-

POLITECNICO
MILANO 1863

AA 2019/2020

RASD – Requirement Analysis and Specification Document

Version 1.2 - 09/12/19

Authors:

Giuseppe Asaro

Giuseppe Italia

Professor:

Elisabetta Di Nitto

Table of contents

1. INTRODUCTION

1.1 PURPOSE

1.1.1 GENERAL PURPOSE

The purpose of this document is to provide a detailed description of the SafeStreets system. 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.

One of the most important problem in the daily life, nowadays, is given by the traffic violations, indeed Safestreets is designed to make people's experience on streets better, but also to help authorities to identify possible traffic violations and help the municipality to improve its road safety. It is a crowd-sourced application that intends to provide users with the possibility to notify the SafeStreets's system when traffic violations occur, and in particular parking violations, but also another important issue that could be solved by the system regards the occupations of parking reserved to people with disabilities (obviously there are many other problems that the system can help solving). The application allows citizens to send pictures, description, position of violations. Through all these reports the system will build some stats, in order to be available to users that want to consult them to take some advantages on streets and, maybe, to choose the best areas in which to drive with their car, possible avoiding fines and bothers. Moreover, if the municipality will make its data about accidents available to SafeStreets, the system will provide stats more detailed involving both accidents and traffic violations so that it also will point out the most unsafe areas in the municipality field. Elaborating this stats SafeStreets will also provide some suggestions to the municipality to improve its road conditions; also the local police takes many advantages from the system indeed on the one hand it will be able to consult every report made by citizens and, in the case they are not truthful, it will be authorized to delete them; on the other hand, it will be helped, accessing to stats, to have a better view of the city so that it will take some advantages in its work knowing the best areas in which take action.

1.1.2 GOALS

1. SafeStreets must allow users to report a violation sending pictures and description of that one.
2. SafeStreets must create automatic statistics about the kind of violations occurred and the class of vehicles with the most violations; these stats must update every 2 weeks;
3. SafeStreets must create statistics providing the most unsafe areas in the municipality territory; these stats must update every 2 weeks;
4. SafeStreets must suggest possible interventions to the municipality (through information sent by municipality and users);
5. SafeStreets must allow both users and local police accessing the available stats;
6. SafeStreets must allow only police to access all reports and in case to delete them if they are not truthful.
7. SafeStreets must allow users to access to their past report.

1.2 SCOPE

This document is addressed to clients, users and to the ones who will work on SafeStreets such as project managers, software developers, testers and requirements analysts.
SafeStreets is a system that allows citizens(once logged in) and not only to hold more security own city signalizing all street violation; the system will give to the citizens the possibility to open in every moment the application on their device through the MobileApp and decides one of the actions provided: “report a violation”, “view stats”, “view past reports” and “tell some problems”; for example when the action is “Report a violation”, the citizen has to fill all forms provided by system; he

takes a photo of street violation and of the violation’s vehicle(with traffic plate visible), he localizes the location both through the inserting of the address and through the use of GPS that offers an accurate location and finally he describes the car and the violation type allowing to system to create the corresponding statistic; any description wrote above are mandatory and furthermore the system must make ensure quality of the information, for example, checking the clarity of the photo since then the system must read the traffic plate from this picture or the checking the correctness of information since if the citizen writes one street that doesn't exist the system must communicate it giving the possibility to reinsert it.
The traffic plate check is done by TrafficScanner, one algorithm that checks the clarity of traffic plate and the right correspondence with traffic plate written by the user in the Report a violation form

