Requirement analysis and specification document
Safestreets

Luca Massini Daniele Nicolò

10 November 2019

Contents

1 Introduction

The purpose of this document is to represent the Requirement Analysis and Specification Document (RASD). This document shows what are the goals and the requirements of the software. It has to represent how the application can be useful for the users and why they are fundamental to improve the quality of the service offered. Secondly, this document can be also used as a support for the testing of the system, for the verification activities and also the validation ones. The RASD can be also used to guide the changes in a already existing system.

1.1 Purpose

1.1.1 Description of the given problem

SafeStreets is a mobile application useful to help people to be safer when they are on the street.
The users can send to the municipality pictures of violations occurring in public streets: the reporting can concern violation on the road, in a parking lot and so on.
The software allows the users to send detailed information about the violation, such as the hour, the date, the type of violation and the position (they can be captured with GPS).
Furthermore, the service can provide to the user information about the streets around which he/she is, such as the number of violations per street and consequently the level of danger of the street.
In addition, the user can have a different service based on the category to which he/she belongs.
The user and the municipality can also find on the application the most "dangerous" vehicles, that are the ones with the highest number of reports from the users.
The service must be different in base of the type of user, such as municipalities, motorists, motorcyclists, bikers, pedestrians, disabled people etc.// Finally, the users must be able to receive recommendations from the system to avoid using streets, parking lots that are risky in general or at a specific hour or date.
The authorities will be informed of the violations through the municipality.

1.1.2 Goals

- [*goal*1]: The service allows the users to report a violation to the municipality.
  - [*goal*1.1]: The user must send a picture of the violation.
  - [*goal*1.2]: The user can specify the date and the hour when the violation has happened and the type of violation.
  - [*goal*1.3]: The municipality must be able to receive the reports.

- [*goal*2]: The service allows the users to have detailed information about the violations and the streets safety.
  - [*goal*2.1]: The users can know which are the most reported streets, areas or parking lots.
  - [*goal*2.2]: The users can see which are the vehicles that commit the highest number of traffic/parking violations.
  - [*goal*2.3]: The user, in base of his/her position, can be recommended to use the safest streets/areas, etc or avoid the dangerous ones.

- [*goal*3]: Safestreets must send to the municipality suggestions to improve the safety of the streets only when the municipality asks for them.
  - [*goal*3.1]: Every time the municipality makes a request in order to obtain suggestions, they must receive them by Safestreets.
- [*goal*4]: The user (both private user and the municipality) must be able to authenticate into the Safestreets platform.

1.2 Scope

SafeStreets will be developed as a mobile application and will be designed only for IOS and Android. The application offers to users different service, that are the possibility to make a report of a violation and send it to the municipality. The user must take a photo of the violation and the system will retrieve the license plate of the photographed vehicle using an algorithm offered by an external API.
Another service offered by SafeStreets is the possibility to check the safety of the streets and the parking areas. The maps are provided by an external API and we will develop an algorithm to highlight them in a different way according to their dangerousness.
The system also offers the possibility to the municipality to receive suggestions in order to improve the safety of the streets. These suggestions are created using an external data mining algorithm that retrieves the necessary data both from the users' violations and the database offered by the municipality.
Furthermore, all the data about the violations, the users, the streets and the parking areas are stored in an external cloud database and in the design document we will show this in detail in a relational schema.

With this app we engage in making the streets safer within the limits that the mobile application has, that are related to the fact that SafeStreets is not designed and developed by the authorities and so the only responsibility that we take is that the reports must be received by the municipality. So, the application does not concern the actions that the municipality will do after having received the report.

In this section we explain what the system can control, what the external world can control and finally what is controllable by the world and observed by the system and vice versa.

World phenomena

This events are controlled by the external world.

From the user point of view

- The user sees one or more violations.
- The user wants to report a violation to the authorities.
- The user wants to know the area/streets where the major violations occur.
- The user wants to know where is safer to park his/ her car/bike/motorcycle.

From the municipality point of view

- The municipality wants to know what to do to improve the safety of the streets.
- The municipality wants to discover which are the most dangerous zones of the city.
- The authorities receive the information about the violations directly from the municipality.

Shared phenomena

Shared phenomena controlled by the world and observed by the machine:

- The user authenticates into the platform.
- The user sends to the system a picture of a violation and all the other data related to it.
- The user sends data requests to the system.
- The municipality sends a data request to the system in order to receive suggestions to improve the safety of the streets.

Shared phenomena controlled by the machine and observed by the world

- The user receives, by the system, all the data related to the safest and most dangerous streets.
- The municipality receives suggestions from SafeStreets to make the city safer.
- The municipality receives violations reported by the users via Safestreets.

Machine phenomena
Events controlled by the machine.

- The system stores the user's data in its database.
- The system stores the data about the municipality in its database.
- The system stores all the data about the violation in its database.
- The system analyses with an algorithm provided by an external API the received picture in order to retrieve the car plate.
- The system checks if the street in which the violation occurs is an existing one with the use of an external API.
- The system runs a data mining algorithm provided by an external API to retrieve suggestions from users' reports and municipality data,it collects elaborated data and provides it to the municipality.
- The system can refuse a request by the user because of insufficient data.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Acronyms

S2B	System to be
GDPR	General Data Protection Regulation
FC	Fiscal Code
WP	World Phenomena
MP	Machine Phenomena
SP	Shared Phenomena
GPS	Global Positioning System

1.3.2 Definitions

- **Violation** = Any kind of infringement of the driving code. Here we distinguish violations concerning the traffic and violation concerning parking.

- **Personal data** = Data belonging to the user, needed in the moment of the signing-up. They are FC, username, password and e-mail.
- **S2B** = System to be developed. It will contain all the services provided by SafeStreets.
- **Report of a violation** = A user can use SafeStreets to report an infringement he/she has seen. In particular he/she can add the type of violation, the date and the hour, the street (that can also be retrieved automatically) and he/she must add a picture of the violation.
- **Municipality request** = The municipality can ask for the same things aforementioned for the user. They will receive more accurate information rather than a private user, including for example data about the violators. They can also request suggestions (provided by SafeStreets) to improve the quality of their service.
- **Fiscal code** = It's a 16-characters code identifying an Italian citizen.
- **Credentials** = username and password used by a user to log-in to the system.

1.3.3 Abbreviations

- Gn = n-th goal;
- Dn = n-th domain assumption;
- Cn = n-th constraint;
- Rn = n-th requirement;
- MP = Machine Phenomena;
- WP = World Phenomena;
- SP0 = Shared Phenomena that are managed by the Machine but that are observable by the World;
- SP1 = Shared Phenomena that are managed by the World but observable by the Machine.

1.4 Revision History

- Work started 13/10/19;
- RASD completed on 8/11/19;
- RASD delivered on 10/11/19.

1.5 Reference Documents

- RASD Slides by professor Matteo Rossi (Polimi);
- Course slides about alloy by professor Rossi;
- $https://web.cs.dal.ca/\ hawkey/3130/srs_template-ieee.doc$.

1.6 Document Structure

This document is organized in the following way:

- **Overall Description:** This section contains a deeper analysis of the world and machine phenomena already described in the scope. It also contains class diagrams which describe the relationship between actors, state chart diagrams used to describe the state of the system from a dynamic point of view, the description of the stakeholders and their needs and finally the key functional requirements and the domain assumption.
- **Specific requirements:** In this section there is a more detailed description of the requirements already described in the previous section. This is thought to help the develop team to understand what the system must ensure.In particular in this part there is a description of the external interfaces requirements,of the functional requirements, of the performance requirements, of the design constraints and finally of the software system attributes.
- **Formal analysis:** This section contains the formal analysis written in alloy. Alloy is a formal notation used to specify model of systems and software. Thanks to alloy it is possible to write a formal models with their own requirement, domain assumptions and goals and then check the correctness of the model.
  There is another very important aspect concerning this section. Such aspect is the fact that alloy gives the possibility to clarify aspects that could be ambiguous. The ambiguity is usually and intrinsically due to natural language.

2 Overall Description

This section is necessary to give a better and deeper description of the shared and world phenomena. In this section, the system will be described also with the help of the class diagrams and state diagrams.

2.1 Product Perspective

SafeStreets is a new software created from scratch. It isn't a follow-on member of an existing product family and it isn't a component of a larger system, but it interfaces with third party components, which are the external cloud database that stores SafeStreets data, the external database provided by the municipality that contains further information about the violations and the streets, a Google cloud API (Vision AI) for the image recognition to retrieve the license plates from the report images, a Google Maps API to use and display the maps, Oracle Data Mining in order to retrieve data from the databases and to provide suggestions for the municipality.

Here there is in detail the description of the shared phenomena to understand how the system interacts with the external world. In addition, there is also an explanation of the other phenomena in order to understand better how the system and the external world act.

The user sees one or more violations: It's the situation in which the user sees a violation which can concern traffic or parking. Such violations could be for example a car/motorcycle parked in a red zone or on a sidewalk area. Another example of violation can be that the user sees a car/motorcycle parked on a pedestrian crossing. All the types of reportable violations are suggested by the municipality and each violation has an index of dangerousness that is given to SafeStreets by the municipality itself.

The user wants to report a violation to the authorities: In this case the user would like to have the possibility to report a violation of which he/she is aware.

The user wants to know the areas/streets where the major number of violations occur. In this type of phenomena the user would like to know what are the most dangerous areas in his/her city/town in terms of traffic or parking violations.

The user wants to know where is safer to park his/ her car/bike/motorcycle: In this case we analyse the case in which the user has a vehicle that needs to be parked. He/she would know what are the safest streets in order to

decide in which area to park. He/she wants to know the safest parking areas whose distance from him/her is less than or equal to a certain value (given by the user).

The municipality wants to know what to do to improve the safety of the streets: In this case the municipality recognizes that in a certain area or street there is a problem with the too high number of violations. In this case the municipality would like to have some suggestions in order to provide a solution to make the number of violations decrease.

The municipality wants to discover which are the most dangerous zones of the city. The municipality would like to know which are the most dangerous streets in order to monitor them with more attention. This could be useful for the municipality if it wants to improve its service quality.

The user signs-in to the platform: In this case the user compiles all the mandatory fields which are: name, surname, username, password, fiscal code and date of birth. Driving license data are not mandatory because this system is designed for people who use the app also to ride a bike or walk.

The user sends to the system a picture of a violation and all the other data related to it: In this case the user sends a report of a park violation or a traffic violation to the authorities (the municipality in this case). The user must take a picture of the violation and add the mandatory information which are needed which are: the hour and the date. The place of the violation can be detected automatically or manually (it's a user's choice).

The user sends data requests to the system: In this case the user could want to retrieve some information from the system. Such information, could concern the most dangerous zones or who commits the major number of violations, the most safe streets where to park and so on. The system will provide these information in different ways, such as a list of the users or a map with the streets highlighted differently in relation to their level of danger.

The municipality sends a data request to the system in order to receive suggestion to improve the safety of the streets: The municipality sends a data request to the system in order to have suggestions to improve the safety. These suggestions are simple phrases which describe what it's possible to do to overcome problems.

The municipality sends a data request to know which are the vehicles that have committed the highest number of violations: In this case the authorities would like to understand which are the ones who commit the major number of violations, so they send a data request to retrieve those information.

The municipality has to specify the street that they want to check and the maximum number of instances to obtain with the query.

The user receives from the system all the data related to the safest and most dangerous streets : In this case the user receives all the data about the streets that are considered the most dangerous or the safest. He/She receives a map with the highlighted streets which are at a certain distance from a point specified by the user.

The user receives from the system all the data related to the vehicles that have committed the major number of violations: In this case the user receives the list of license plates that have committed the highest number of violations. The user receives a number of license plates less than or equal the number he/she has specified as input.

SafeStreets provides the suggestion in order to give them to the authorities: SafeStreets runs an algorithm to retrieve information and suggestions about the future improvement of the streets safety from the users' reports. These data will be sent to the municipality so that they will be able to use them to enhance the safety.

The municipality receives suggestions to keep the streets safer: The municipality receives a list where each element is composed on one side by the problem and on the other by the suggestion to overcome it.

The municipality receives the list of the most dangerous vehicles: The municipality receives all the license plates belonging to the vehicles that have committed the highest number of violations. The authorities select the maximum number of plates to receive (That they can choose each time as an input) and they obtain by the system a less than or equal number of license plates.

The municipality receives violations reported by the user via Safestreets: In this case the Municipality will receive the violations and then they will be reported to the authorities.

The system stores the user's data: The system will store permanently and safely all the user data. They will be stored in the system databases in such a way that data corruption is rather difficult.

The system stores all the data about the municipality: The system will also store all the data about the municipality in its databases.

The system stores all the data about the violation: The system will store all the information concerning the violation. Such information are the picture, the data, the hour, the license plate of the vehicle, and an unique identifier to distinguish identical violations eventually reported by different users.

The system analyses the received picture in order to retrieve the car plate: The system, after having received a picture of the violation, will run an algorithm to recognize the license plate of the vehicle that committed the violation so that it is uniquely identified. The algorithm takes the picture sent by the user as an input and works on it to retrieve the license plate. If it is recognizable from the picture, the system saves the data about it in its database. Otherwise it only keeps the information about the violation.

The system checks if the street in which the violation occur is an existing one: if the place of the violation is manually entered by the user there is the possibility that such place doesn't exist. In this case the system must automatically report an error to the user. Otherwise the system will store the street in the database together with all the other information related to the violation.

The system runs an algorithm to retrieve suggestions from users' reports and municipality data, it collects elaborated data and provides it to the municipality: Safestreets will collect data from the users and also from the municipality. The system then will elaborates the different types of data using a data mining algorithm and will send the resulting suggestions to the municipality.

The system can refuse a request by the user because of insufficient data: In the case that the data on the databases in insufficient the system abort and will report the fail to the user.

