POLITECNICO
MILANO 1863

SafeStreets

Software Engineering 2 Project - Prof. Matteo Rossi

Requirements Analysis and Specification Document

Salvatore Fadda - 944786
Adriano Mundo - 944684
Francesco Rota - 948714

A.Y. 2019/2020
Version 2.0

December 9, 2019

Contents

1 Introduction

This document represents the *Requirements Analysis and Specification Document* (RASD). It aims at providing an overview of the project *SafeStreets*. It illustrates the purpose of the project, starting from its goals and how these can be reached with certain non functional requirements, functional requirements and constraints. This document is intended for all the people involved in the project life-cycle, from planning and estimation to development and validation.

1.1 Purpose

SafeStreets is a crowd-sourced application that aims at keeping safe the city's streets. The application allows *Users* to notify the *Municipality* when a violation occur on the streets under its jurisdiction. The *User* can notice and notify the violation by sending a photo of the violation including date, time and position. These violations are for the majority parking violation, such as double parking or vehicle parked in the middle of bike lanes.

SafesStreets, once the *User* has notified the violation, stores all the data, completing them with all the necessary metadata. In order to be sure that the violation is correctly elaborated, the application uses an image recognition algorithm. The *User* is notified if something goes wrong during the whole process, so an alternative solution can be found. All the data stored by *SafeStreets* are provided by the *Users* or they can be retrieved directly from the device, like the position from the GPS.
All the data collected by *SafeStreets* can be mined by *Users* and the *Authorities*, so they can be provided with statistics built from this data. The application can show different statistics based on different visibility level, so the *Authorities* can access some information that the *Users* can not access. This is the purpose of the **Basic Service**.

The application has also two specific Advanced Functions. In the **AF1**, the application *SafeStreets* can cross the information about the road accident, thanks to a service offered by the *Municipality*, with the data stored and retrieved from the *User*. So *SafeStreets* elaborates them in order to identify potentially unsafe areas and suggest possible interventions to solve the issues founded. The suggestions can be various and depend on each case.

In the **AF2**, *SafeStreets* allows the *Municipality* to generate traffic tickets directly from the application data, derived from the *User* notifications'. Additionally, starting from the application data about issued tickets, *SafeStreets* can build, looking for trends in the data, statistics and provide insights to the *Municipality*. This can help *Municipality* to improve the process of issued tickets and understand the effectiveness of *SafeStreets*, finding some information useful to improve their services.

From the description above about the purpose of *SafeStreets*, we can summarise those goals:

- **[G1]** Allowing application users to be identified as simple *User* or *Authority*.
- **[G2]** Allowing *Users* to report a traffic violation.
- **[G3]** Allowing *Users* to enter data/information about the violation.
- **[G4]** Providing both *Users* and *Authorities* with statistics built from notifications data.

With regards to *Advanced Function 1*, we identify:

- **[G5]** Identifying potentially unsafe areas and making suggestions to address those issues.

With regards to *Advanced Function 2*, we identify:

- **[G6]** Allowing *Municipality* to generate tickets based on the users notifications.
- **[G7]** Providing statistics built on data from issued tickets to the *Municipality*.

1.2 Scope

In this section we will distinguish between the *Machine*, that's the System to be and the *World*, that's the portion of the real world affected by the *Machine*. This separation, according to the *World and Machine* paradigm by Jackson and Zave, leads to a classification of the phenomena in three different types, depending on where they occur.

1.2.1 World Phenomena

World phenomena are events that take place in the real world and do not have a direct impact on the *System*.

- A generic traffic violation occurs on the streets under the *Municipality's* jurisdiction.
- An accident occurs on the streets under the *Municipality's* jurisdiction.
- A *User* notices the violation and takes action.
- A ticket is generated by the *Municipality*.
- An intervention is made by the *Municipality* to address possible issues on the streets under its jurisdiction.

1.2.2 Shared Phenomena

Shared phenomena are world phenomena that are shared with the Machine. These are further divided in two categories.

Controlled by the world and observed by the machine

- A *Guest* signs up on the system giving all the personal data needed and/or signs in with his credentials.
- A *User* sends a violation notification to the *System*, including type of violation, position, date, time and picture.
- *Municipality* transmits data about the accidents occurring on the streets to the *System*.
- *Municipality* transfers data about tickets generated to the *System*.
- *Authority*, as a representative of the *Municipality* evaluates a violation notification coming from the *System*.
- *Authority*, as a representative of the *Municipality*, evaluates a suggestion for an intervention coming from the *System*.

Controlled by the machine and observed by the world

- The *System* notifies the *User* that the image recognition tool was not successful, in order for the user to re-load the picture.
- The *System* transmits a violation notification to the *Municipality*.
- The *System* provides suggestions on possible interventions to address issues to the *Municipality* through *Authorities*.
- The *System* shows statistics mined from violation notifications data to *Users* and/or the *Authorities*.
- The *System* shows statistics obtained from tickets data to the *Authorities*.
- Data from all the tickets generated by the *Municipality* are stored by the *System*.

1.2.3 Machine Phenomena

Machine phenomena are events that entirely take place inside the System and cannot be observed in the real world.

- The plate recognition algorithm is ran on the picture of the violation.
- The complete set of data are stored and can be retrieved by the *System's* DBMS.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Guest:** someone who needs to sign up to the application in order to access its functionalities, otherwise they can't access to the service.
- **Authorities:** representatives of the Municipality considered as a kind of application user
- **User:** citizen or city's tourist that is registered to the application, so he/she can access the functionality offered by the application. Even if the term User can be considered as a generic term for every application user, in this document we'll consider user as explained above.
- **Municipality:** entity composed of both men and the information system that has authority on the streets, where it has the responsibility to enforce the rules and to guarantee safety.
- **Violation:** the action of violating traffic laws.
- **Ticket:** administrative sanction established by law for a traffic violation.
- **Plate Recognition Algorithm:** algorithm that automatic recognise vehicle's' plate by the images sent from users to report a violation or an accident.
- **Notification Data:** information that is provided by the user when he reports a violation. This includes picture, license plate, date, time, position.
- **Ticket Data:** information that is provided by the municipality when it adds a new ticket in the system. This includes violation type, license plate, date, time, position.
- **Accident Data:** information that usually is provided by the municipality when it stores data about an accident. This can includes accident type, picture, multiple license plate, date, time, position.
- **Intervention:** action taken by the municipality to prevent further issues in the city traffic.
- **Notification:** message sent by the user to advise the system about a violation.
- **Safe Map:** map indicating the violation level of the city. The level is obtained from the users' data.
- **Authority Map:** map indicating the violation level of the city. The level is obtained from the users' data. Also contains sensible data for authorities.

- **Statistics:** all the insights obtained from the analysis of the notifications' data, aggregated and showed to the user or to the authority, as user of the application.
- **Ticket Statistics:** all the insights obtained from the analysis of the tickets' data, aggregated and showed to the authority, because they contains sensible information and are not useful to the user.

1.3.2 Acronyms

- **GPS:** Global Positioning System
- **API:** Application Programming Interface
- **RASD:** Requirements Analysis and Specification Document
- **DBMS:** Data Bases Management System
- **GDPR:** General Data Protection Regulation

1.3.3 Abbreviations

- **[Gn]:** n-th goal
- **[Rn]:** n-th functional requirement
- **[Dn]:** n-th domain assumption
- **AF1:** Advanced Function One
- **AF2:** Advanced Function Two
- **SP1:** Shared Phenomena controlled by the World and observed by the Machine
- **SP2:** Shared Phenomena controlled by the Machine and observed by the World
- **WP:** World Phenomena
- **MP:** Machine Phenomena

1.4 Revision History

Version	Date	Description
1.0	10/10/2019	First Delivery
2.0	09/12/2019	Second Delivery

Table 1: Revision History

1.4.1 Changes

- Changed [**G1**] and [**G2**] to ensure they are described in terms of world/shared phenomena.
- Refactor of the Alloy models, adding comments and graphics.
- Fixed semantic errors to be consistent across the documentation in Use Cases, Class Diagrams, Tables.
- Removed the old [**R17**] and added [**R30**]. Corrected a double [**R23**].
- Fixed Statistics and Interventions Sequence Diagrams.

1.5 Reference Documents

- Mandatory Project Assignment
- Alloy Official Documentation: http://alloy.lcs.mit.edu/alloy/documentation.html
- ISO/IEC/IEE 29148: System and Software engineering - Life cycle process - Requirements engineering

1.6 Document Structure

The rest of the document is organised as follows:

- **Overall Description** (Section 2): it will be given a general description of the application, with an in depth analysis of the phenomena according to the *World and the Machine* paradigm, and the User Characteristic. There will be provided class diagram and state charts in UML Language and also domain assumptions, dependencies and constraints.
- **Specific Requirements** (Section 3): in this section all the Functional Requirements of the application are explained in details and related to use case scenarios and sequence diagrams for clarifying processes and interactions between the actors and the *System*. Also, there are descriptions of all the non-functional requirements and external interfaces.
- **Formal Analysis** (Section 4): description and creation of simulation using a formal model, the Alloy specification language in order to address the critical aspects of the *SafeStreets System*.

2 Overall Description

2.1 Product perspective

In this section we will provide a detailed descriptions of World and Shared Phenomena. Also, there are a Class Diagram and State Charts to clarify the modelling of the application.

A generic traffic violation occurs on the streets under the Municipality's jurisdiction (WP)
A violation of traffic laws occurs on the streets on which the municipality has responsibilities. Violations includes (but are not limited to) parking where it is forbidden (outside parking lanes, on pedestrian crossings, on sidewalks, etc).

An accident occurs on the streets under the Municipality's jurisdiction (WP)
A car accident occurs on the streets on which the municipality has responsibilities. Because guaranteeing safety is one of those, the municipality aims the number of accidents to be the lowest possible. Car accidents do not only occur between two vehicles, but may also include pedestrians, bikes and urban infrastructure.

A User notices the violation and takes action (WP)
A user of the system acknowledges a traffic laws violation and contributes, through the system, to the street safety and rules enforcing effort by reporting the type of violation together with a picture of the vehicle involved.

A ticket is generated by the Municipality (WP)
Authorities, as representatives of the municipality confirm a ticket and municipality issues a ticket to the owner of the vehicle involved in a traffic laws violations.

An intervention is made by the Municipality to address possible issues on the streets under its jurisdiction (WP)
In order to fulfil its responsibility of guaranteeing safety, the municipality makes a modification in the viability or to the urban infrastructure layout on the streets under its jurisdiction to address a safety issue.

The System notifies the User that the image recognition tool was not successful, in order for the user to re-load the picture (SP2).
As soon as the system receives from a user a notification of a traffic laws violation, it instantly runs the iamge recognition algorithm and in case that fails, the system notifies the user asking him to take a new picture in which the plate is more readable.

The System transmits a violation notification to the Municipality (SP2)
The system shares every notification that is reported by users with the municipality, which has the responsibility to take action if necessary.

The System provides suggestions on possible interventions to address issues to the Municipality through Authorities (SP2)
After crossing the information about accidents, which has been transmitted from the municipality, with the data of the violations reported, the system is able to individuate areas and traffic dynamics that are particularly dangerous and to compute possible solutions that would improve safety.

The System shows statistics mined from violation notifications data to Users and/or the Authorities (SP2)
The system allows both users and authorities, as representatives of the municipality to view statistics mined from the notification data. Of course users cannot visualise some of this statistics for privacy reasons (for example the plate of the most egregious violators).

The System shows statistics obtained from tickets data to the Authorities (SP2)
The system allows authorities, as representatives of the municipality to view statistics obtained from tickets' data. Users cannot visualise this insights.

Data from all the tickets generated by the Municipality are stored by the System (SP2)
The municipality keeps detailed track in the database of its information system of every ticket issued, including date, time, position, type of violation and plate number.

A Guest signs up on the system giving all the personal data needed and/or signs in with his credentials (SP1)
A guest registers on the system sharing his identity, date of birth and email address. He then gets valid credentials to login into the System and use the service.

