POLITECNICO
MILANO 1863

RASD - REQUIREMENT ANALYSIS AND SPECIFICATION DOCUMENT

Version 1.0 - 10-10-1019

Computer Science and Engineering – Software Engineering 2

Amedeo Carrioli – 866256, Andrea Ceruti – 945604

Professor Elisabetta Di Nitto

POLITECNICO DI MILANO

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

1.1.1 General Purpose

This document represents the Requirement Analysis and Specification Document (RASD). In this document we will explain SafeStreets. This will be done by a detailed presentation of the proposed solution and its purpose, listing its goals, and the requirements and assumptions through which they will be achieved.

SafeStreets is a public interface aimed to public-spirit citizens who want to help keeping the streets clear. This S2B intends to provide users with the possibility to notify authorities when traffic violations occur. This materializes using a platform through which users can upload pictures of streets violations, in particular parking violations.

There can also be other types of issues that a customer can report, for example speed violations, accidents, non-respected traffic lights. They're reported in different ways.

The S2B also has a map, based on Google Maps, on which some areas are highlighted with different colors according to the number and types of violations reported (for example, firstly, the user can choose the type of violation, then the map shows different areas with different colors: a red area means that a lot of the chosen violation have occurred, yellow is medium quantity, green one means very few).

Once the violation is sent, its data are stored in SafeStreets data center and analyzed by the software, in order to retrieve information to update the map.

The customers of the application are both singular users and authorities, for example the Police Department, that can find the S2B useful in order to maintain the public order.

An important point is that a user can eventually report fake violations. First, the application allows the user to report a violation even if the user's geographical position and the violation's positions are different (for example the user sees an illegal parked car while he is jogging but doesn't want to stop his run to make the signalization, so he updates the violation once he gets home. This means that, when he is asked for the parking violation position, he could give a non-accurate position of the illegal parking). Secondly, there could actually be some users that find funny to make reportings fake violations, for now there is nothing we can do to fix this.

Finally, SafeStreets wants to offer a service exploiting the information of the municipality, if it allows users to retrieve the required information. When the municipality sends the data to SafeStreets (only accident info), the application crosses their data, which are reliable, and the ones given by the users, which are not, in order to have more reliable information for

the users. Then it updates the map and make suggestions regarding possible solutions to prevent parking violations, which will be made by considering the exact type of parking violations that occurred in a specific red area (high frequency).

1.1.2 Goals

- [G1] The application must allow a visitor to become a registered user after providing credentials.
- [G2] The application must allow private users to send reports of streets violations.
- [G3] The application must allow both end users and authorities to mine the information stored. This will be done by selecting areas and reports on the map.
- [G4] The system must send suggestions to authorities to minimize the illegal parking frequency in certain areas. When an area is red for a specific type of parking violation a suggestion is made.
- [G5] The system must be able to send the reliable parking violation to the municipality.
- [G6] The system must allow authorities to send accidents information to SafeStreets.

1.2 Scope

As already mentioned, the SafeStreets system is made to provide users a map where is possible to watch the areas and the reportings made by other users and a service to report streets violations. Once the user signs in with a proper username and password, is in the system and can use all the services offered by the application.
After the sign in the system will ask two questions to the user. One is if the user wants to share the GPS signal with the application, in order to be more precise about the areas close to him and allowing the user to select his current position in case of making a reporting. The second one is about crossing data with authorities. The idea is that SafeStreets sends reliable reporting of parking violations, made by the users, to authorities. In order to do that the S2B needs to ask the users if they allow the application to eventually share their reportings with authorities (their username won't be shared, only reporting).
Once the users are logged in, the system will allow to look at the map, view the user's personal information and report a street infringement. As mentioned before, on the map are highlighted different areas with different colors, based on the frequency of violations. The user can interact with the map. If a highlighted area is typed on, the number of infringements will be shown.

It is also possible to select the type of violation and the interval of time (today, last week, last month) in which the user is interested in watching the map. By default, if the user doesn't select any type of violation or interval of time, the map of all types of violations together occurred in the last month will be shown.

Regarding to the reporting of violations, the systems allows to select the type of reporting (parking, speed, traffic light violation or accident), only in case of parking is possible to send a picture. The system will ask for a picture with visible license plate, if the user doesn't send one, he can type the license plate by himself (not mandatory).

When parking violation is selected, it is also possible to choose the specific type of parking. The possible choices are:

- Crosswalk
- Sidewalk
- Double-parking
- Street crossing
- Red zone
- Taxi zone
- Bus zone
- Handicap spot
- Other

SafeStreets has a software that recognizes the areas in which a specific type of parking violation is above a certain number, this means that a lot of the same type of infringement occurred in the same area. A suggestion is made by the software, based on the type of parking violation. For example, if there are many sidewalk violations, the suggestion made will be "add a barrier between the street and the sidewalk to prevent unsafe parking". These suggestions are sent to authorities.

After any reporting, the user is asked if he wants to keep anonymous his reporting or not. If he chooses to be anonymous, on the map won't be shown his username along with his reporting, otherwise his username will be shown.

To prevent any misuse of the reporting violation system, if the user sends a picture with visible license plate and the application analyzing pictures algorithm fulfills in recognizing it, then that reporting will be shared with authorities. Otherwise, either if the user only types the license plate without sending a picture of it or the application analyzing pictures algorithm can't recognize the plate, it is considered not reliable, so the reporting won't be shared with authorities. It will only be added on the map.

For speed infraction, traffic light violation or accident reporting, is not possible to send a picture, in order to prevent the use of phones while driving. The user can only choose the type of violation.

The customer has also to choose the time and geographical position of the infraction, which is independent from the type of violation. It's either possible to select the user's position (using the GPS) or selecting a position on the map or typing the address.
The user can't open the gallery on his smartphone to update a picture. This is for preventing the update of fake parking pictures that were shot in a time we can't know. The user can open the camera through SafeStreets application and, either take the picture and send it, or take a picture and, if he doesn't have time to send it, the S2B puts a timer of 2 hours on the photo, during which the user can re-open the application and upload the picture. If the timer expires the picture gets deleted.
If the municipality offers a service through which is possible to retrieve information about the accidents, SafeStreets, when receives the date, stores them in the database of the application and analyzes them. This is done in order to update the map, which means identify unsafe areas (SafeStreets sees the municipality information as they were reliable reportings) and to give suggestions to the municipality itself to prevent more violations.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- User: the costumer of the application that send reports. It could be a private citizen or an authority like municipality. In this case the use of the application will be different.
- Application analyzing pictures algorithm: the algorithm that SafeStreets uses for recognizing the license plate of the car object of the violation.
- Municipality: this is the municipal police section that collaborates with SafeStreets for preventing accidents and violations.

1.3.2 Acronyms

- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface.
- GPS: Global Positioning System.
- S2B: Software To Be.
- GDPR: General Data Protection Regulation.

1.3.3 Abbreviations

- Gn: nth goal.
- Rn: nth requirement.
- Dn: nth domain assumption.

1.4 Revision history

- Version 1.0:
  - First release.
- Version 2.0:
  - This RASD changes from the previous version for the presence of the Alloy section.
- Version 3.0:
  - This is the final version of the RASD. The content has remained the same, we have edited the document template and other minor changes.

1.5 Reference documents

- Specification document: “SafeStreets Mandatory Project Assignment”.
- IEEE Std 830--1998 IEEE Recommended Practice for Software Requirements Specifications.
- Examples documents:
  - RASD Sample from A.Y. 2015-2016.pdf
  - RASD Sample from A.Y. 2016-2017.pdf

1.6 Document structure

The RASD is divided in 5 sections.

- **Section 1**:

  it is the introduction of the RASD in which the problem is presented informally with natural language. It provides base information such as the product to develop and the application domain. The scope part is an analysis of the world and the shared phenomena.

- **Section 2**:

it presents an overall description of the project. It describes external interfaces, summary of major functions, constraints, assumption and dependencies of the S2B. Furthermore, a class diagram and some state diagrams are provided to make stakeholders better understand the project, but even for giving more details on shared phenomena and the domain model.

- **Section 3**:

  this is the body of the document. It first describes the interfaces requirements. Then it lists some scenario to show how system works in real life situations and functional requirements. Lastly, we have nonfunctional requirements such as performance requirements and design constraints. This section will be useful for the development team.

- **Section 4**:

  here we have the Alloy formal description of the problem that includes all the relevant details.

- **Section 5**:

  it presents the effort spent for every member of the group.

2. Overall Description

2.1 Product perspective

The SafeStreets system is a completely new software application, designed from scratch. It is intended to be used as a mobile application for both private and authorities and as a web application for authorities only, in order to make easier the data crossing.

The S2B receives reporting of accidents and parking, speed and traffic lights violation. It stores the information on the application data center. Then updates the map every time a reporting occurs. The map as already said is based on Google Maps' APIs because it offers a very large library of APIs.

The application also works along with authorities. Authorities can only acquire reliable data from SafeStreets. A reporting is reliable only if it is a parking violation reporting, with a picture, and the algorithm to read the license plate fulfils.

Anonymity is an important aspect that must be respected. When authorities acquire data, the username of the costumer that made the reporting won't be sent. Only the reporting along with the picture, geographical position and time will be sent.

The municipality can also send information about accidents to SafeStreets. The information will be used as they were reliable reportings. The data is sent through the application in a specific format, provided to the authorities by SafeStreets. The authorities will have to compile some fields for every reporting they want to send: the data, time, geographic position of the accident, number of vehicles involved, and the license plates (not mandatory).

Now we will explain some parts the application, how it is constructed and how it works.

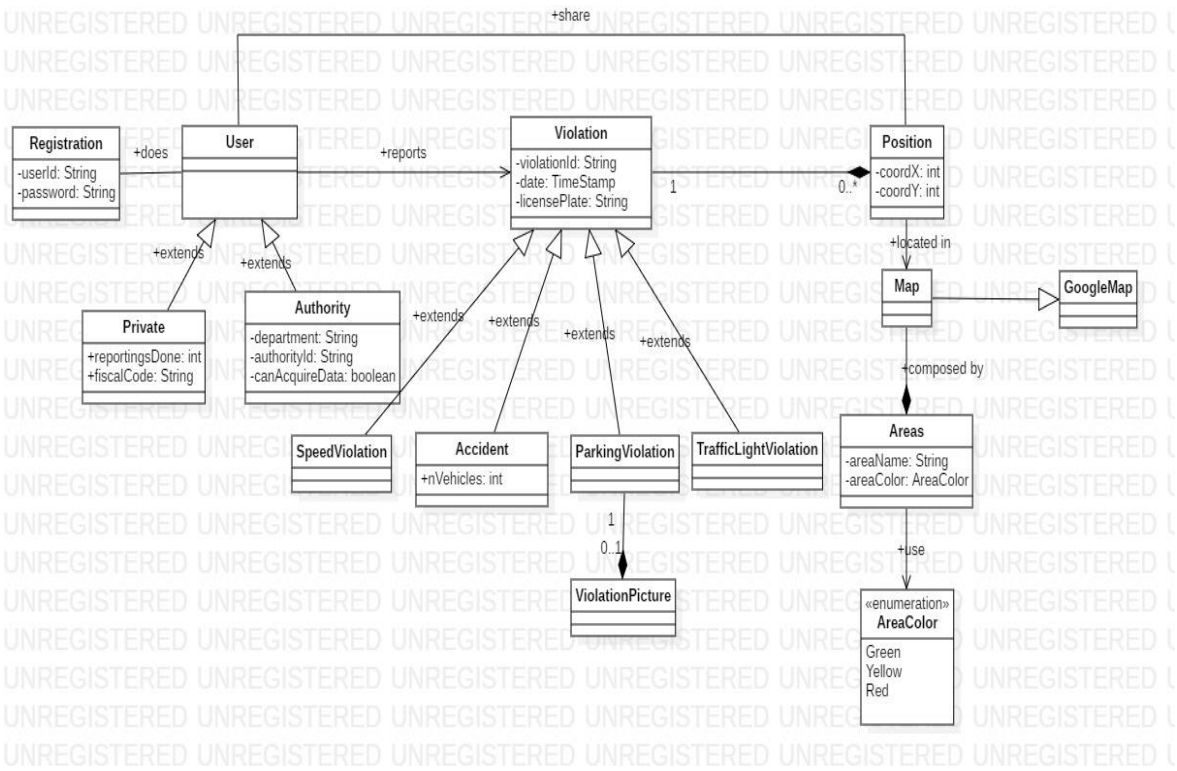


Figure 1 - Class Diagram

The class diagram (figure 1) shows the front-end side of the application. Users can be either private costumers or authorities, both of them must make a registration. Each private costumer has an attribute that stores the number of reportings that he has made.

