the violation;
- Mark areas with an high rate of violations;
- System can fullfill data requests;

### 1.2.3 Machine Phenomena

The Machine Phenomena are the events that occur inside the machine and are not affected by the real world.

We identify:

- Storing permanently collected data;
- Encryption of sensitive data;
- Retrieving data for a request;

## 1.3 Definitions, acronyms, abbreviations

### 1.3.1 Definitions

- *Users*: can be either *Citizen* or *Authority*
- *traffic violation*: generic violation that can occur in a street
- *parking violation*: a violation caused by a bad parking
- *violation*: general violation, identity both traffic or parking violation
- *unsafe areas*: areas with an high rate of violations

### 1.3.2 Acronyms

Table with all acronyms used in document.

ACRONYM	COMPLETE NAME
RASD	Requirements Analysis and Specification Document
GPS	global positioning systems
S2B	Software to be
GDPR	General Data Protection Regulation
FC	Fiscal code
UC	Use Case

### 1.3.3 Abbreviations

- **Gn**: n-th Goal
- **Rn**: n-th Requirement
- **Dn**: n-th Domain Assumption
- **Cn**: n-th Constraint
- **UCn**: n-th Use Case

## 1.4 Revision History

### 1.5 Reference documents

- ISO/IEC/IEEE 29148: <https://www.iso.org/standard/45171.html>
- Specification Document: "SafeStreets Mandatory Project Assignment"
- Diagrams: <https://www.draw.io/>
- Mockups: <https://www.figma.com/>
- Alloy Official Documentation: <http://alloy.lcs.mit.edu/alloy/documentation.html>

## 1.6 Document Structure

- **Chapter 2:** Presents an overall description of the system explaining in more detailed way Phenomena described in chapter 1. Provides some diagrams useful to understand key aspects and general behavior of the system and possible type of *Users* with respective functions that they are allowed to do. This chapter is also focused on defining functional requirements such as constraints, domain assumption and dependencies that will be covered later.
- **Chapter 3:** This chapter is intended for developers, dives deeper on the aspects of chapter 2 using use cases and sequence diagrams in order to clarify process and interaction between *Users* and *System*. Describe the interfaces for the application, focusing on system's design constraints and software systems attributes.
- **Chapter 4:** Uses Alloy to generate a Formal Model for the application.

## 2 Overall Description

### 2.1 Product perspective

This section aims to explain in more detail the World, Machine and Shared Phenomena described in the previous Chapter.

#### 2.1.1 World Phenomena

- **Citizen sees a parking violation and wants to report it:** While the *Citizen* is quietly walking, he sees a parking violations like a double parking or a car parked in the middle of bike lane and wants to report it.
- **Users want to know about some violations that have been occurred:** An *User* has the needs to check some statistics about parking violations on a certain area for some purpose.
- **A parking violation occurs:** Someone in the city decides to not follow parking rules and doesn't park his car in a proper way.

#### 2.1.2 Machine Phenomena

- **Storing permanently collected data:** The *System* needs to store, in a secure way, all the data submitted. In order to achieve this purpose and guarantee the best service the *System* needs to use a DBMS.
- **Encryption of sensitive data:** Personal user's data and all the data relative to the violations that can only be seen by authorities need to be encrypted in order to protect it from non-allowed third parties.
- **Retrieving data for a request:** *System* has to fulfill the data request from the *Users*. Data requests can be of two types, a *Citizen* request to see statistics of a certain city area or data request by *Authorities* who want to receive the violation reports collected by *SafeStreets*

### 2.1.3 Shared Phenomena

Controlled by the World observed by the Machine

- **A Citizen reports a violation:** Situation in which a *Citizen* spots a generic violation and wants to report it through the application. Using the phone camera he can take the photo of the violation.
- **User can enter data for registration/login:** A *User* decide to use the application and provides his personal data in order to register if it's the first time he use the app, or to identify himself.
- **Users can request data:** In this phenomena we make a distinction between *Citizen* and *Authorities*. A *Citizen* may want to see violation statistics of a certain area, *Authorities* can request violation statistics and most egregious offender's vehicles statistics.

Controlled by the Machine observed by the World

- **Track position of the violation:** The *System* can retrieve the position where the violation occurred by fetching it from GPS service.
- **Mark areas with an high rate of violations:** Once some violations have occurred, the *System* mines the information that it has in order to highlight the areas with the highest frequency of violations.
- **System can fulfill data requests:** After processing a request, the *System* will show to the *User* the result of the DBMS query in a proper way.