A User sends a violation notification to the System, including type of violation, position, date, time and picture (SP1)
A User reports a violation through the System, taking a picture of the vehicle and selecting the type of violation. The System autonomously includes to that information the date, time and position of the User in the instant he loads the picture.

Municipality transmits data about the accidents occurring on the streets to the System (SP1)
The municipality shares complete data about every accident occurring on the streets with the system.

Authority, as a representative of the Municipality evaluates a violation notification coming from the System (SP1)
Representatives of the municipality evaluate a notification transmitted by the system to verify the actual violation of the traffic laws and to generate, if the conditions are right, a ticket.

Authority, as a representative of the Municipality evaluates a suggestion for an intervention coming from the System (SP1)
Once received a suggestion for an intervention from the system, representatives of the municipality evaluate whether it is consistent and feasible and make a cost and benefits analysis.

Municipality transfers data about tickets generated to the System (SP1)
The municipality shares complete data about every ticket issued with the system.

The plate recognition algorithm is ran on the picture of the violation (MP)
Once received the picture taken by the user, the system runs his tool to read the plate number directly from the picture, to integrate the set of data of the notification.

The complete set of data are stored and can be retrieved by the System's DBMS (MP)
The systems DBMS assures integrity and durability of every piece of data managed by the system: every notification, information on issued tickets and statistics about the previous are available to be retrieved with a proper query.

2.1.1 Class Diagrams

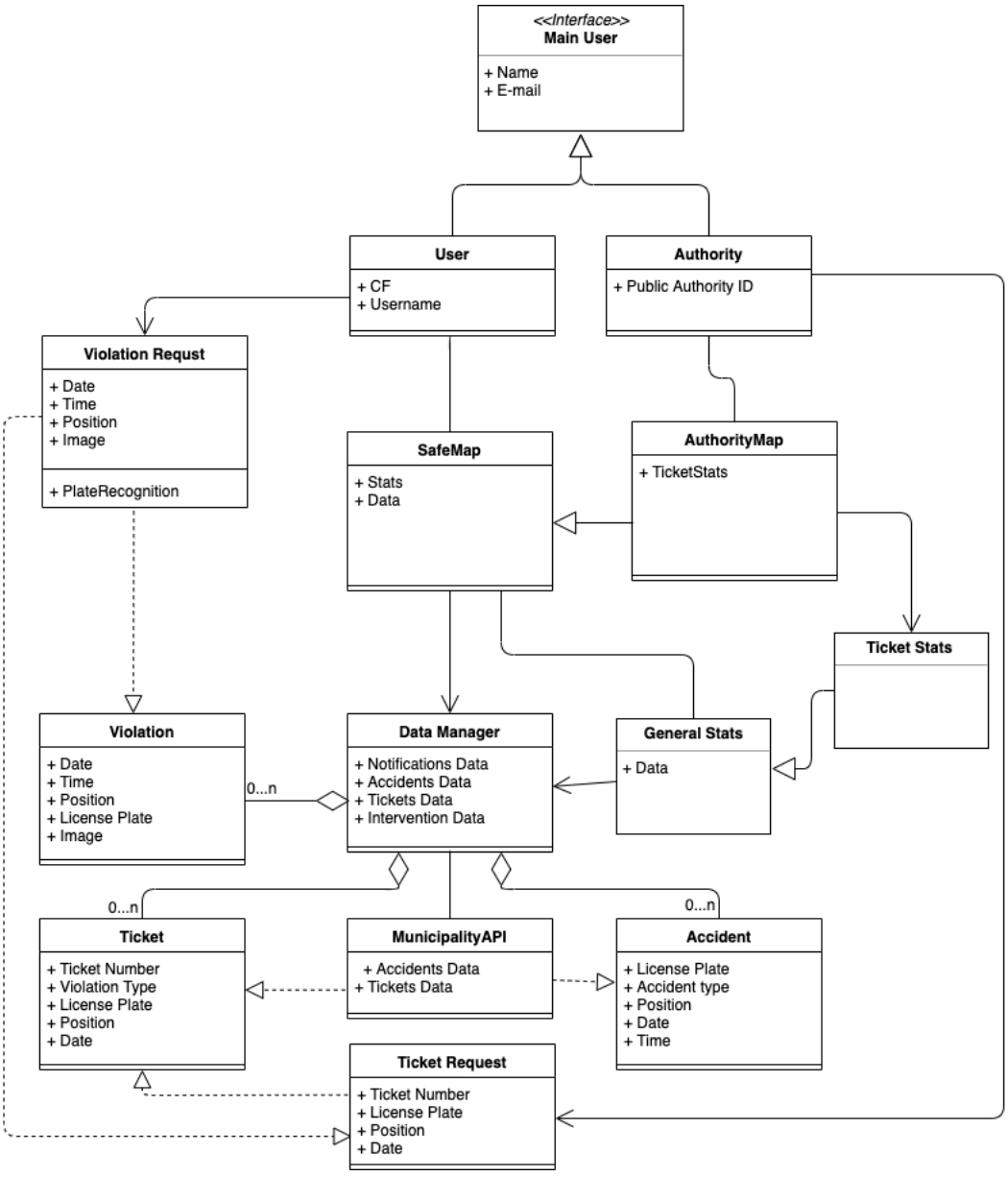


Figure 1: *SafeStreets* Class Diagram

2.1.2 State Charts

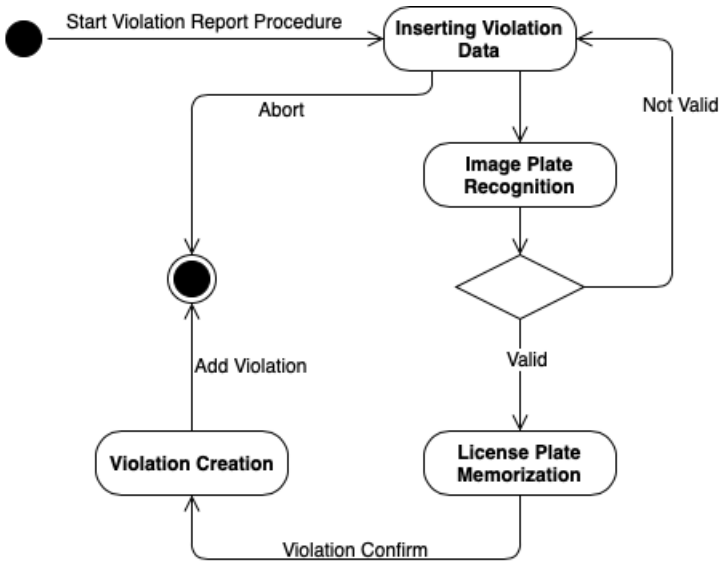


Figure 2: *Violation Report* State Chart

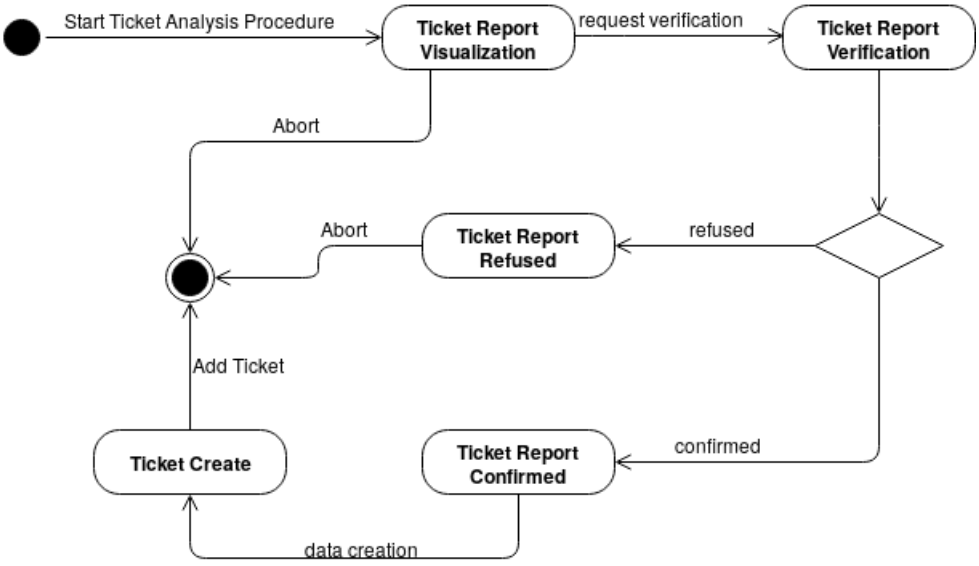


Figure 3: *Ticket Analysis* State Chart

2.2 Product functions

- **User Functions:**

  Report a traffic violation
  The User noticing a violation on the streets is able to, once accessed the System, fill in the information to report a violation: he takes a picture of the vehicle, with the plate decently readable, and select the type of violation from the list provided. The date, time and position are automatically taken by the system when the image is loaded (the picture of the violation can only be taken using the application, not retrieved and uploaded from the photo archive) and integrated to the information given by the User. All the data are eventually sent to the Systems DBMS and the image recognition algorithm is ran to complete the data with the plate number. If the tool fails, the User is instantly notified and asked to take a better picture.

  Access statistics from notifications' data
  The User, once logged in, is able to enter the section of the application where statistics from the notifications received are available: number of violations per period (week, month, year), violations average per day of the week, the previous two per type of violation and a map showing where the violations were reported. Of course no plate number and no information about who reported the violations is shown to the User.

- **Authority Functions:**

  Access statistics from notifications' data
  The Municipality is able to enter the section of the system interface where statistics from the notifications received from its jurisdiction are available: number of violations per period (week, month, year), violations average per day of the week, the previous two per type of violation and a map showing where the violations were reported. As a representative of the municipality, Authority can view the plate number of the offender of any violation notification but has no information on the User that reported it.

  Obtain suggestions about unsafe areas
  The system provides to the Municipality possible solutions to issues individuated through statistics, which are being mined from data on accidents and violations. This is something really valuable for the Municipality because most of the time it doesnt have the tools itself to individuate connections and cause-effect relationship in the series of events (ex. a lot of accidents happening at a corner where a lot of second row parking violations are notified).

Generate tickets
The Municipality receives every notification of violations occurring under its jurisdiction and evaluates every one of them to determine whether a violation actually occurred, if the type of violation reported is correct and if there are the conditions to issue a ticket, in which case the ticket generated is no different from any ticket coming from a physically-verified violation. By exploiting the Systems information the municipality is able to issue a much larger number of tickets in its jurisdiction and to gain effectiveness in enforcing the rules. Every information about generated tickets is then shared with the System.

Access statistics from issued tickets
The Municipality is able to enter the section of the system interface where statistics from the tickets generated are available: number of tickets per period (week, month, year), tickets average per day of the week, the previous two per type of violation sanctioned, a map showing where the violations were reported.

2.3 User characteristics

SafeStreets has two kinds of users. There are **simple** *Users* of the application and *Authorities* (for details about the semantics about the term User refers to the section 1.3.1 of this document). The first type is usually a citizen or a tourist of the city where the *SafeStreets'* service is activated. The *User* needs to notify authorities of a violation, and in order to do this, he/she has to install the application on his/her smartphone. The second type of user are *Authorities*, they are representative of the *Municipality*. They can access the application from their smartphone or any device provided by the *Municipality*. Also their access to the service is provided by the *Municipality* too because they have a different level of visibility in respect to the simple *Users*. They need to access statistics provided by the service or they have to generate tickets.
The characters involved are:

- *Guest:* a user who has downloaded the application on his/her smartphone, but has to create an account. No functionalities can be accessed without registering to the service
- *User:* a user who has created an account, but has to log-in to access the feature. Signing-up with all the necessary data or inserting his credentials, he/she can access all the functionalities offered by the application.
- *Logged-in User:* a user who has logged-in the application providing the credentials established during the registration phase. He's able to access all the functionalities.
- *Authorities:* a user that's a representative of the Municipality, with its own privileges and levels of visibility. Logging-in with his/her credentials, provided directly from the Municipality, has the possibility to access the functionalities.
- *Logged-in Authorities:* a user that's a representative of the Municipality, that has provided the credentials and can access all the functionalities.

