SafeStreets
RASD document

Dario Miceli Pranio
Pierriccardo Olivieri

Academic year: 2019 - 2020

POLITECNICO
MILANO 1863

Contents

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-funcional. 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

1.3.3 Abbreviations

- **Gn**: n-th Goal

- **Rn**: n-th Requirement

- **Dn**: n-th Domain Assumption

- **Cn**: n-th Constraint

1.4 Revision History

1.5 Reference documents

- ISO/IEC/IEEE 29148: https://www.iso.org/standard/45171.html

- Specification Document: "SafeStreets Mandatory Project Assignement"

- 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 datailed way Phenomena described in chapter 1. Provides some diagrams usefull 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 defininng 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*.

- **Chapter 4**:

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 so he logs in, takes a picture, inserts the type of violations and sends it to the *System*.

- **Authorities want to know abount some violations that have been occurred:** An Authority logs in and retrieves all the violations that have been send by the Citizen

- **Parking violations occurrs:** Someone in the city decides to not follow parking rules and does not 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 achive 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 proctect it from non-allowed third parties.

- **Retrieving data for a request:** System have to fullfill the data request from the users. Data requests can be of two types, a Citizen request who want 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 take a photo of 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 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 or may want to see violation statistics of a certain area, or in the case of Authorities they can, request violation statistics and additional informations

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 lots of violations occured, the *System* mines the information that it has in order to highlight the areas with the highest frequency of violations

- **System can fullfill data requests** After processing a request, the *System* will show to the user the result of the DBMS query in a proper way
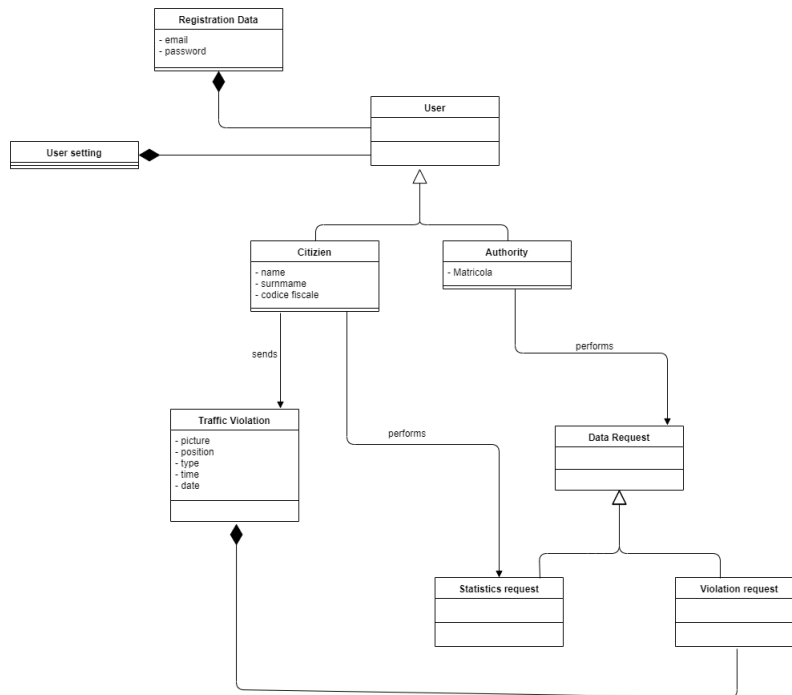
2.1.4 Class Diagram

Figure 1: Safestreets' Class diagram
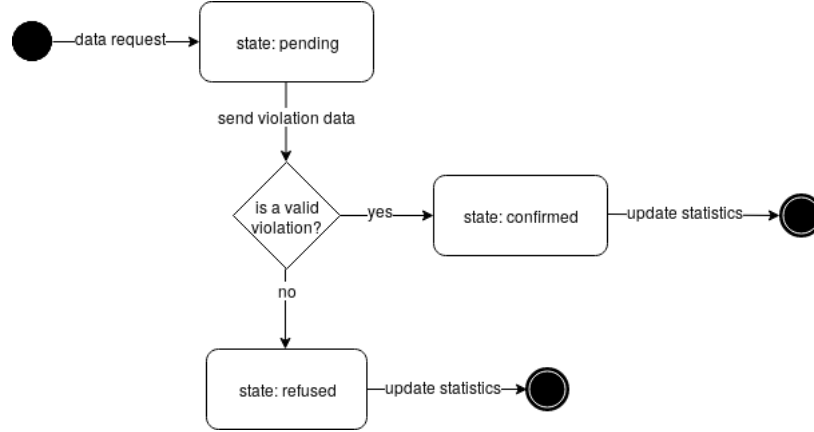
2.1.5 State Charts
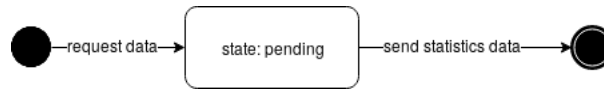