### 2.1.4 Class Diagram



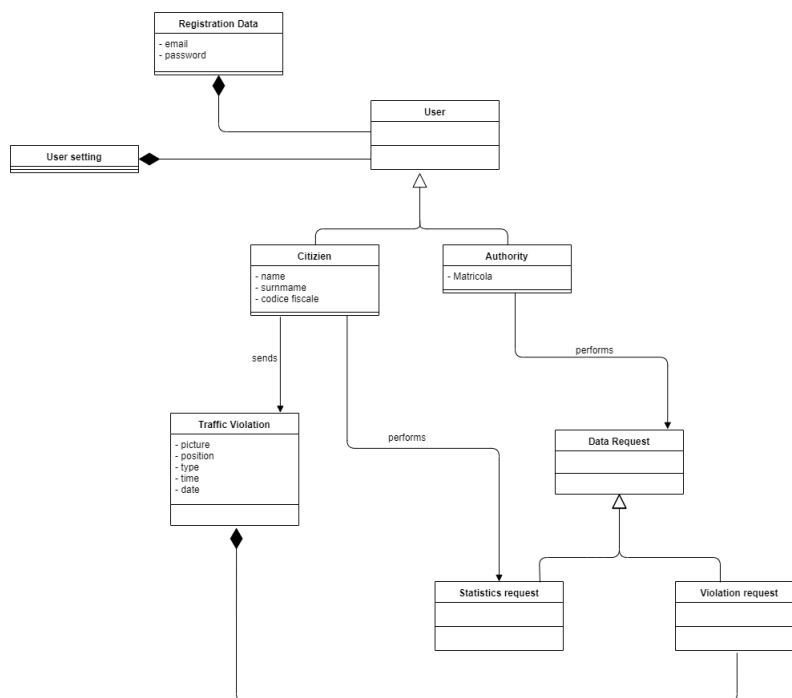


Figure 1: Safestreets' Class diagram

### 2.1.5 State Charts

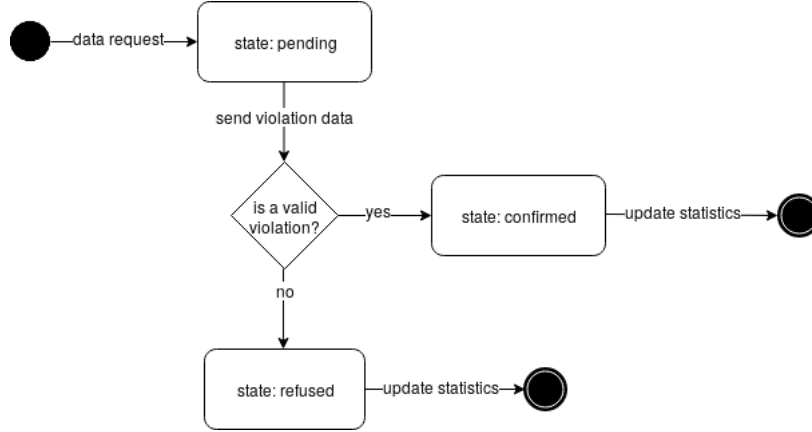


Figure 2: Authority requests for violations state chart

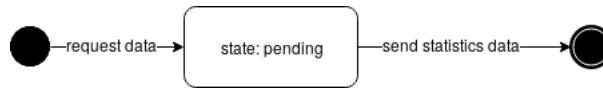


Figure 3: Users request statistics state chart

## 2.2 Product functions

In this section are explained the functions associated to User.

- **Citizen functions:**

### Report a violation

When a *Citizen* sees a parking violation, he takes a picture of the vehicle paying attention to focus on the license plate, inputs the type of the violation and sends it. The System will provide to add the position retrieving from GPS, to add the right time and date and to add the license plate obtained through the algorithm and confirmed by the *Citizen*.

### Retrieve statistics about unsafe areas

*Safestreets* enables *Citizen* to visualize statistics about *unsafe areas*. *SafeStreets* mines the informations it has and let the *Citizen* retrieves the result through a clear interface containing significant plots, tables and charts.

- **Authority functions:**

### Retrieve statistics about unsafe areas

*Safestreets* enables *Authority* to visualize statistics about unsafe areas. *SafeStreets* mines the informations it has and let the *Authority* retrieves the result through a clear interface containing significant plots, tables and charts.

### Retrieve statistics about vehicles

*SafeStreets* enables *Authority* to visualize statistics about vehicles. *SafeStreets* mines the informations it has and let the *Authority* retrieves the result through a clear interface containing significant plots, tables and charts about most egregious offenders.

#### **Request violations data for traffic tickets**

*SafeStreets* enables *Authority* to retrieve all the parking violations sent by *Citizens*. For each parking violation *Authority* can accept it or decline it. In the first case he can generate a traffic ticket, in the second case he discards the informations about the parking violations. In both cases *SafeStreets* records response in order to build statistics.

### **2.3 User characteristics**

Below we describe the convention used to identify the *Users* of the application and the function that those *Users* are allowed to perform.

- **Guest:** A user that has downloaded the application but is not registered yet. This type of user is not allowed to access the application functionalities.
- **Citizen:** is a generic user app not related to authorities, a common *Citizen* that wants to use the application. After the registration process, he can log in the application and use the functionalities such as report a violation or request informations about the statistics of a certain area.
- **Authority:** This user is associated to the local municipal police district, any traffic warden, once registered with his matricola number and logged in has full access to statistics, both violations and vehicles, and can request all the violations reported from *Citizens* in order to generate traffic tickets.
- **User:** can be both a *Citizen* or *Authority* type, in this document this name is used when it's not necessary to make a distinction between the two.

### **2.4 Assumption and Dependencies Constraints**

#### **2.4.1 Domain Assumption**

The following list presents all the domain assumptions made.

- [D1]: *Users* can't make more than one account.
- [D2]: The personal informations provided by *User* are valid and belong to the him.
- [D3]: The *Citizen* assumes all responsibility for misrepresentation on a violation report.
- [D4]: *Citizen* who use the application are evenly distributed in any city area.
- [D5]: The Matricola provided by *Authority* is valid and related to him.
- [D6]: Position data has an accuracy of 10 meters.
- [D7]: The *System* can access internet whenever it needs it.
- [D8]: Permission to access GPS data is always allowed.
- [D9]: Permission to take a photo is always allowed.

### 2.4.2 Dependencies

This list below represent all the dependencies that S2B need in order to work properly.

- Smartphone needs an internet connection.
- Smartphone needs an a Photocamera.
- Smartphone needs a GPS system.
- SafeStreet needs a trusted external storage for violations data and personal data.

### 2.4.3 Constraints

- The S2B must guarantee the European data protection GDPR for user's sensitive data.
- The S2B will be used only in Italy due to personal data type like (fiscal code and police matricola).
- The S2B will be developed as a smartphone application.
- The *Citizen* can only take photos from the application.

## 3 Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

**Login or register page** This is the first page that *Users* see after downloading and installing the SafeStreets application. Both *Authorities* and *Citizens* can log in from this page without distinction because they have to provide only email & password. If *User* hasn't been registered already in SafeStreets can go to register page by click on register button and the *System* will show the default register for *Citizen*.