2.4 Assumptions, dependencies and constraints

2.4.1 Constraints

- *Users* are located on the streets under the *Municipality* jurisdiction when they notify the violation.
- *Users* must have a smartphone application to access the *System*.
- The *System* must respect the GDPR regulation.
- The *System* must ask the permission to process personal data about *Users*.
- The *System* must guarantee that the information is never altered during the whole process.

2.4.2 Dependencies

- The *System* needs a DBMS service to retrieve and store data.
- The *System* makes use of the GPS provided by the smartphone.
- The *System* uses a map visualisation service.
- The *System* makes use of internet connection provided by the smartphone.

2.4.3 Domain Assumptions

- [**D1**] Each user, simple one or authority, has only one account.
- [**D2**] Data provided during the registration are correct and belong to the person who created the account.
- [**D3**] A violation should be clearly visible and identifiable from the picture.
- [**D4**] A violation is processed if everything is correct, otherwise the Municipality fixes the errors.
- [**D5**] Data from Municipality about incidents or issued tickets are correct.
- [**D6**] GPS system of the device has an accuracy of 50 meters.
- [**D7**] Permission to access GPS and device data is granted to the System.
- [**D8**] Each car has a license plate.
- [**D9**] Municipality is entitled to issue tickets even if the violation is not physically acknowledged.
- [**D10**] Suggestions from the System are not ignored from the Municipality, but they are evaluated by the Authority.
- [**D11**] User takes responsibility for the reports of violation he/she submits.
- [**D12**] Violation data have a relative error of 5%.

3 Specific Requirements

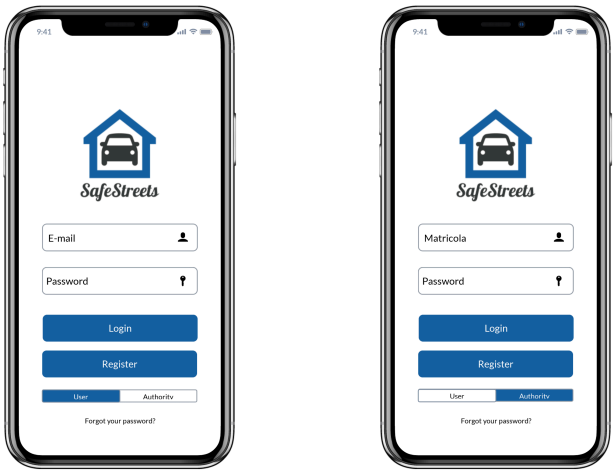
3.1 External Interface Requirements

3.1.1 User Interfaces

- **Logo**
  The logo of *SafeStreets*, that's also the logo icon for the smartphone application, captures the value of the service provided by the application. The logo is a safe home for a car, like the idea behind the development of the application: make streets safe thanks to the notification of car violation by citizens and city's tourists.

Figure 4: *SafeStreets* Logo

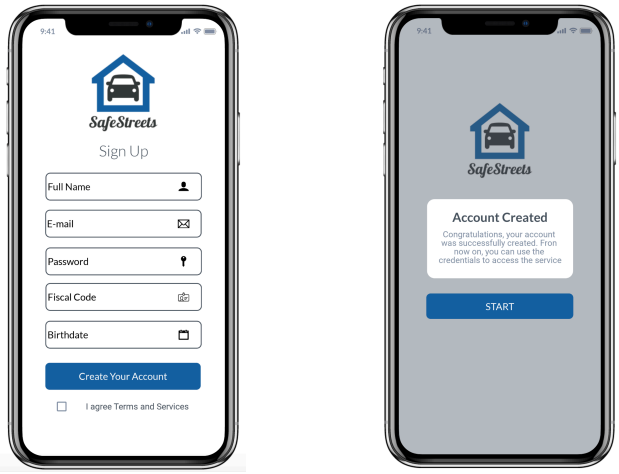
- **Login/Sign-up**
  A *Guest* downloads the applications and opens it for the first time. The *System* shows an interface that asks to choose between log-in and sign-up. If the *Guest* has not an account will click register to create an account in order to obtain access to all the functionalities, while a *User* of the application will click login button after that he/she filled the form with his/her credentials.



(a) *User* login (b) *Authority* login

Figure 5: Login interfaces

- **Account Creation**
  A *Guest* downloads the applications and opens it. The *Guest* has not an account yet, so clicks the register button. Then the *System* shows a form to fill with username, e-mail, password and fiscal code needed to identify the *User* as unique.

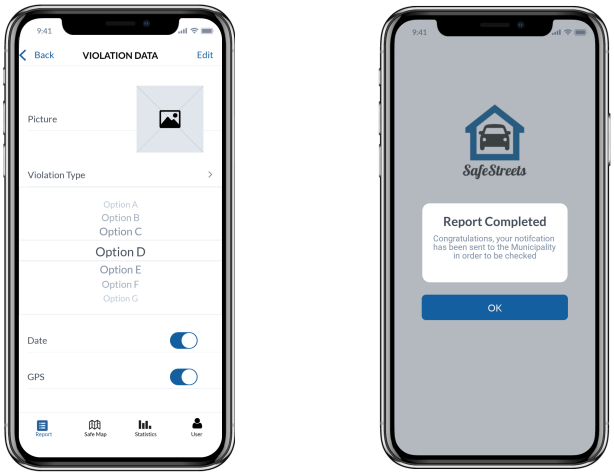


(a) Registration form (b) Registration confirmation

Figure 6: Account creation interfaces

- **User Interface User**
  After the Log-in, the *User* enters in the application, where the *System* shows an interface with four tabs, each one representing a function of the application: report a violation, see the safety map, see statistics about violation and user setting. Below we will describe each section in details.

  – **Report**
    The Report tab is the primary tab for the *User*. Here the *User* can report a violation that noticed on the streets under the *Municipality's* jurisdiction. He can load a picture of the violation, that will be processed by the plate recognition algorithm to be sure that the vehicle is identified and associated to the offender. From the drop down menu the *User* can enter the type of the violation. The GPS and Date switch should be allowed to retrieve the metadata and the position. The second interface is an alert that advise the *User* the report is successfully completed.



(a) Report violation (b) Reported completed

Figure 7: Report Violation *User* Interface

– **Safe Map**
The Safe Map tab is the second tab of the application. The *User* can see the map of the city where is located. The streets are highlighted with different colours (red, yellow, green). Each one indicates the violation level of the highlighted street, as stated by the legend. So, the tab give the *User* the possibility to access information about streets' safety. Clicking the blue points distributed on the map, the *User* can see the reported violation with its data.

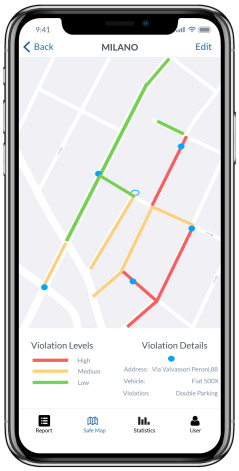


Figure 8: Safe Map *User* interface

– **Statistics**
The Statistics tab is the third tab of the application. The *User* can see the statistics obtained by the notifications' data. The interface shows some examples, such as a violation risk index of the city, the most frequent violations, the average per week day and so on and so forth. These are some of the statistics that can be generated. Others can be created and added to the same interface.

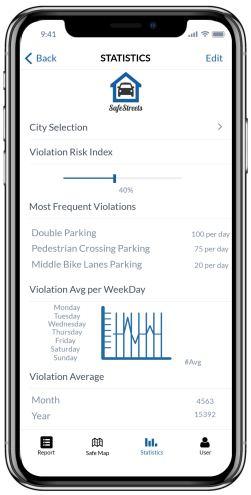


Figure 9: Statistics about violation *User* interface

– **User Settings**
The User settings tab is the fourth tab of the application. It allows the *User* to manage personal data associated with the account. He may modify the password, name and birthdate but e-mail and fiscal code can't be changed because they are the identifiers for the *User*. The interface shows all the information in a list view that can be changed anytime by the *User*. If there's a change the *System* notifies the *User* to check the change.

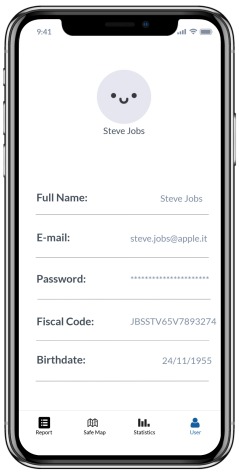
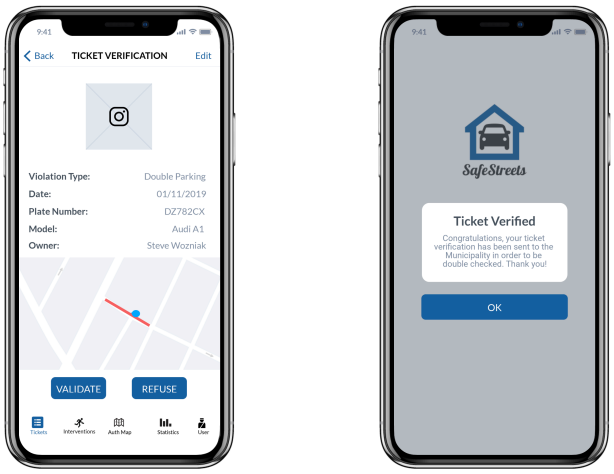


Figure 10: Settings *User* interface

- **User Interface Authority**
  - **Ticket**
    The Ticket tab is the first tab for the *Authority*. The *Authority* can see a violation that was notified by a *User* on the streets. *Authority* can verify the picture of the violation and see all the data regarding the violation type, vehicle and owner data with a map highlighting the street where the violation happened. The *Authority* can decide to validate the ticket or to refuse it if it's not a traffic law violation. The second interface shows that the ticket verification has been sent.



(a) Ticket verification (b) Ticket verification completed

Figure 11: Ticket Verification *Authority* Interface

– **Interventions**
The Interventions is the second tab of the application. The *Authority* can access all the interventions suggested by *SafeStreets* to improve the streets' security. The *Authority* can select the city, see the interventions as a list with the street and a brief motivation of the suggestion. The switch button let the *Authority* advise the *Municipality* if the *System* suggestion is feasible or not.

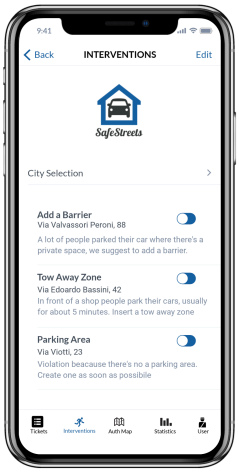


Figure 12: Interventions *Authority* interface

– **Authority Map**
The *Authority* Map tab is the third tab of the application. The *Authority* can see the map of the city where is located. The streets are highlighted with different colours (red, yellow, green). Each one indicates the violation level of the highlighted street, as stated by the legend. So, the tab give the *Authority* the possibility to access information about streets' safety. Clicking the blue points distributed on the map, the *Authority* can see the reported violation with its data and the plate number.

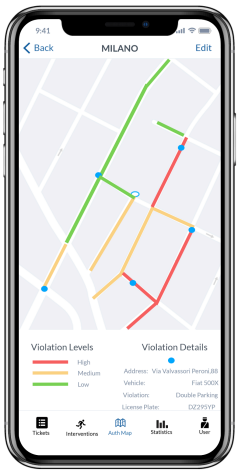


Figure 13: Map *Authority* interface

– **Statistics**
The Statistics tab is the fourth tab of the application. The *Authority* can see the statistics obtained by the notifications' data. The interface shows some examples, such as a violation risk index of the city, the most frequent violations, the average per week day and so on and so forth. The *Authority* can see also the statistics derived from the tickets' data. These are just example of possible insights from data. Others can be generated and inserted in this tab.