2.1.1 Class diagrams

This is the user class diagram:

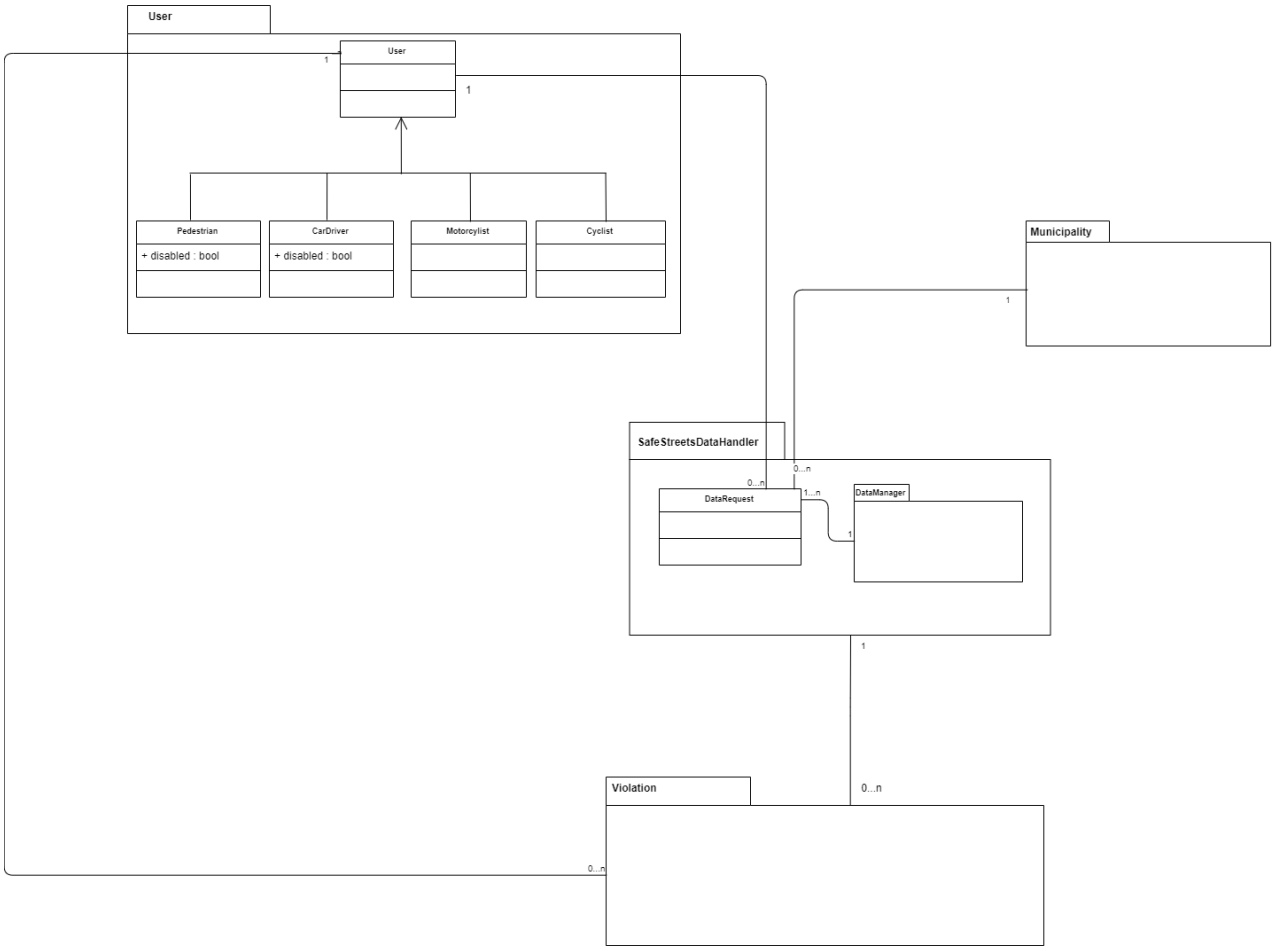


Figure 1: User class diagram

This is the violation class diagram:

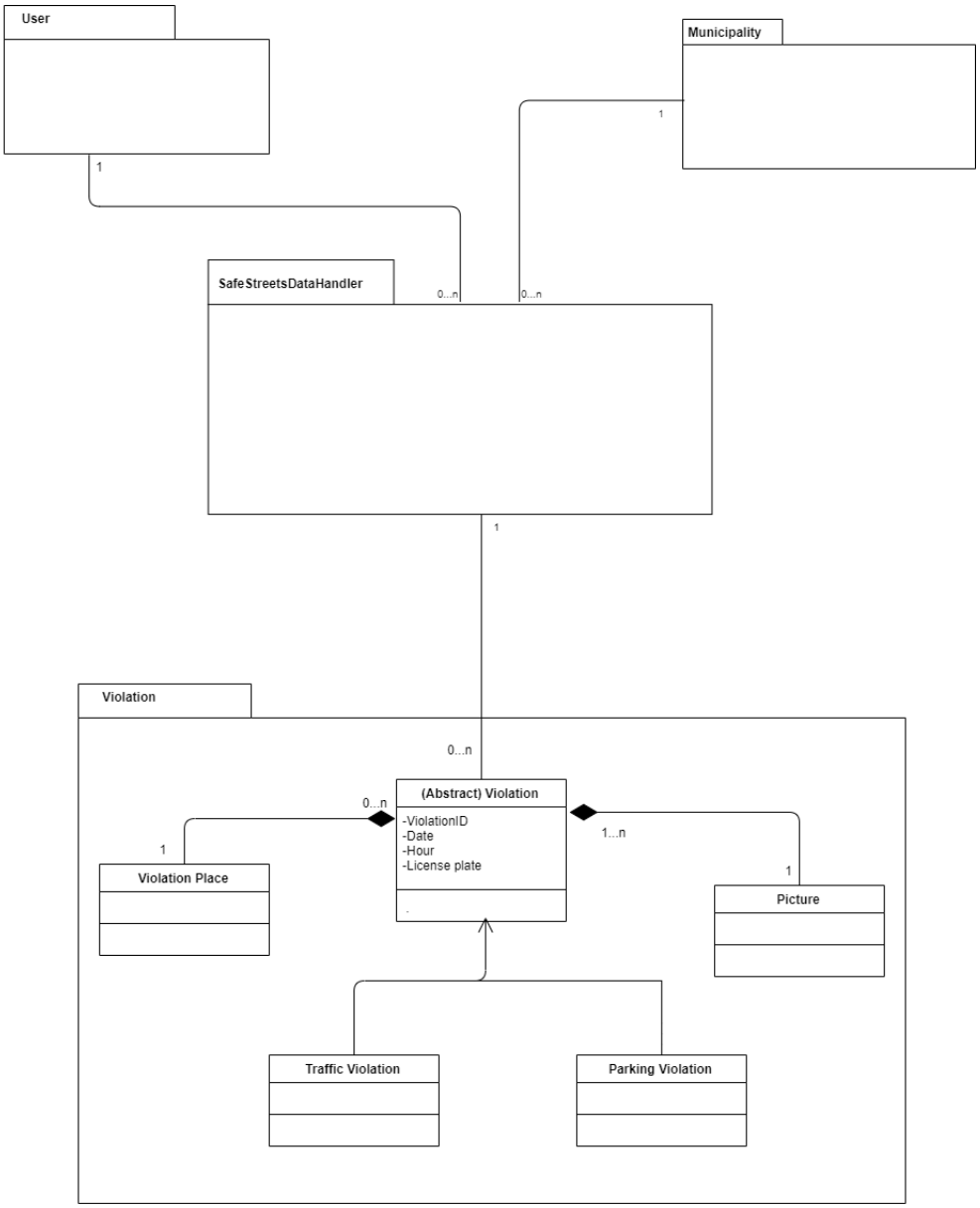


Figure 2: Violation class diagram

2.1.2 State charts

Here you can find a state chart diagram of the system:

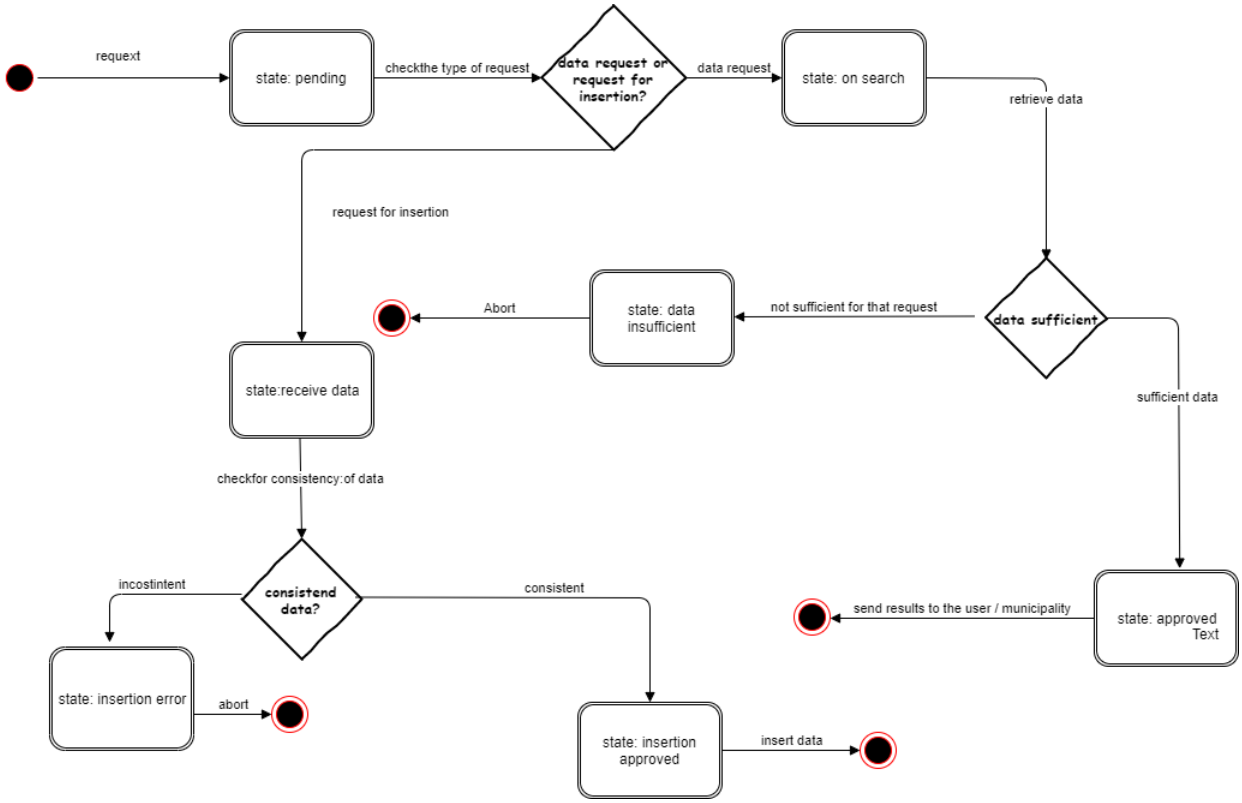


Figure 3: System State chart diagram

2.2 Product functions

In this section there is a description of the main functions that the system will make available to the final user.
These functions are the following:

- **Report of a violation:** Safestreets allows the user to report a violation.This function must permit the user to take a picture directly and only from the application. It must also allow to insert the hour and the date. The street where the violation occur can be detected automatically or manually. In the first case the user decides that he/she wants to be localized through the GPS coordinates and in the last case the user must insert the street and the city manually into the application. It's very important to remark the important fact that the way in which the user wants

to insert the street is only a choice of the private user.

- **Get the list of most dangerous streets:** The most dangerous streets are the ones with the highest violations and they can be retrieved specifying the maximum number of streets that the request has to show.
  This function must also permit the user to insert the city of which he/she wants to know the dangerous roads and also a distance (expressed in kilometres). This function will provide a map with the dangerous streets highlighted. This function must take into consideration only the streets whose distance is less or equal to the one specified in the request by the user.

- **Send a request to obtain suggestions to make the streets safer:** This function is only designed for the municipalities. In this case the municipality has to insert the place about which it wants to have suggestions. The system will return a list containing on one side the problem and on the other side a possible solution to overcome it. Obviously, the list is not static but it will be updated over time thanks to the reported violations.

- **The user send a request to know where to park his/her vehicle in a safe area:** This function allows the user to explicitly ask the system to suggest places where to park his/her vehicle in a secure way. The user must specify how far he/she wants to park from his/her position or from a position given as an input. The system will not guarantee the fact that in the proposed place there will be at least one parking lot but it only will suggest which roads are considered more secure than others to do so.

- **The user will receive the suggested areas where to park:** After having requested the safest areas where to park, the system will provide them to the user in a map form. In this map there will be highlighted the streets considered more secure than the others in the specified area given by the user as input.

2.3 User characteristics

In this subsection there is a description of the application users and an analysis of their needs. We will categorize the people who use the app in order to understand their necessities in a better way.

Description of the users of the app:
The user of the app can be divided in a first categorization which is the following:

- Private user;
- Municipality.
- Authorities.

It is very important to say that the authorities are the only ones will not directly use the Safestreets system. They will be informed of the reports of violations by the municipality. The municipality is the entity which will receive all the reports and then it will pass them to the authorities. This choice is due to the fact that the authorities are part of the municipality, which manages them and Safestreets must not change the way in which the authorities are managed and organized internally.

Description of the private users:
It is possible to divide again the private users in other classes. For every class there will be a different concept of dangerousness.

- Pedestrians;
- Motorists;
- Motorcyclists;
- Cyclists.

A private user can be also a disabled person belonging to the pedestrians category or to the car drivers one (We suppose that it's rare that a person with motor disabilities drives a bicycle or a motorbike). Because of this fact these type of users will require a different type of service and a different concept of dangerousness. There are a lot of things that are not dangerous for non-disabled people and on the contrary others which are. We can make some brief examples:

1) Streets which are not equipped with facilities for disabled pedestrians are for sure dangerous than for a non-disabled one.