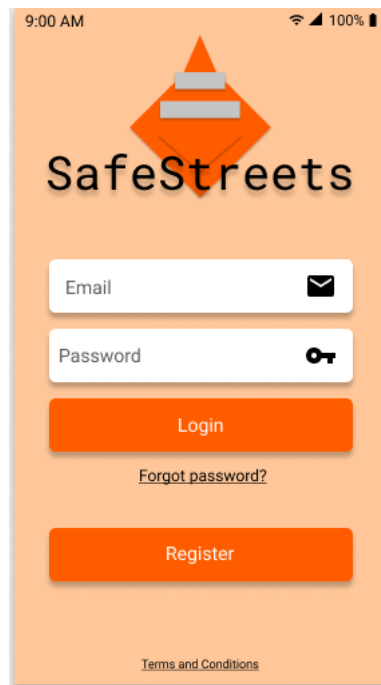


Figure 4: Login or Register page

**Registration page** Registration pages ask *Guests* to input name, surname, email and a password. If the *Guest* is a *Citizen* he must also input his Fiscal Code otherwise if he is a *Authority* he must input his Matricola. The default page that the *System* shows when the register button is clicked is the *Citizen* registration page. If the *Guest* wants to register him as *Authority* he must click the "Register as Authority" button. From this page it is possible to return in the *Citizen* registration page by clicking the "Register as Citizen" button.



9:00 AM 100%

**SafeStreets**

Email 

Password 

Matricola 

Name Surname

Register

Login Register as Citizen 


[Terms and Conditions](#)


(a) Register for *Authorities*



9:00 AM 100%

**SafeStreets**

Email 

Password 

Fiscal code 

Name Surname

Register

Login Register as Authority 

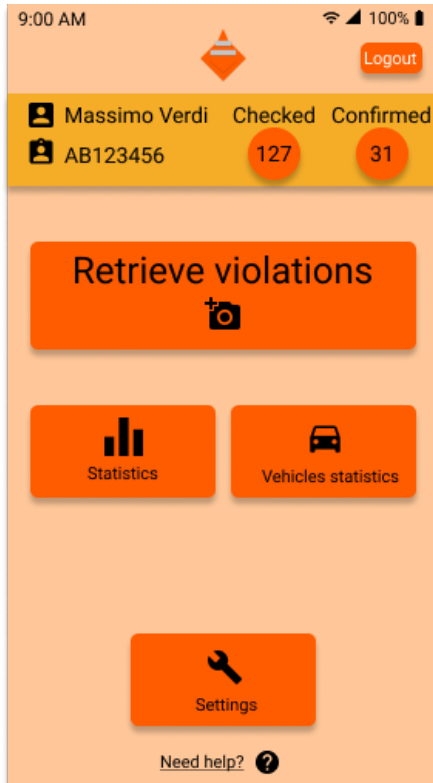
[Terms and Conditions](#)

(b) Register for *Citizens*

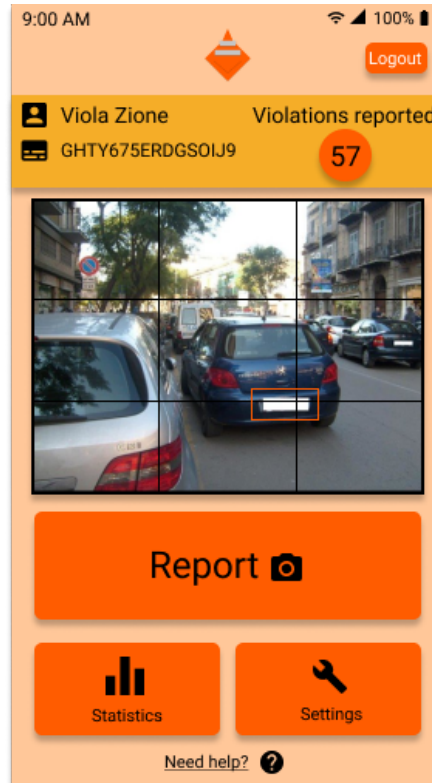
Figure 5: Registration pages

## Home pages

- **Citizen:** Home page shows a bar at the top of screen with some data such as name, surname, FC and the number of violations reported. In the center of screen *System* shows the open photcamera ready to take a picture by clicking the report button. It is possible to take a picture only if a license plate is framed with the photcamera. Once the report button is clicked, a picture is taken and the *Citizen* is redirected to the *Citizen* report info page. The two button at the bottom allow *Citizen* to access statistics and account's settings.
- **Authority:** Home page shows a bar at the top of screen with some data such as name, surname, Matricola, the number of violations checked and the number of violation confirmed. In the center of screen there are 3 buttons: Retrieve Violation, Statistics, Vehicles statistics. The first one allow *Authority* to access *Authority* report info page, the second one allow to access violation statistics and the last one allow to access vehicles statistics. It is also present the settings button to access account's settings.



(a) Home page for *Authorities*



(b) Home page for *Citizens*

Figure 6: Home pages

**Settings** This two pages below represents the settings page in which the *Citizen* and *Authority* can change their personal data or visualize it. Only some informations can be modified, those who can't be modified are showed with grey color.

9:00 AM

< Settings

Credentials

Massimo Verdi

Password

..... Change

Email

massimo.verdi@municipale.it

Matricola

AB123456

Save

(a) Settings for Authorities

9:00 AM

< Settings

Credentials

Viola Zione

Password

..... Change

Email

viola.zione@polimi.it

Fiscal code

HGTU1239IHY67GF

Save

(b) Settings for Citizens

Figure 7: Settings Pages



**Statistics for Citizens** In questa pagine sono disposte le statistiche utili per il citizen vi sono principalmente 3 grafici il primo riguarda learns violazioni accadute durante un determinato anno e divise per mese a seconda della posizione specificata nella mappa di sotto la quale permette di visualizzare le aree con maggior frequenza di violazioni e permette di spostare il cursore di posizione per avere dettagli in una determinata zona. in the botton grahp possiamo il citizen può vedere invece un grafico che mostra le commit che ha fatto nell'ultimo anno di violazioni.