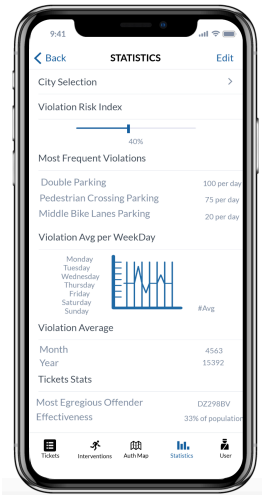


Figure 14: Statistics *Authority* interface

– **Authority Settings**
The *Authority* settings tab is the fifth tab of the application. It allows the *Authority* to manage personal data associated with the account. He may modify the password but e-mail and matricola code can't be changed because they are the identifiers given by the *Municipality*. The interface shows all the information in a list view.

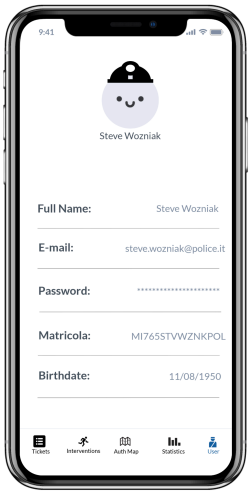


Figure 15: Settings *Authority* interface

3.1.2 Hardware Interfaces

The *SafeStreets' System* is a software application that does not offer any Hardware Interfaces, the User needs a smartphone to access the service.

3.1.3 Software Interfaces

- **Operating System**
  - Android
  - iOS
- **Web Server**
- **DBMS**
- **API**: the *System* will use an API based communication system with external services to serve both User and Authority. Data will be exchanged in multi-platform data formats such as JSON or XML.

3.1.4 Communication Interfaces

- **HTTPS Protocol:** the protocols permits a safe communication over the Internet between the application and the Web Server and/or the DBMS.

3.2 Functional Requirements

3.2.1 Use Case Diagrams

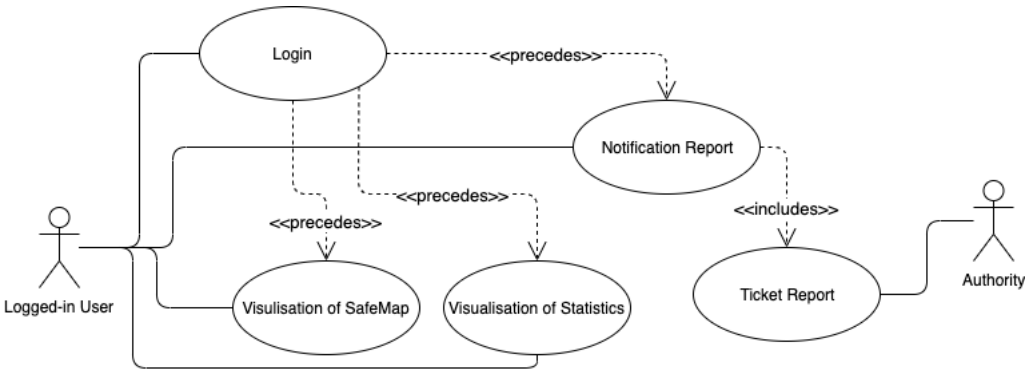


Figure 16: *User* Use Case Diagram

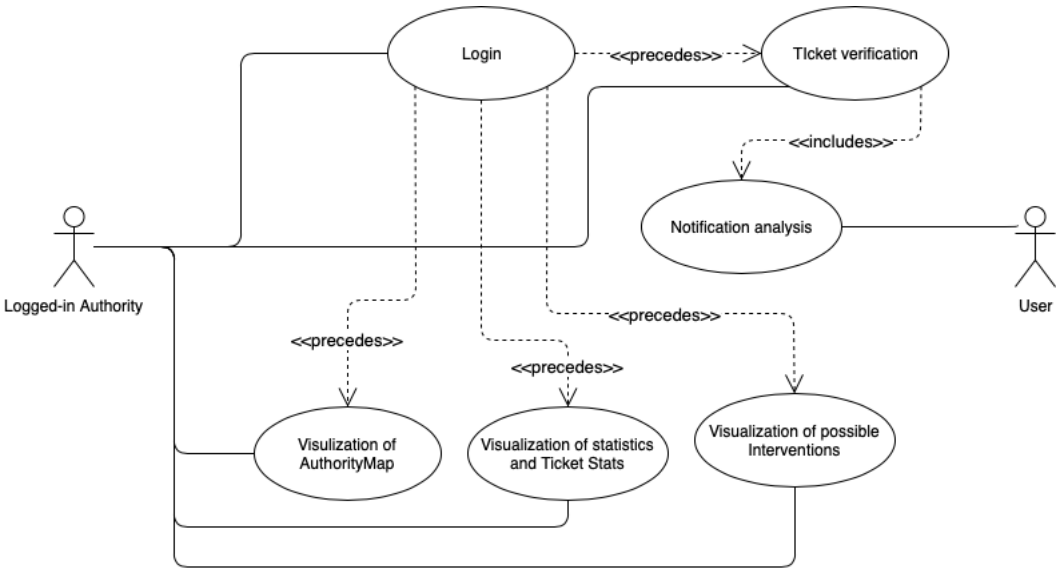


Figure 17: *Authority* Use Case Diagram

3.2.2 Scenarios

Scenario 1
Marco lives next to San Siro stadium in Milan; he has been living here for 10 years now and he is used to the fact that if he needs the car in proximity and during the one or two matches that every week for most of the year are played at San Siro, he has to move it out the garage few hours in advance, park it somewhere else and most of the time even pay for the parking spot. Thats because when Inter or Milan play a game at the stadium people dont mind using the passage at the exit of Marcos building garages (same with every other building in the area) as it was a regular parking spot. Its become an usual behavior because during the matches every resource of the municipality is busy providing security and traffic control right outside the stadium and along the streets that leads outside the area. Its then very good news for Marco the announcement made by the Municipality of Milan regarding the new partnership with SafeStreets: by reporting the parking violations to the system he can make sure the offenders get sanctioned even if the authorities are busy in the exact moment of the violation, so that people will understand, ticket after ticket, to leave the car somewhere else or to go to the matches using public transportation.

Scenario 2
The Municipality of Pavia has to deal with an annually increasing number of car accidents causing injuries in the city center. The city council has tried to limit the traffic by setting up a limited traffic area in which most of the vehicles cannot enter in certain time slots, and that is pretty effective in reducing congestion and improving safety. However the accidents problem is just slightly affected by the measure, especially when traffic limitations are not in force. The municipality then partners with SafeStreets and start receiving violation notification from the System, which collects them from the citizens (Users). Furthermore the Municipality makes data of all past accidents available to the System and shares details of new accidents as well. In a very few months SafeStreets is able to identify a pattern, which is that half of all accidents occurring are located in just two spots, both of them intersections, not controlled by traffic lights, between two narrow streets; these are also locations of a lot of wild parking notifications, especially sidewalks parking and pedestrian crossing parking. Moreover, most of this accidents are collision between two cars and between one car and a bike in the middle of the intersections. SafeStreets informs the Municipality of the observation and suggest a low-effort and cheap solution: placing, in proximity of the intersections, dividing posts on the edge of the sidewalk, one every one or two meters, to make it impossible to leave the car without blocking the narrow street viability. That will certainly make it a lot easier to see vehicles coming from the other way.

Scenario 3
The Municipality of Torino has to manage one of the most chaotic urban area of the country and struggles quite a bit in keeping order in the citys viability. Knowing about the success of the SafeStreets initiative in other cities, the Municipality decides to stipulate a year-long trial partnership. During the first four months the notification statistics show an exponential increase of the violation reports from the Users. Statistics from issued tickets follow this trend with few weeks of delay. The system is then clearly crucial in allowing the Authorities to sanction more offenders and to be more effective in enforcing the rules. The trends continue during the rest of the year, convincing the Municipality of the effectiveness of the System, but it is during the last two months that a very interesting statistics breaks out: despite the number of active Users keeps increasing, the number of violation notification slightly decreases week by week. This is observed mostly in the city center, where the concentration of notifications is higher, and is interpreted as a general decrease in the violation occurring. The higher probability of getting sanctioned keeps more and more potential offenders from breaking the rules, as SafeStreets appeals to the civic sense of citizens.

3.2.3 Use Cases

ID	UC1
Description	A *Guest* creates a *User* account for the application.
Actors	*Guest*
Preconditions	• The *Guest* downloaded the application on the smartphone. • The *Guest* opens the app and does not have an account yet.
Flow of events	1. The *Guest* opens the application. 2. The *System* shows the interface to login or register. 3. The *Guest* clicks the register button. 4. The *System* shows a form to fill with the user personal information: username, e-mail, password, fiscal code that will be used for future logins. 5. The *Guest* fills the form with all the information. 6. The *System* checks if all the informations are correct. 7. The *System* shows an interface to confirm the account is created. 8. The *System* sends an e-mail to the *Guest* to confirm the account. 9. The *Guest* receives the e-mail and confirms the account.
Postconditions	• The *Guest* is now able to sign-in as a *User*. • The *System* has stored the information of a new *User*.
Exceptions	• The *Guest* enters an e-mail that's associated with an existing account. • The *Guest* enters a fiscal code that's associated with an existing account. The *System* shows an error message and the the flow restarts from point 4.

Table 2: Create a *User* account Use Case

ID	UC2
Description	A *User* logs-in to the application.
Actors	*User*
Preconditions	• The *User* downloaded the application on the smartphone. • The *User* has already created an account.
Flow of events	1. The *User* opens the application. 2. The *System* shows the interface to login or register. 3. The *User* clicks the user tab. 4. The *System* shows a form to fill with the credentials. 5. The *User* fills the form with e-mail and password. 6. The *User* clicks the login button. 7. The *System* checks if the credentials are correct.
Postconditions	• The *User* is logged-in. • The *System* shows the *User* interface.
Exceptions	• The *User* selects the *Authority* tab entering the wrong credentials. • The *User* enters the wrong credentials. The *System* shows an error message and the flow restarts from point 2.

Table 3: *User* logs-in Use Case

ID	UC3
Description	An *Authority* logs-in to the application.
Actors	*Authority*
Preconditions	• The *Authority* downloaded the application on the smartphone. • The *Authority* has already been provided of an account by the *Municipality*.
Flow of events	1. The *Authority* opens the application. 2. The *System* shows the interface to login or register. 3. The *Authority* clicks the authority tab. 4. The *System* shows a form to fill with the credentials. 5. The *Authority* fills the form with matricola and password. 6. The *Authority* clicks the login button. 7. The *System* checks if the credentials are correct.
Postconditions	• The *Authority* is logged-in. • The *System* shows the *Authority* interface.
Exceptions	• The *Authority* selects the *User* tab entering the wrong credentials. • The *Authority* enters the wrong credentials. The *System* shows an error message and the flow restarts from point 2.

Table 4: *Authority* logs-in Use Case

ID	UC4
Description	A *Logged-in User* reports a violation
Actors	*Logged-in User*
Preconditions	• The *User* has logged in with his credentials. • The location of the violation (therefore the *User's*) is within the *Municipality's* jurisdiction
Flow of events	1. The *User* notices a traffic laws violation 2. The *User* decides to do something about it and accesses the *System*. 3. The *User*, in the Report a violation tab, takes a picture of the violation and submits it to the *System*. 4. The *System* runs the image recognition algorithm on the picture to obtain the plate number. 5. The *User* selects the type of violation from the drop down menu. 6. The *User* submits the notification to the *System*. 7. The *System* retrieves date, time and position from the mobile device.
Postconditions	• The *System* forwards the notification to the *Municipality*
Exceptions	• The plate of the of the vehicle involved in the violation is not recognised by the *System* specific tool. The *System* sends an error message to the *User*, that need to retake the photo. The flow of the events restarts from point 3.