The citizen will ever have the possibility to view all his past report sent to SafeStreets and he can it through the way "View past reports"
SafeStreets allows both citizens and Local police to view all statistics provided by the system so information about the type of vehicle, type of violation and the information about accidents(recovered from municipality) are really important to create all stats.
All stats are available(the user can view them) if and only if the system has enough dates to analyze: the minimum number of reports to analyze is of 50 units
All stats are continuously updated to allow citizens to have good and truthful information; so every 2 weeks SafeStreets takes all report(olds and news), receives new accident information from the municipality and recreate new stats with new dates.
When one citizen or Local Police wants to view the stats for one particular area he can filter the research with the appropriate form provided by the system; the form provided is one map or one list with *X* different city's areas defaults which can be considered.
SafeStreets analysing all accident and all reports sent by users, must be able to create a suggestion to send to the municipality for interventions on the streets such as one particular problem area can improve; for example, if in a street there are many violations about the car parked on the pedestrian crossing so the suggestion could be "Arrange the barriers in the sides of the pedestrian crossing".

SafeStreets uses a particular recognition mechanism for Local Police: the system will make “Police code” available to the Local police of every city(where SafeStreets is available) through which every policeman will be recognized at the login; so SafeStreets will offer a different level of visibility to a different kind of user.
When a policeman is logged the system in addition to giving him all stats provided has to allow him to view and check all reports stored in the system and if a policeman considered not true one violation report can decide to remove it with a form provided by the system.

Local Police if signs in with mobile phone can only read the stats instead if he Local Police signs in by web he can both check and remove the reports and also view stats.
The application is available for the mobile phone to customers, it will also be available from PC to Local Police.

1.3 WORD AND MACHINE

Phenomenon	**Shared**	**Who controls it**
User sees a traffic violation	N	W
User wants to send a report	Y	W
User get location through GPS	Y	W
System checks the report(traffic plate and location)	N	M
System rejects/accepts the report	Y	M
System builds stats	N	M
User get stats	Y	W
Local Police check report	Y	W
Local Police delete report	Y	W
System builds suggestions	N	M
Municipality gets suggestions	Y	W
Municipality makes accidents information available	Y	W
Municipality improves traffic condition	N	W
Citizen reviews his old report	Y	W

1.4 Definitions, acronyms, abbreviations

1.4.1 Definitions

- **User**: general SafeStreets customer that can make a violation report with his device and can view all stats that the system provided.
- **Citizen:** actor that can only make report and see stats.
- **Local police:** actor of the system that intervenes checking all report and in some case removing it; furthermore can view all stats provided by SafeStreets.
- **Traffic violation:** a road infraction reported from user.
- **Municipality:** the entity that will provide accident information and will get suggestion from SafeStreets.
- **Accident:** sent by municipality, will involved into stats.

1.4.2 Acronyms

- **API :** Application Programming Interface
- **GPS :** Global Positioning System
- **UI :** User Interface
- **PC:** Personal Computer
- **GDPR:** General Data Protection Regulation
- **EU:** European Union
- **EEA:** European Economic Area

1.4.3 Abbreviations

- **Stat:** Statistic

1.5 Document structure

- **CHAPTER 1:** it is an introduction, it provides a general overview of SafeStreets through the purpose and the scope: listing goals, domain assumptions and the main phenomena.
- **CHAPTER 2:** In this section a general description of the system is provided. Both the behavior of users and main objects is specified through some state diagrams and the class diagram, there are also described the main functionalities of the application (Registration & Login, Make reports, Building & Access to stats, Check report and Provide Suggestions).
- **CHAPTER 3:** it represents the core of the document, it contains the interface requirements that are: user interface, software interface and hardware interface. Moreover are listed domain assumptions and requirements to achieve the goals, through some scenarios are also described how the system acts in the real world. Through the use-case diagrams and the sequence diagrams are defined the interaction between objects. Last but not least in the end of the section are presented the non-functional requirements.
- **CHAPTER 4:** in this function a formal analysis using alloy is given, focusing mainly on the constraints given by how objects should interact between themselves and how they should behave in the context of the real world.
- **CHAPTER 5&6:** in these last chapters is reported the effort spent and the reference documents.

1.6 Revision History

- Version 1.0: first release
- Version 1.1: final release, the first version was modified according the DD document, in particular changes have been made on sequence diagrams, class diagram and on the alloy model; also some requirements have been better specified.