A user can make a reporting violation. The violation will have a violationID, a date and a license plate, which is not mandatory. A violation occurs in one position, and in one position can occur many violations.

Only with a parking violation a picture can be sent. This was decided because in the other cases of violations, the probability of the user is driving while noticing the violation is very high (almost every accident is sees while driving, for example on the highways. Also speed violation: while driving it's easy

to notice a car that speeds, for example if it overtakes you.) Taking a picture while driving is very risky, besides being illegal.

A user can share the position of an infringement (using GPS, if he allows to). All the violations's geographical position are shown on the map, grouped in areas, that are differently colored (green, yellow, red) based on the frequency of the violations.

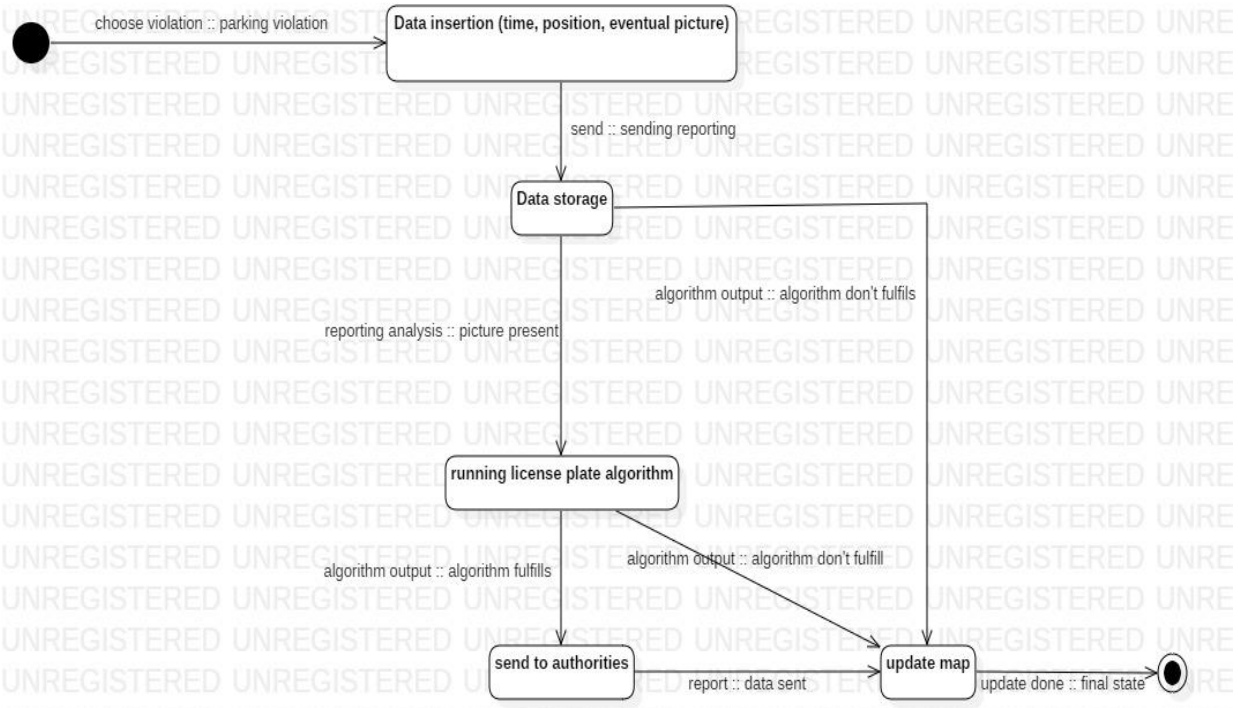


Figure 2 - State diagram 1: parking violation

This first state diagram (Figure 2) shows how the application works on a parking violation reporting. The user chooses the type of violation, then inserts the information about it.

Once it is sent to SafeStreets, the data is stored, then, if a picture is present, the license plate recognizing algorithm is ran. If the reporting is reliable (algorithm fulfills) the data are sent to authorities, otherwise it is only used to update the map.

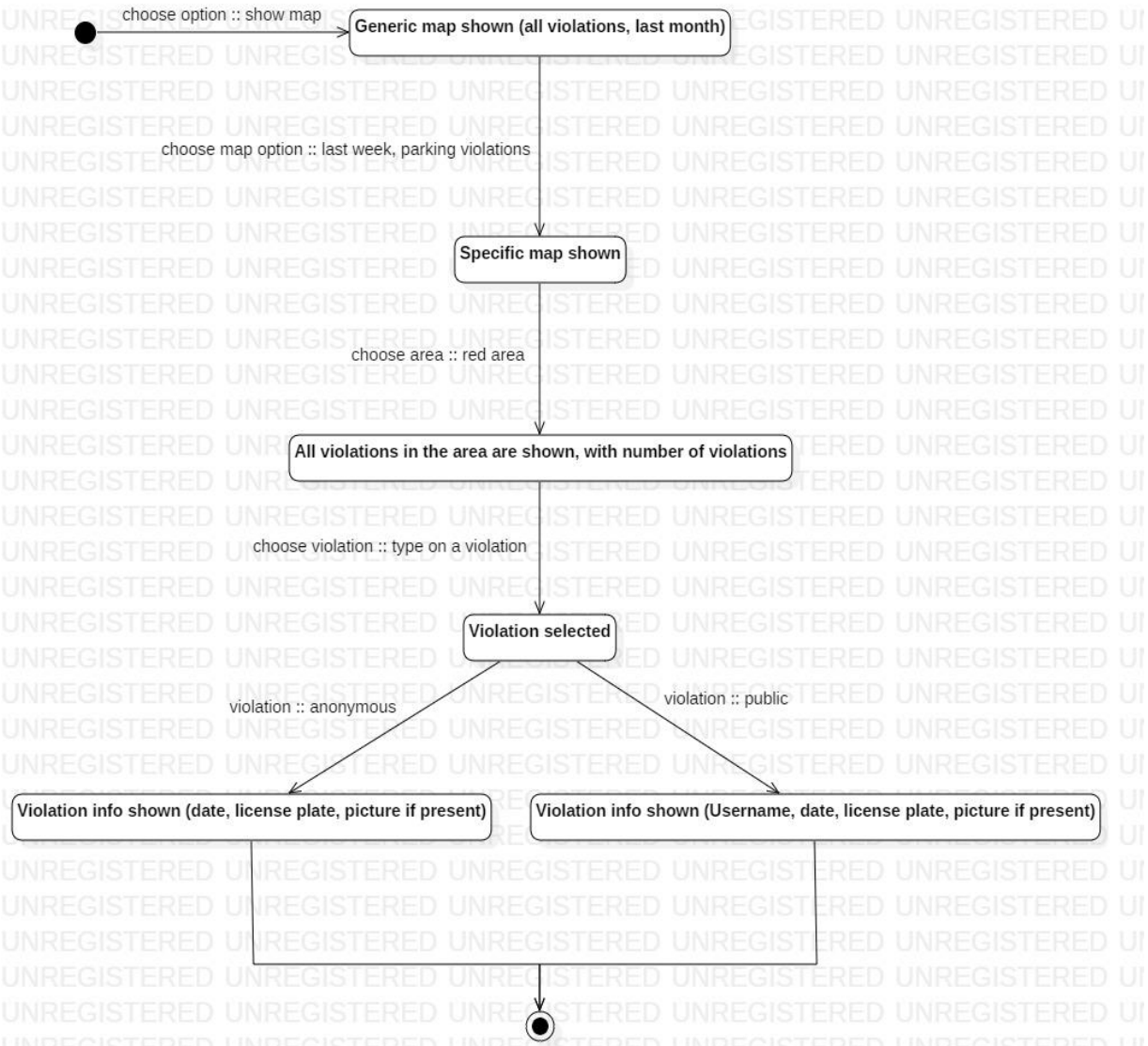


Figure 3 - State diagram 2: map usage

Figure 3 describes how the map works along with interaction of the user. Firstly, the map shows all types of violations together which occurred since a month ago from now. Then, the user can choose a specific type of violation, parking violations in the diagram, which occurred in a chosen time interval, last week.

Now the areas on the map regards parking violations that occurred last week, and the user can select by typing on it a colored area, in this example a red one. The area shows the exact positions of the violations (the position selected by the costumer that sent it) and the number of them. The user can type on a specific violation, information about it will be shown, including the date and time of the reporting, along with the picture if present.

As already said, after every reporting, the user is asked if wants the reporting to be an anonymous or not. If he chooses it to be anonymous, his username won't be shown to other users that select his violation on the map.

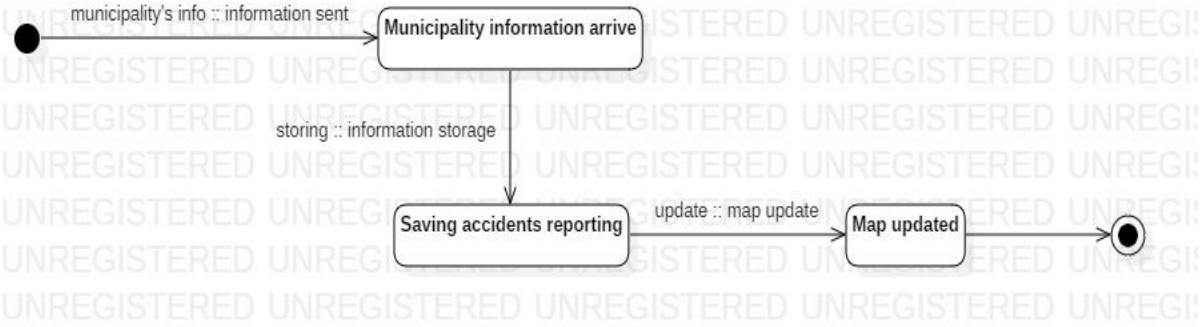


Figure 4 - State diagram 3: behavior on municipality info

State diagram 3 shows how SafeStreets works on the information the municipality shares. Information are stored on SafeStreets data center and considered reliable information. Then the map is updated with the violations arrived from the municipality. The reportings will be anonymous. The geographical position is given by the municipality, along with the time and the license plate can either be sent or not.

2.2 Product functions

In this section will be analyzed all the funcions that SafeStreets offers. Some have already been described in the previous parts. Here are listed and more precisely specified, with respect of the already mentioned goals of the system.

2.2.1 Reporting system management

This is the main functionality of the SafeStreets. It's based on the idea of safeness; therefore, the goal is to make streets more secure, with less violations, working along with authorities.
All users help each other to know better the street-violation view of the territory.
The system allows both private customer and authority to use the application. They are allowed to sign up entering a username, password and data.
The user is now signed in and can start using the application functions. For the reporting system function, on the main menu it's possible to click either on "Make a reporting" or "Camera". The first option allows to make any type of street reporting (accident, speed

violation, ecc), while the second is specifically for parking reportings (remember that a picture can be sent only for parkings). The user can now insert all the information about the violation he wants to report. Mandatory information are: type of violation, geographical position, which can be either typed in as an address or chosen on the map, and time (license plate not mandatory).
Before the upload of the reporting, the system asks if the reporting wants to remain anonymous, this means that, when the reporting is selected on the map by a generic customer, the username of the user that made it will not be shown.
Now the user can finish the reporting, unloading it on the application. SafeStreets will store the reporting and update the map.

2.2.2 Map management

On the map are shown colored areas, based on the frequency in which the violations occur. By default, the map of all types of violations that occurred during last month is shown. The user can select a specific type of violation and interval of time in which he is interested in watching the map, for example, parking violations that occurred today.
The user can click on a specific colored area, the map will zoom in and the spots of single violations is now on the map. The user can select a single violation, the information about it will pop up (time, position, license plate and picture if present, username if not anonymous).