Figure 2: Authority request state chart

Figure 3: Citizen request 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 occurs, 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. **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. For istance Authority can understand which are the vehicles that commit the most violations or who are the 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 accepts it or declines it. In the first case he can generates

8

traffic ticket, in the second case he discard 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 user of the application and the function that those users are allow to perform.

- **Guest:** A user that have donwload 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 want to use the application. After the registration process and the validation of the ID card provided He can log in the application and use the functionalities: - report a violation - request informations about the statistics of a certain area.

- **Authorities:** This user is associated to the local municipal police district, any traffic warden, once registered with is matricola number and logged in have access to those functionalities: - request informations about the statistics of a certain area. - request all the violations reported from Citizens.

- **User:** can be both a Citizen or Authority type, in this document this name is used when it's not necessary make a distinction between the two.

2.4 Assumption and Dependencies Constraints

2.4.1 Assumption

The following list present all the domain assumption made.

- **[D1]:** Users can't make more than one account.

- **[D2]:** The Citizen assumes all responsibility for misrepresentation.

- **[D3]:** Citizens who use the application are evenly distributed in any city area.

- **[D4]:** The ID card present by the Citizen during the registration is valid.

- **[D5]:** The S2B allows to take photos from the application.

2.4.2 Dependencies

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

- A internet connection.

- A Photocamera (with a minimum precision of 3Mp ?(non so se va messo, algoritmo può non riconoscere low quality foto))

- A GPS.

- A Trusted external Storage for violations 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).