Table 5: *Logged-in User* report violation Use Case

ID	UC5
Description	A *Logged-in User* accesses to the Safe Map.
Actors	*Logged-in User*
Preconditions	• The *User* downloaded the application on the smartphone. • The *User* has logged in with his credentials.
Flow of events	1. The *User* selects the Safe Map tab in the application. 2. The *System* shows the Safe Map interface. 3. The *User* clicks on a SafeMap's point to see more details. 4. The *System* shows the violations of the selected zone.
Postconditions	• The *User* can see all violations in the selected point of the SafeMap.
Exceptions	• The *User* selects a point that doesn't contain violations. The *System* show a message explaining that there are no violation in that point. The flow of events restarts from point 2.

Table 6: *Logged-in User* accesses Safe Map Use Case

ID	UC6
Description	A *Logged-in User* accesses to the statistics.
Actors	*Logged-in User*
Preconditions	• The *User* downloaded the application on the smartphone. • The *User* has logged in with his credentials.
Flow of events	1. The *User* selects the Statistics tab in the application. 2. The *System* shows the Statistics interface. 3. The *User* clicks on a specific Statistic to see more details. 4. The *System* shows the user all the insights about the selected statistic.
Postconditions	• The *User* can see all the details about a certain statistic.
Exceptions	• There's no data in the database that let to create Statistics. The *System* shows a message explaining there are no sufficient data at the moment to provide Statistics.

Table 7: *Logged-in User* accesses Statistics Use Case

ID	UC7
Description	A *Logged-in Authority* accesses to the Authority Map.
Actors	*Logged-in Authority*
Preconditions	• The *Authority* downloaded the application on the smartphone. • The *Authority* has logged in with his credentials.
Flow of events	1. The *Authority* selects the Authority Map tab in the application. 2. The *System* shows the Authority Map interface. 3. The *Authority* clicks on a AuthorityMap's point to see more details. 4. The *System* shows the violations of the selected zone with detailed data about vehicles and offenders that are not visible to the *User*.
Postconditions	• The *Authority* can see all violations in the selected point of the Authority Map.
Exceptions	• The *Authority* selects a point that doesn't contain violations. The *System* show a message explaining that there are no violation in that point. The flow of events restarts from point 2.

Table 8: *Logged-in Authority* accesses Authority Map Use Case

ID	UC8
Description	A *Logged-in Authority* validates the notification to issue a ticket
Actors	*Logged-in Authority*
Preconditions	• The *Authority* downloaded the application on the smartphone. • The *Authority* has logged in with his credentials.
Flow of events	1. The *Authority* accesses the tickets tab on the *System*. 2. The *System* shows to the *Authority* a list of notifications reported by *Users*. 3. The *Authority* takes under examination a notification. 4. The *Authority* recognises the violation in the picture. 5. The *Authority* confirms the information provided by the *User*. 6. The *Authority* establishes the case matches the criteria needed to guarantee an effective violation and validate it.
Postconditions	• The *Municipality* transmits data about the ticket to the *System*. • The *Municipality* generates a real ticket to the vehicle reported in the violation notification.
Exceptions	• The picture does not show the type of violation selected by the *User*, but another kind (the evaluation of the violation to generate a ticket still takes place) • The picture does not actually show a violation, or the violation is not clear enough in the picture to issue a ticket (in this case no ticket is generated)

Table 9: *Logged-in Authority* ticket generation Use Case

ID	UC9
Description	A *Logged-in Authority* accesses to Statistics, including Ticket Statistics.
Actors	*Logged-in Authority*
Preconditions	• The *Authority* downloaded the application on the smartphone. • The *Authority* has logged in with his credentials.
Flow of events	1. The *Authority* selects the Statistics tab in the application. 2. The *System* shows the Statistics and Ticket Statistics interface. 3. The *Authority* clicks on a specific Statistic to see more details. 4. The *System* shows all the insights about the selected Statistics.
Postconditions	• The *Authority* can see all details about a certain Statistic or ticket statistic.
Exceptions	• There's no data in the database that let to create Statistics. The *System* shows a message explaining there are no sufficient data at the moment to provide Statistics.

Table 10: *Logged-in Authority* accesses Statistics Use Case

ID	UC10
Description	A *Logged-in Authority* accesses suggestions on possible interventions
Actors	*Logged-in Authority*
Preconditions	• The *Authority* downloaded the application on the smartphone. • The *Authority* has logged in with his credentials. • The *Municipality* sent data about accidents to the *System*.
Flow of events	1. The *System* shows the *Authority* interface. 2. The *Authority* selects the Interventions tab. 3. The *System* shows the Interventions list. 4. The *Authority* selects an intervention. 5. The *System* shows all the details regarding the intervention. 6. The *Authority* evaluates the utility and feasibility of the suggestion. 7. The *Authority* check an intervention if it's considered feasible.
Postconditions	• The *Municipality* evaluates the check of the *Authority* and can decide to realise one of the suggestions.
Exceptions	• The *System* was not able to elaborate any specific suggestion. The *System* shows a message explaining there are no suggestions for interventions and the flow of events restarts from point 2.

Table 11: *Authority* accesses to Interventions Use Case

3.2.4 Sequence Diagrams

The first sequence diagram models the interaction for the creation of a new account. The actors are a *Guest* and the *System*.

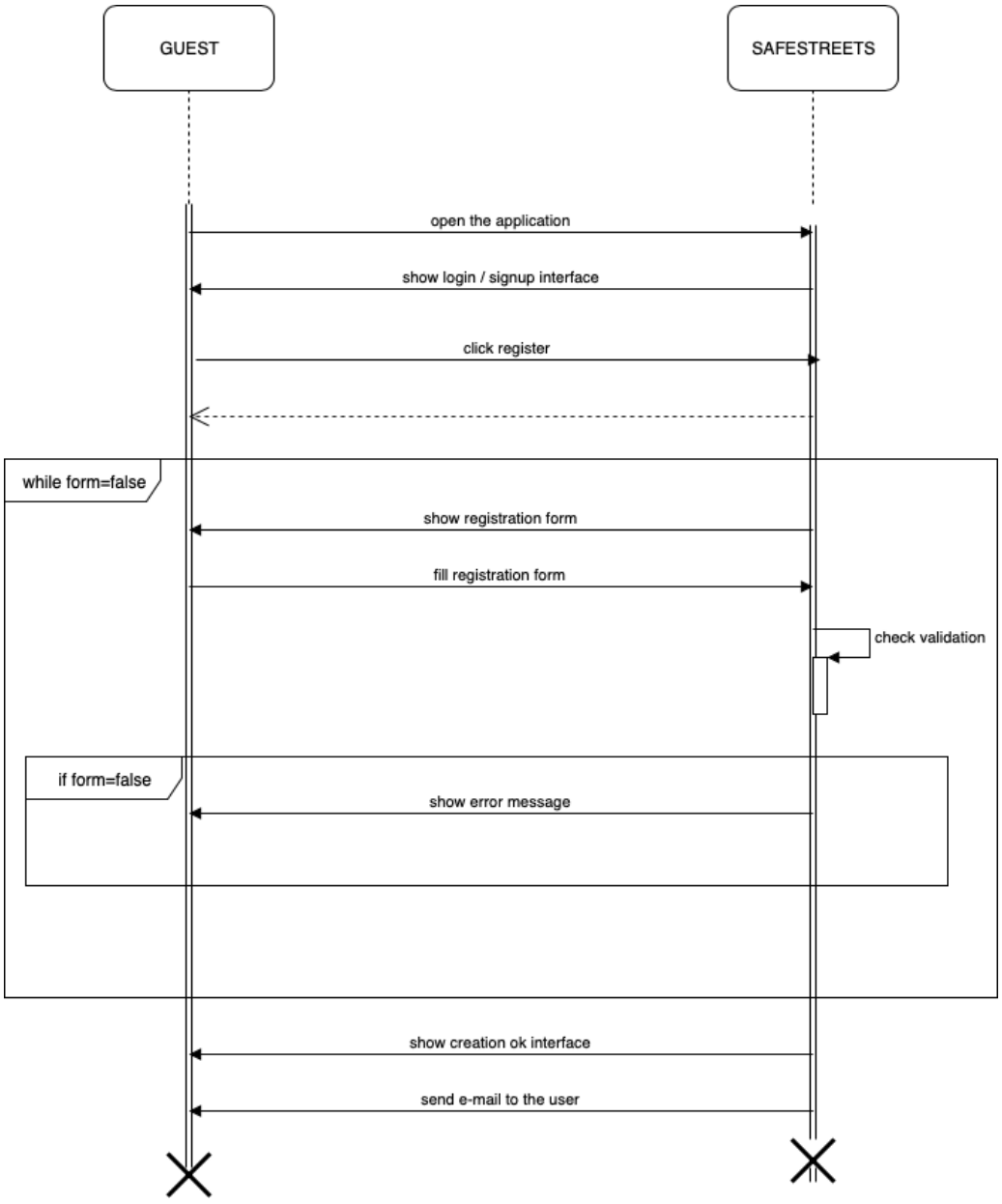


Figure 18: Account creation sequence diagram

The second sequence diagram models the interaction for the log-in to the application. We created a single sequence diagram for both *User* and *Authority* because the procedure is almost the same.

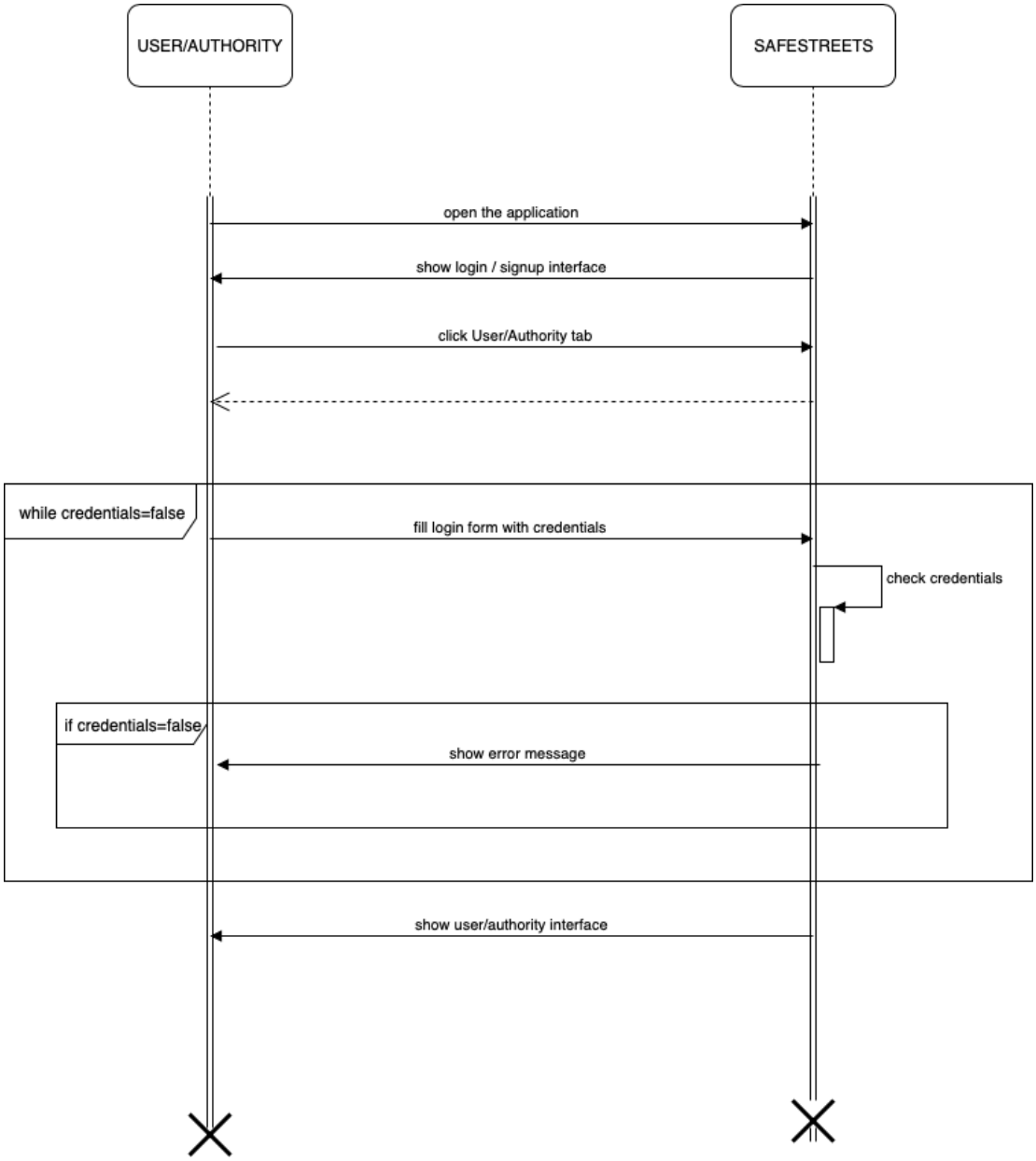


Figure 19: Application log-in sequence diagram

The third sequence diagram models the interaction for reporting a violation. The actors are a *User* and the *System*.