2.2.3 Authorities management

An authority can sign in inserting its authority ID and email address. Authorities, in addition of all the base functions that a private user has, can also send and receive information to/from SafeStreets.
Authorities can only send information about accidents, when this happens, SafeStreets stores the information and updates the map.
SafeStreets sends not only reliable parking violations to authorities, but also suggestions. As previously mentioned, when an area exceeds a certain number of a specific type of parking infringement, a suggestion is made and sent to authority.

2.3 User characteristics

- **User**: a person who is registered in SafeStreets and is allowed to use its services.

- **License plate algorithm**: the algorithm that runs on the SafeStreet data center, used to analyze the pictures uploaded along with parking reportings, in order to recognize the vehicle license plate.

- **Authorities**: police departments and municipality. They can use basic SafeStreets services in addition of specific services for authorities (information cross).

- **Violation**: a street infringement that can be reported on SafeStreets. The possible violations are parking violation, speed violation, traffic light violation, accident.

- **Reporting**: a description of the street violation, which is sent to SafeStreets.

2.4 Assumptions, dependencies and constraints

- [D1] The usernames used in the system are unique to every user.

- [D2] The device used by the users on which SafeStreets runs, has a camera.

- [D3] The device used by the users on which SafeStreets runs, can provide an accurate GPS signal.

- [D4] The license plate algorithm works.

- [D5] The user inserts the right type of violation.

- [D6] The user does not use the application for fun use (making fake reportings).

- [D7] The reportings are reliable.

- [D8] The authority ID is unique.

- [D9] The authority sends reliable information.

- [D10] The internet connection works properly without failure.

- [D11] The device used by the users on which SafeStreets runs, have internet connection (2G/3G/4G/5G or WiFi).

- [D12] SafeStreets application does not crash.

- [D13] The geographical position of the violation, specified by the user, is accurate.

- [D14] The time of the violation, specified by the user, is accurate.

3. Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

The following mockups represent a basic idea of what the mobile and web app will look like in the first release.

The web mockup shows the authority downloading some data from SafeStreets.