3 Specific Requirements

3.1 External Interface Requirements
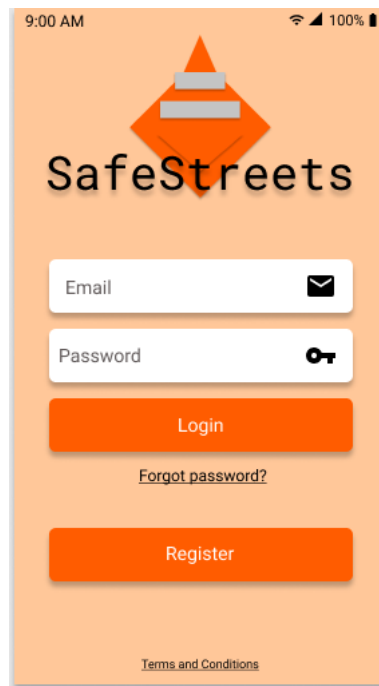
3.1.1 User Interfaces

Figure 4: photos

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

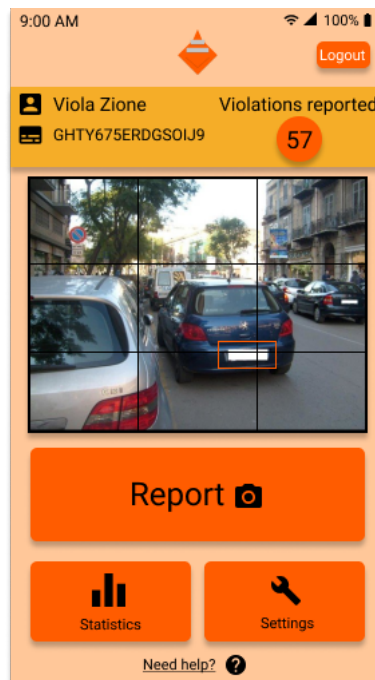Figure 5: Register authority

Figure 6: Register citizen

Figure 7: Home citizen
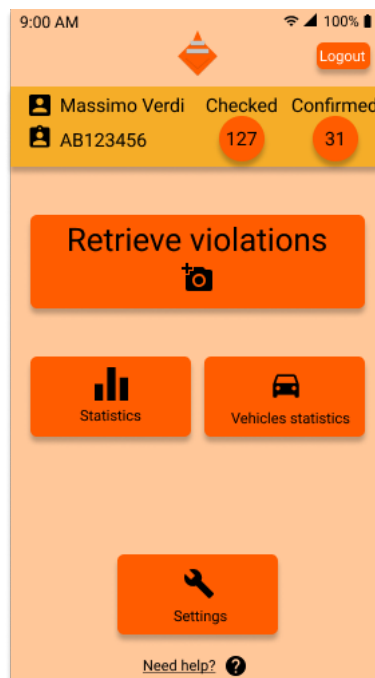

Figure 8: Home authority
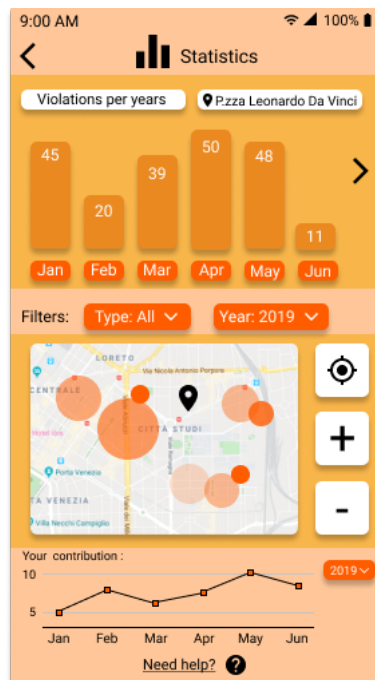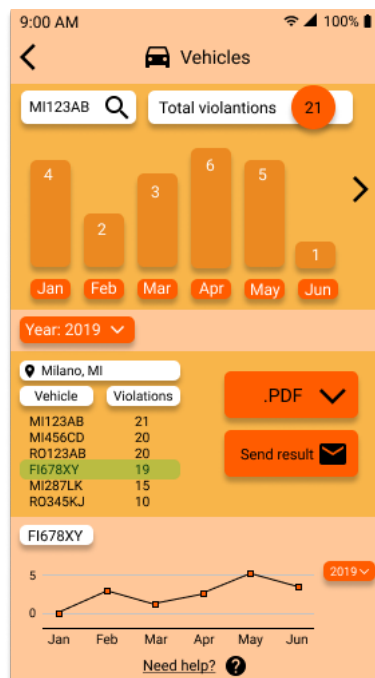
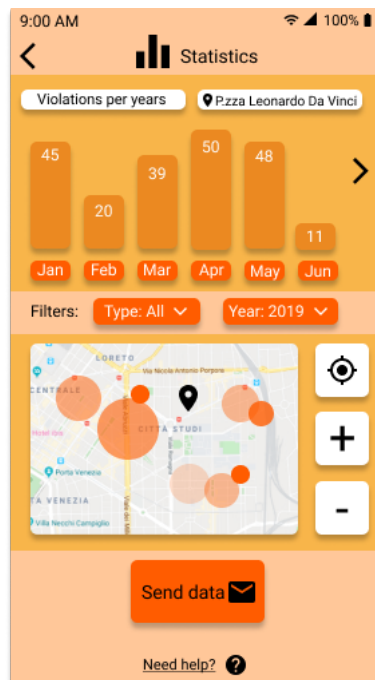Figure 9: statistics citizen