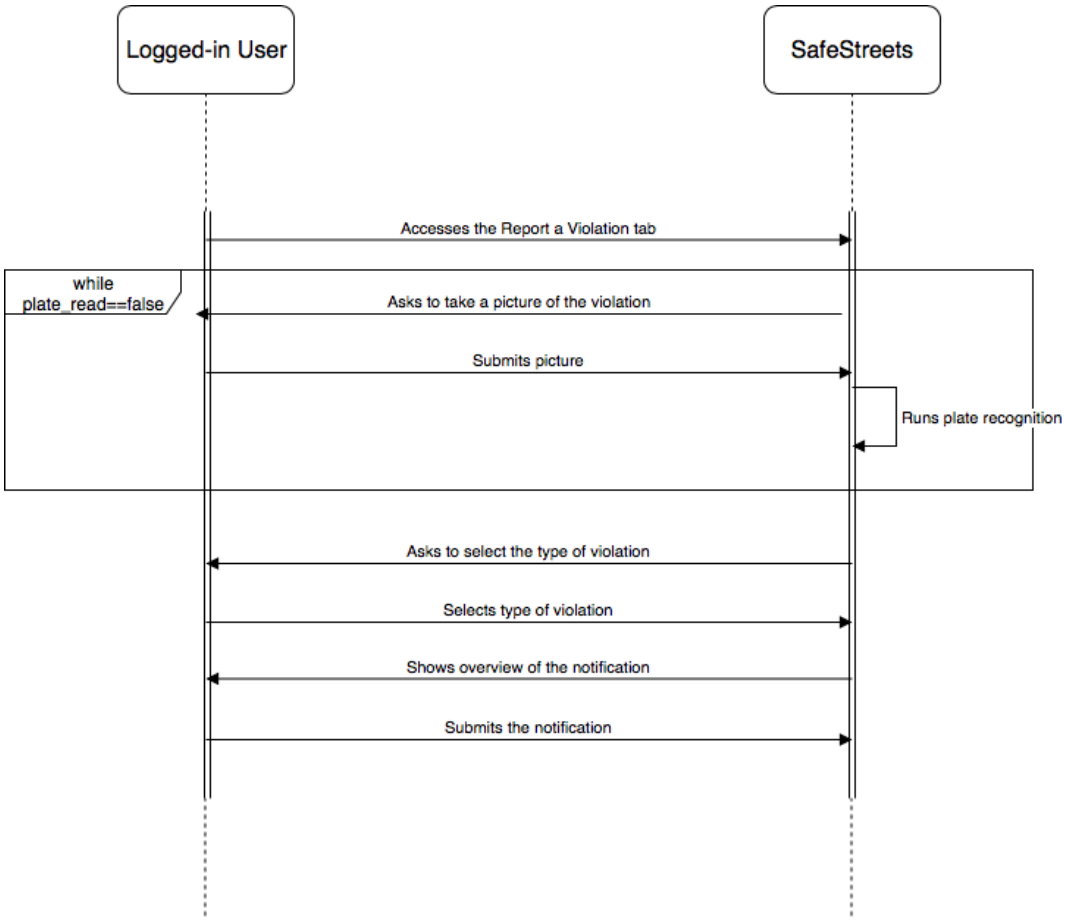


Figure 20: Report violation sequence diagram

The fourth sequence diagram models the access and interaction between the application *User* and the Map function. We created a single sequence diagram for *User* and *Authority* because the procedure is almost the same for both Safe Map and Authority Map.

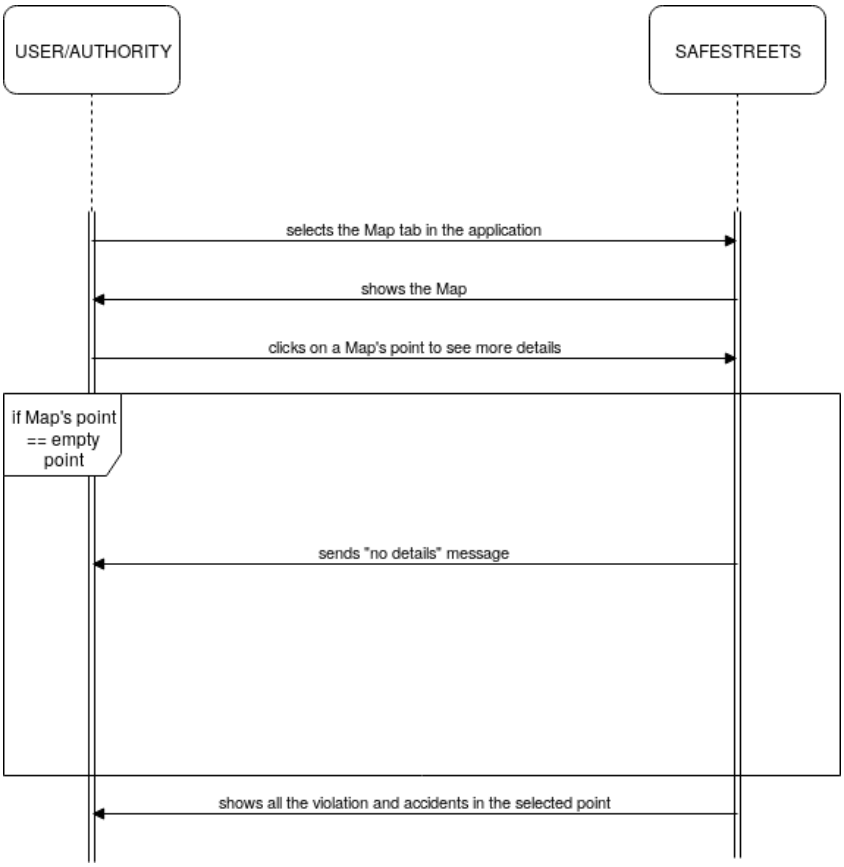


Figure 21: Map access sequence diagram

The fifth sequence diagram models the access and interaction between the application *User* and the Statistics function. We created a single sequence diagram for *User* and *Authority* because the procedure is almost the same for both. The difference between the two is the level of visibility.

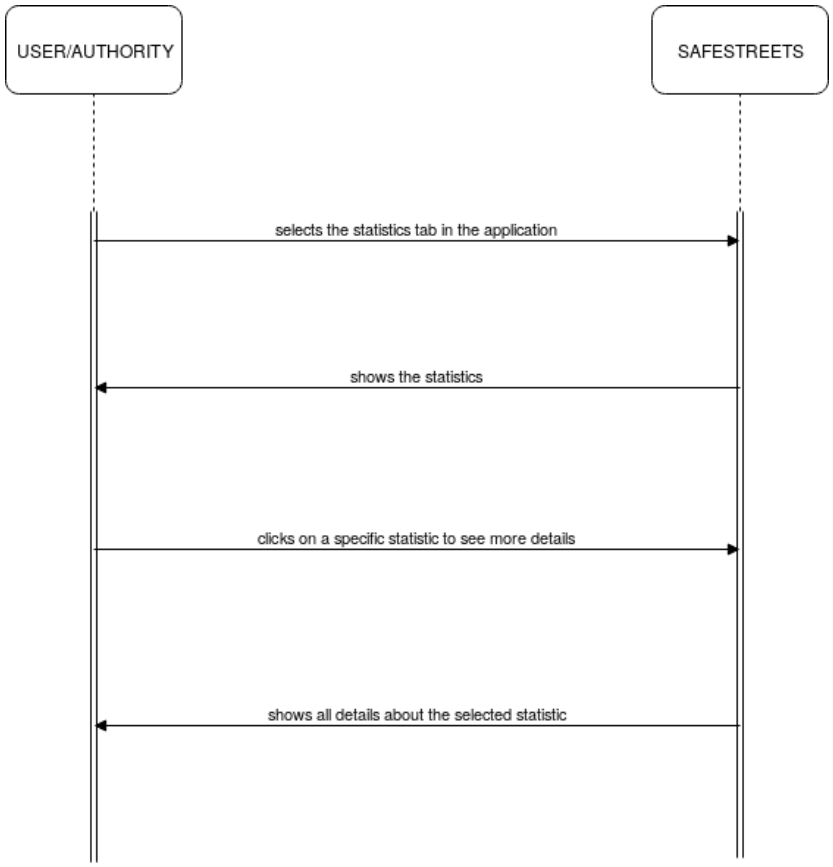


Figure 22: Statistic access sequence diagram

The sixth sequence diagram models process of ticket generation using data from notifications. The actors are the *Authority* and the *System*.