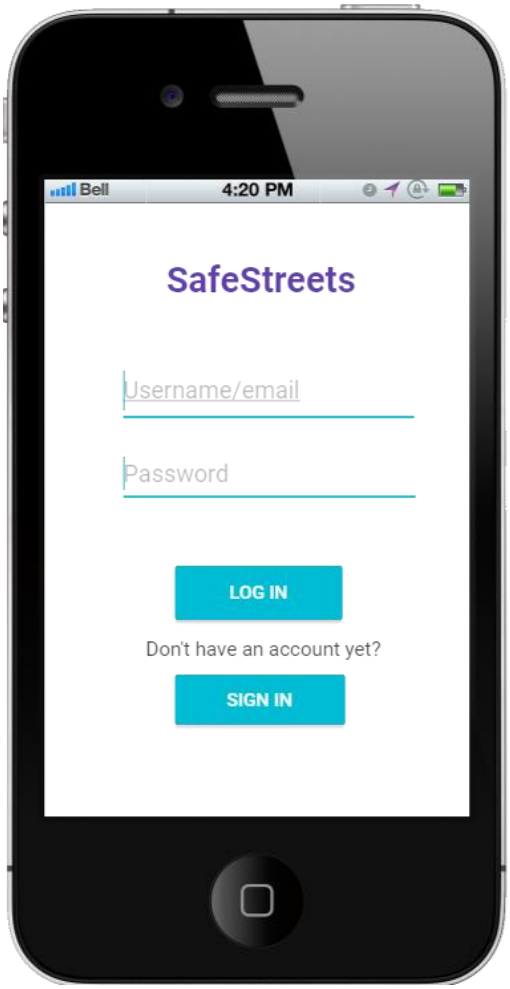


Figure 5 – Mockup: Login

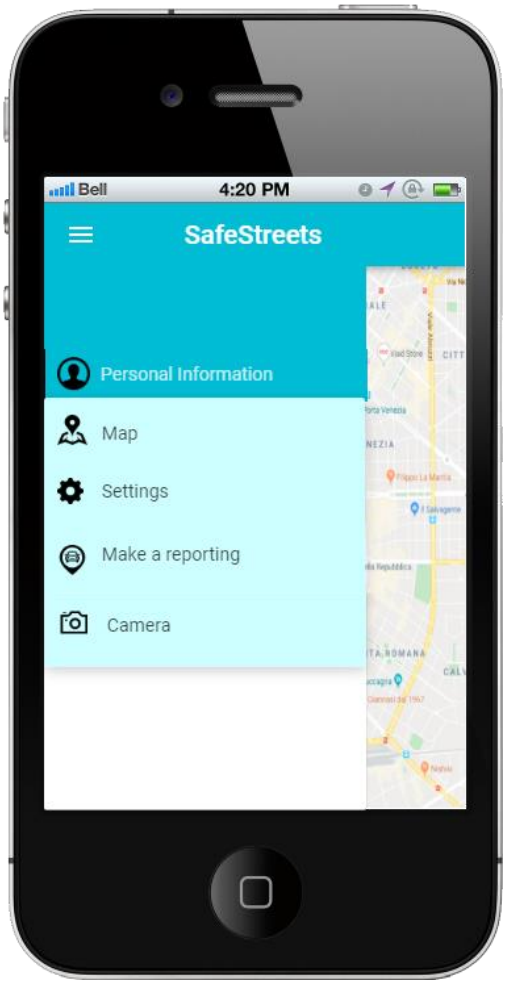


Figure 6 - Mockup: Menu

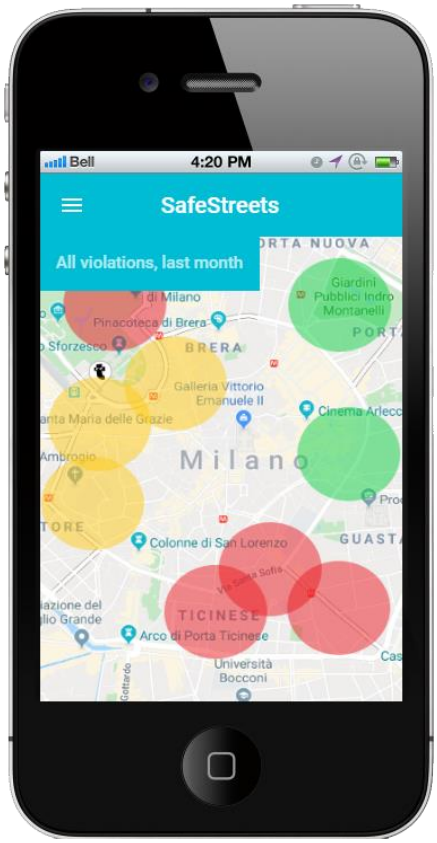


Figure 7 – Mockup: Default map

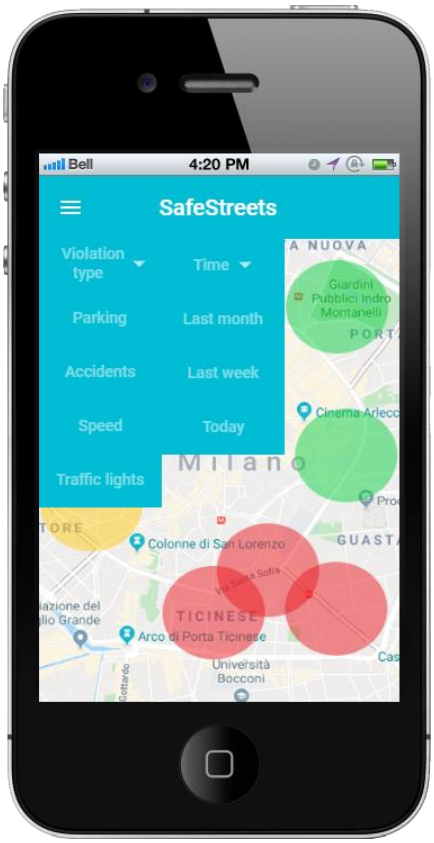


Figure 8 – Mockup: Options on map

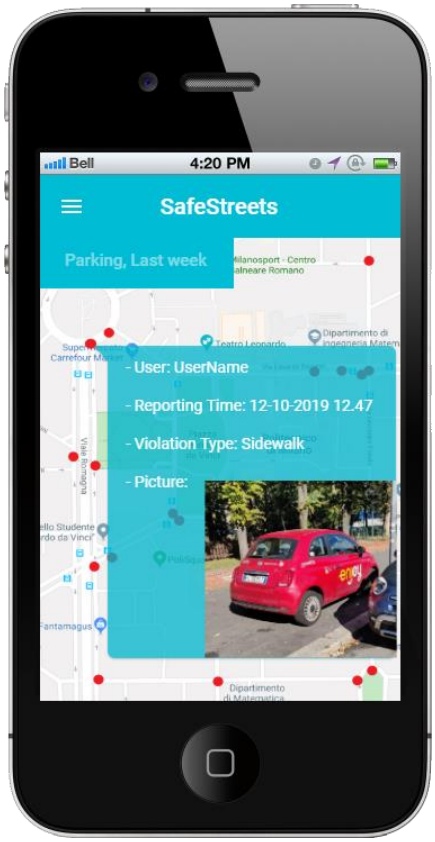


Figure 9 – Mockup: Reporting on map

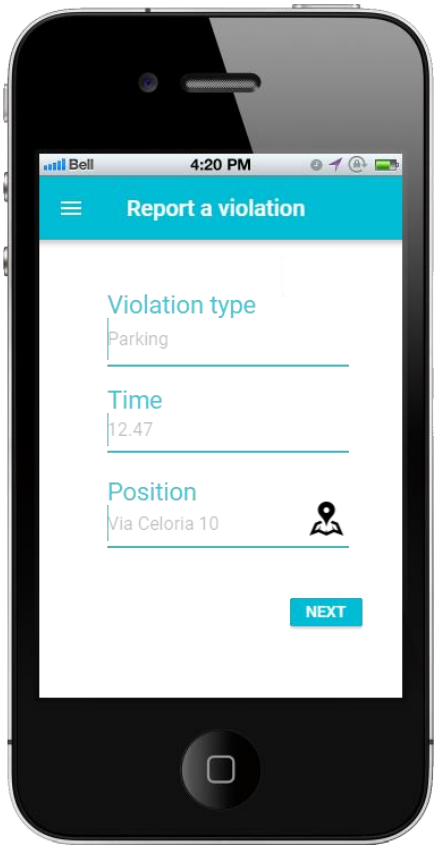


Figure 10 – Mockup: Violation reporting

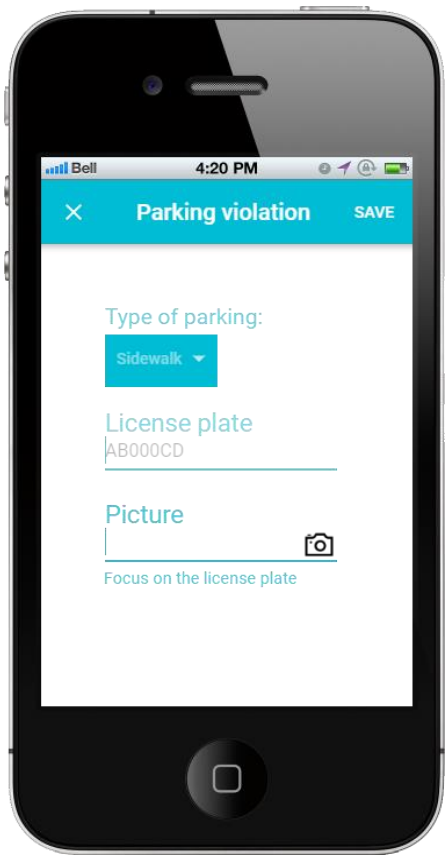


Figure 11 – Mockup: Reporting options

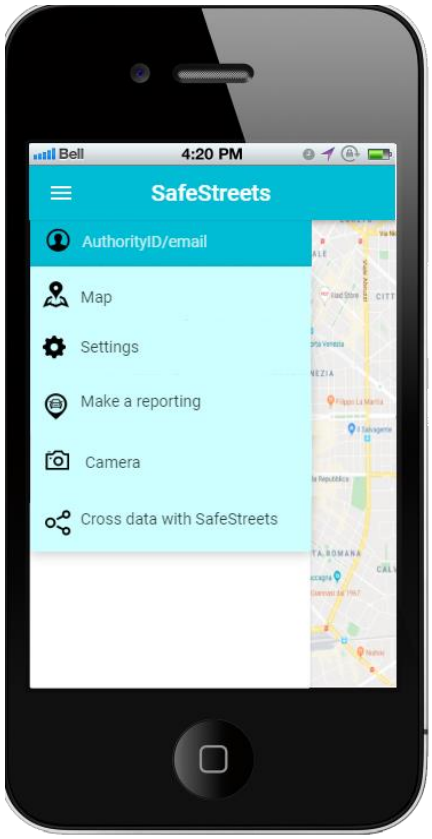


Figure 12 – Mockup: Authorities menu

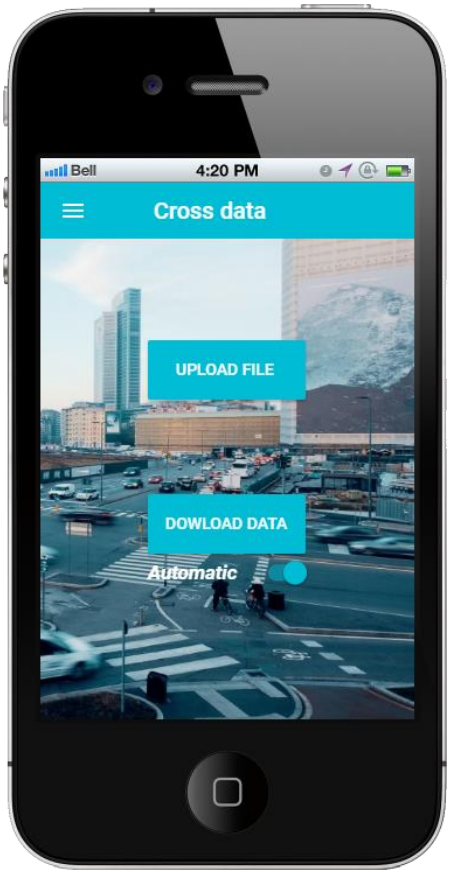


Figure 13 – Mockup: Authorities data cross

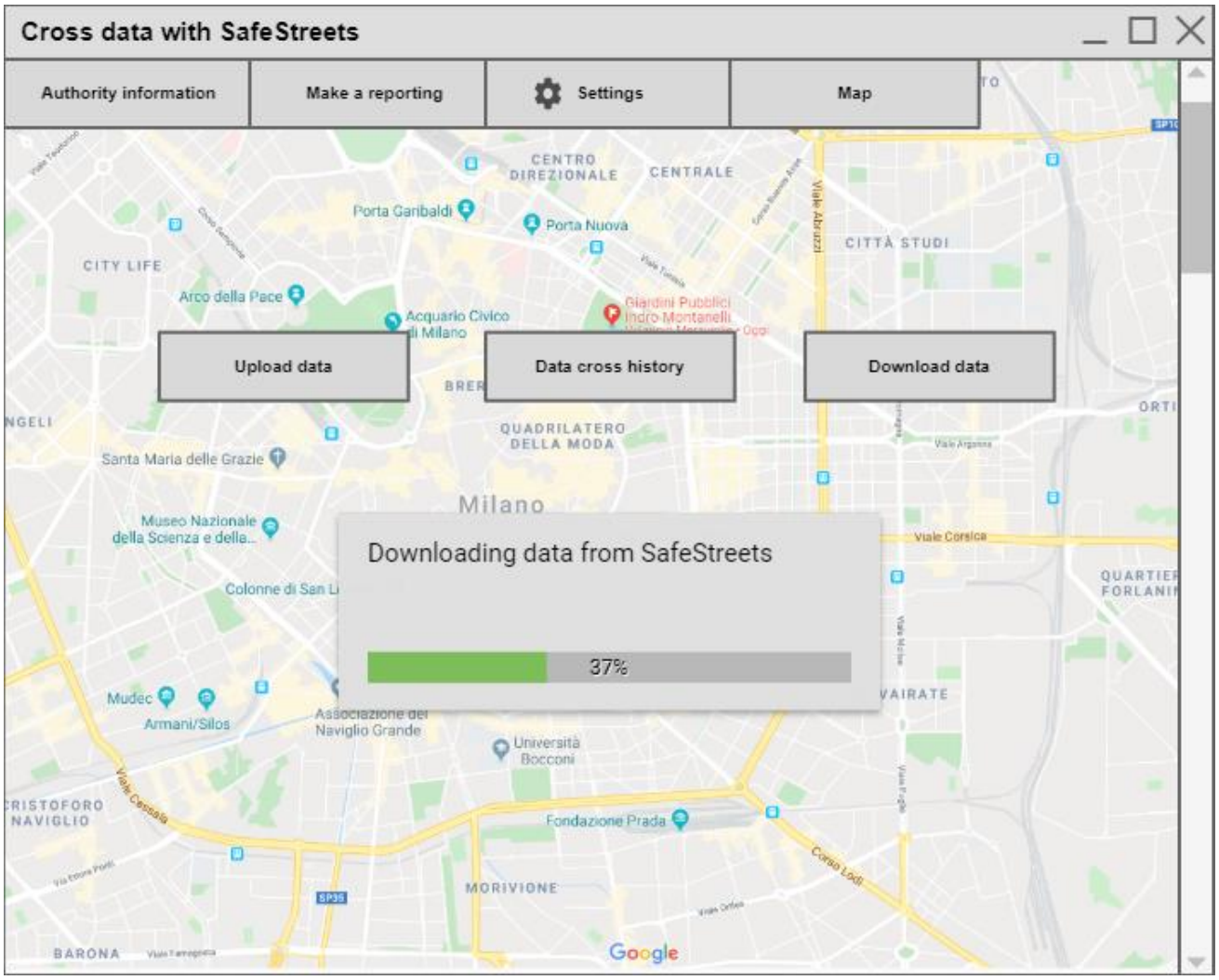


Figure 14- Mockup: SafeStreets web app: authority downloading data

3.1.2 Hardware interfaces

The system has a application server and a database server on which the information is stored, and the license plate algorithm runs.
The users need a smartphone with internet connection in order to use the SafeStreets application. The authorities can access the system through the web application from their pc.

3.1.3 Software interfaces

The system does not provide any API to external application.

3.2 Functional Requirements

3.2.1 Private

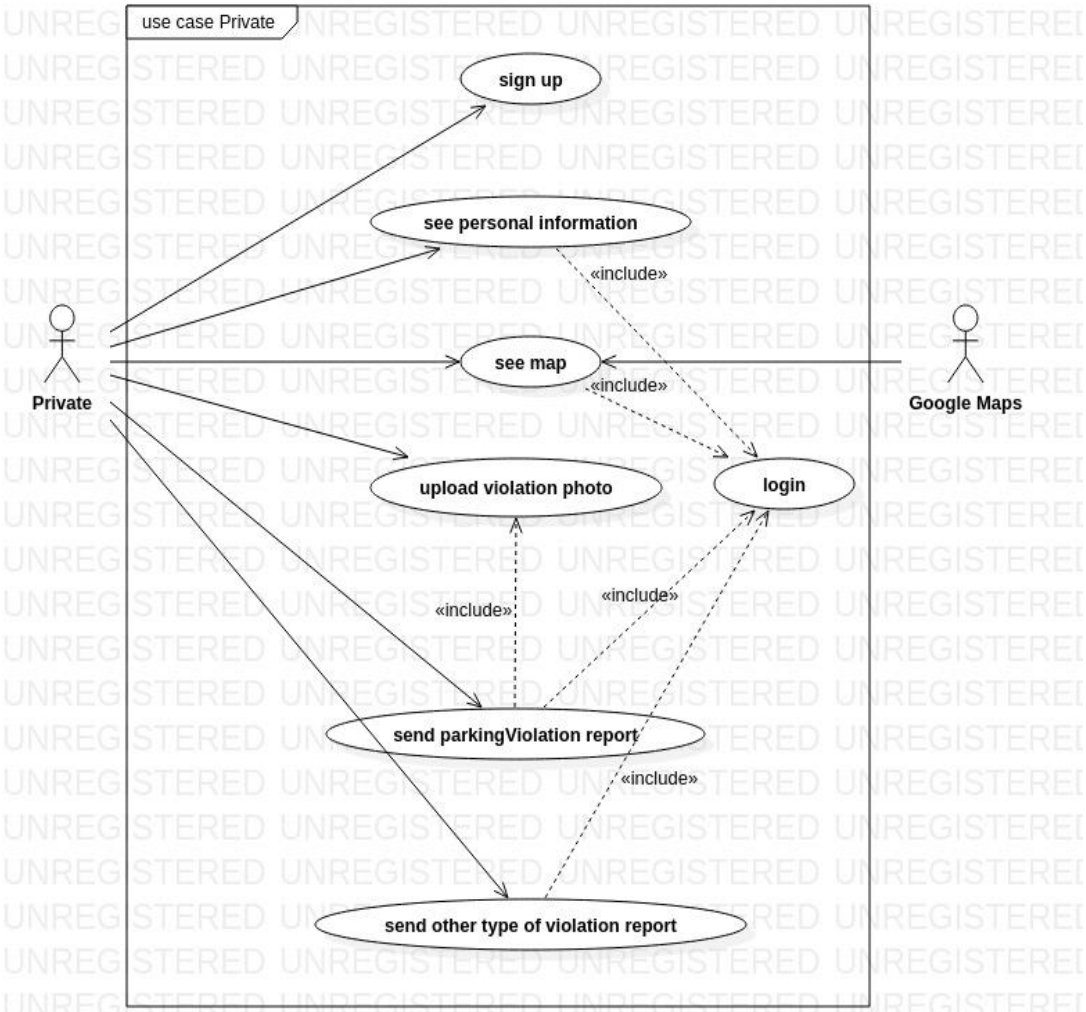
- **Scenario 1**

  Mhysa, a foreign student studying Computer Science at Politecnico di Milano, needs to go to the student office to bring some documents. She has a great sense of civic duty, so on her way from Lambrate to the office she notices a car parked on the crosswalk impeding an elderly woman with the groceries to cross the street safely. Mhysa decides to report the car with a parking violation. She takes a picture of the vehicle and types the license plate on the reporting she's about to send. Then shares her GPS position (which agreed to share at her signup) and time, deciding to remain anonymous for this reporting. Finally, she chooses crosswalk as type of parking violation.

- **Scenario 2**

  Alfonso lives in the Navigli area in Milan. On Saturday he has an exam at Politecnico di Milano Leonardo, in Città Studi. He knows there is a public transportation strike organized during the weekend, so he chooses to go by car at the exam. He first looks at Google Maps which gives him two possible alternative routes of the same duration. So, to decide which one to take, he opens SafeStreet application on his smartphone and takes a look at the map. He knows that SafeStreet colors the map based of the violations reported from users therefore he selects accidents as type of violation, and today as time interval. He notices there are reportings accidents on one route he could have possibly chosen. So, he decides to take the other route, the one without any accidents reported.

Use Case Diagram



Use Cases

Name	Sign Up
Actor	Private
Entry condition	The user has opened the application on his smartphone.
Events flow	1. The user chooses the “sign up” option. 2. The user fills all the mandatory fields and provide the necessary information. 3. The user clicks on confirm option. 4. The system checks all the user fields.

	5. The system saves the user data. 6. The system asks to the user if he wants to share his GPS position. 7. The system keeps track of user position if he accepts the request.
Exit conditions	The user is registered, he is able to use the application.
Exceptions	1. The user is already registered. The system suggests the user to do the login. 2. The user inserts not valid information in one or more fields. 3. The username is already taken. 4. The mail is already taken. Except point 1, the system handles other exceptions returning at the start of point 2 of the events flow, so the user re-enters all the fields (mandatory or not).

Name	Login
Actor	User
Entry condition	1. The user has already downloaded the application. 2. The user has already done the “Sign up” activity.
Events flow	1. The user chooses the “Login” option. 2. The user enters his email. 3. The user enters his password. 4. The user clicks on confirmation option.
Exit conditions	The user is logged in and he can use the application services.
Exceptions	1. The user enters the wrong email. 2. The user enters the wrong password. In each case the system tells the user what field is wrong and let him re-enter the wrong field.