Figure 10: statistics authority

Figure 11: statistics authority
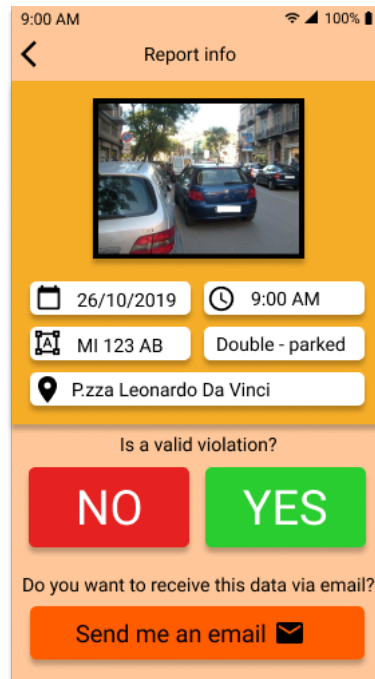

Figure 12: Page citizen
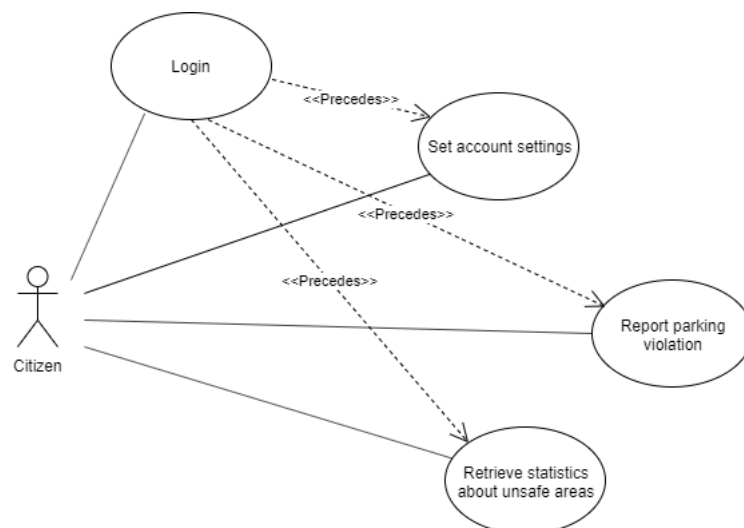
Figure 13: Page authority

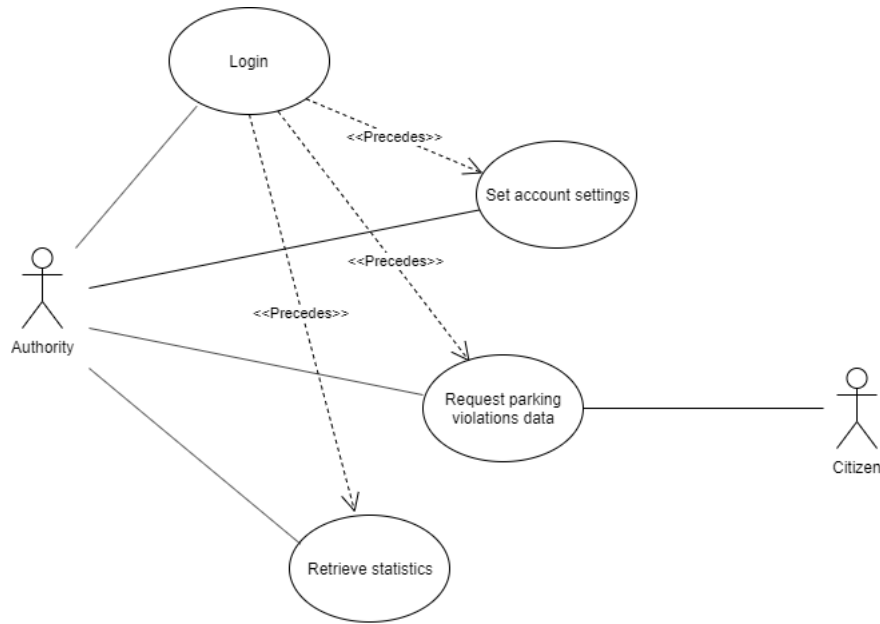

Figure 14: Citizen Use Case Diagram

Figure 15: 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 clik 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

ID	UC1
Description	A *Guest* creates a Citizen account
Actors	*Guest*
Precondition	*Guest*'s smartphone satisfies hardware limitations
	Guest has downloaded the app from the store
	Guest has not an account
Flow of events	1. *Guest* opens the app
	2. *Guest* clicks the registration button
	3. *System* asks which type of account he wants to create
	4. *Guest* selects Citizen account
	5. *System* shows the form
	6. *Guest* fills the form with his personal data plus mail and password
	7. *System* checks the validity of the data inserted
	8. *System* sends confirmation email
	9. *Guest* receives the email and clicks the URL to complete the registration
Postconditions	*System* has stored a new Citizen account
	Guest can login as Citizen
Exceptions	*Guest* inserts an email that has been used by another account
	Guest inserts a FC that has been inserted by another account
	In these case *System* shows user an error message and the flow of events restart from point 5