2) Streets with no parking lots for disabled can become impracticable for disabled car drivers. Maybe he/she could not have enough space for a wheelchair in a non-disabled car park. Instead for non-disabled ones this aspect does not generate any problem for sure.

For example they will have information about cars parked in a handicap parking areas or on ramps for disabled people.
All the private users are supposed to have downloaded the app from the store,to have done the sign-up entering all the mandatory data and finally to have accepted all the permissions.
The private user, according to his/her category, will take advantage of a different type of service.
A car driver, for example, will have information about parking lots or parking violation, that is not required in a biker service.
A cyclist will have informations about bicycle paths, while pedestrians will have detailed data about sidewalks or pedestrian crossing, with all the possible violations related to them. The location of the pedestrian crossings and of the bicycle paths must be provided by the municipality.

The authorities will receive Safestreets information by the municipality that will share this data with them. We have done this choice because the authorities are part of the municipality of a city. This means that the authorities are managed by the municipality and so it is better that the system sends the data to the central body.

Furthermore, the application can encounter:

- Users that have downloaded the app but that haven't already signed-up in the system;
- Users that have already signed-up the system but that haven't logged-in yet. Users logged-out from the system belong to this category;
- Users also logged-in the system.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

- [**D1**]: The user is supposed to submit a correct e-mail.
- [**D2**]: The user creates one and only one account.
- [**D3**]: Every municipality which has an account on the system certifies its own account as valid.
- [**D4**]: The GPS signal has a relative error of 10 meters.
- [**D5**]: The memory where the data is stored is persistent.
- [**D6**]: The user allows the app to have access to his/her position and the camera of his/her device.
- [**D7**]: The internet connection has to be enabled when the app needs it.
- [**D8**]: The municipality must provide the reports (sent by the users) to the authorities automatically.
- [**D9**]: The email identifies uniquely a private user.
- [**D10**]: The fiscal code identify uniquely a person.
- [**D11**]: It is impossible that 2 roads that are in the same city have also the same name.
- [**D12**]: A street can be identified with its name and the city to which it belongs.
- [**D13**]: A disabled user belongs only to the pedestrians category or the car drivers one (Because we suppose that the users with motor disabilities rarely ride a bicycle or a motorbike).

- [**D14**]: SafeStreets has in its database a unique ID associated to a city (that only the municipality of that city knows).

2.4.2 Constraints

- The system must treat the users' personal data in according to the GDPR regulation.
- The system will be designed and implemented for Android and IOS smart-phones.
- The application will be available only in the Italian app stores.

2.4.3 Dependencies

- The system will use the GPS services provided by the smart-phones.
- The system will use the internet services offered by the smart-phones.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

- **SafeStreet Interfaces**
  - **SafeStreets Logo:**
    SafeStreets has a simple but explicative logo. It represents a street, because the main theme of the application concerns public roads. It also has a green tick, that represents the correct functioning and the safety of the streets.

Figure 4: SafeStreets Logo

– **SafeStreets log-in page:**
This is the page a user sees when he/she downloads the app and opens it for the first time or when he/she has done a log-out. In this page there is the possibility to log-in (inserting username and password) or to sign-up. The user can also recover his/her credentials using the function below the log-in button. When logged-in, the user will be redirected to the Home Page, that will be different for private users and authorities.

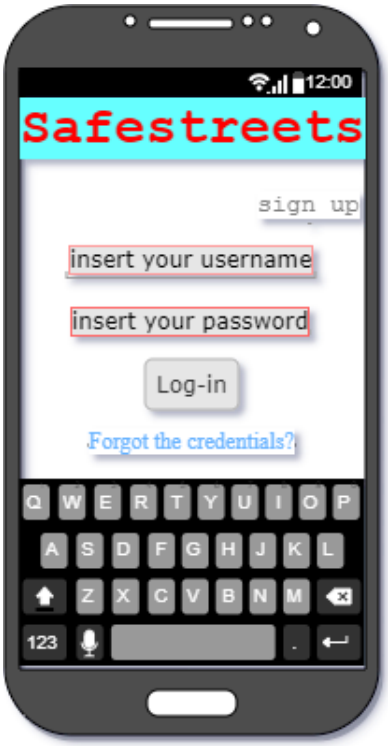


Figure 5: SafeStreets Log-in

– **SafeStreets sign-up account choice:**

  Here the user can choose the type of account he/she wants to create. There is the account reserved to the municipality and the one available to a private user. By choosing one or the other, the user will be redirected to the second step of the sign-up.

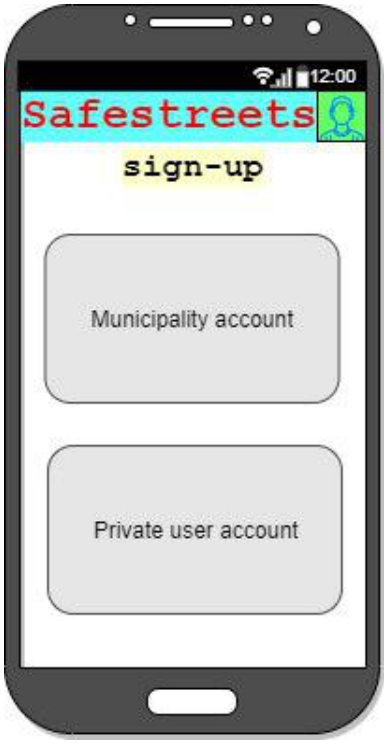


Figure 6: SafeStreets Account Choice

– **SafeStreets private user sign-up:**

This is the first part of the private user's sign-up. Here the user has to put his/her username he/she will use in SafeStreets (that will also identify him uniquely) and the password, that has to comply the security policy. Then he/she can continue to the second part of the registration.

Figure 7: SafeStreets private user sign-up, first part

– **SafeStreets private user sign-up (second part):**

This is the second and last part of the private user's sign-up in the SafeStreets platform. Here the user has to fill the fields with his/her personal e-mail and his/her FC. Furthermore he/she has to choose which category he(she belongs to (car driver, cyclist, motorcyclist, pedestrian, and so on) and he/she can also indicate if this account will be created for a disabled person or not. Finally he/she can add the id of his/her drive license if he/she drives a vehicle that needs it (It's not mandatory). After doing that the sign-up is complete and the user will be redirected to the log-in page.

Figure 8: SafeStreets private user sign-up, second part

– **SafeStreets municipality sign-up:**
If the user has chosen the municipality sign-up he/she will be redirected here. The authorities must indicate the city of their competence and must insert an ID code to prove that they have the authorization to represent the municipality. Then they have to insert a username and a password, in the same way as explained for private users. Then they will be redirected to the log-in page.

Figure 9: SafeStreets Municipality sign-up

– **SafeStreets Private User's Home Page:**

This is the Home Page reserved to private users. Its functions are very easy to understand and they are: the possibility to report a violation to the system, the chance to see which are the most reported streets in a selected area, the possibility to see the condition (in terms of safety) of a selected street, parking lot or area. Finally there is also the log-out button and the accessibility button, that allows the user to change his/her category.

Figure 10: SafeStreets Private Users' Home Page

– **SafeStreets Private User's Violation report:**

This is the first function presented in the Home Page. Here the user can report a violation that he/she has seen. The first field to fill is the one that allows the user to describe the violation, indicating the type of violation that he/she has seen (for example motorcycle parking areas occupied by cars, or traffic accidents, and so on). Then he/she has to indicate the date and the hour of the violation. Furthermore a user has to indicate (manually or with GPS signal) the street and the city of the violation and finally he/she can take a photograph of the violation (this is mandatory).

Figure 11: SafeStreets Violation Reporting interface

– **SafeStreets Most reported streets interface :**

This interface allows the user to check which are the most reported streets in a city. The city can be inserted manually by the user or retrieved automatically by the system using the GPS signal of the user's smartphone. The user has to specify how many streets the list should be made up. Then he/she must submit the request and he/she will receive the list.

Figure 12: SafeStreets Most reported streets interface

– **SafeStreets safest streets interface :**

Here the user can check which is the condition of a selected area. He/she can specify the street manually, inserting its name and the name of the city. The street can also be retrieved automatically by the system with the GPS signal of the user's smartphone. Then the user has to insert a radius (expressed in KM) within which the system will provide the condition of the contained streets, highlighting the safest and the most dangerous ones.

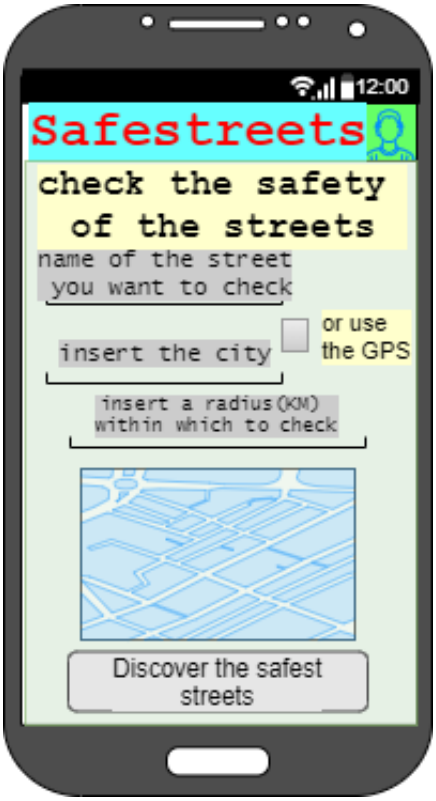


Figure 13: SafeStreets safest streets interface

– **SafeStreets safest parking areas :**

The user can use this functionality to find the safest place where to park his/her vehicle. He/she has to specify his/her position, that can be done manually (inserting the name of the city and the street) or, as in the other functionalities, automatically (using the GPS signal of the smartphone). Then the user has to insert the maximum distance (expressed in kilometres) from the selected position to which he/she is willing to get. After have inserted all the needed information, the user will see the safest parking area where to go, highlighted differently from the less safe ones.

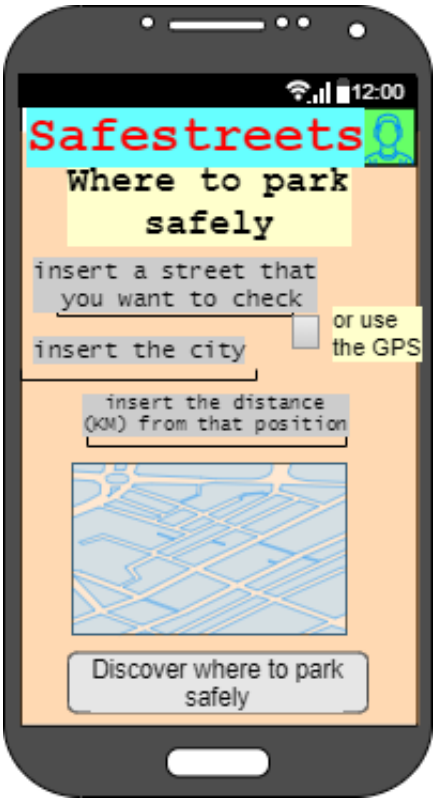


Figure 14: SafeStreets safest parking area interface

– **SafeStreets User Category Change :**

Here the user can change his/her category. There are some buttons, that represents a driver, a motorcyclist, a cyclist and a pedestrian. The user has to pick his/her new category. Then he/she can also indicate if the account will be used by a disabled person or not.

Figure 15: SafeStreets User Category change

– **SafeStreets Municipality Home Page :**

This is the Home Page reserved to the authorities. Here the municipality has the possibility to choose between four functionalities: three are structured in the same way of the ones designed for the private users and it's the possibility to check the most reported streets. The third is the one to check the conditions of a selected area. The interfaces for the requests are the same than the ones for the users. The municipality also has the possibility to do a request to obtain suggestions in order to improve the quality of their service and, so, the safety of the city.

Figure 16: SafeStreets Municipality Home Page

– **SafeStreets Municipality suggestions request :**

Here the municipality can make a request in order to discover which are the suggestions provided by SafeStreets to enhance the safety of the city. The authorities have to insert the name of the city, or provide it using the GPS signal of their device.

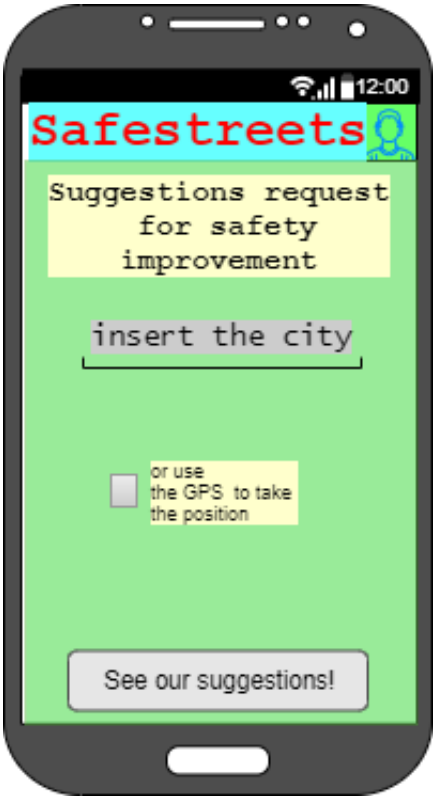


Figure 17: SafeStreets Municipality Suggestions Request

After having submitted the information, the municipality will receive the suggestions provided by SafeStreets concerning the indicated city. The data is retrieved by SafeStreets mining the information received by the users with their reports.

Figure 18: SafeStreets Municipality Suggestions List

3.1.2 Hardware Interfaces

The system doesn't offer any kind of hardware interface to the user. In fact the user can use the application simply thanks to his/her phone.