Name	See personal information
Actor	User
Entry condition	The user has already done the “Login” activity.
Events flow	1. The user clicks on the “Personal information” button. 2. The user selects the type of data he wants to consult. 3. The system shows all the information stored at the registration of the user.
Exit condition	The user sees what he has requested from the system.
Exceptions	None

Name	See map
Actor	User
Entry condition	The user has already done the “Login” activity.
Events flow	1. The user clicks on “See map” button. 2. The system shows the map to the user, with the default options. 3. The user selects the type of violation he is interested in. 4. The user selects the interval of time he is interested in.
Exit condition	The system shows to the user the updated map with the options chosen by the user, who can click on the area and he can sees other information.
Exceptions	None

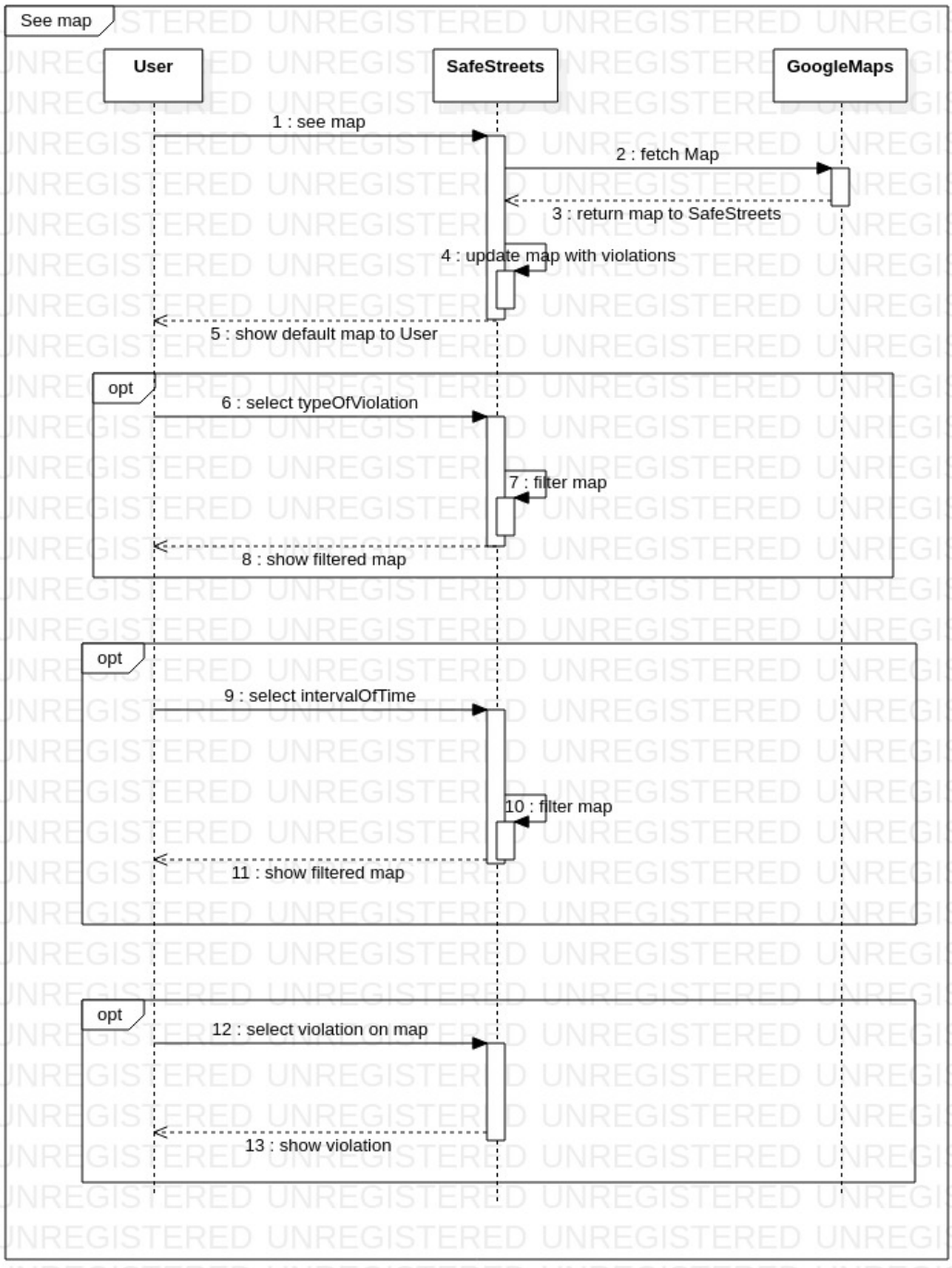
Name	Send parking violation report
Actor	User

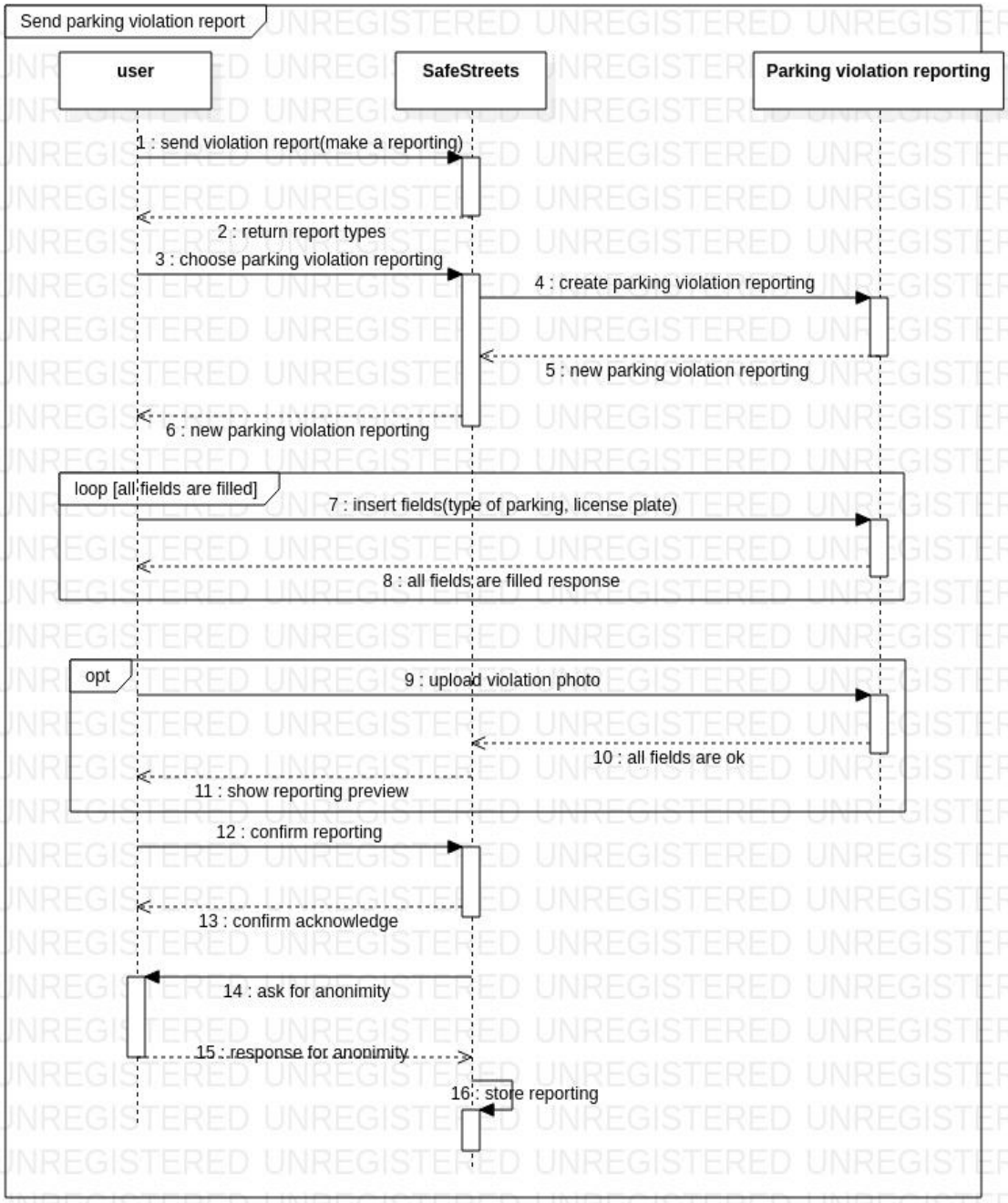
Entry condition	The user has already done the “Login” activity.
Events flow	1. The user clicks on the “send violation reporting” button. 2. The user clicks on the “parking violation” option. 3. The user selects the type of parking violation he wants to report (crosswalk, bus zone, etc). 4. The user inserts the position of the violation. 5. The user inserts the date of the violation. 6. The user inserts the license plate of the vehicle. 7. The system asks to the user if he wants to upload a picture of the violation. 8. The system asks to the user if he wants to remain anonymous.
Exit condition	The user sends the reporting to SafeStreets.
Exceptions	1. If SafeStreets can’t retrieve user position. it asks to the user to insert his position manually. 2. If SafeStreets read an invalid field, it asks to the user to re-insert the data.

Name	Send other type of violation report
Actor	User
Entry condition	The user has already done the “Login” activity.

Events flow	1. The user clicks on the “send violation reporting” button. 2. The user clicks on the “other type of violation” option. 3. The user selects the type of violation he wants to report. 4. The user inserts the position of the violation. 5. The user inserts the date of the violation. 6. The user inserts the license plate of the vehicle. 7. The system asks the user if he wants to remain anonymous.
Exit condition	The user sends the reporting to SafeStreets.
Exceptions	1. If SafeStreets can’t retrieve user position. it asks to the user to insert his position manually. 2. If SafeStreets read an invalid field, it asks to the user to re-insert the data.

Sequence diagrams




Send parking violation report
user
SafeStreets
Parking violation reporting
1 : send violation report(make a reporting)
2 : return report types
3 : choose parking violation reporting
4 : create parking violation reporting
5 : new parking violation reporting
6 : new parking violation reporting
loop [all fields are filled]
7 : insert fields(type of parking, license plate)
8 : all fields are filled response
opt
9 : upload violation photo
10 : all fields are ok
11 : show reporting preview
12 : confirm reporting
13 : confirm acknowledge
14 : ask for anonimity
15 : response for anonimity
16 : store reporting