2. Overall Description

The following high-level class diagram provides a model of the application domain: it obviously doesn't contain every class that will be necessary to define the model of the System, there are only few attributes that help to get a better understanding of the system, also methods are not specified. As we can see the focal points of the diagram are the *ReportViolation*, that is also involved in most of the sequence diagram in the next section, and the *Stats* that will provide many different functions.

The particular choice to highlight the class *Violation and the class Model(vehicle)* as an enumeration is made in order to make the system able to catalog each different kind of violation and kind of vehicle's model through stereotyped type, so that will be easier build stats and look up specific information. To help user providing an exactly location the alternative to entering an address is given by the GoogleMaps API.

Between Accident and ReportViolation we used an aggregation relation

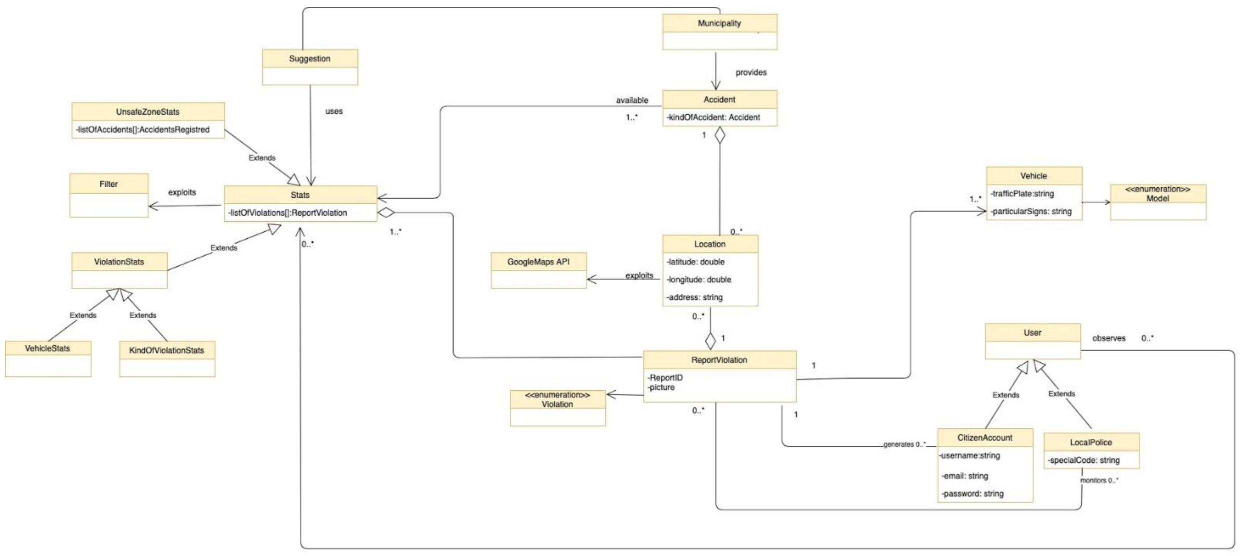


Figure 1 - Class diagram

2.1 State Diagrams

Now the most critical aspects of the system will be modelled, showing their behaviors and the evolution over time of their states.