3.1.3 Software interfaces

The system offers its service through the use of the following software interfaces:

Operating systems:
The system will have to be designed for the most common smart-phones operating system:

- Android
- IOS

3.1.4 Communication interfaces

Each smart-phone that will use Safestreets will communicate with the servers thanks to the HTTPS protocol. Also the databases will communicate using the same protocol. We've chosen this protocol because thanks to it, it is possible to obtain a secure connection between nodes. Furthermore it is a very commonly used protocol for the connections over internet.

3.2 Functional Requirements:

3.2.1 Use case diagrams:

Use case diagrams are very important because they allow the reader to understand the flow of events between actors and the system itself. We have done two different use case diagrams to make them more explanatory and less confusing.

The first diagram:

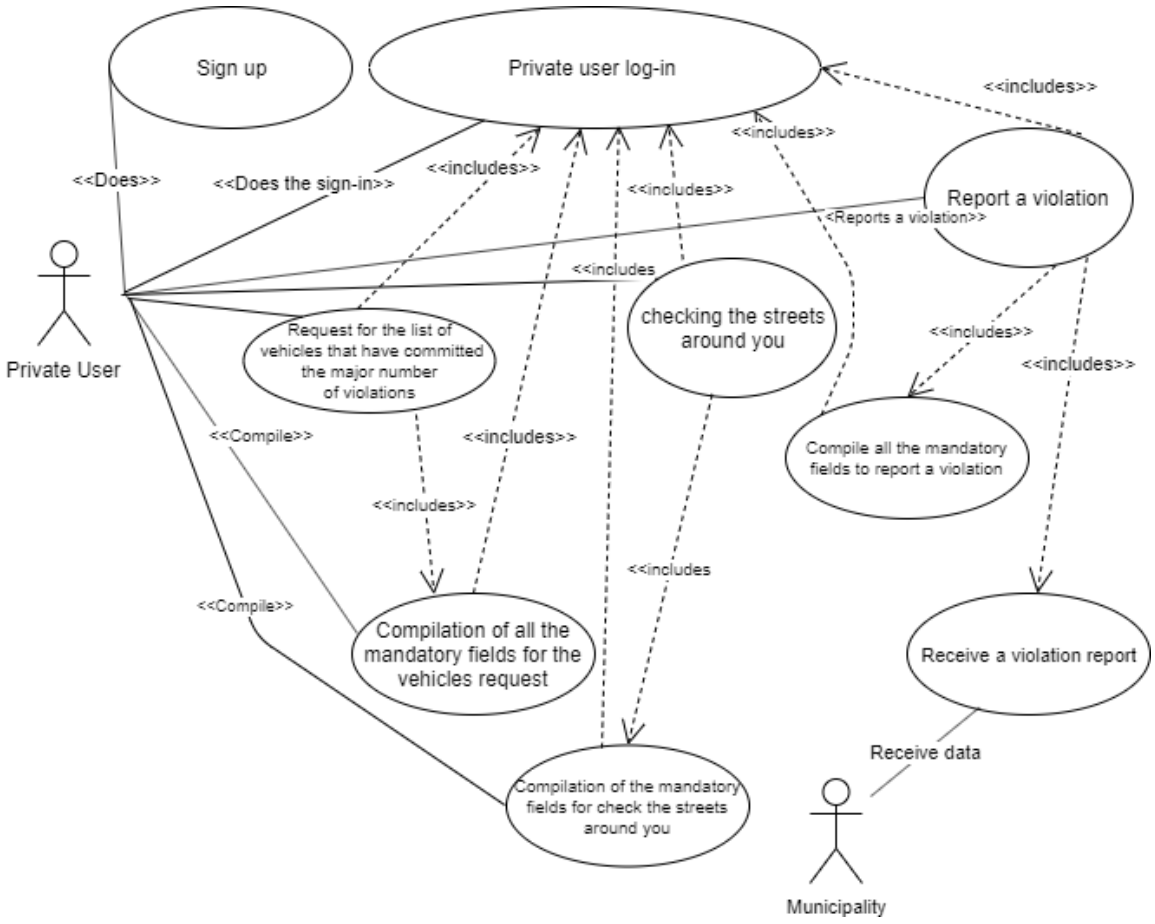


Figure 19: First use case diagram

The second diagram:

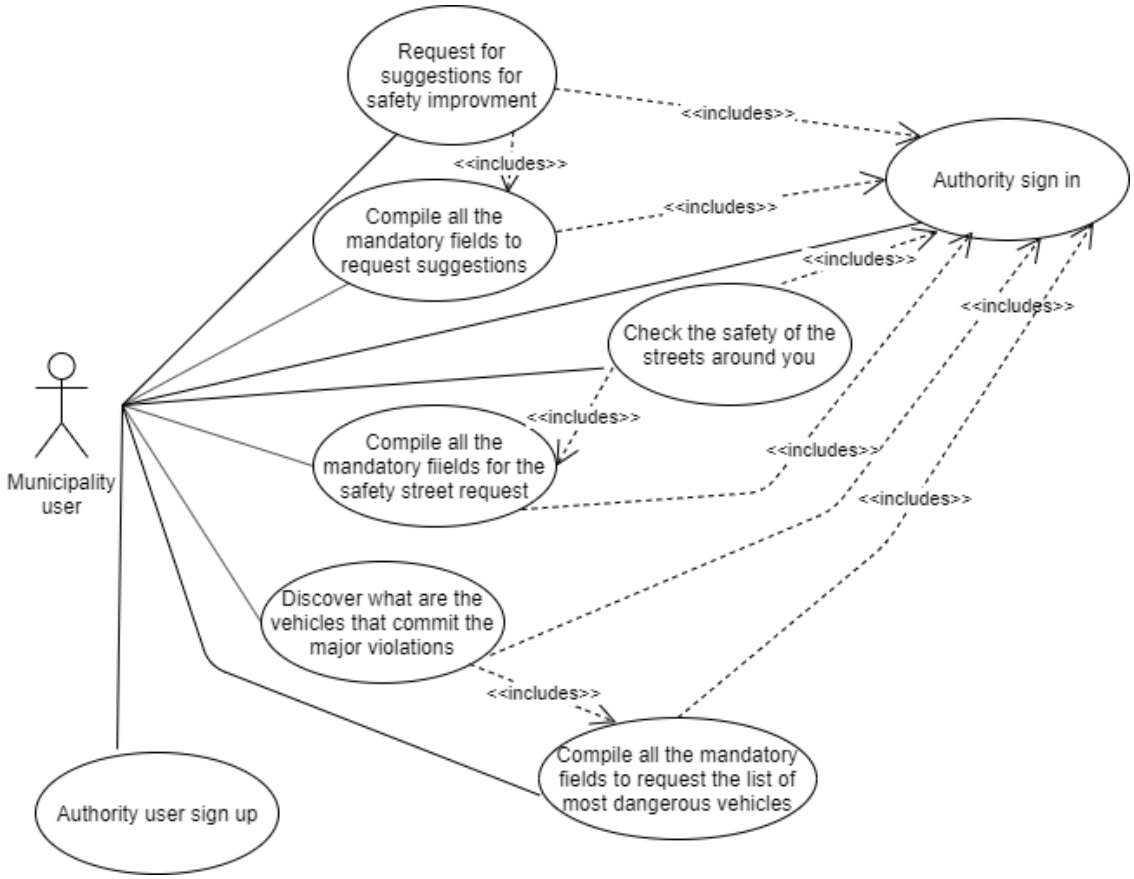


Figure 20: Second use case diagram

3.2.2 Scenarios

From the private user point of view:

Scenario 1: Luigi is a middle-aged man who lives in a very busy area of Milan. This district is very known to be a zone with a strong lack of parking. People usually park their vehicles in illegal zones. Because of this, he has found cars parked in front of his home gate several times. He has called the municipality and the authorities many times and now he is fed up about this situation. He usually calls the authorities with his phone but this process takes a lot of time because the telephone line is always busy. He would like the authorities to have a better communication channel.
Luckily he has downloaded an application called Safestreets. He has read in the

description the fact that the application allows a user to report a violation via smart-phone.
Thanks to this mobile application he is now able to save time in reporting violations. Furthermore, he has observed that the authorities have started to check more frequently even without any report (because of the Safestreets suggestions to the municipality that have suggested to authorities to check more frequently).

Scenario 2: Giovanni lives very close to the place where he/she works but he usually goes to work by car even if he would like to go there by bicycle. The problem is that the zone is very busy and unfortunately accidents involving cyclists happen very often. Because of this, he does not feel like taking the risk. He notices an interesting application on the market called Safestreets. He is particularly interested in the function which suggest the safest streets around the user from a point of view of the category in which the user has signed-up. This interest is caused by the idea of being able to choose a route that minimizes the danger of making an accident against a car taking in consideration the suggested streets given by the application.
Now, he is finally able to go to work by bicycle in a secure way.

Scenario 3: Andrea is a disabled person who strolls very often in his district. Unfortunately not all the roads are equipped with facilities for disabled people and also a lot of people park their cars there.
He has downloaded the application which suggest him the streets which are the most secure to be travelled for a person with his problem. The app allows Andrea to avoid roads where it is more probable to have cars parked on disabled facilities. He can compute himself the route from a place to another taking in consideration the suggestions given by the application. When he sees a violation he can report it to the authorities. Thanks to this feature Andrea is able to report violations, to do his strolls in the city almost in an autonomous way.

Scenario 4: An employee of an insurance has the task of understanding which are the most dangerous areas of Milan in order to change the price of insurance. He does not know where to find the data necessary to do this task.Furthermore it has just been decided by the boss that funding a research would be too costly. Fortunately the insurance company has just downloaded the Safestreets applications which can be used to retrieve dangerous areas. Thanks to Safestreets the company saved a lot of money by doing a free data search.

From the municipality point of view:

Scenario 5: The municipality of Milan is worried about the safety of the streets in a particular zone. The zone that we are considering is full of pedestrian crossings and unfortunately a lot of cars exceed the speed limit constantly making the street very dangerous for pedestrians. The municipality to improve the safety taking in consideration various projects.
It has recently discovered that there exist the application Safestreets that helps the municipalities to do changes in the streets to improve the safety. The municipality has taken in account the suggestions and it has noticed that the projects now have improved and become more detailed.

3.2.3 Use Cases

ID code	Use Case 1
Name	Private user sign-up
Actor	Private user
Preconditions	• The private user has downloaded SafeStreets on his/her device. • The private user has not already created an account on SafeStreets.

Events flow	1. The private user opens Safestreets; 2. The system shows the private user the log-in page; 3. The private user selects sign-up; 4. The system asks the user to choose between private account and municipality account; 5. The private user chooses private account; 6. The system shows the user the first page of the private account registration; 7. The private user inserts a username; 8. The private user inserts a password; 9. The private user submits his/her data; 10. The system checks the correctness of this first part of the sign-up; 11. The system shows the user the second part of the registration; 12. The user inserts his/her personal e-mail; 13. The user inserts his/her FC; 14. The user indicates his/her category; 15. The user indicates if the account will be used by a disabled person; 16. If he/she is a car driver or a motorcyclist, the user can insert his/her driving license; 17. The user submits these other data; 18. The system checks the new data; 19. The system sends the user an e-mail to confirm the registration; 20. The user confirms the sign-up;

Postconditions	• The system stores the private user's account data in its database; • The user has a personal account and can do the log-in, accessing the functionalities of SafeStreets.
Exceptions	• The private user inserts an email that is already associated to an account; • The private user inputs a password that isn't valid because it doesn't comply the security policy; • The private user inserts a FC already associated to an account.; • The private user inserts personal data that don't match with the FC; • The user doesn't specify his/her category; For all these exceptions: The system shows the user an error message and the events restart from the first page of private user registration (Event 6). • The user selects the login button inserting not existing credentials. The system shows the user an error message and the user will be redirected in the log-in page (Event 2).

ID code	Use Case 2
Name	Municipality sign-up
Actor	Municipality
Precondition	• The municipality has downloaded Safestreet on a smartphone. • The municipality doesn't already have an account on SafeStreets.
Events flow	1. The municipality opens SafeStreets; 2. The system shows the municipality the log-in page; 3. The municipality selects sign-up; 4. The system asks the municipality to choose between private account and municipality account; 5. The municipality chooses municipality account; 6. The system shows the municipality the account registration page; 7. The municipality inserts the city of its competence; 8. The municipality inserts its unique ID code; 9. The municipality inserts a username; 10. The municipality inserts a password; 11. The municipality submits its data; 12. The system checks the correctness of the data inserted by the municipality; 13. The system automatically confirms the registration.
Postconditions	• The system stores the data about the new municipality account on its database; • The municipality now has an account and can log-in, accessing the municipality functionalities of SafeStreets.

Exceptions	• The municipality inserts a password that isn't valid because it doesn't comply the security policy; • The municipality inserts an ID code already associated to an account; • The municipality inserts a wrong ID code. The ID code is considered wrong if and only if it doesn't match the code corresponding to its city in SafeStreets database. • The municipality inserts a city that has already an account associated to it; For all these exceptions: The system shows the user an error message and the events restart from the municipality registration page (Event 6). • The municipality selects the login button inserting not existing credentials. The system shows the user an error message and the user will be redirected in the log-in page (Event 2).

ID code	Use Case 3
Name	Log-in
Actor	User (private or municipality)
Precondition	• The user has downloaded the application; • The user has already done the sign-up.
Events flow	1. The user opens SafeStreets; 2. The system shows the user the log-in page 3. The user fills the form inserting his/her personal credentials; 4. The user submits the credentials tapping on the log-in button; 5. The system checks the correctness of the credentials.
Postconditions	• The user is logged-in, so he/she is redirected to the Home Page and is able to use SafeStreets with its functionalities, that are different for private and municipality accounts.
Exceptions	• The user inserts wrong or not existing credentials; The system sends notifies the user with an error message and redirects him/her to the log-in page (Event 2).