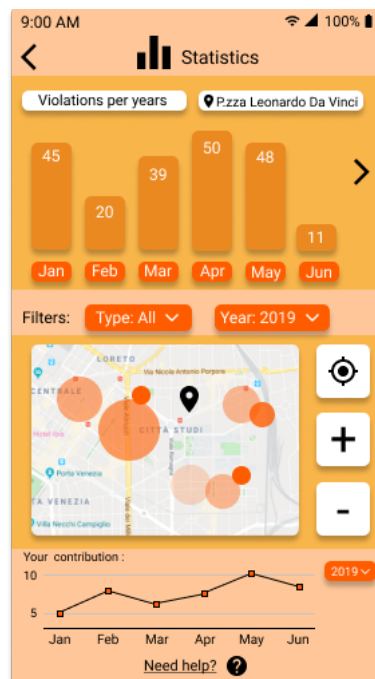
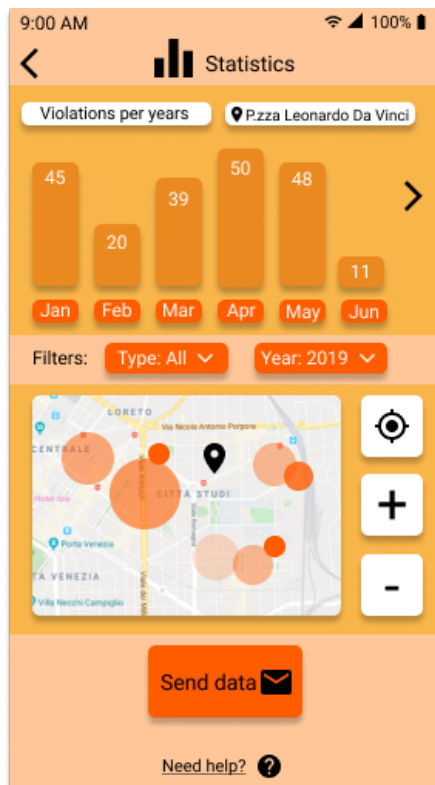


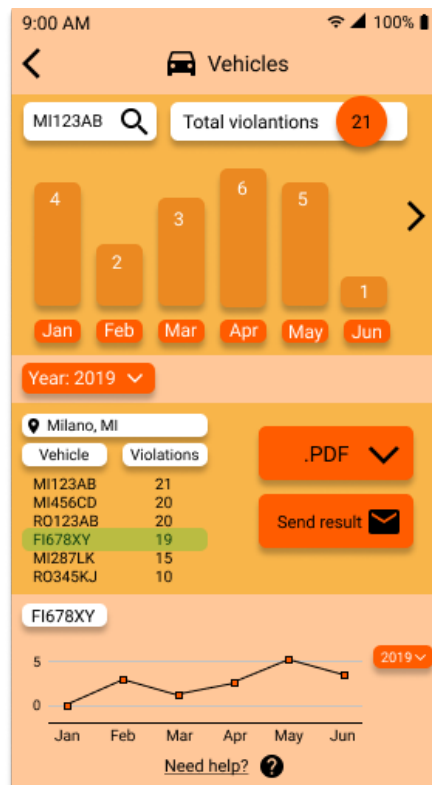
Figure 8: statistics citizen

## Statistics for Authorities

- **Violation's statistics:** I primi due grafici hanno la stessa funzione e sono identici a quelli del citizen lo stesso per la modalità di funzionamento. non è presente ovviamente il grafico delle violazioni riportate ma vi è un bottone per ricevere via mail tali statistiche.
- **Vehicles statistics:** In questa pagina sono invece mostrate le statistiche dei veicoli, nella prima parte è possibile cercare una determinata targa, e se questa ha commesso violazioni si può osservare l'andamento delle violazioni commesse durante i mesi dell'anno selezionato. al di sotto vi è una lista delle targhe dei most egregious offenders con la possibilità di ricevere le suddette targhe via mail. Cliccando su una targa tra queste nel grafico di sotto appariranno mostrate le violazioni commesse durante i vari mesi.



(a) Violations Statistics




(b) Vehicle Statistics


Figure 9: Statistics for *Authorities*


**Violation's report Citizen** Below we can see the page that is displayed when a *Citizen* clicks on the report button taking a photo of a violation. In this page is showed the photo taken and some metadata retrieved automatically by the *System* like date, time, position and the license plate that is read by the algorithm. *Citizen* can change the license plate if the algorithm doesn't read it properly he can also choose the type of violation. And the confirm button, if no error occurs, will send the data collected to the *System*.



9:00 AM


< Report info




 26/10/2019

 9:00 AM

 MI 123 AB Correct? 

 P.zza Leonardo Da Vinci

Please insert the violation type:

Double - parked 

Confirm

Figure 10: Page citizen

**Violation's check page for Authority** In this page showed below the *Authority* can confirm the violations reported by *Citizen* and they can generate traffic tickets from the data. In the page are showed the photo and the data necessary for the traffic ticket. The *Authority* can also receive thi data by clicking on the button in the bottom.