State diagram 1: Report's life

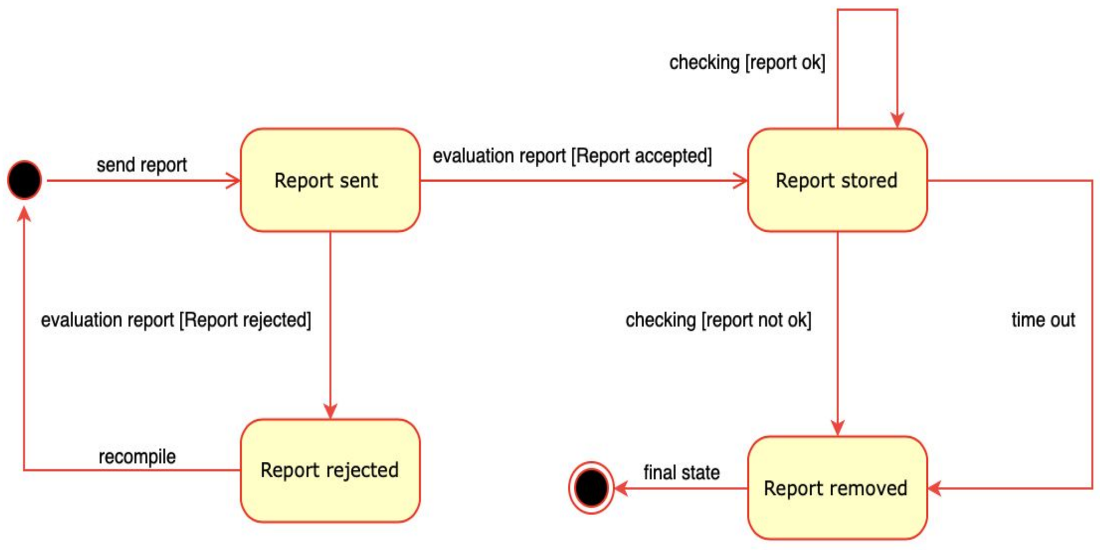


Figure 2

Report's life: it represents the lifecycle of a report, including the possibility to recompile the form if something goes wrong; obviously a report can be removed either if the local police check it and observe that it is not truthful or if it passed enough time from is stored, so that it is deleted in order to keep only the recentest reports and so stats more accurate.