3.2.2 Authority

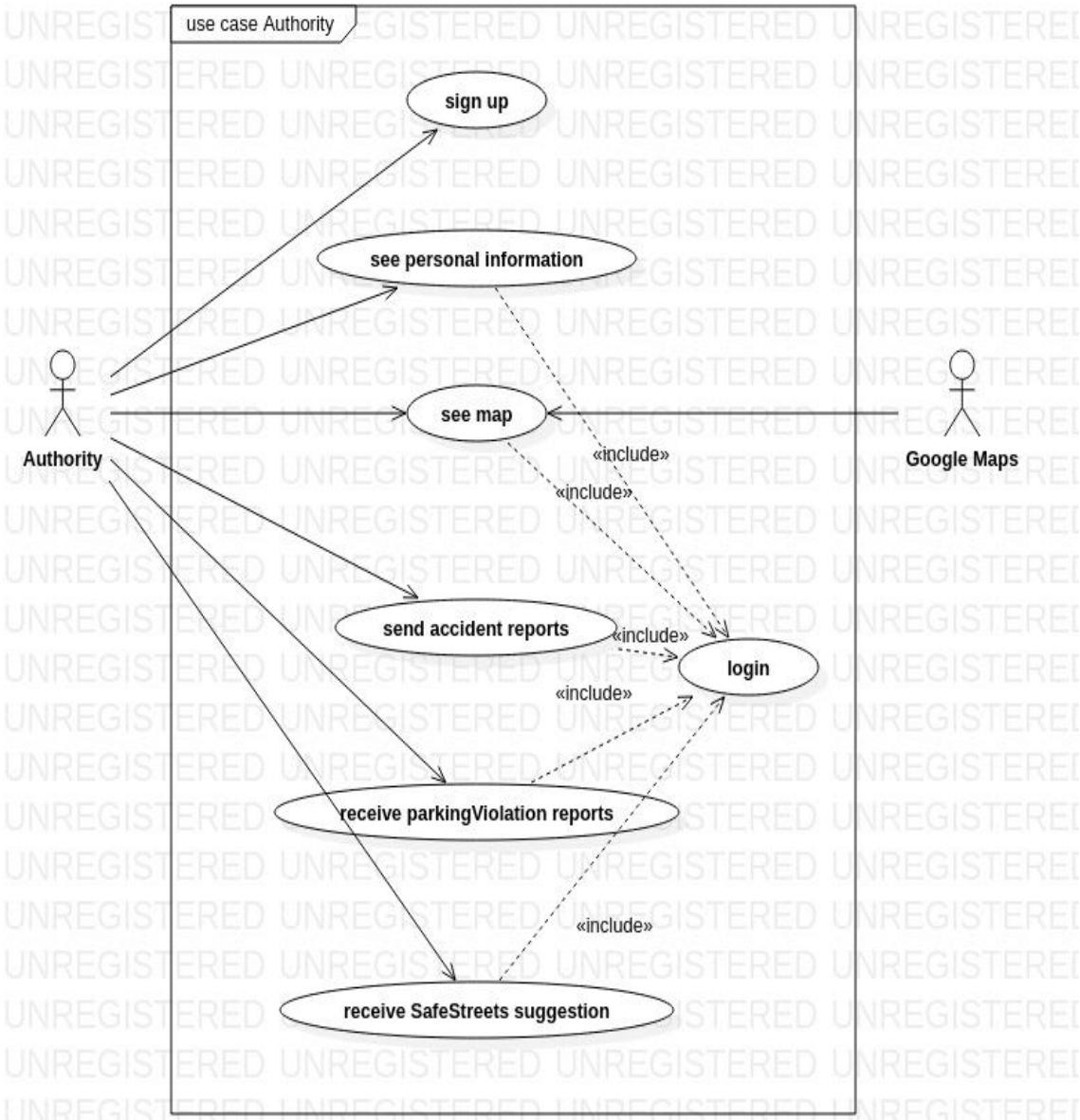
Scenario 1

Donald, an employee at the Milan municipal police, is in charge of choosing the destination for the traffic auxiliaries daily work. In order to do this, Donald consults the reports sent by SafeStreets. Then he opens the SafeStreets application.
He notices that there are three red areas on the map. He chooses parking violation, last week. The three red areas remain on the map. He decides to go there. There will be many tickets to do. Thanks to our application Donald will be helped in his decisions regarding the work of the auxiliaries.

Scenario 2

SafeStreets does not work with the municipality yet, its launch was only a week ago. Frank, an employee at the Milano municipality, receives an email from SafeStreets, in which was asked to work together and cross information in order make things work better and more efficiently. Frank understands the importance of crossing information. He decides, with the approval of the head of the municipality, to share the data with the application. After just a week Frank is happy about the decision. SafeStreets help the municipality very much. Frank has a salary increase.

Use Case Diagram



Use Cases

Name	Sign up
Actor	Authority
Entry condition	The authority has opened the application

	on his working device.
Events flow	1. The authority inserts the “Sign up” option. 2. The authority clicks on “Authority” button. 3. The authority enters his department code. 4. The authority enters his department email. 5. The authority chooses the “Confirm” option.
Exit condition	The authority is registered and can use the application.
Exceptions	1. The authority is already registered. The system suggests the authority to do the login. 2. The authority inserts not valid information in one or more fields. 3. The mail is already taken. The excpetions are handled returning to the starting point of the “Sign Up” option.

Name	Login
Actor	Authority
Entry condition	The user is already registered in the system.
Events flow	1. The user clicks on the “login” option. 2. The user inserts his username/email. 3. The user clicks on the confirmation option.
Exit condition	The user is logged in and can use the application services.
Exceptions	1. The user inserts the wrong department code. 2. The user inserts an email that

	does not correspond to the one stored under the department. These exceptions are handled by allowing the user the possibility to reenter the wrong field.

Name	See personal information
Actor	Authority
Entry condition	The authority has already done the "Login" activity.
Events flow	1. The authority clicks on the personal information (authorityID/email) button. 2. The system shows all the authority information.
Exit condition	The user sees what he has requested from the system.
Exceptions	None

Name	See map
Actor	Authority
Entry condition	The user has already done the "Login" activity.
Events flow	1. The user clicks on the "Map" option. 2. The system shows the map with the default option. 3. The user selects the type of violation he is interested in. 4. The user selects the interval of time he is interested in.
Exit condition	The system shows to the user the updated map with the options chosen by the user, who can click on the area and he can see other type of information.

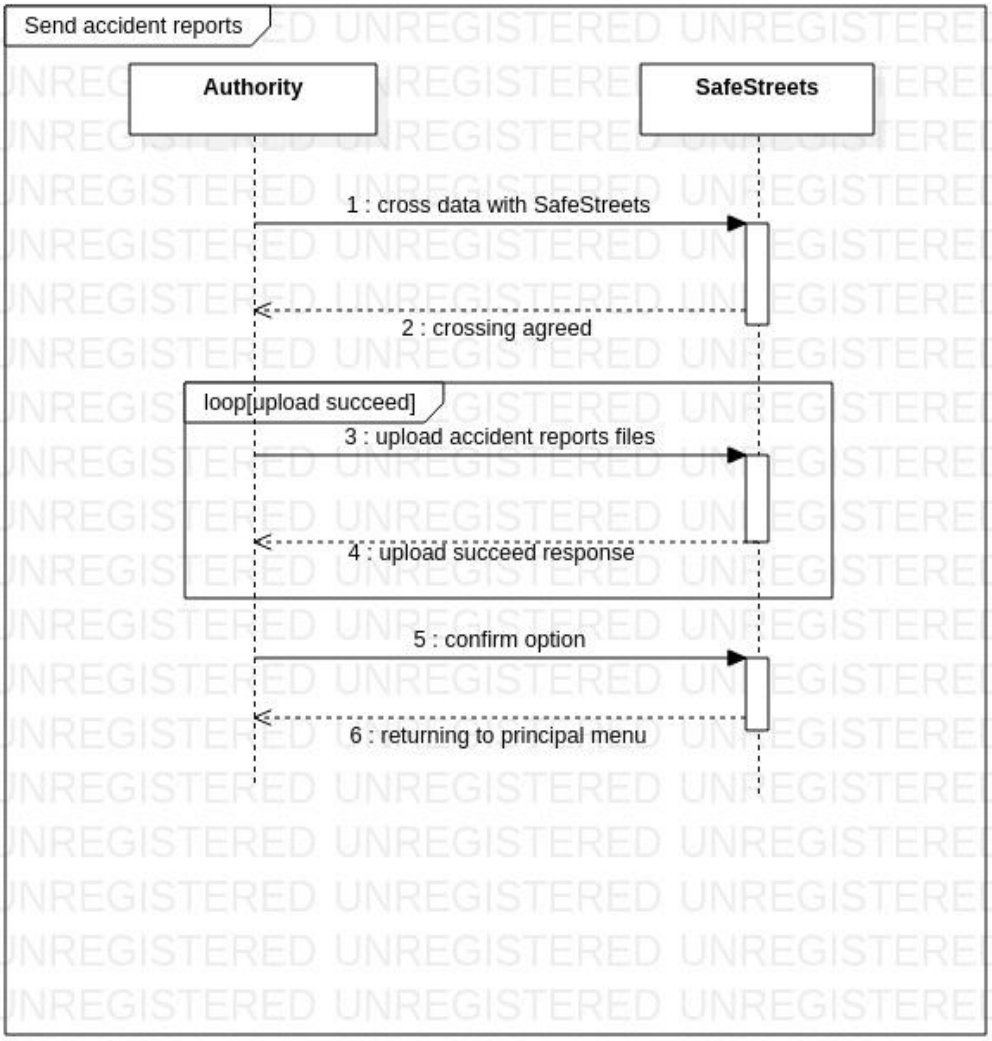
Exceptions	None

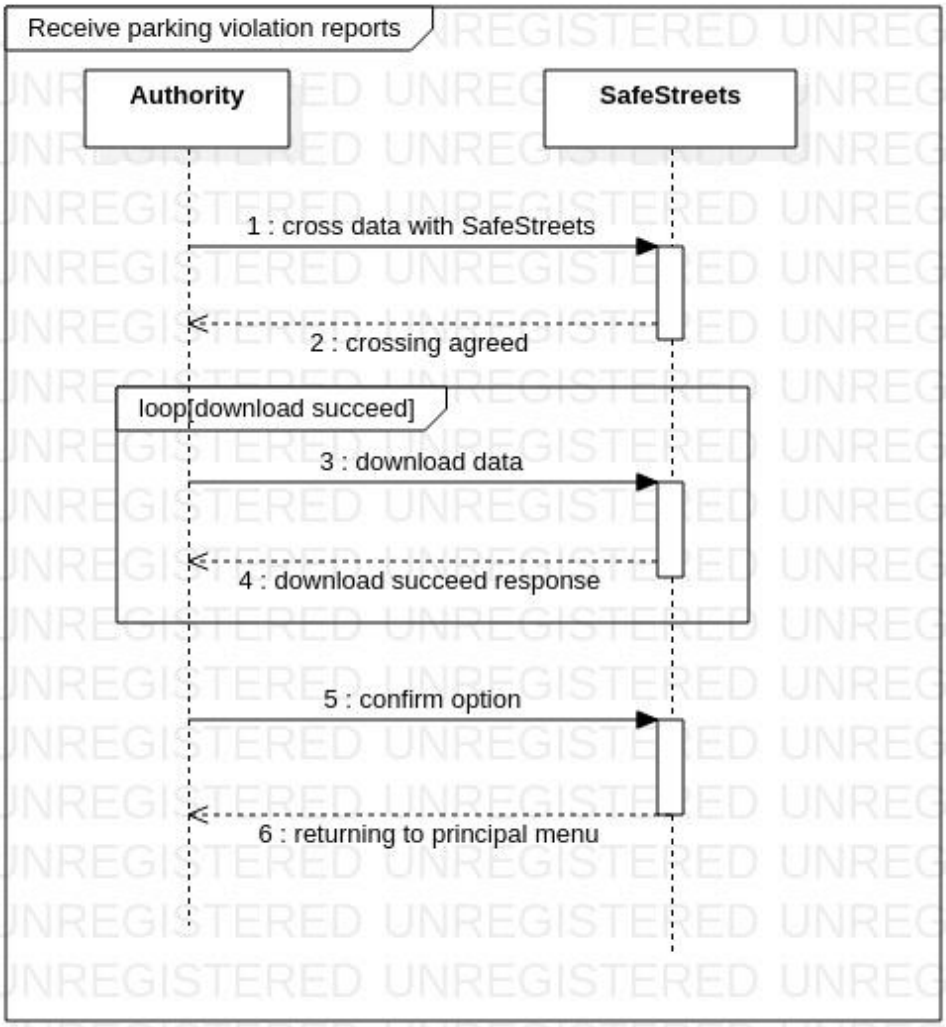
Name	Send accident reports
Actor	Authority
Entry condition	1. The user has already done the "Login". 2. The user has the permits to send the accident reports.
Events flow	1. The user clicks on the "cross data with SafeStreets". 2. The user selects the reports he wants to send by clicking on "UPLOAD FILE". 3. The user selects the confirmation option.
Exit condition	The reports are sent to SafeStreets.
Exceptions	The user doesn't select any report. So, the system warns the user to redo the operation.

Name	Receive parking violation Reports
Actor	Authority
Entry condition	1. The reports are sent periodically from SafeStreets to the municipal police. 2. The authority has already done the "Login".
Events flow	1. The user clicks on the "cross data with SafeStreets". 2. The user selects the reports he wants to download by clicking on

	"DOWNLOAD DATA". 3. The user selects the confirmation option.
Exit condition	The reports are downloaded correctly
Exceptions	The reports information are damaged, in these case there are two operations that the authorities can do. One is to reboot the system and try to download the reports again. The other is to contact SafeStreets which will give them suggestions on how to download the reports manually.