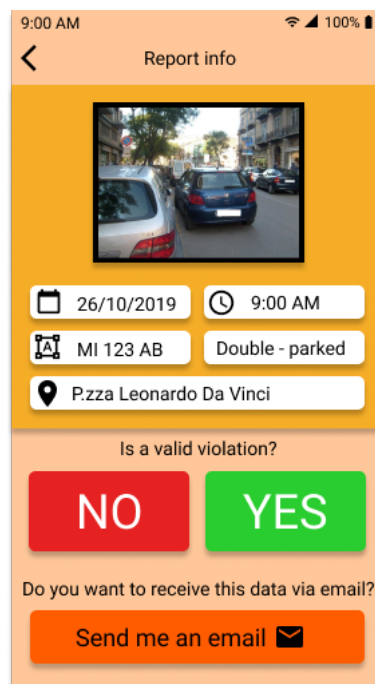


Figure 11: Page authority

### 3.1.2 Hardware Interfaces

The System does not offer any Hardware Interfaces

### 3.1.3 Software Interfaces

As mobile applications, the main software interfaces are:

- iOS
- Android

### 3.1.4 Communication Interfaces

**HTTPS protocol:** to safely communicate through the internet

## 3.2 Functional Requirements

### 3.2.1 Use Case Diagrams

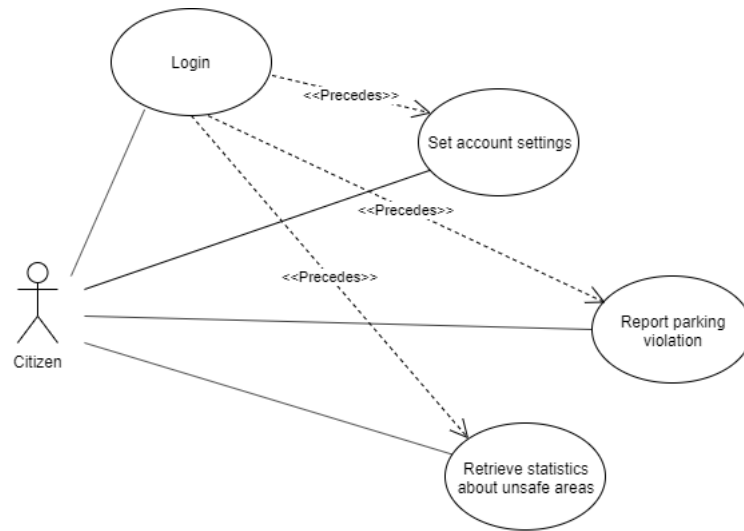


Figure 12: Citizen Use Case Diagram

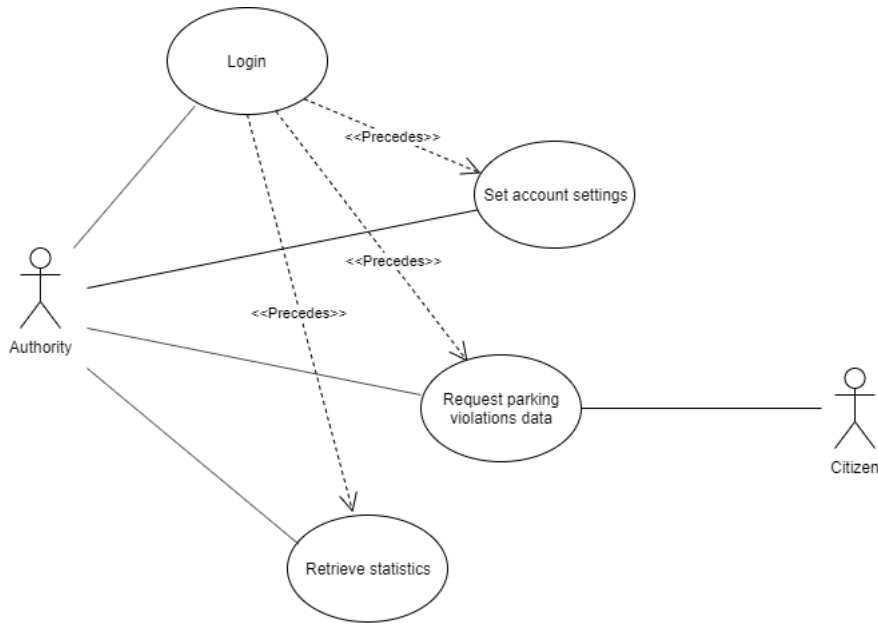


Figure 13: Authority Use Case Diagram

### 3.2.2 Scenarios

- Scenario 1:** Luca is walking towards work when outside, in the street of his house, he sees that many cars are parked badly. Some of these are parked near the strips, obstructing the view of the pedestrian who must cross the strips. Luca tired of the situation, that endangers him and many other citizens, decides to report the incident. With his smartphone he opens the SafeStreets application and after logging in, click on the report button to report the fact. He take the photo by click on the report button in the application's home, then he receives the data retrieved, the plate is correctly recognized and, after inserted the type of violation, Luca clicks on confirm button.
- Scenario 2:** Andrea is a disabled boy, he is perfectly able to drive the car but it is difficult for him to walk for long stretches. The area in which he works is very busy and it is very rare to find parking nearby, fortunately there are parking spaces reserved for disabled people near the entrance of the building. One morning he finds the place occupied and looking better the machine parked he notece that lack of the certificate necessary for parking in the places reserved for the disabled. Thanks to *SafeStreets* after logging in as a citizen, Andrea can report this violation directly to the authorities. Andrea can now take a photo directly from the application's home by click on report button. Safestreest retrieves information such as location and a timestamp with date and time. The application tries to recognize the plaque from the photos and shows the result to Luca, who after confirming the correctness can click on confirm button and officially send the violation.
- Scenario 3:** The command of the municipal of the municipality of Milan wants to optimize his patrols, aiming at the most problematic areas of the city. This targeted surveillance is essential and would bring significant benefits including:

a potential reduction of violations in these areas and reduction of unnecessary patrols in areas with fewer violations. Fortunately, having joined the *SafeStreets* initiative, thanks to the contribution of citizens, they can use the application to receive these statistics directly from smartphones. After having registered as an authority and logged in, they can access the violations statistics. From this page they can see not only a map with a general perspective of the areas but also check for a specific location by moving the pointer on the map.