ID code	User Case 4
Name	Private User Violation Report
Actor	Private user and Municipality
Preconditions	• The user has logged into SafeStreets.
Events flow	1. The private user selects "Report a violation" in his/her Home Page; 2. The system shows the user the report page; 3. The user takes a photo and uploads it; 4. The user describes the violation, indicating in particular the type (parking, traffic, etc.); 5. The user inserts the date and the hour of the violation. 6. The user inserts manually the place where the violation has occurred or , alternatively, uses his/her GPS signal to provide it automatically. 7. The user submits his/her report. 8. The system checks the correctness of the report; 9. The system retrieves the license plate with an algorithm from the photograph and then stores the data about the violation in its database; 10. The system sends the report to the municipality; 11. The municipality receives the report and forwards it to the authorities.
Postconditions	• The system database now contains the information about the violation; • The authorities now have the report.

Exceptions	• The system can't manage to retrieve the license plate from the picture; • The user doesn't put complete data about the violation; In this cases the system sends an error message and redirect the user to the report page (Event 2).

ID code	Use Case 5
Name	Most reported streets
Actor	User (private or municipality)
Preconditions	• The user has logged into SafeStreets.
Events flow	1. The user selects "Which are the most reported streets?" in his/her Home page; 2. The system shows the user the most reported streets interface; 3. The user inserts, manually or automatically (with GPS signal), the city he/she wants to check; 4. The user must specify how many streets the system must provide him; 5. The user submits his/her request; 6. The system checks the correctness of the request; 7. The system provides the list of the most reported streets.
Postconditions	• The user will see the list of the most reported streets.
Exceptions	• The user provides a non existing city name; • The user provides a non valid number of streets he/she wants to receive; The system sends an error message and redirects the user to the most reported streets request interface (Event 2).

ID code	User Case 6
Name	Checking the safety of an area
Actor	User (private or municipality)
Preconditions	• The user is logged into SafeStreets.
Events flow	1. The user selects "Check the safety of the streets" in the Home Page; 2. The system show the user the interface to do a street check request; 3. The user specifies the street and the city to check (manually or automatically with GPS signal); 4. The user indicates how far from the indicated point the system has to provide the information about the safety of the area; 5. The user submits the request to the system; 6. The system checks the correctness of the request; 7. The system provides the map with the requested area highlighted.
Postconditions	• The user will see a map of the indicated area with the safest and the most dangerous streets highlighted in a different way.
Exceptions	• The user provides a non existing street or city; • The user indicates a non valid radius; The system sends an error message and redirects the user to the streets safety request interface (Event 2).

ID code	Use Case 7
Name	Check the safety of a parking area
Actor	Private user
Preconditions	• The user is logged into SafeStreets.
Events flow	1. The user selects "Check for a safe parking area" in the Home Page; 2. The system shows the user the page to do the request for safe parking areas; 3. The user specifies the street and the city near which to check (manually or automatically with GPS signal); 4. The user indicates how far from the indicated point the system has to provide parking areas; 5. The user submits the request to the system; 6. The system check the correctness of the request; 7. The system provides the map with the requested parking areas highlighted.
Postconditions	• The user will see a map, provided by the system, that contains the parking areas in the zone requested by him, with the safest and the most dangerous ones highlighted with different colours.
Exceptions	• The user provides a non existing street or city; • The user inserts a non valid distance from which to check; For this exceptions the system sends to the user an error message and redirects him/her to the request page (Event 2).

ID code	Use Case 8
Name	Suggestions for municipality
Actor	Municipality
Preconditions	• The municipality is logged into SafeStreets;
Events flow	1. The municipality user selects "See the suggestions for safety improvements" in the Home Page; 2. The system shows the municipality user the page to make the suggestion request; 3. The municipality user insert the city for which he/she wants to receive suggestions (with GPS signal or manually); 4. The user submits the request for the city of competence of his/her municipality; 5. The system checks the correctness of the request; 6. The system, with a data mining algorithm, retrieves useful information from the users' reports regarding that city and adds it to the information that it has from the municipality side; 7. The system sends the result of its research to the municipality user.
Postconditions	• The municipality will see a table showing the suggestion that the system has provided for their city. This table is structured to have on one side the problem reported by the users, and on the other side the solution provided by SafeStreets.
Exceptions	• The municipality user provides a non existing city; The system will send to the user an error message and will redirect him/her in the suggestions request page.

ID code	Use Case 9
Name	Change of the user type
Actor	Private user
Preconditions	• The user is logged into SafeStreets.
Events flow	1. The user taps the accessibility button (represented with a hand) in the Home Page; 2. The system shows the user the interface for selecting his/her new type; 3. The user selects his/her new type and submits his/her selection by tapping it on the touch screen; 4. The system stores the new data about the user in its database;
Postconditions	• The system has the new information about the user's type in its database.
Exceptions	

ID code	Use Case 10
Name	Log-out
Actor	User (private or municipality one)
Preconditions	• The user is logged into SafeStreets.
Events flow	1. The user taps the log-out button in the Home Page; 2. The system asks the user for the confirm of the log-out; 3. The user confirms the log-out.
Postconditions	• The user will be logged out from SafeStreets and will be redirected to the log-in page.
Exceptions	

3.2.4 Sequence Diagrams

- The first sequence diagram represents the registration process. The actors are the user (municipality or private one) and the system.

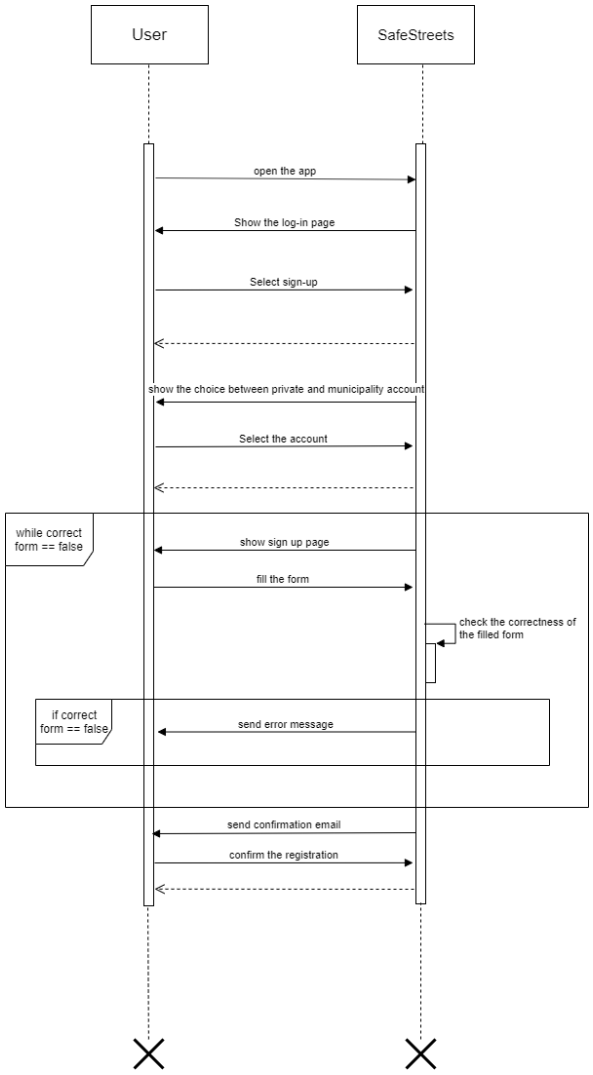


Figure 21: First Sequence Diagram

- The second sequence diagram represents the log-in process. The actors are the user (municipality or private one) and the system.

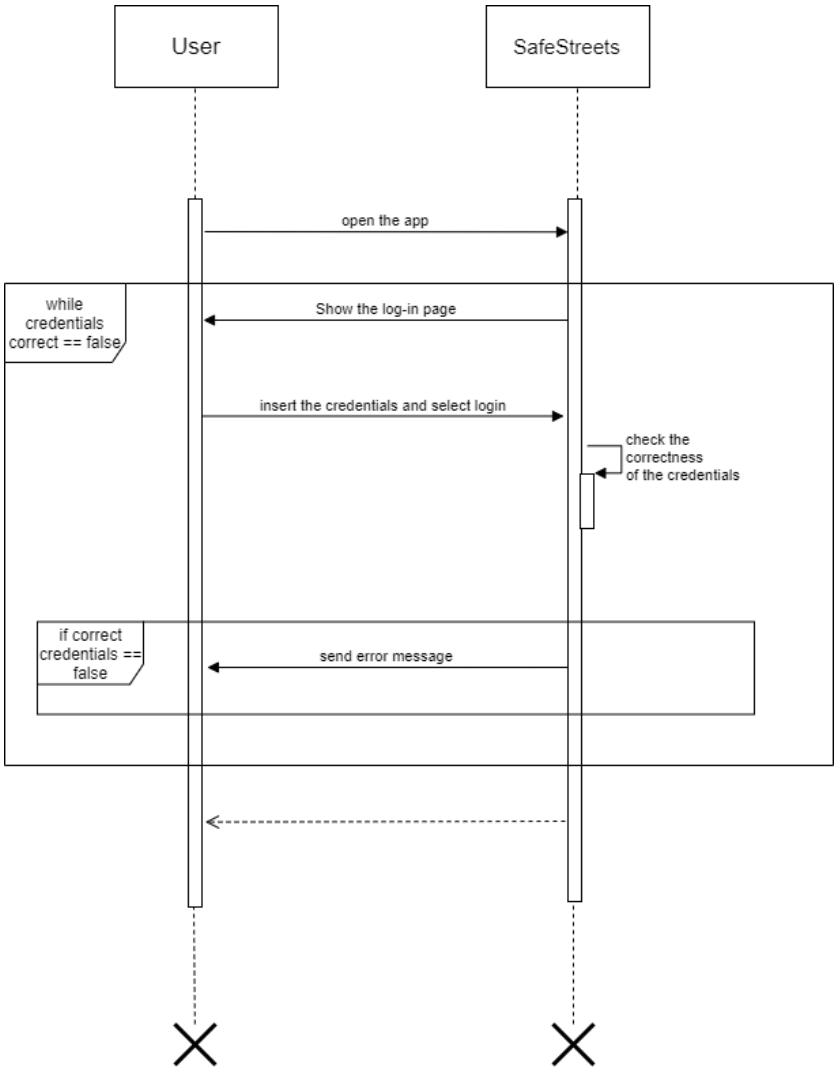


Figure 22: Second Sequence Diagram

- The third sequence diagram represents the violation reporting process. The actors are the private user that sends it, the system and the municipality that receives it.

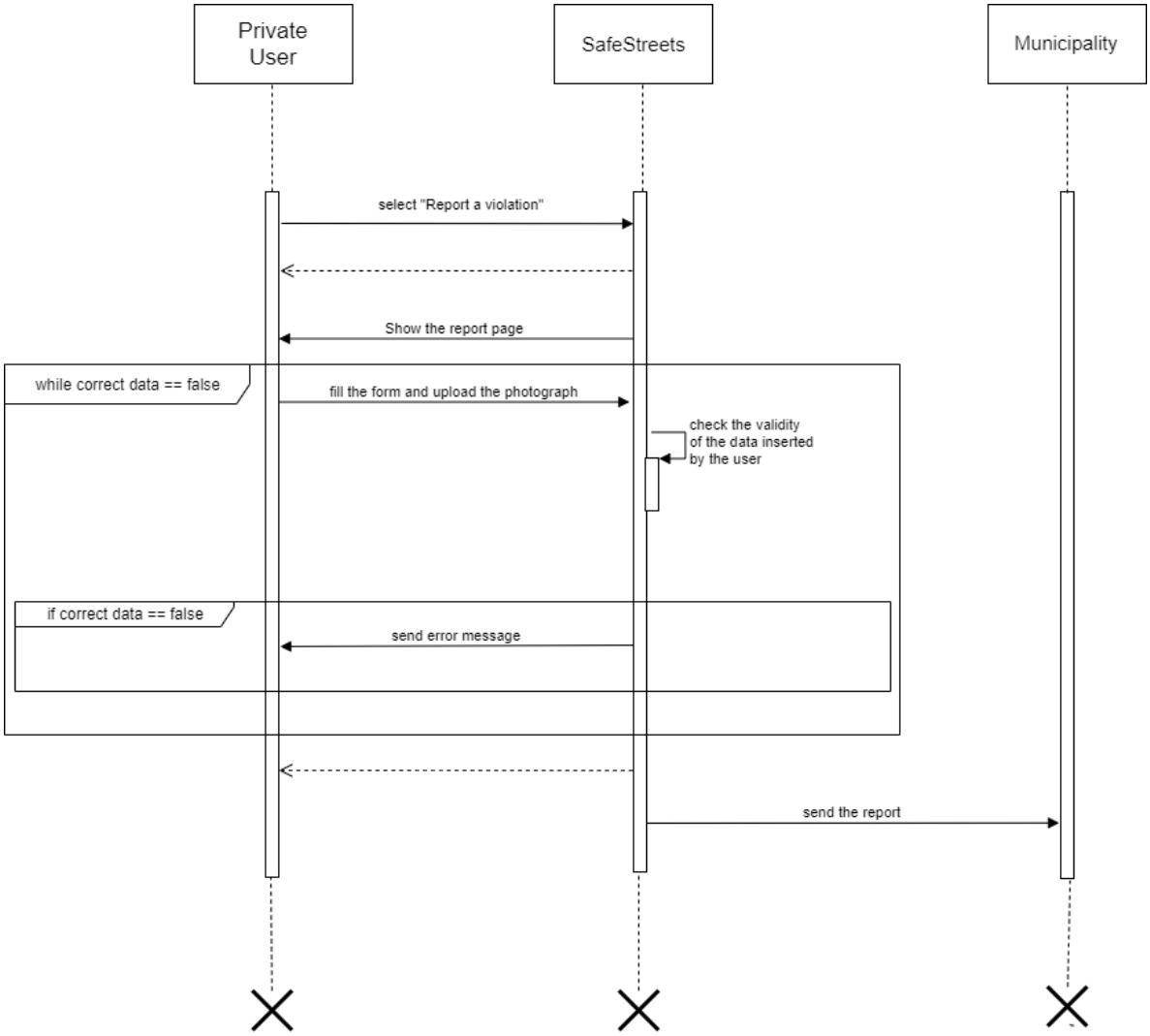


Figure 23: Third Sequence Diagram

- The fourth sequence diagram represents process to make a request to the system. The actors are the user (private and municipality) and the system. The private user uses this process to request for the safest parking areas, the condition of the streets, the most reported streets and the most reported vehicles, while the municipality can make the request for the most reported vehicles and streets, the condition of an area and to receive suggestions for improvements.