Sequence Diagrams





3.2.3 Requirements

[G1] The application must allow a visitor to become a registered user after providing credentials.

- [R1] A visitor must be able to begin the registration process. During the process the system will ask him/her to provide credentials.
- [D1] The usernames used in the system are unique to every user.
- [D8] The authority ID is unique.

- [D10] The internet connection works properly without failure.
- [D12] Safe Streets application does not crash.

- [G2] The application must allow users to send reports of streets violations.
  - [R1] A visitor must be able to begin the registration process. During the process the system will ask him/her to provide credentials and payment information.
  - [R2] The user can log in to the application by providing the combination of a username and a password that match an account.
  - [R3] The systems must allow the user to compile the reporting form with all the information.
  - [R4] The system must allow the user to see his violation reporting immediately after he confirms to send the reporting to SafeStreets.
  - [R5] The systems must allow user to remain anonymous after the reporting.
  - [R6] The system must allow a user who send a reporting both to share his position through GPS and to select his position on the map manually.
  - [R7] The application will have to store the information about violations and complete them with suitable metadata.
  - [D2] The device used by the users on which SafeStreets runs, has a camera.
  - [D3] The device used by the users on which SafeStreets runs, can provide an accuret GPS signal.
  - [D10] The internet connection works properly without failure.
  - [D11] The device used by the users on which SafeStreets runs, have internet connection (2G/3G/4G/5G or WiFi).

- [D11] The device used by the users on which SafeStreets runs, have internet connection (2G/3G/4G/5G or WiFi).

- [D13] The geographical position of the violation, specified by the user, is accurate.

- [D14] The time of the violation, specified by the user, is accurate.

- [G3] The application must allow both end users and authorities to mine the information stored. This will be done by selecting areas and reports on the map.

  - [R2] The user can log in to the application by providing the combination of a username and a password that match an account.

  - [R3] The systems must allow the user to compile the reporting form with all the information.

  - [R4] The system must allow the user to see his violation reporting immediately after he confirms to send the reporting to SafeStreets.

  - [R7] The application will have to store the information about violations and complete them with suitable metadata.

  - [R8] The system must be able to color the map based on the number of violations occurred in each zone.

  - [R9] The system must update the map after every reporting.

  - [R10] The system must be able to cross information received from municipality with its own data.

  - [D7] The reportings are reliable.

  - [D10] The internet connection works properly without failure.

  - [D11] The device used by the users on which SafeStreets runs, have internet connection (2G/3G/4G/5G or WiFi).

- [D12] SafeStreets application does not crash.

- [G4] The system must send suggestions to authorities to minimize the illegal parking frequency in certain areas. When an area is red for a specific type of parking violation a suggestion is made.
  - [R2] The user can log in to the application by providing the combination of a username and a password that match an account.
  - [R3] The systems must allow the user to compile the reporting form with all the information.
  - [R7] The application will have to store the information about violations and complete them with suitable metadata.
  - [R8] The system must be able to color the map based on the number of violations occurred in each zone.
  - [R9] The system must update the map after every reporting.
  - [R11] the system must be able to generate suggestions for each type of violation parking.
  - [D4] The license plate algorithm works.
  - [D5] The user inserts the right type of violation.
  - [D6] The user does not use the application for fun use (making fake reportings).
  - [D10] The internet connection works properly without failure
  - [D12] SafeStreets application does not crash.
  - [D13] The geographical position of the violation, specified by the user, is accurate.
  - [D14] The time of the violation, specified by the user, is accurate.

- [G5] The system must be able to send the reliable parking violation to the municipality.
    - [R2] The user can log in to the application by providing the combination of a username and a password that match an account.
    - [R3] The systems must allow the user to compile the reporting form with all the information.
    - [R5] The systems must allow user to remain anonymous after the reporting.
    - [R6] The system must allow a user who send a reporting both to share his position through GPS and to select his position on the map manually.
    - [R7] The application will have to store the information about violations and complete them with suitable metadata.
    - [R10] The system must be able to cross information received from municipality with its own data.
    - [D4] The license plate algorithm works.
    - [D5] The user inserts the right type of violation.
    - [D6] The user does not use the application for fun use (making fake reportings).
    - [D7] The reportings are reliable.
    - [D8] The authority ID is unique.
    - [D10] The internet connection works properly without failure.
    - [D12] SafeStreets application does not crash.
- [G6] The system must allow authorities to send accidents information to SafeStreets.
    - [R7] The application will have to store the information about violations and complete them with suitable metadata.

- [R8] The system must be able to color the map based on the number of violations occurred in each zone.
- [R9] The system must update the map after every reporting.
- [R10] The system must be able to cross information received from municipality with its own data.
- [D9] The authority sends reliable information.
- [D10] The internet connection works properly without failure.
- [D12] SafeStreets application does not crash.

Traceability Matrix

R1	Sign Up
R2	Login
R3	Send parking violation report Send other type of violation report
R4	Send parking violation report Send other type of violation report See map
R5	Send parking violation report Send other type of violation report
R6	Send parking violation report

	Send other type of violation report
R7	See map
R8	See map
R9	See map
R10	See map Send parkingViolation reports Receive SafeStreets suggestions
R11	Receive SafeStreets suggestions

3.3 Performance Requirements

The application must be able to save data and violations on SafeStreet data center (a database). It also must be able to analyze the data by querying the database. Our software must be able to send and receive specific type of data in order cross information with authorities. In particular, it must adapt to legacy systems managed by the public administration and smartphone devices.

3.4 Design constraints

3.4.1 Standards compliance

- The application provides a daily incremental backup and a full weekly back up.
- The data are stored with regard of the privacy of data and the accesses of system administrators are monitored. The entire project is subject to the General Data Protection Regulation (GDPR), a regulation in EU law on data protection and privacy for all individuals within the EU and the EEA (European Economic Area).
- The reports received from authorities must have a specific format in order to analyze the data and update the map.

3.4.2 Hardware limitations

This is a software application which works with a smartphone that can use GPS services for users. This software doesn't depend from any platform.
The smartphone must have an internet access and can provide GPS service (not mandatory).

3.4.3 Other costraints

The system must respect privacy policy according to actual GDPR law. In particular, when a user agrees to share his name after a reporting it will appear on SafeStreets map. So, we will only report the username to avoid retaliation.
E-mail addresses will not be used for commercial uses.

3.5 Software system attributes

3.5.1 Reliability

The application requires the use of fault tolerance and balancing procedures in order to avoid service interruptions.
The data are duplicated in cluster environments, to avoid accidental loss of data and to guarantee the continuity of the service provided 24/7.

3.5.2 Availability

The application as described in the previous section is available to the user 24/7 and there is a customer care system for those who encounter problems on the user side.

3.5.3 Security

The security protocols provide for the encryption of the stored data.
An encrypted communication protocol between client and server must be used to avoid Man in the Middle and other types of attack.

3.5.4 Maintainability

The application for future developments that it envisages has been designed to be easy to maintain, in order to instruct future teams that will follow its evolution.

3.5.5 Compatibility

The application adapts to the various systems on the smartphone market (Android, IOS, etc).

4. Formal analysis using Alloy

```
open util/integer

//-------------------------------------------------------------SIGS-----------------------------------------------------

// Date and time of every reporting
sig TimeStamp{}

//Position of reportings and users
sig Position{

        latitude: one Int,
        longitude: one Int
}
//latitude >= -90 and latitude <= 90 and longitude >= -180 and longitude <= 180
{latitude >= -3 and latitude <= 3 and longitude >= -6 and longitude <= 6}

//String, name of private or authority
abstract sig Username{}


//String, only authotities have one
sig AuthorityID extends Username{}

//String, only private users have one
sig PrivateUsername extends Username{}

//Both privates and authorities have one
sig Email{}

//Both privates and authorities have one
sig Password{}

//String, every vehicle has one
sig LicensePlate{}

sig Vehicle{

    licensePlate: one LicensePlate
}
```

```
//Picture of parking reportings
sig Picture{}

sig Registration{

	password: one Password,
}

//Privates must make a Registration
sig PrivateRegistration extends Registration{

	privateUsername: one PrivateUsername,
	GPS_RequestResult: one Request_Result
}


//Authoritues must make a Registration
sig AuthorityRegistration extends Registration{

	authorityID: one AuthorityID
}

sig ReportingID{}

//Any tipe of reporting
abstract sig Reporting{

	reportingID: one ReportingID,
	position: one Position,
	timestamp: one TimeStamp,
	username: one Username,
	anonimity: one Anonimity_Request_Result
}


sig ParkingReporting extends Reporting{

	picture: lone Picture,
	licensePlate: lone LicensePlate
}

sig AccidentReporting extends Reporting{

	numCarsInvolved: Int,
	carsInvolved: set LicensePlate
}

sig SpeedReporting, TrafficLightReporting extends Reporting{

	licensePlate: lone LicensePlate
}
```

```
//User can be both private or authoritie
sig User{

	registration: one Registration,
	reportings: set Reporting,
	position: lone Position,
	email: one Email,
	see : Map
}

sig Private extends User{

	privateUsername: one PrivateUsername,
}

sig Authority extends User{

	authorityID: one AuthorityID,
	downloadedData: set ParkingReporting,
	uploadedData: set AccidentReporting
}


//Made at the sign up
sig GPS_Request {

	subjectOfRequest: one User
}

//Result of the GPS request
abstract sig Request_Result{

    request: one GPS_Request
}

one sig GPS_Request_Accepted extends Request_Result{}

one sig GPS_Request_Refused extends Request_Result{}

one sig Map{

	reportings: set Reporting
}
```

```
//made for every reporting
sig Anonimity_Request{

	reporting: one Reporting
}

abstract sig Anonimity_Request_Result{

	request: one Anonimity_Request
}

sig AnonimityTrue extends Anonimity_Request_Result{}

sig AnonimityFalse extends Anonimity_Request_Result{}
```

```
//------------------------------------------------------------ FACTS------------------------------------------------------------

//All PrivateUsername have to be associated with a Private
fact UserNamePrivateConnection{

	all u: PrivateUsername | some p: Private | u in p.privateUsername
}

//All reportings are unique
fact UniqueReportings{

	no disjoint rep1, rep2: Reporting | rep1.reportingID = rep2.reportingID
}

//All AuthorityID have to be associated with a Authority
fact AuthorityIDAuthorityConnection{

	all ID: AuthorityID | some a: Authority | ID in a.authorityID
}



//All PrivateUsernames have to be associated to a PrivateRegistration
fact UsernameRegistrationConnection{

	all u: PrivateUsername | some p: PrivateRegistration | u in p.privateUsername
}

//All AuthorityID have to be associated to a AuthotityRegistration
fact AuthotityIDRegistrationConnection{

	all ID: AuthorityID | some a: AuthorityRegistration | ID in a.authorityID
}


//All Passwords have to be associated to a Registration
fact PasswordRegistrationConnection{

	all p: Password | some r: Registration | p in r.password
}
```

```
//Every request can be either accepted or refused, not both.
fact GPS_requestAcceptedOrRefused{

	all GPS: GPS_Request | (some rr: Request_Result  | GPS in rr.request) and
	(no disjoint rr1, rr2: Request_Result | GPS in rr1.request and GPS in rr2.request)
}

//All Reportings have to be associated with one and only one user
fact ReportingUniqueUser{

	all r: Reporting | some u: Username | u in r.username
}


//All reportings on map have to be anonymous or not, not both
fact ReportingAnonymousOrNot{

	all a: Anonimity_Request | (some arr: Anonimity_Request_Result  | a in arr.request) and
	(no disjoint arr1, arr2: Anonimity_Request_Result | a in arr1.request and a in arr2.request)
}


//Every request can be either accepted or refused, not both.
fact GPS_requestAcceptedOrRefused{

	all GPS: GPS_Request | (some rr: Request_Result  | GPS in rr.request) and
	(no disjoint rr1, rr2: Request_Result | GPS in rr1.request and GPS in rr2.request)
}

//All Reportings have to be associated with one and only one user
fact ReportingUniqueUser{

	all r: Reporting | some u: Username | u in r.username
}


//All reportings on map have to be anonymous or not, not both
fact ReportingAnonymousOrNot{

	all a: Anonimity_Request | (some arr: Anonimity_Request_Result  | a in arr.request) and
	(no disjoint arr1, arr2: Anonimity_Request_Result | a in arr1.request and a in arr2.request)
}
```

```
//All GPS_request have to be associated with one user
fact GPS_requestOneUser{

	all GPS: GPS_Request | some u: User | u in GPS.subjectOfRequest
}

//All vehicles have one and only one license plate
fact vechicleUniqueLicensePlate{

	no disjoint vehicle1, vehicle2: Vehicle | vehicle1.licensePlate = vehicle2.licensePlate
}

//All reportings on the map correspond with all reportings done by the users
fact ReportingsOnMapEqualsToReportingsDone{

	all rep: Map.reportings | some reportings: User.reportings | reportings in rep
}



//There aren't two identical license plates in an accident reporting
fact DifferentLicensePlatesInAccidentReporting{

	no disjoint plate1, plate2: AccidentReporting.carsInvolved | plate1 = plate2
}

//The downloaded data from authorities correspond to parking reportings that are one the map
fact DownloadedDataPresentOnMap{

	all rep: Map.reportings | some data: Authority.downloadedData | data in rep
}

//A picture must be in one parking reporting
fact PictureInReporting{

	all pic: Picture | some p: ParkingReporting.picture  | pic in p
}
```

```
-----------------------------------------------------------ASSERTS-----------------------------------------------------------


//A reporting can't be in two positions on the map (don't exist two reportings with same reportingID with different positions)
assert NoReportingInTwoPositions{

	no disjoint rep1, rep2: Reporting | rep1.reportingID = rep2.reportingID and rep1.position = rep2.position
}

check NoReportingInTwoPositions for 5


//Two reportings can't have the same reportingID
assert ReportingsWithDifferentID{

	no disjoint r1, r2: Reporting | r1.reportingID = r2.reportingID
}

check ReportingsWithDifferentID for 5

//All Anonimity request have to be associate with one reporting
assert Anonimity_RequestOneReporting{

	all an: Anonimity_Request | some rep: Reporting | rep in an.reporting
}

check Anonimity_RequestOneReporting for 5


-----------------------------------------------------------PREDICATES-----------------------------------------------------------


pred world1{

	#Private = 1
	#Authority = 1
	#Reporting = 0

}

run world1 for 5


pred world2{

	#Private = 0
	#Reporting = 3
	#Authority = 1
}

run world2 for 3
```

```
pred world3{

        #Authority = 3
        #Private = 3
        #Reporting = 3
}

run world3 for 6
```