- **Scenario 4:** Maurizio is a young policeman from the city of Milan. He loves putting a lot of passion into his work and to do so he often learns about new technologies. After downloading SafeStreets and registering as an Authority, he immediately takes an interest in the function to generate fines thanks to the reports made by users. From the home of SafeStreets Maurizio clicks on retrieve violations button, then the application starts showing to him some violations, once a time with a photo and the related data. Maurizio then needs only to analyze the photo and check if it's a valid violations or not. Once he decided he can generate a ticket for that violation and then by confirming click on yes/no button the next violation will be showed. Every answer provided allow SafeStreets to update statistics and give more precise information to users.

### 3.2.3 Use Cases

<b>ID</b>	UC1
<b>Description</b>	A <i>Guest</i> creates a <i>Citizen</i> account
<b>Actors</b>	<i>Guest</i>
<b>Precondition</b>	<i>Guest</i> 's smartphone satisfies hardware limitations <i>Guest</i> has downloaded the app from the store <i>Guest</i> has not an account
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>Guest</i> opens the app</li> <li>2. <i>Guest</i> clicks the registration button</li> <li>3. <i>System</i> shows the <i>Citizen</i> registration form</li> <li>4. <i>Guest</i> fills the form with his personal data plus mail and password</li> <li>5. <i>System</i> checks the validity of the data inserted</li> <li>6. <i>System</i> sends confirmation email</li> <li>7. <i>Guest</i> receives the email and clicks the URL to complete the registration</li> </ol>
<b>Postconditions</b>	<i>System</i> has stored a new <i>Citizen</i> account <i>Guest</i> can login as <i>Citizen</i>
<b>Exceptions</b>	<i>Guest</i> inserts an email that has been used by another account <i>Guest</i> inserts a FC that has been inserted by another account <i>Guest</i> inserts an invalid FC In these case <i>System</i> shows user an error message and the flow of events restart from point 3

Table 1: *Guest* creates a Citizen account



<b>ID</b>	UC2
<b>Description</b>	A <i>Guest</i> creates an <i>Authority</i> account
<b>Actors</b>	<i>Guest</i>
<b>Precondition</b>	<i>Authority</i> 's smartphone satisfies hardware limitations <i>Guest</i> has downloaded the app from the store <i>Guest</i> has not an account
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>Guest</i> opens the app</li> <li>2. <i>Guest</i> clicks the registration button</li> <li>3. <i>System</i> shows the <i>Citizen</i> registration form</li> <li>4. <i>Guest</i> clicks on register for <i>Authority</i> button</li> <li>5. <i>System</i> shows the <i>Authority</i> registration form</li> <li>6. <i>Guest</i> fills the form with his personal data plus Matricola, mail and password</li> <li>7. <i>System</i> checks the validity of the data inserted</li> <li>8. <i>System</i> sends confirmation email</li> <li>9. <i>Guest</i> receives the email and clicks the URL to complete the registration</li> </ol>
<b>Postconditions</b>	<i>System</i> has stored a new <i>Authority</i> account <i>Guest</i> can login as <i>Authority</i>
<b>Exceptions</b>	<i>Guest</i> inserts an email that has been used by another account <i>Guest</i> inserts a Matricola that has been inserted by another account <i>Guest</i> inserts an invalid Matricola In these case <i>System</i> shows user an error message and the flow of events restart from point 5

Table 2: *Guest* creates a *Authority* account

<b>ID</b>	UC3
<b>Description</b>	A <i>User</i> logs in
<b>Actors</b>	<i>Citizen, Authority</i>
<b>Precondition</b>	<i>User</i> has already created the account
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>User</i> opens the app</li> <li>2. <i>System</i> shows login/register interface</li> <li>3. <i>User</i> inputs his credentials</li> <li>4. <i>User</i> clicks login button</li> <li>5. <i>System</i> checks the validity of the data inserted</li> </ol>
<b>Postconditions</b>	<i>User</i> can use properly the app
<b>Exceptions</b>	<i>User</i> inserts wrong credentials In this case <i>System</i> shows user an error message and the flow of events restart from point 2

Table 3: *User* login

<b>ID</b>	UC4
<b>Description</b>	A <i>Citizen</i> reports a parking violation
<b>Actors</b>	<i>Citizen</i>
<b>Precondition</b>	<i>Citizen</i> has already logged in
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>System</i> opens the photocamera</li> <li>2. <i>Citizen</i> clicks the report button</li> <li>3. <i>System</i> shows the report info page for <i>Citizens</i></li> <li>4. <i>Citizen</i> inputs the type of violation</li> <li>5. <i>Citizen</i> clicks the send button</li> </ol>
<b>Postconditions</b>	<i>System</i> 's DB stores the violation
<b>Exceptions</b>	<i>Citizen</i> takes a bad picture In this case <i>System</i> discards the picture and the flow of events restart from point 1

Table 4: *Citizen* reports a parking violation

<b>ID</b>	UC5
<b>Description</b>	A <i>Authority</i> retrieves a legitimate parking violation
<b>Actors</b>	<i>Authority</i>
<b>Precondition</b>	<i>Authority</i> has already logged in
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>Authority</i> clicks the retrieve button</li> <li>2. <i>System</i> shows the report info page for <i>Authorities</i></li> <li>3. <i>Authority</i> checks that it is a real parking violations</li> <li>4. <i>Authority</i> clicks the YES button</li> </ol>
<b>Postconditions</b>	<i>Authority</i> generates a traffic ticket and <i>System</i> uploads statistics
<b>Exceptions</b>	

Table 5: Legitimate parking violation retrieved by *Authority*

<b>ID</b>	UC6
<b>Description</b>	A <i>Authority</i> retrieves a wrong parking violation
<b>Actors</b>	<i>Authority</i>
<b>Precondition</b>	<i>Authority</i> has already logged in
<b>Flow of events</b>	<ol style="list-style-type: none"> <li>1. <i>Authority</i> clicks the retrieve button</li> <li>2. <i>System</i> shows the report info page for <i>Authorities</i></li> <li>3. <i>Authority</i> checks that it is not a real parking violations</li> <li>4. <i>Authority</i> clicks the NO button</li> </ol>
<b>Postconditions</b>	<i>Authority</i> discards the picture and <i>System</i> uploads statistics
<b>Exceptions</b>	