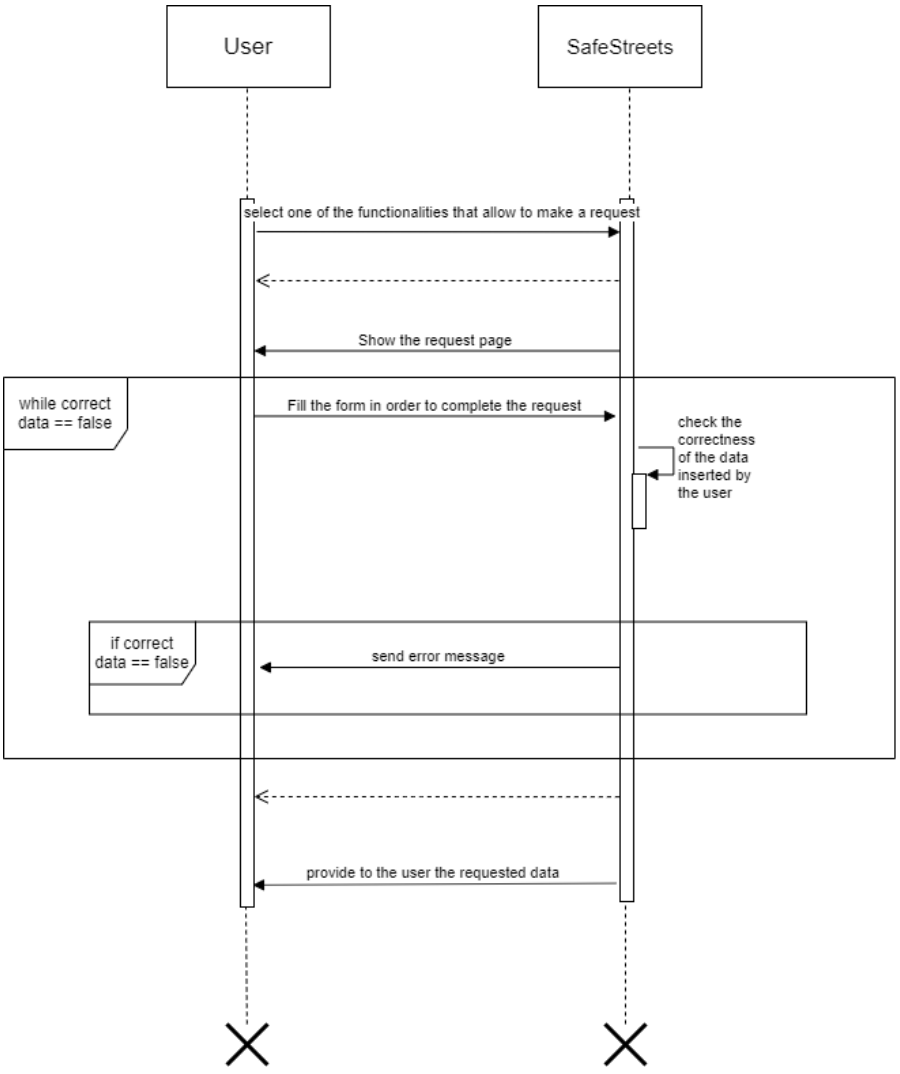


Figure 24: Fourth Sequence Diagram

- The fifth and last sequence diagram represents the process for changing the category of the private user. The actors are the private user that wants to change his/her category and the system.

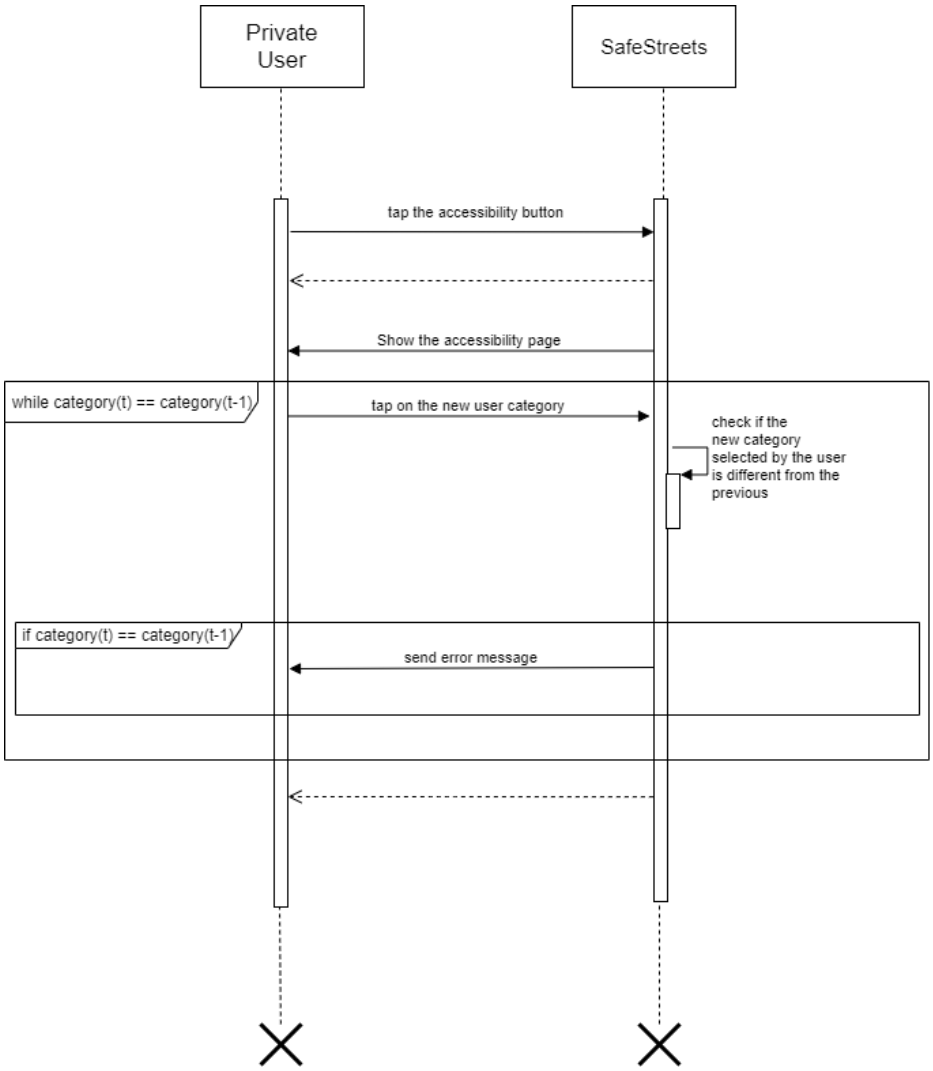


Figure 25: Fifth Sequence Diagram

3.2.5 Goal mapping on requirements

- $[G1]$: The service allows the users to report a violation to the municipality:
  - $[R1]$: The system must allow the user to take a picture.
  - $[R2]$: The system must allow the user to indicate the street where the violation occurred.
  - $[R3]$: The system must allow the user to indicate the hour and the date when the violation occurred.
  - $[R4]$: The system must allow the user to insert a brief description of the violation.
  - $[R5]$: The system can localize his/her position automatically as alternative to the manual insertion.
  - $[R6]$: The system must allow the user to finally submit the report.
  - $[R7]$: The system must run an algorithm to retrieve the license plate from the violation picture.
  - $[R8]$: The chosen username must identify the private user uniquely.
  - $[R9]$: The pictures must be taken only through the app camera and not imported from the device gallery in order to take photos in real time.
  - $[R10]$ :: The system must store the report if it is correct.
  - $[R11]$ :: The system must check if the reported street exists in reality.
  - $[D4]$: The GPS signal has a relative error of 10 meters.
  - $[D6]$: The user allows the application to have access to his/her position and to the camera of his/her device.
  - $[D7]$: The internet connection has to be enabled when the app needs it.
  - $[D8]$: The municipality must provide the reports (sent by the users) to the authorities automatically.
- $[G2]$: The service allows the users to have detailed informations about the violations and the streets safety.
  - $[R12]$: The system must allow the user to insert the street around which he/she wants to check the safety.
  - $[R13]$: The system must be able to retrieve users' position data from the GPS service of their smartphones.
  - $[R14]$: The system must allow the user to insert the city in which the violation occurred.
  - $[R15]$: The system must allow the user to insert a number representing the radius within which the system must provide the condition of the streets.

- [$R16$] : The system must report the safest streets in a map form. These streets must be highlighted on the map.
- [$R17$] : The system must run a data mining algorithm to update the information (contained in the database) about the violations and the streets safety.
- [$R18$] : The system must retrieve the information from its database.
- [$D4$] : The GPS signal has a relative error of 10 meters.
- [$D7$] : The internet connection has to be enabled when the app needs it.

- [$G3$] : Safestreets must send to the municipality suggestions to improve the safety of the streets whenever the municipality asks for them.
  - [$R19$] : The system must retrieve the information about violations from the users' reports.
  - [$R20$] : The system must use both data received by the users and by the municipality.
  - [$R21$] : The system must run a data mining algorithm to update the database of the system. It must update the suggestion for the municipality.
  - [$R22$] : The system must retrieve municipality suggestions from its database.
  - [$D5$] : The memory where the data is stored is persistent.
- [$G4$] : The user (both private user and the municipality) must be able to authenticate into the Safestreets platform.
  - [$R23$] : The system must allow the user to choose between municipality account or private user one.
  - [$R24$] : The system must allow the municipality to sign-up if and only if they insert a valid ID code.
  - [$R25$] : The system must allow the user (both private user and the municipality) to sign-up if and only if he/she inserts a username that is not already used.
  - [$R26$] : The system must allow the user (both private user and the municipality) to sign-up if and only if he/she inserts a password that complies the security policy.
  - [$R27$] : The system must allow the user to recover his/her password if he/she has forgotten it.
  - [$R28$] : The system must allow the municipality users and the private ones to access the corresponding Home Page after the log-in.
  - [$R29$] : The system must allow the user to log-out from the platform in every moment by the exit button placed in the Home Page.

- [$R30$] : The system must check that the chosen username (chosen during the sign-up phase) must be unique.
- [$D1$] : The user is supposed to submit correct e-mail, and data matching his/her FC.
- [$D2$] : The user creates one and only one account.
- [$D3$] : Every municipality which has an account on the system certifies its own account as valid.
- [$D5$] : The memory where the data is stored is persistent.
- [$D7$] : The internet connection has to be enabled when the app needs it.
- [$D9$]: The email identifies uniquely a private user.
- [$D10$]: The fiscal code identify uniquely a person.

+

3.3 Performance Requirements

In this part we will indicate the performance requirements of SafeStreets. Performance requirements:

- A response time (from the point of view of the user) of at most 2 seconds for the violation reporting.
- A response time (from the point of view of the user) of at most 3 seconds to answer to a user request.
- The state of the streets (dangerous or not) must be updated at least every 24 hours.
- The municipality suggestions must be updated at least every 2 days.

3.4 Design Constraints

3.4.1 Standard Compliance

- The system will comply a standard that allows to test the software automatically from the requirements (D0-178C).
- For the distances we use the standard SI measurement unit, that is the meter.

3.4.2 Hardware Limitations

A user, to be able to use SafeStreets with all its functionalities, must run the application on a smartphone that has at least:

- 200Mb of mass memory;
- 2Gb of RAM;
- an internet connection of 15 Mb/s.

3.5 Software System Attributes

3.5.1 Reliability

The Reliability of the system must be high, because the system has to provide to the users a service that has to be as much as possible fast and correct. So the fault tolerance and the tolerance when a component of the system goes down has to be very low.

3.5.2 Availability

The system, in order to provide an efficient service, must be available at a good level. We think that it must have the target value of 95%, because we want that the app will be sufficiently available to every user.

3.5.3 Security

The security is fundamental in our system, as it contains the users' sensitive data. In particular:

- The system uses HTTPS in order to allow to communicate with users, municipality and the DBMS in safety;
- The sensitive data, such as passwords, has to be encrypted before being stored in the DB.

3.5.4 Maintainability

The software will have to be developed in order to be maintained in future and, eventually, to add some new features. So it has to be programmed well, avoiding code duplication and using good design principles and design patterns.

3.5.5 Portability

The application is designed for IOS and Android, so it will have to be interchangeable from one OS to the other, preserving the accounts and the personal data. In the back-end part, it has to be designed in a way that it will be independent from the OS.

4 Formal analysis with alloy

4.0.1 Overview of this section

In this section there is a description of the system using alloy. Alloy is a declarative specification language which is used to model systems in a formal way. This model can be used to reduce ,as much as possible, the intrinsically ambiguity due to natural language used so far. Another very important aspect to consider is the validation of the model. Through the use of formal models we can establish if our requirements and domain assumptions are together sufficient to achieve our goals.
In this model we have not shaped all aspects of the system but only the ones that we consider as very important aspects of the system.The non represented aspects are not so many. We have decided so because we didn't want to make a too much complex alloy model. A too much complex alloy model would have been too much difficult to understand for many readers and in our opinion one of the scope of this document is also to be sufficiently understandable for any interested person.