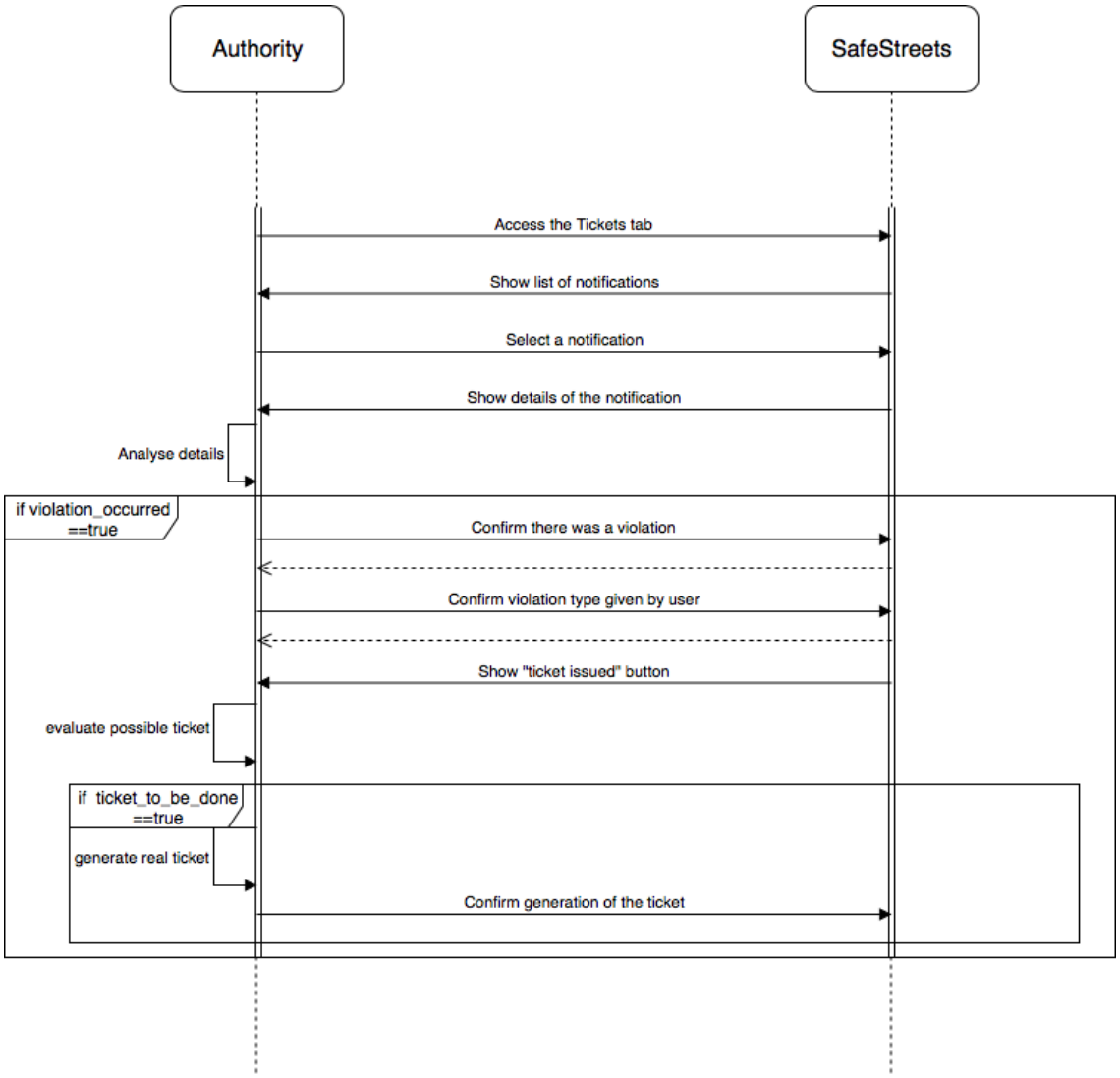


Figure 23: Ticket generation sequence diagram

The seventh sequence diagram models the interaction occurring when the *System* provides suggestions on possible intervention to the *Authority*.

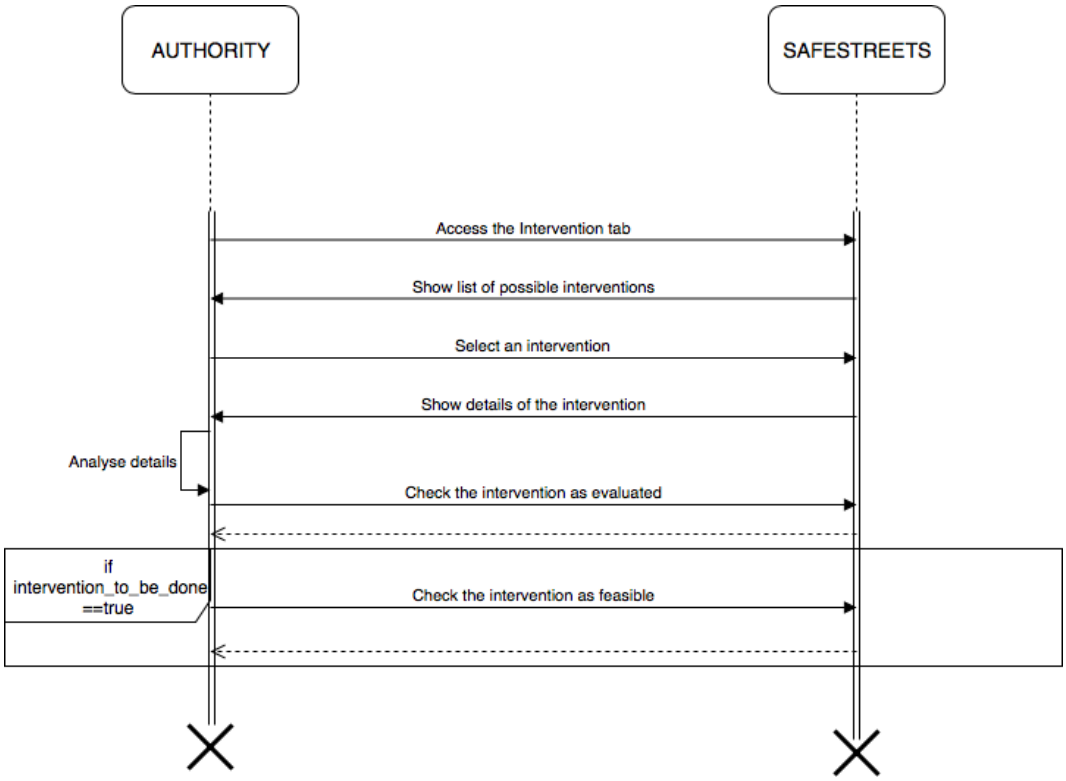


Figure 24: Intervention suggestions sequence diagram

3.2.5 Mapping on Requirements

- **[G1]** Allowing application users to be identified as simple *User* or *Authority*.
  - **[R1]** The *System* allows account to be created as a simple *User* or *Authority*.
  - **[R2]** A *User* account can be created if the *User* provides the correct data: unique email and fiscal code.
  - **[R3]** An *Authority* account exists if Municipality granted *Authority* with credentials.
  - **[R4]** *Users* and *Authority* can access the service if they log-in with their credentials.
  - **[R30]** A *User* can modify his credential and/or his/her personal information, expect for the fiscal code.
  - **[R5]** The *System* must be able to check if the credentials are valid.
  - **[R6]** The *System* must store all *User* credentials.
  - **[D1]** Each user, simple one or authority, has only one account.
  - **[D2]** Data provided during the registration are correct and belong to the person who created the account.
- **[G2]** Allowing *Users* to report a traffic violation.
  - **[R7]** The *System* must be able to analyse the picture and recognize the plate number.
  - **[R8]** The *System* gives a feedback to the *User* if the report process is done correctly.
  - **[R9]** Only *User* with an account can create and send a report.
  - **[D3]** A violation should be clearly visible and identifiable from the picture.
  - **[D4]** A violation is processed if everything is correct, otherwise the Municipality fixes the errors.
  - **[D8]** Each car has a license plate.
  - **[D11]** A *User* takes responsibility for the reports of violation he/she submits.
- **[G3]** Allowing *Users* to enter data/information about the violation.
  - **[R7]** The *System* must be able to analyse the picture and recognize the plate number.
  - **[R10]** The *System* must allow the *User* to take a picture of the violation.

  - [**R11**] The *System* can accept or refuse the image loaded from the *User*.
  - [**R12**] The *System* must allow the *User* to select the violation type.
  - [**R13**] Metadata, such as date and time, must be automatically retrieved from the device.
  - [**R14**] The *System* must allow the *User* to edit information before sending the report.
  - [**R15**] The *System* must be able to store all the notifications' data.
  - [**D3**] A violation should be clearly visible and identifiable from the picture.
  - [**D4**] A violation is processed if everything is correct, otherwise the Municipality fixes the errors.
  - [**D6**] GPS system of the device has an accuracy of 50 meters.
  - [**D7**] Permission to access GPS and device data is granted to the System.
  - [**D8**] Each car has a license plate.
  - [**D11**] A *User* takes responsibility for the reports of violation he/she submits.
- [**G4**] Providing both *Users* and *Authorities* with statistics built from notifications data.
  - [**R15**] The *System* must be able to store all the notifications' data.
  - [**R16**] All violation data used for statistics are validated by the *Municipality*.
  - [**R18**] The *System* must allow the *User* to visualise statistics derived from the data.
  - [**R19**] The *System* must allow *Authority* to see sensible information that *User* can't access.
  - [**D7**] Permission to access GPS and device data is granted to the System.
  - [**D12**] Violation data have a relative error of 5%.
- [**G5**] Identifying potentially unsafe areas and making suggestions to address those issues.
  - [**R15**] The *System* must be able to store all the notifications' data.
  - [**R19**] The *System* must allow *Authority* to see sensible information that *User* can't access.
  - [**R20**] The *System* must be able to identify viability issues based on data.

  - **[R21]** The *System* must suggest solution to address viability issues.
  - **[R22]** The *System* must allow the *Authority* to access the list of possible interventions.
  - **[R17]** The *System* must allow the *Authority* to establish if the intervention is feasible or not.
  - **[D5]** Data from Municipality about incidents or issued tickets are correct.
  - **[D10]** Suggestions from the System are not ignored from the Municipality, but they are evaluated by the Authority.
  - **[D12]** Violation data have a relative error of 5%.

- **[G6]** Allowing *Municipality* to generate tickets based on the *Users'* notifications.
  - **[R15]** The *System* must be able to store all the notifications' data.
  - **[R19]** The *System* must allow *Authority* to see sensible information that *User* can't access.
  - **[R23]** The *System* must allow the *Authority* to access information about the violation notification.
  - **[R24]** The *System* must allow the *Authority* to validate a ticket.
  - **[R25]** The *System* must allow the *Authority* to access sensible data about the violation.
  - **[R26]** The *System* must inform the *Authority* that the violation details process is done correctly.
  - **[D5]** Data from Municipality about incidents or issued tickets are correct.
  - **[D9]** Municipality is entitled to issue tickets even if the violation is not physically acknowledged.

- **[G7]** Providing statistics built on data from issued tickets to the *Municipality.*
  - **[R15]** The *System* must be able to store all the notifications' data.
  - **[R19]** The *System* must allow *Authority* to see sensible information that *User* can't access.
  - **[R23]** The *System* must allow the *Authority* to access information about the violation notification.
  - **[R27]** The *System* must be able to store all the tickets' data.
  - **[R28]** All tickets data used for statistics are validated by the *Municipality*
  - **[R29]** The *System* must allow the *Authority* to visualise statistics derived from tickets' data.
  - **[D5]** Data from Municipality about incidents or issued tickets are correct.

3.3 Performance Requirements

In this section we will explore the requirements that the application should respect to work properly. The *System* is implemented as a mobile application, *SafeStreets*, with a two-tier architecture, and designed to be used by citizens (or tourists) and authority in a city where the *Municipality*, in collaboration with *SafeStreets*, activated the service.
The back-end of the application must be able to store all the information about the different kind of Users, with all their credentials, preferences, reported violations. Every User has a space of 50MB on the server to store all the data. All the images stored will be compressed in JPEG format to not exceed the space reserved for the User. All the reported violation and request for issuing tickets must be processed in maximum 3 days.
The *System* reserves a maximum space for 75% of the number of people living in a city where *SafeStreets* is activated. These requirements could be fixed, for example increasing the percentage in order to reach more people, but it depends on how the initiatives is perceived from the population.
All the data must be stored for years in order to access a lot of data from which you can derive significant statistics and insights.
The reported violation image and its analysis must be done in in less than 3 seconds, so the user could be notified of an error and do again the picture.
The front-end is responsible for the computational tasks related to the graphical-user-interface.

3.4 Design Constraints

3.4.1 Standards Compliance

The *System* will be compliant with the following standard, as listed below:

- Italian law for everything concerning issued tickets, for whom is in charge the Authority, as a representative of the Municipality.
- GPS related information in form of latitude and longitude.
- ISO/IEC 27701 for privacy control and management.
- GDPR regulation for personal data processing and use.

3.4.2 Hardware Limitations

The application to perform as intended, should run under the following hardware conditions:

- 2Gb of RAM
- 60Mb of memory mass space
- Internet connection (Wi-Fi, 3G, 4G, LTE), at least 10Mb/s
- GPS connectivity

3.5 Software System Attributes

3.5.1 Reliability

The application and the external services offered by the application must reach a Reliability of 99%. Also, the architecture should be solid enough to prevent unscheduled downtimes. Servers' maintenance, communicated to the User in advance, are the only exception admitted.

3.5.2 Availability

The Availability, due to a strong Reliability, should be high enough, with a target value equals to 99%, in order to offer a 24/7 service.

3.5.3 Security

The Security is a crucial factor for every application. Because our *System* has a lot sensitive data, Security should be prioritised. The communication protocol, as stated before, is HTTPS. This guarantees that communication with the Server and DBMS are encrypted, while ensuring integrity of the exchanged data.
All sensitive data of the registered User, such as password, are not stored in the DB without being hashed with a proper function.
The application needs to distinguish the data accessed by different kind of User because of their privileges. So, a simple User (citizen or tourist) can not access some information that are owned by the Municipality.