Table 1: *Guest* creates a Citizen account

ID	UC2
Description	A *Guest* creates a Authority account
Actors	*Guest*
Precondition	*Authority*'s smartphone satisfies hardware limitations
	Guest has downloaded the app from the store
	Guest has not an account
Flow of events	1. *Guest* opens the app
	2. *Guest* clicks the registration button
	3. *System* asks which type of account he wants to create
	4. *Guest* selects Authority account
	5. *System* shows the form
	6. *Guest* fills the form with his personal data plus P.IVA mail and password
	7. *System* checks the validity of the data inserted
	8. *System* sends confirmation email
	9. *Guest* receives the email and clicks the URL to complete the registration
Postconditions	*System* has stored a new Authority account
	Guest can login as Authority
Exceptions	*Guest* inserts an email that has been used by another account
	Guest inserts a P.IVA that has been inserted by another account
	In these case *System* shows user an error message and the flow of events
	restart from point 5

Table 2: *Guest* creates a Authority account

ID	UC3
Description	A *Citizen* logs in
Actors	*Citizen*
Precondition	*Citizen* has already created the Citizen account
Flow of events	1. *Citizen* opens the app 2. *Citizen* clicks the login button 3. *System* shows the form 4. *Citizen* inputs his credentials 5. *System* checks the validity of the data inserted
Postconditions	*Citizen* can use properly the app
Exceptions	*Citizen* inserts wrong credentials In this case *System* shows user an error message and the flow of events restart from point 3

Table 3: *Citizen* login

ID	UC4
Description	A *Authority* logs in
Actors	*Authority*
Precondition	*Authority* has already created the Authority account
Flow of events	1. *Authority* opens the app
	2. *Authority* clicks the login button
	3. *Authority* clicks the login button for Authority
	4. *System* shows the form
	5. *Authority* inputs his credentials
	6. *System* checks the validity of the data inserted
Postconditions	*Authority* can use properly the app
Exceptions	*Authority* inserts wrong credentials
	In this case *System* shows user an error message and the flow of events restart from point 4

Table 4: *Authority* login

ID	UC5
Description	A *Citizen* reports a parking violation
Actors	*Citizen*
Precondition	*Citizen* has already logged in
Flow of events	1. *Citizen* clicks the report button
	2. *System* opens the photocamera
	3. *Citizen* takes a picture
	4. *System* shows the summary
	5. *Citizen* inputs the type of violation
	6. *Citizen* clicks the send button
Postconditions	*System*'s DB stores the violation
Exceptions	*Citizen* takes a bad picture
	In this case *System* discards the picture and the flow of events restart from point 3

Table 5: *Citizen* reports a parking violation

ID	UC6
Description	A *Authority* retrieves a legitimate parking violation
Actors	*Authority*
Precondition	*Authority* has already logged in
Flow of events	1. *Authority* clicks the retrieve button 2. *System* shows summary of a parking violation 3. *Authority* checks that it is a real parking violations 4. *Authority* clicks the ok button
Postconditions	*Authority* generates a traffic and *System* uploads statistics
Exceptions	

Table 6: Legitimate parking violation retrieved by *Authority*

ID	UC7
Description	A *Authority* retrieves a wrong parking violation
Actors	*Authority*
Precondition	*Authority* has already logged in
Flow of events	1. *Authority* clicks the retrieve button
	2. *System* shows summary of a parking violation
	3. *Authority* checks that it is not a real parking violations
	4. *Authority* clicks the discard button
Postconditions	*Authority* discards the picture and *System* uploads statistics
Exceptions	

Table 7: Wrong parking violation retrieved by *Authority*

ID	UC8
Description	A *User* retrieves statistics
Actors	*Authority, Citizen*
Precondition	*User* has already logged in
Flow of events	1. *User* clicks the retrieve statistics button
	2. *System* shows summary of statistics
Postconditions	*User* increases his knowledge about parking violations of his city
Exceptions	