4.0.2 Code

```
/*all the signature necessary to define the fundamentals ones with respect to
    ↪  the model*/
sig Date{} // a simple date
sig Username{}// a username of a person in the application
sig Password{}// the password
sig fc{} //signature of the fiscal code
sig Email{}// this is a signature of an Email
sig Drivelicense{}// signature of the driving License
abstract sig boolDisabled{}
abstract sig bool{}
sig logged extends bool{} // a signature which is associated to the fact that
    ↪  a user is logged into the system
sig notLogged extends bool{}//a signature which is associated to the fact
    ↪ that a user is not logged into the system
sig Disabled extends boolDisabled{} // signature associated to the fact that
    ↪ a person is disabled
sig Notdisabled extends boolDisabled{}//signature associated to the fact that
    ↪  a person is not disabled
abstract sig dangerousBool{}
sig DangerousInGeneral extends dangerousBool{}//a signature that indicates
    ↪ that the street is considered as dangerous
sig NotDangerousInGeneral extends dangerousBool{}//a signature that indicates
    ↪  that the street is not considered as dangerous
sig Picture{} // a signature of a picture
sig City{} // a signature of a city
sig MunicipalityIdCode{} // this is a signature of a code associated to the
    ↪ identity of the municipality
sig Violationdescription{}// a signature associated to the description of the
    ↪  violation
sig HourViolation{}// a signature associated to the hour when the violation
    ↪ occurred.
sig LicensePlate{}// a signature of a license plate of a vehicle


/*This represent one single data given by the municipality. This data is used
    ↪  to mine them in order to retrieve
more detailed informations. This data is mined by Safestreets*/
sig MunicipalityDataAccidents{
```

```
        street:  one Street,
        hour: one  HourViolation,
        date: one  Date,
        licenseplate: one  LicensePlate
}



/*this is the signature of the private user. It is abstract because the
    ↪ instances are the categories of the
user*/
abstract sig privateUser{
        username:one Username,
        password:one  Password,
        fiscalcode:one fc,
        email:one  Email,
        islogged: one bool
}


/*A category of user that is a car driver. This signature contains also a "
    ↪ flag" that indicate if the user is
disabled and the driving License. This signature contains also the driving
    ↪ license of the user.*/
sig cardriver extends privateUser{
        disabledChoice:one boolDisabled,
        license:one Drivelicense
}

/*This is a category of user which is a pedestrian user.He/she can also be a
    ↪ disabled person or not. */
sig pedestrian extends privateUser{
        disabledChoice:one boolDisabled
}

/*This is the motorcyclist user. In this signature it is also included  his/
    ↪ her driving license*/
sig motorcyclist extends privateUser{
                license: one Drivelicense
}

/*This is the signature of the user of the municipality who have a code to
    ↪ identify the municipality and
a password  to access his/her account. It is also present the city of which
    ↪ the user belongs*/
sig MunicipalityUser{
        city: one City,
        municipalityIdCode: one MunicipalityIdCode,
        password: one Password
}

/*Fake integer.It is necessary only to represent numbers*/
sig integer{}

/*This signature represents a user who belongs to the cylists category*/
sig cyclist extends privateUser{}

/*This is latitude and longitude that determine a place*/
sig Place{
        latitude: one integer,
        longitude: one  integer
}

/*This signature represents a street with its city, its longitude and
    ↪ latitude and if it is considered dangerous or not*/
sig Street{
        city: one City,
        place: one Place,
```

```
        dangerousInGeneral:one  dangerousBool
}

/*This signatire represents a violation report. It includes: the date, the
    ↪ hour and the street related to the violation.
It also include the user who has reported the violation, the picture of the
    ↪ violation and the license plate of the vehicle related to the
violation.*/
sig violationReport{
        dateViolation: one Date,
        hour : one  HourViolation,
        description: one  Violationdescription,
        street: one  Street,
        user: one  Username,
        picture: one Picture,
        licenseplate: one LicensePlate
}

/*This contains all the streets of the same city that are considered
    ↪ dangerous. All the dangerous streets of  a city are contained in
this signature */
sig DangerousStreetsInCity{
        street: set Street,
        city:one City
}{
        (all s1,s2:street | s1.city=s2.city and s1.city=city)and
        street.dangerousInGeneral=DangerousInGeneral
        #street>0
}
/*This fact ensures that if a dangerous street exsists, then there must be a
    ↪ DangerousStreetsInCity that contains it with the same city*/
fact dangerousStreetsGroupExsistence{
        (all s:Street | (s.dangerousInGeneral=DangerousInGeneral) implies
        (some dsc:DangerousStreetsInCity | s in dsc.street ))
}
/*This fact ensures the fact that there cannot exsist 2
    ↪ DangerousStreetsInCity with the same city and the intersection of the
    ↪  streets of 2 different
DangerousStreetsInCity is the empty set.  */
fact uniquenessOfDangerousStreetsInCity{
        (no disj dsc1,dsc2: DangerousStreetsInCity | dsc1.city=dsc2.city) and
        (all dsc1,dsc2:DangerousStreetsInCity | (dsc1≠dsc2)implies(dsc1.
            ↪ street & dsc2.street = none) )
}

// this implies that if 2 streets have the same longitude and latitude, then
    ↪ they are in the same city and they are the same city
fact notSameCoordDifferentCity{
        (all s1,s2:Street | (s1.place.latitude=s2.place.latitude and s1.place
            ↪ .longitude=s2.place.longitude) implies
        (s1.city = s2.city and s1=s2))
}


/*This alloy function returns the average number of reports per street*/
fun averageNReportPerStreet[c:City]: one Int{
        div[#{r:violationReport|r.city=c} ,#{s:Street|s.city=c}]
}

/*This function returns the total number of reports in a street of a city*/
fun NReportInStreet[s:Street]: one Int{
        #{r:violationReport|r.street=s and r.street.city=s.city}
}

/*This alloy function returns true if the street is considered dangerous and
    ↪ false if not.
A street is considered dangerous iff the number of reporting in that street
    ↪ is 10% major than the average per street. If this condition holds,
```

```
    ↪ then
the street is considered as dangerous*/
fun IsStreetDangerous[s: Street]: one dangerousBool{
        {d:dangerousBool | (d=DangerousInGeneral) iff(mul[NReportInStreet[s
            ↪ ],10]>mul[11, averageNReportPerStreet[s.city]])}
}

/*For each street this fact establishes if it is considered dangerous or not
    ↪ */
fact dangerousnessOfStreet{
        (all s:Street |(s.dangerousInGeneral=IsStreetDangerous[s])) or
    (all s1:Street | (some dc:DangerousStreetsInCity |dc.street=s1)or
        (all s2:Street |(some mda:MunicipalityDataAccidents | mda.street=s2 )
            ↪ ))
}


/*Here we want to ensure that latitude and longitude identify uniquely a
    ↪ street*/
fact positionStreet{
        (all s1,s2:Street|( (s1.place.latitude=s2.place.latitude and s1.place
            ↪ .longitude=s2.place.longitude)implies
        (s2=s1 and s1.place=s2.place)))and
        (all p1,p2:Place| (p1.latitude≠p2.latitude or p2.longitude≠p1.
            ↪ longitude)iff(p2≠p1))
}

/*Here we want to ensure that there is a 1 to 1 corrispondence between the
    ↪ city and the code related
to the municipality. This is like to say that the code related to the
    ↪ municipality identifies both the municipality itself
and the city of the municipality.*/
fact linkBetweenCityAndCode{
        (all mu1,mu2:MunicipalityUser|((mu1.city≠mu2.city)iff(mu1.
            ↪ municipalityIdCode≠mu2.municipalityIdCode)))
}

/*Here we want to ensure that there are not standalone city instance. In fact
    ↪  a city can be linked only to a MunicipalityUser or to a street.
A Municipality cannot be standalone and it can be linked only to
    ↪ MunicipalityUser. A password is linked to a MunicipalityUser.
In this fact we are also ensuring the fact that there are not standalone
    ↪ MunicipalityIdCode and Passwords. A password is linked
only to a MunicipalityUser or a privateUser and the MunicipalityIdCode is
    ↪ linked only to a MunicipalityUser*/
fact everythingLinkedToMunUsr{
        ((all c:City|(some m1:MunicipalityUser| m1. city=c ))or
        (all c1:City|(some s:Street|  s.city=c1))or
        (all c2:City |(some  dc:DangerousStreetsInCity |dc.city=c2)))
        and (all m:MunicipalityIdCode|(some m2:MunicipalityUser|m2.
            ↪ municipalityIdCode=m))and
        (all p:Password|(some m2:MunicipalityUser|m2.password=p)or
        (all p1:Password |(some u1:privateUser | u1.password=p1)))
}

/* here wa want to ensure that an email identifies a user in an unique way.
    ↪ It is also like this for the
fiscal code  and the username*/
fact userIdentifiers{
   (no disj u1,u2:privateUser | u1.email=u2.email or u1.fiscalcode=u2.
       ↪ fiscalcode
        or  u1.username=u2.username)
}

// we ensure that the driving license can uniquely identify a motorcyclist or
    ↪  a carDriver
fact uniqueDrivingLicense{
        (no disj u1,u2:cardriver | u1.license=u2.license) and
```

```
        (no disj u3,u4:motorcyclist | u3.license=u4.license) and
    (no u5:cardriver,u6:motorcyclist | u5.license=u6.license)
}


/*Here we want to ensure that every instance of Name,Surname,Username,
    ↪ Password,fc,Email,Date,
Drivinglicense will be linked to at least one user.*/
fact everythingLinkedToUser{
        ((all us: Username|(some u3:privateUser |u3.username=us))or
        (all us1: Username|(some v4:violationReport|v4.user=us1)))and
        (all pa: Password|(some u4:privateUser | u4.password=pa)) and
        (all f: fc|(some u5:privateUser |u5.fiscalcode =f)) and
        (all em: Email|(some u6:privateUser |u6.email=em ))and
        (((all d1: Date|(some v:violationReport | v.dateViolation=d1)))or
        (all d2:Date | (some md: MunicipalityDataAccidents |md.date=d2) ))and
        (all b:bool|(some u8:privateUser|u8.islogged=b)) and
        (all p:Place|(some s:Street|s.place=p))and
        (all i:integer|(some p:Place|p.longitude=i or p.latitude=i))and
        (all h:HourViolation|(some v2:violationReport|v2.hour=h)or
        (all h1:HourViolation|(some mda:MunicipalityDataAccidents | mda.hour=
            ↪ h1)))and
        (all de:Violationdescription|(some v3:violationReport|v3.description=
            ↪ de))and
        (all p:Picture|(some v4:violationReport|v4.picture=p))and
        (all lp: LicensePlate|(some v5:violationReport| v5.licenseplate=lp)or
        (all p1:LicensePlate |(some mda:MunicipalityDataAccidents | mda.
            ↪ licenseplate=p1)))and
        (all dg:dangerousBool|(some std:Street|std.dangerousInGeneral=dg))
}

/*Here we want to ensure that every boolDisabled is associated to a
    ↪ pedestrian or a cardriver and it cannot be standalone.*/
fact DisabledInstancesLinkedToUser{
        (all b:boolDisabled |(some c1:cardriver| c1.disabledChoice = b ))or
        (all b1:boolDisabled |(some p1:pedestrian| p1.disabledChoice = b1 ))
}



/*Here we want to ensure that every drivingLicense instance will be linked to
    ↪  at least one motorcyclist or one
Cardriver*/
fact drivingLicenseLinkedToUser{
        ((all dr:Drivelicense|(some m:motorcyclist|m.license=dr)) or
        (all dr1:Drivelicense|(some cd:cardriver|cd.license=dr1)))
}



--
    ↪ ------------------------------------------------------------------------------------------------------------------------
    ↪


/*This are the predicates that allow to see some of the several possible
    ↪ static views of the alloy model.
These are some of the possible static worlds:*/


pred showStaticViewOfUsers{
        #privateUser=2
        #MunicipalityUser=1
        #MunicipalityDataAccidents=0
        #violationReport=0
        #DangerousStreetsInCity=0
        #Street=0
}
```

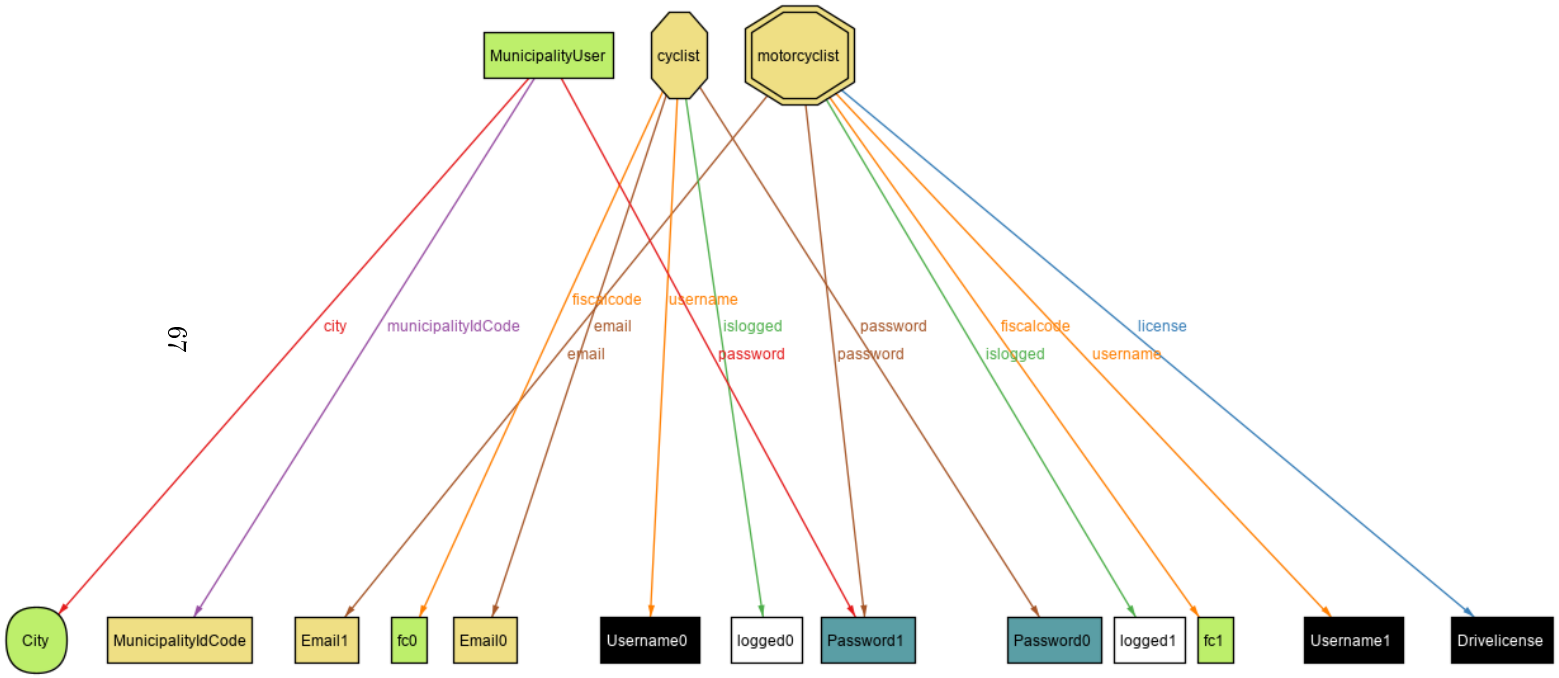
```
pred showStaticViewOfViolationsAndStreets{
        #privateUser=0
        #MunicipalityUser=0
        #MunicipalityDataAccidents=2
        #violationReport =1
        #Street = 4
        #City=2
}

pred showStaticCompleteModel{
        #MunicipalityDataAccidents=1
        #violationReport =1
        #Street = 2
        #privateUser=1
        #MunicipalityUser=1
        #City=2
}


run showStaticViewOfUsers for 7
run showStaticViewOfViolationsAndStreets for 4
run showStaticCompleteModel for 4
```

4.0.3 Validation

In this subsection we will show 3 different worlds. Each generated world will show an important aspect of the model and so of the system itself.