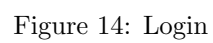
Table 6: Wrong parking violation retrieved by *Authority*

<b>ID</b>	UC7
<b>Description</b>	A <i>User</i> retrieves statistics
<b>Actors</b>	<i>Authority, Citizen</i>
<b>Precondition</b>	<i>User</i> has already logged in
<b>Flow of events</b>	1. <i>User</i> clicks the retrieve statistics button 2. <i>System</i> shows summary of statistics
<b>Postconditions</b>	<i>User</i> increases his knowledge about parking violations of his city
<b>Exceptions</b>	

Table 7: statistics retrieved by *User*

### 3.2.4 Sequence Diagrams

**Login** The following diagram shows how a generic *User* can login into the application. The actors involved are both *Citizen* and *Authority*.



30

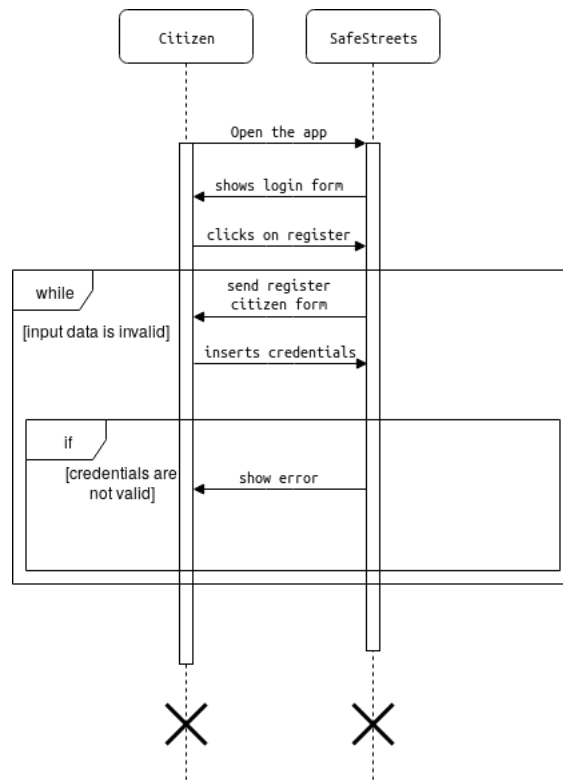


Figure 15: Register for *Citizen*

**Register Authority** This sequence diagram represents the same case as above but referred to *Authorities*, so is necessary another step in order to reach the register form. Then like in the previous case it's possible to complete correctly the registration only by inserting valid credentials.



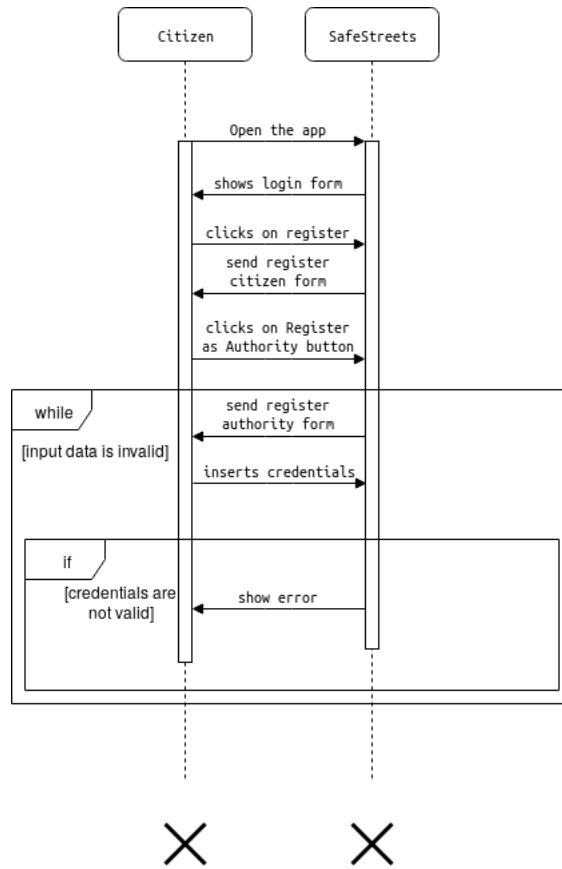


Figure 16: Register for *Authority*

**Retrieve Violation** The following diagram shows how an *Authority* can retrieve violations. Two cases are considered: in the first case the violation is legitimate, he accepts it and generates traffic ticket. In the second case the violation is wrong so he discards it.