Table 8: statistics retrieved by *User*

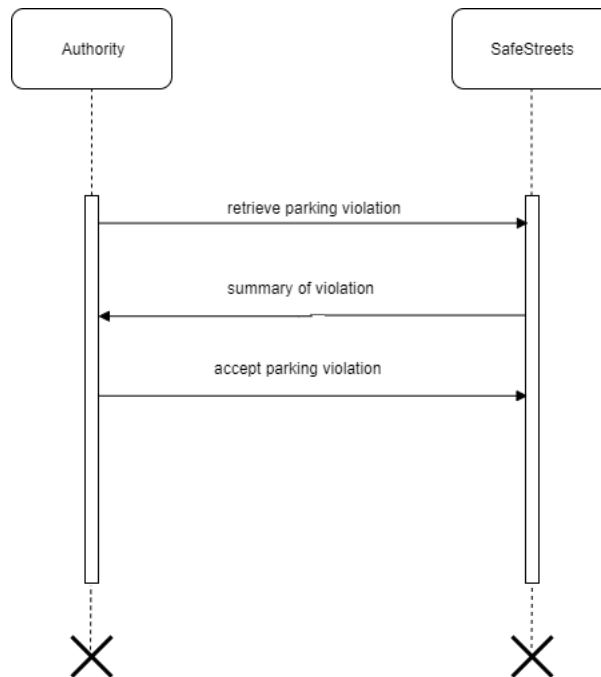### 3.2.4 Sequence Diagrams

Figure 16: Accept retrieve violation

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 1 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 1 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.
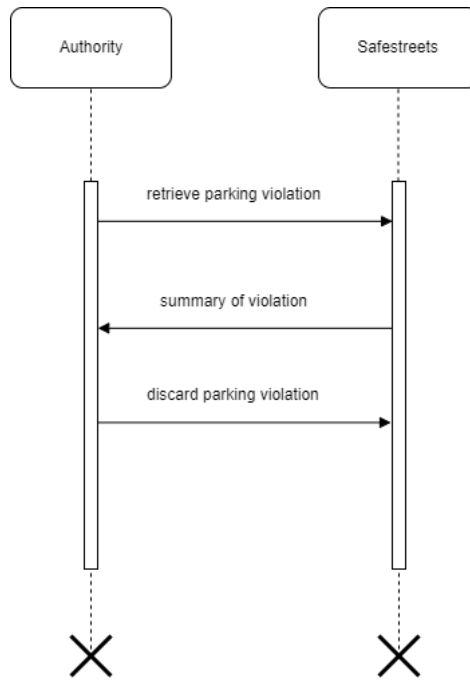
Figure 17: Discard retrieve violation