World 1:

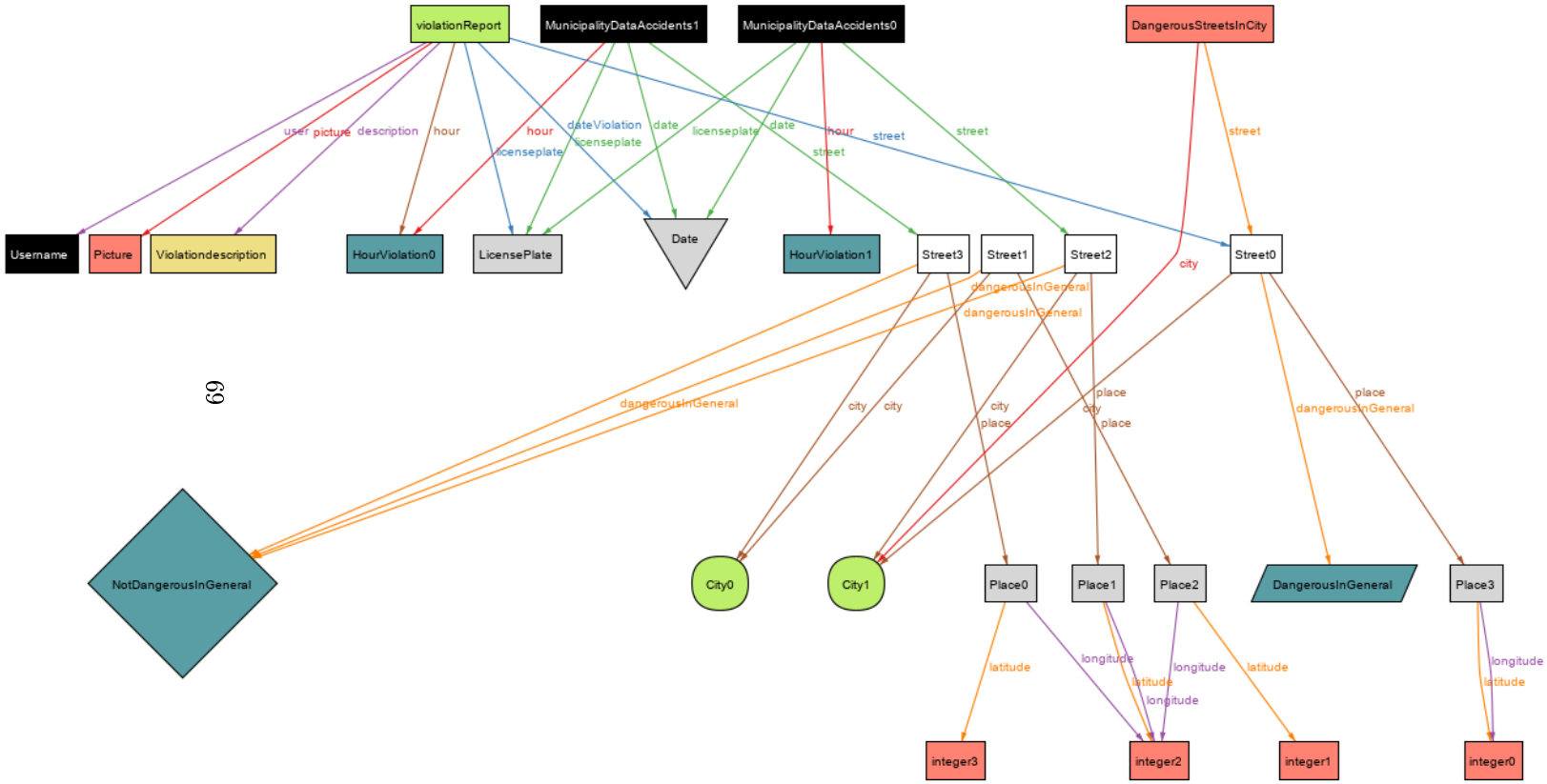
This world is generated by the run of the predicate **showStaticViewOfUser**. This world represents a view of the possible users of the application. Such a users are private users(privateUser in our model) and the municipality users (MunicipalityUser in our model) In this model there is also the representation of the characteristics of the users.


MunicipalityUser
cyclist
motorcyclist
city
municipalityIdCode
fiscalcode
email
email
username
islogged
password
password
password
fiscalcode
islogged
license
username
City
MunicipalityIdCode
Email1
fc0
Email0
Username0
logged0
Password1
Password0
logged1
fc1
Username1
Drivelicense

As you can see in the image above that each private user has a unique fiscal code, a unique email and a unique user-name. The other aspect can be equal to the ones of other users (they can be shared between different private users in the alloy model). The aspects that can be non unique are: the password and finally the state of the logging.
The Municipality users have a password, a code which identifies uniquely the municipality and finally the city of which the municipality refers. In this case the municipality users can have the same id code and the same city. This is the case in which they belong to the same municipality.

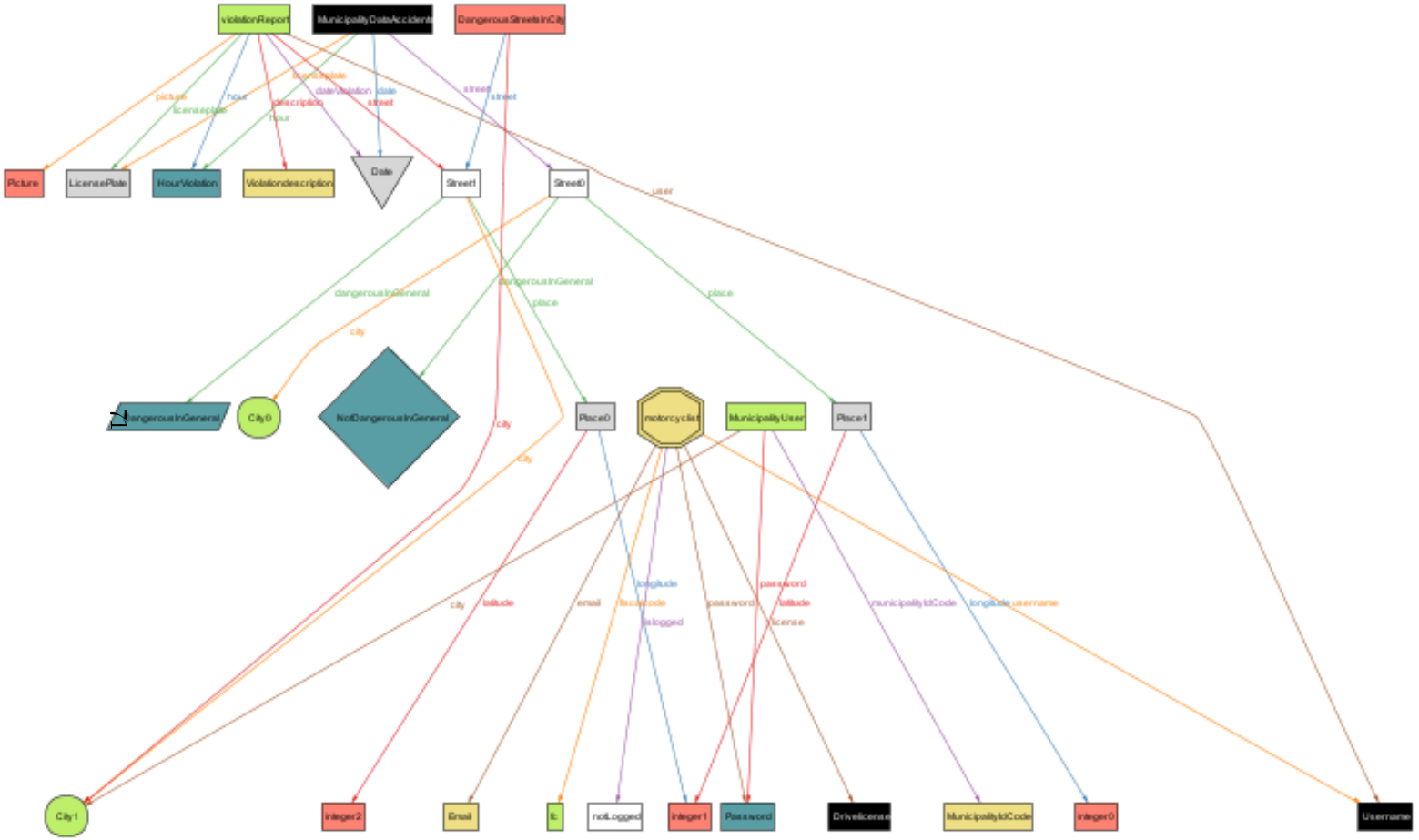
World 2: In this world we will show a view of how the violation reports are managed. We will show how the S2B will manage some aspect of the system which are the following:

- The reports of the violations.
- The concept of dangerousness of the streets in the system.
- The traceability of the streets on the system.
- The set of the most dangerous streets.
- The mining of the information given by the municipality and the one given by the reports.


violationReport
MunicipalityDataAccidents1
MunicipalityDataAccidents0
DangerousStreetsInCity
user picture
description
hour
hour
licenseplate
dateViolation
licenseplate
date
licenseplate
date
hour
street
street
street
street
Username
Picture
Violationdescription
HourViolation0
LicensePlate
Date
HourViolation1
Street3
Street1
Street2
Street0
city
dangerousInGeneral
dangerousInGeneral
dangerousInGeneral
city
city
city
place
city
place
place
dangerousInGeneral
NotDangerousInGeneral
City0
City1
Place0
Place1
Place2
DangerousInGeneral
Place3
latitude
longitude
latitude
longitude
longitude
latitude
longitude
latitude
integer3
integer2
integer1
integer0

As you can see in the image above, a street can be considered dangerous or not. It depends on the number of reports. It is considered dangerous if and only if the number of reports is 10% greater than the average number of violations per street reported by the users or 10% greater than the average violation per street retrieved from the municipality DB. This is respected in the world above. The streets which have the same name but they are different are distinguished expressing also the city of which they belongs.
The system mine also the information about accidents of the municipality in order to have better information about the safety.

World 3: In this world we will show a complete view of the system in its completeness. Every aspect of the two worlds above will be shown.


violationReport
MunicipalityDataAccident
DangerousStreetsInCity
picture
licensePlate
hour
violationDescription
date
street
user
Picture
LicensePlate
HourViolation
ViolationDescription
Date
Street1
Street0
dangerousInGeneral
place
city
NotDangerousInGeneral
City0
Place0
motorcyclist
MunicipalityUser
Place1
latitude
email
longitude
password
isLogged
municipalityIdCode
username
license
City1
integer2
Email
E
notLogged
integer1
Password
DriveLicense
MunicipalityIdCode
integer0
Username

Consistency:

Figure 29: Consistency

5 Effort

Task	Hours
Git setup	1
Group work on chapter 1	3
Writing in latex of sections 1.1 and 1.2 (purpose and scope) (already done in group)	2
Section 2.1	2,5
State chart diagram	1
Section 2.2 (Product functions)	2
Half of section 2.3 (User characteristics)	2
Half of section 2.4	2
Section 3.1	2
Use case diagrams	3
Mockups	6
Scenarios	3
Alloy	15
Chapter 4 review	3
	Total
	47.5

Table 11: Luca Massini's effort

Task	Hours
Git setup	1
Group work on chapter 1	3
Sections 1.3, 1.4, 1.5	1
Section 1.6	1
Section 2.1.1 (class diagrams)	2
Half of section 2.3 (User characteristics)	2
Half of section 2.4	2
Review of chapters 1 and 2	5
Section 3.1 (Mockups insertion in latex and description)	7
Use cases tables	8
Sequence diagrams	8
Section 3.2.5	3
Sections 3.3, 3.4, 3.5	1.5
Chapter 3 review	2
	Total
	46.5

Table 12: Daniele Nicolò's effort