Asserts:

```
Executing "Check NoReportingInTwoPositions for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  13005 vars. 1335 primary vars. 23102 clauses. 24ms.
  No counterexample found. Assertion may be valid. 16ms.


Executing "Check ReportingsWithDifferentID for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  12931 vars. 1335 primary vars. 22868 clauses. 19ms.
  No counterexample found. Assertion may be valid. 12ms.


Executing "Check Anonimity_RequestOneReporting for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  12851 vars. 1330 primary vars. 22625 clauses. 20ms.
  No counterexample found. Assertion may be valid. 29ms.
```

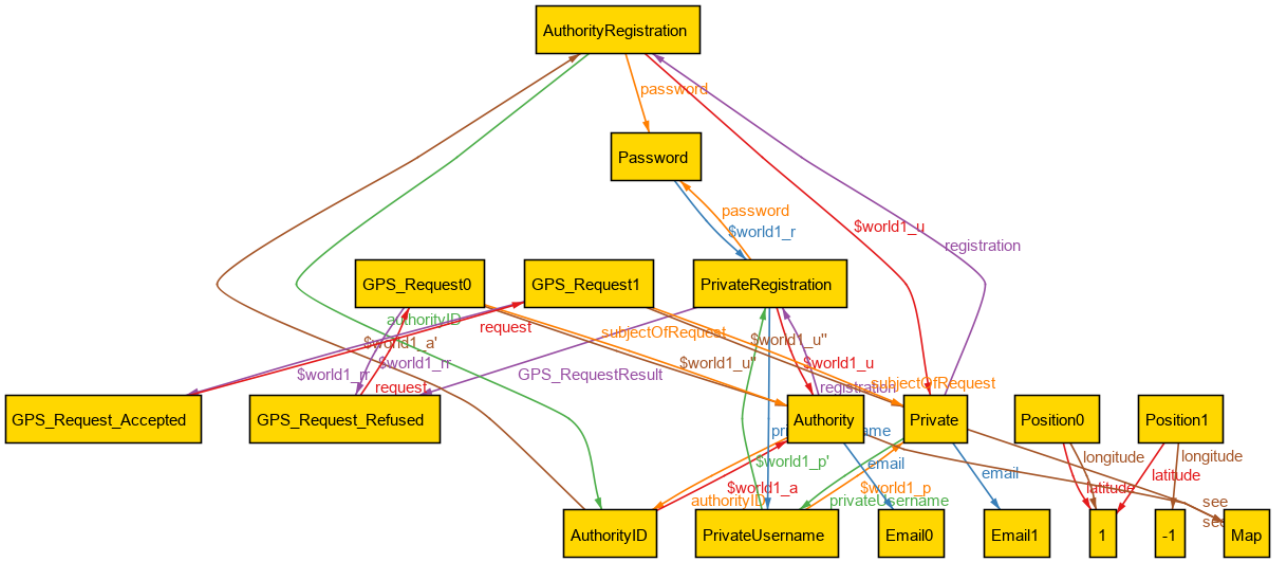
Predicates:

```
Executing "Run world1 for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  12830 vars. 1325 primary vars. 22708 clauses. 21ms.
  Instance found. Predicate is consistent. 29ms.
```

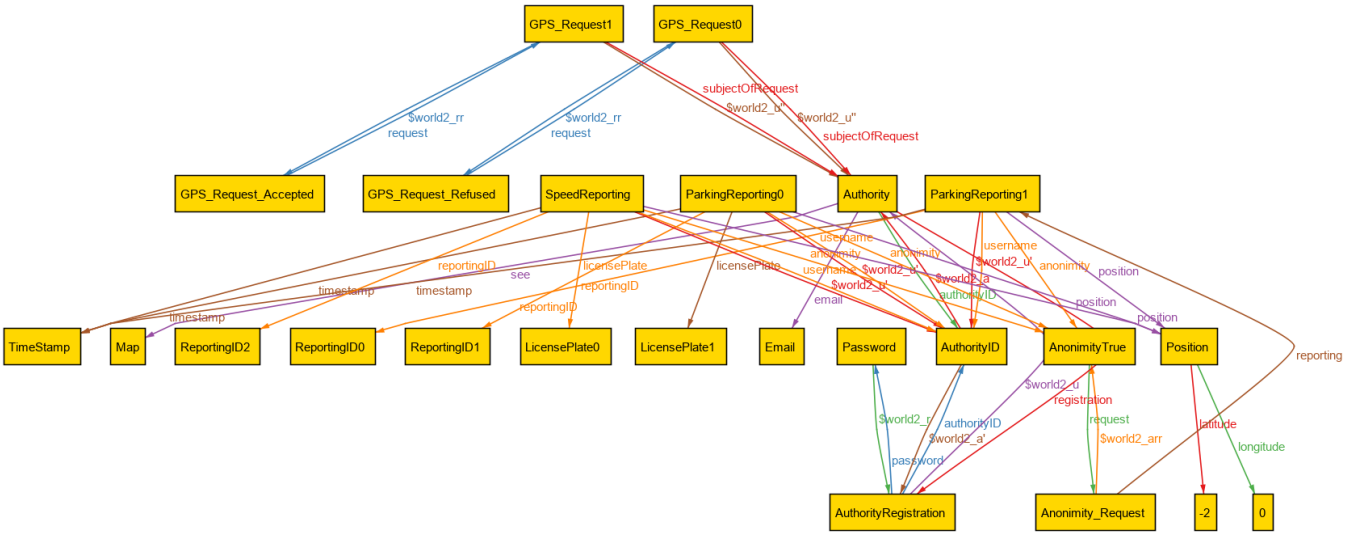
```
Executing "Run world2 for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  5502 vars. 573 primary vars. 10404 clauses. 12ms.
  Instance found. Predicate is consistent. 27ms.
```

```
Executing "Run world3 for 6"
  Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
  17781 vars. 1812 primary vars. 30600 clauses. 28ms.
  Instance found. Predicate is consistent. 47ms.
```

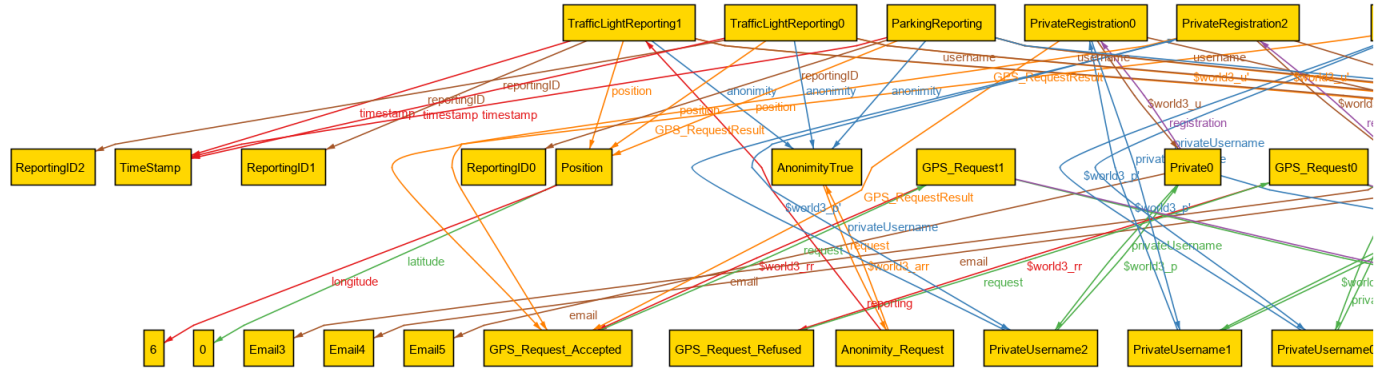
World1:

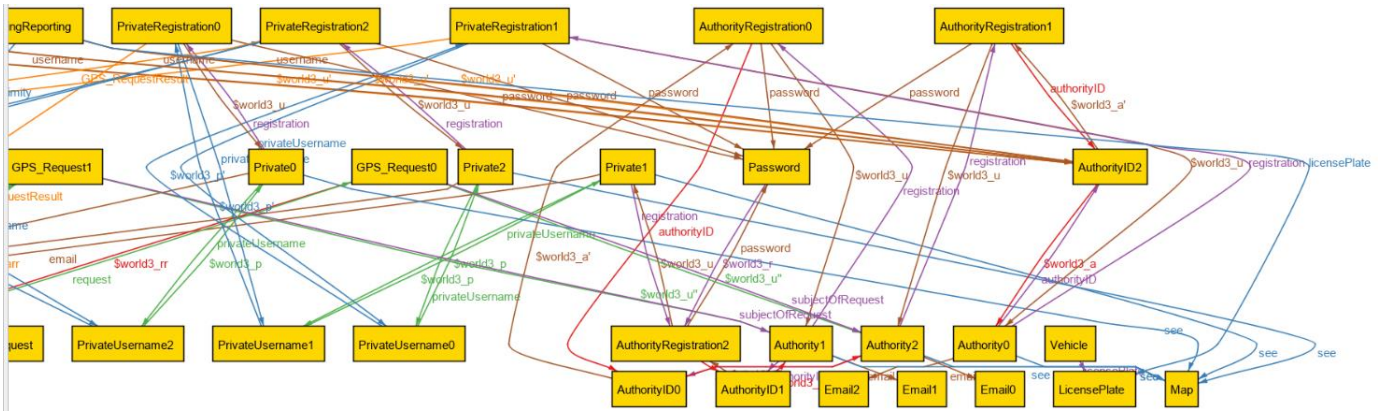


World2:



World3:





5. Effort spent

5.1 Carrioli

Section 1	11
Section 2	15
Section 3	4
Section 4	16

5.2 Ceruti

Section 1	4
Section 2	2
Section 3	11
Section 4	12