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,
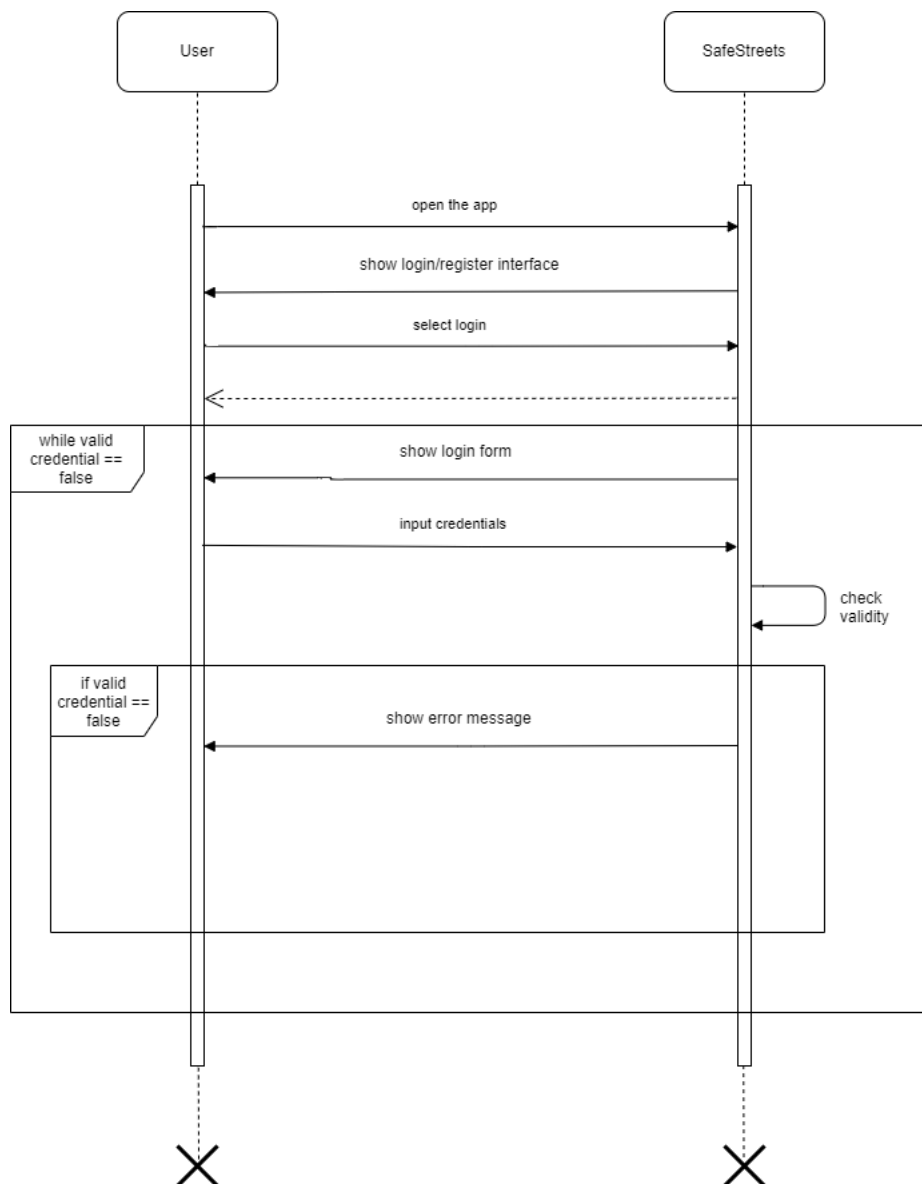
Figure 18: Login

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 shoud be good.
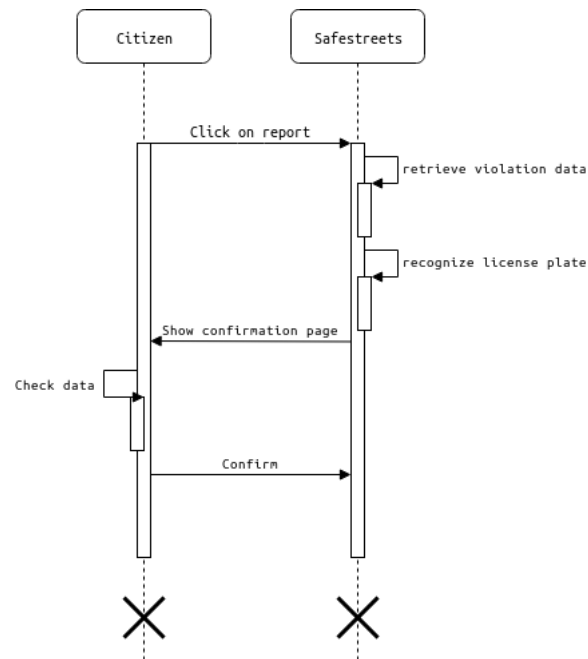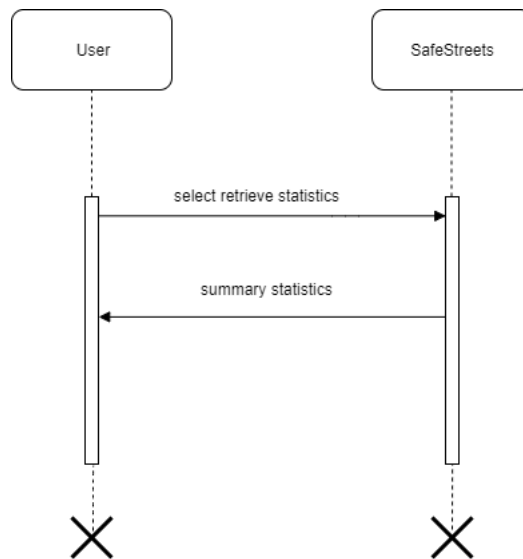
Figure 19: Report violation

Figure 20: Retrieve statistics

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