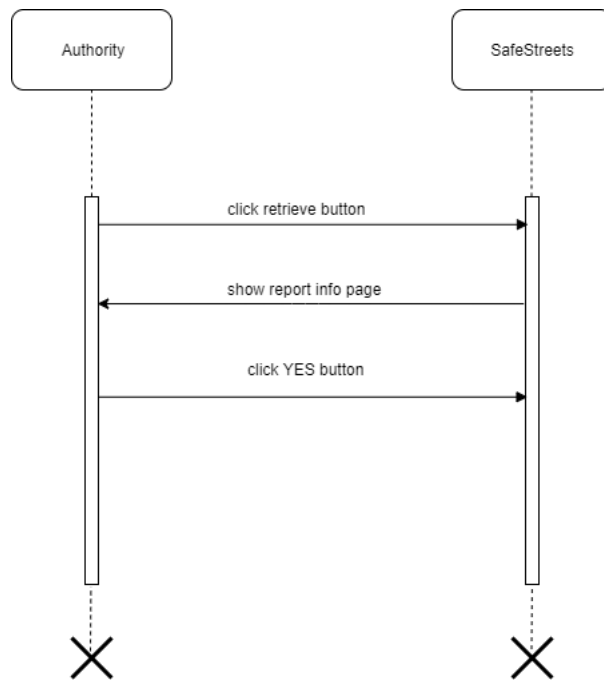


Figure 17: Accept retrieve violation

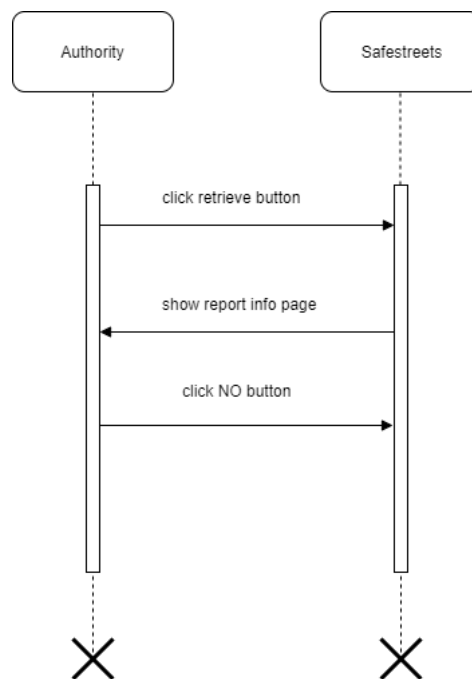


Figure 18: Discard retrieve violation

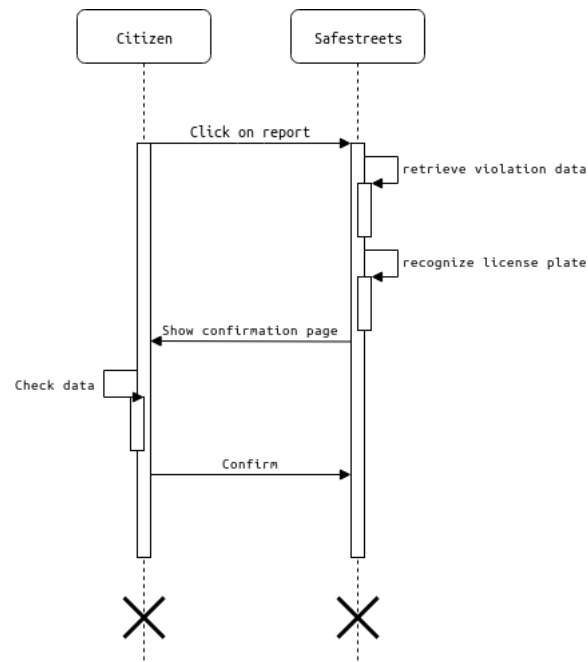


Figure 19: Report violation

**Retrieve statistics** The following diagram shows how a generic *User* can retrieve statistics. The actors involved are both *Citizen* and *Authority*.

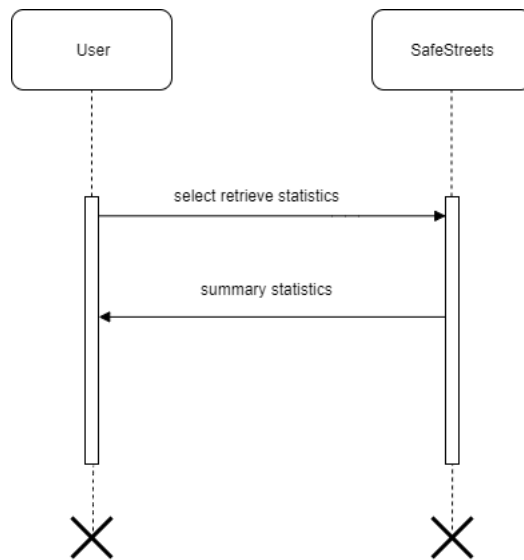


Figure 20: Retrieve statistics

### 3.2.5 Goal Mapping on Requirements

## 3.3 Performance Requirements

In this section we discuss requirements for what regards performance. The System must be able to support up to 5 million of registered users. This limitation is not posed by the front-end of the System, but rather by the back-end part, specifically the DB. For the same reasons it must be able to handle up to 5 million of parking violations sent by the *Citizen*. In order to avoid any kind of saturation, every parking violation that has not been taken into account by any *Authority* for 30 days, must be automatically discarded. This operation does not update the information about statistics.

Requests about statistics shall be processed in less than 5 seconds. Requests about parking violations, instead, shall be processed in less than 1 second.

## 3.4 Design Constraints

### 3.4.1 Standards compliance

The S2B will use certain measures as:

- Standard longitude and latitude measures for the position

For what concerns the privacy, the S2B is subject to GDPR, a regulation in EU law on data protection and privacy for all individual citizens of UE

### 3.4.2 Hardware limitations

In order to work properly the application must rely on hardwares that have certain requirements such as:

- GPS
- internet connection (4G/3G/2G)
- Photocamera with a minimum precision of 5Mp

## 3.5 Software System Attributes

### 3.5.1 Reliability

The system must be able to run continuously without any interruptions. In order to do that, it must be ensured that the system is fault tolerant. To prevent downtime, one of the main goals of architecture design must be ensuring graceful degradation of the System

### 3.5.2 Availability

SafeStreets does not present any critical functions so 99% availability with 3.65 days/year as downtime should be good.

### **3.5.3 Security**

Security is a key aspect of SafeStreets because it is very important that the informations are never altered. The S2B must:

1. use HTTPS to safely communicate with the Server and DBMS
2. Hash the passwords so that they are not stored in clear in the DB
3. Encrypt sensitive data before storing it
4. digital sign the parking violation sent by Citizen and then hash it

### **3.5.4 Maintainability**

In order to achieve maintainability some good practices must be followed to reduce coupling and avoid code duplication

### **3.5.5 Portability**

S2B, as it stated previously, will work both in Andorid and iOS and this ensures itself portability. For the back-end part, it should be OS independent

## **4 Formal Analysis with Alloy**

## **5 Efforts**