3.5.4 Maintainability

Following the standard software life cycle process and good software engineering practices, the application should have an hierarchical structure, avoid anti-pattern, code duplication and lack of cohesion. Use of design pattern and good abstractions is necessary to ensure modules are open to future changes and continuous refinements.

3.5.5 Portability

The *System*, implemented as a mobile application, ensures a natural Portability between the two main operating systems: iOS and Android. If the services will be popular, a portability of the app to a web application will be taken into consideration even if it implies a lower degree of usability for the final User.

4 Formal Analysis Using Alloy

```
open unit/boolean
----------Signatures----------
sig Psw {}
sig Email {}
sig FC {} //fiscal code
sig Plate {}
sig Matricola {}//identifier of the authority
sig Name {}
sig Image {}
sig Position {}
sig ViolationT {}//identifier that allows to recognise the different types of violetions
sig Date {} //datestamp format

abstract sig Utilizer {
    password: one Psw,
}

sig Authority extends Utilizer {
    matricola: one Matricola,
}

sig User extends Utilizer {
    fullname: one Name,
    email: one Email,
    fc: one FC,
}

sig Notification {
    notificationID: one Int,//unique identifier that allows to identify the notification in the DB
    image: one Image,
    position: one Position,
    date: one Date,
    violationType: one ViolationT,
    plate: one Plate,
    fcUser: one FC,
    beTicket: one Bool, // it is true if the notification is tranformed in ticket, else false.
}{ notificationID > 0}

sig Ticket {
    ticketID: one Int,//unique identifier that allows to identify the ticket in the DB
    matrAuthority: one Matricola,
    plate: one Plate,
    date: one Date,
    position: one Position,
    violationType: one ViolationT,
} { ticketID > 0 }

//contains all the tickets and notifications of the application
sig dataManager {
    tickets: set Ticket, //tickets list of the DB
    notifications: set Notification, //notifications list of the DB
}

//specity the type of configuration that we use for our application
one sig Configuration {
    data: one dataManager,
    tickets: set Ticket,
    notifications: set Notification,
}
```

```
----------Facts----------

fact existence{

    no p: Psw | no u: Utilizer | u.password = p //all passwords are linked to at least one utilizer
    no m: Matricola | no a: Authority | a.matricola = m //all matricolas are linked to at least one authority
    no n: Name | no u: User | u.fullname = n //all names are linked to at least one user
    no e: Email | no u: User | u.email = e //all emails are linked to at least one user
    no f: FC | no u: User | u.fc = f //all fc are linked to at least one user
    no i: Image | no n: Notification | n.image = i //all images are linked to at least one notification
}

fact uniqueUtilizer{

    no disj a1, a2: Authority | a1.matricola = a2.matricola //code of two different matricolas must be different
    no disj u1, u2: User | u1.fc = u2.fc //value of two different fc must be different
    no disj u1, u2: User | u1.email = u2.email //value of two different matricolas must be different
}


fact uniqueNotification {

    no disj n1, n2: Notification | n1.notificationID = n2.notificationID //value of two different notificationIDs must be different
}


fact uniqueTicket {

    no disj t1, t2: Ticket | t1.ticketID = t2.ticketID //value of two different ticketIDs must be different
}


fact notificationProperties{

    all n: Notification | some u: User | (n.fcUser = u.fc) //all notifications are linked to the fc of the user
}


fact ticketProperties{

    all t: Ticket | some a: Authority | (t.matrAuthority = a.matricola) //all tickets are linked to the matricola of the authority
}


fact configurationCostrains{ //specify the costraints of the configuration

    //tickets of one configuration have to stay in the data manager tickets and notifications too
    no c: Configuration | some d: dataManager | (c.data = d) and
        (d.tickets != c.tickets)

    //notifications of one configuration have to stay in the data manager tickets and notifications too
    no c: Configuration | some d: dataManager | (c.data = d)  and
        (d.notifications != c.notifications)

}
```

```
fact allDataInDM{

	no t: Ticket | some d: dataManager | t not in d.tickets //all tickets are contained in at least one data manager
	no n: Notification | some d: dataManager | n not in d.notifications //all notifications are contained in at least one data manager


}

fact notificationToTicket { //if a notification is tranformed in a ticket it must have the same plate, position, date and violation type

	all n: Notification | n.beTicket = True iff one t: Ticket |
		(n.plate = t.plate and n.position = t.position and n.date = t.date
		and n.violationType = t.violationType)

}


----------Predicates----------

pred show {}

//predicate used to add an user
pred userAccountCreation[u: User, n: Name, e: Email, f: FC, p: Psw] {

	u.fullname = n
	u.fc = f
	u.email = e
	u.password = p

}

//predicate used to add an authority
pred authorityAccountCreation[a: Authority, m: Matricola, p: Psw] {

	a.matricola = m
	a.password = p

}

//predicate used to add a notification in the data manager
pred addNotification[n: Notification,c: Configuration, d,d': dataManager] {

	d'.notifications = d.notifications + n
	d'.tickets = d.tickets
	c.data = d'
	c.tickets = d'.tickets
	c.notifications = d'.notifications

}
```

```
//predicate used to modify the notification to allow the correction of it, it is possible to correct only the violation type
pred correctNotification[n, n' :  Notification, d,d': dataManager,c: Configuration, v: ViolationT] {

	n'.notificationID = n.notificationID
	n'.image = n.image
	n'.position = n.position
	n'.date = n.date
	n'.fcUser = n.fcUser
	n'.beTicket = n.beTicket
	n'.plate = n.plate
	n'.violationType = v
	d'.notifications = d.notifications - n + n'
	d'.tickets = d.tickets
	c.data = d'
	c.tickets = d'.tickets
	c.notifications = d'.notifications
}

//predicate used to create a new ticket starting from an existing notifiation
pred ticketFromNotification[n: Notification, t: Ticket, d,d': dataManager,c: Configuration, a: Authority, id: Int] { //rifare

	t.ticketID = id
	t.matrAuthority = a.matricola
	t.plate = n.plate
	t.date = n.date
	t.position = n.position
	t.violationType = n.violationType
	d'.tickets = d.tickets + t
	d'.notifications = d.notifications
	c.data = d'
	c.tickets = d'.tickets
	c.notifications = d'.notifications

}

-------Run Predicates----------

run userAccountCreation for 3 Psw,5 Email,5 FC,3 Plate,5 Matricola,4 Name,5 Image, 4 Position,
	3 ViolationT,2 Date,5 Authority,5 User,5 Notification, 5 Ticket,2 dataManager,1 Configuration, 2 Int

run authorityAccountCreation for 3 Psw,5 Email,5 FC,3 Plate,5 Matricola,4 Name,5 Image, 4 Position,
	3 ViolationT,2 Date,5 Authority,5 User,5 Notification, 5 Ticket,2 dataManager,1 Configuration, 2 Int

run addNotification for 3 Psw,5 Email,5 FC,3 Plate,5 Matricola,4 Name,5 Image, 4 Position,
	3 ViolationT,2 Date,5 Authority,5 User,5 Notification, 5 Ticket,2 dataManager,1 Configuration, 2 Int

run correctNotification  for 3 Psw,5 Email,5 FC,3 Plate,5 Matricola,4 Name,5 Image, 4 Position,
	3 ViolationT,2 Date,5 Authority,5 User,5 Notification, 5 Ticket,2 dataManager,1 Configuration, 2 Int

run ticketFromNotification for 3 Psw,5 Email,5 FC,3 Plate,5 Matricola,4 Name,5 Image, 4 Position,
	3 ViolationT,2 Date,5 Authority,5 User,5 Notification, 5 Ticket,2 dataManager,1 Configuration, 2 Int
```

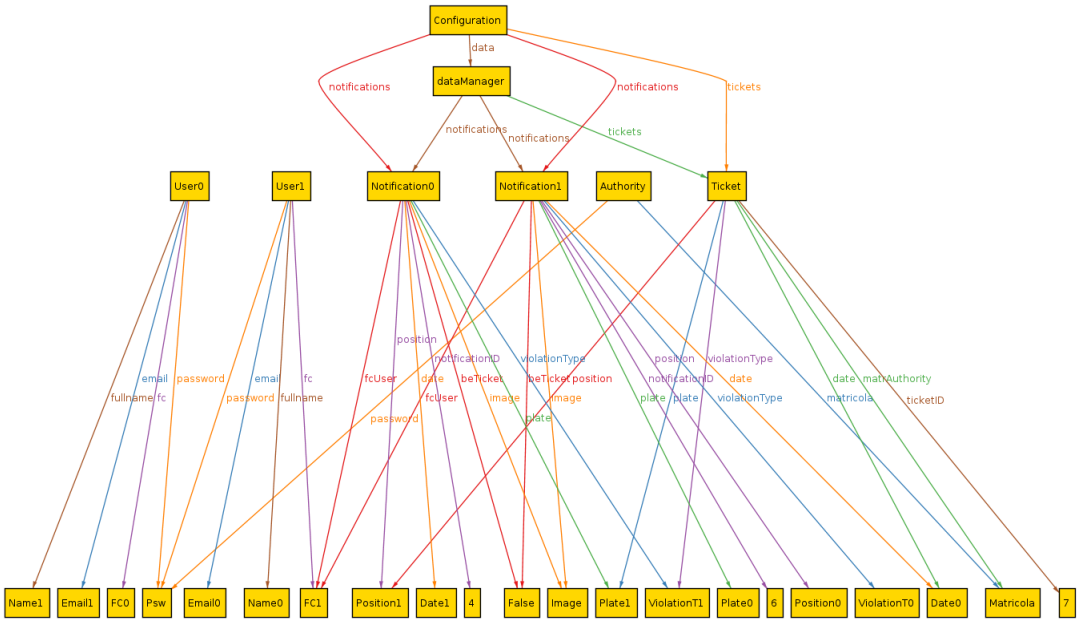


Figure 25: Alloy General Graph

5 Effort Spent

Date	Task	Hours
27/10/19	First Meeting	3
29/10/19	Github setup	2
30/10/19	Second Meeting	4
01/11/19	Introduction, Scope, Purpose	3
02/11/19	Constraints	2
03/11/19	User Characteristics	2
04/11/19	Section 3.3,3.4,3.5	3
05/11/19	External Interface	6
06/11/19	External Interface	6
06/11/19	Use Cases	2
07/11/19	Sequence Diagrams	1
08/11/19	Requirements	4
09/11/19	Meeting	2
10/11/19	Refining and Corrections	2
		Total
		42

Table 12: Adriano Mundo's effort

Date	Task	Hours
27/10/19	First Meeting	3
29/10/19	Github setup	2
30/10/19	Second Meeting	4
02/11/19	Phenomena	2
03/11/19	Product Functions and Perspective	3
04/11/19	Use Cases	3
05/11/19	Scenarios	3
05/11/19	Use Cases	2
07/11/19	Assumptions	3
08/11/19	Sequence Diagrams	2
08/11/19	Alloy Discussion	4
09/11/19	Meeting	2
09/11/19	Alloy	8
10/11/19	Refactoring e Refinements	1
		Total
		39

Table 13: Francesco Rota's effort

Date	**Task**	**Hours**
27/10/19	First Meeting	3
29/10/19	Github setup	2
30/10/19	Second Meeting	4
31/10/19	Section 1.3	2
02/11/19	Section 1.4	1
04/11/19	Class Diagrams	4
05/11/19	State Diagrams	2
05/11/19	Use Cases	3
06/11/19	Use Cases	1
08/11/19	Sequence Diagrams	2
08/11/19	Alloy	6
09/11/19	Meeting	2
09/11/19	Alloy	8
10/11/19	Corrections	1
		Total
		41

Table 14: Salvatore Fadda's effort