Class diagram 2: Stats violation creation

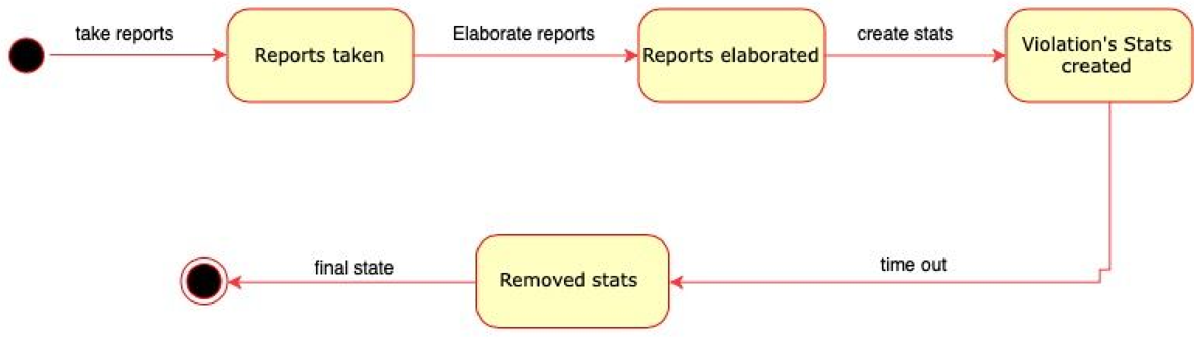


Figure 3

Class diagram 2: Stats streets safety creation

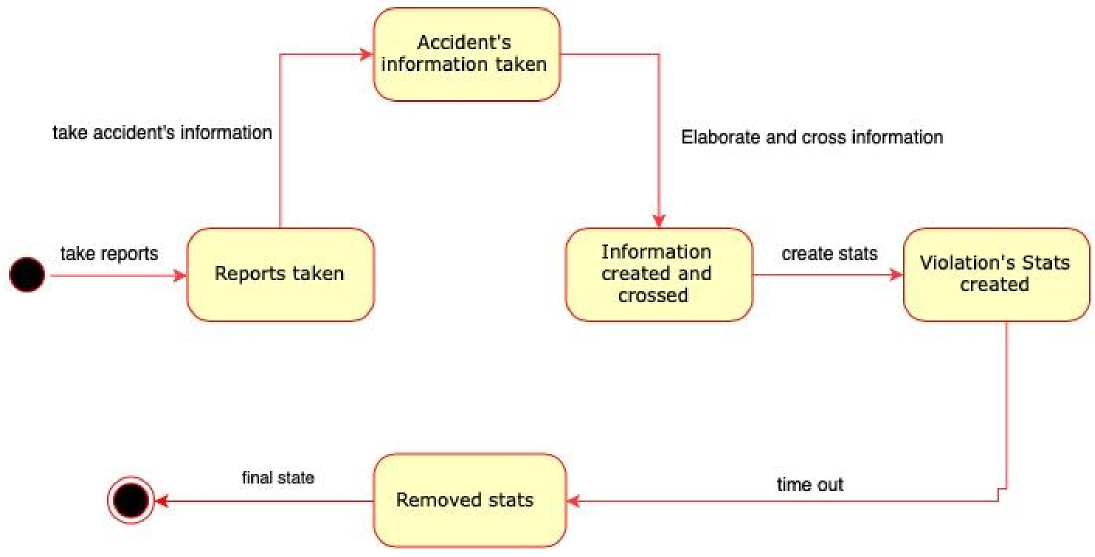

Accident's information taken
take accident's information
Elaborate and cross information
take reports
Reports taken
Information created and crossed
create stats
Violation's Stats created
final state
Removed stats
time out

Figure 4

Diagrams in figure 3 and figure 4: they shows the lifecycle of the two kind of stats, as we can see they are very similar because each kind of stats extends the same class. It's important to point out that each time there is the time out, there will be an update of stats, so the old stats will be removed and they will be regenerated so that lifecylce will start again and again.

2.2 Product functions

Considering the goals listing before is possible to define the mainly functions of the system that are: the possibility to make reports, the building & access to stats, the possibility to check and eventually delete reports by LocalPolice and the suggestion that the System can provide to Municipality.

2.2.1 REGISTRATION AND LOGIN

The system will allow customers to access in different ways: system will let citizens to sign up and then to insert them username and password to login every time they want; instead regarding Local Police they will get a special code to access to the system. Every time a new user will sign up is data will be stored into a relational DB creating a new tuple.

2.2.2 MAKE REPORTS

This is the base function of SafeStreets, it's available for Citizens.
Every Citizen correctly logged in can signalize a street violation filling a form provided by the system; to do that he has to provide a photo well visible, so that the system can recognize the traffic plate, moreover the location, that can be capture by GPS and through inserting manually the address, must be enough accurate (instead the date will be assigned automatically by the system). In case of the location is not accurate or the traffic plate is not visible, the system will ask user to refill the form. When the report will be correct, the data will be stored into the DB and will keep in memory untill the LocalPolice will decide to delete it or will be passed a defined time frame so that the report will be too old and it could make stats not so efficient. Moreover, everytime he wants, citizen will be able to review his past reports.

2.2.3 BUILDING & ACCESS TO STATS

Another very important function of SafeStreets is given by the access to stats, that are available for all kind of user (both citizens and Local Police).
There are two mainly class of stats: the one provided only by report violations and the one that cross the first one with the accidents data maka available by municipality.

The first class is also divided into two subclass that are the class of vehicles with most infractions and the type of violation most often committed, these stats will make available to user through appropriate list with relative percentages.
The second class will show the most unsafe areas in the municipality highlighting the areas with the highest frequency of violations.
To have an easier access to the stats in which user are interested in, it will be possible to filter stats with some parameters; it’s important to underline that stats will not be always available, indeed the instance referred to users’ request must be more than fifty to be available.

2.2.4 CHECK REPORT

This function is available only for Local Police; through this function a policeman has the possibility to request some particular report and to check if it is truthful or not, in the case it’s not truthful he has the possibility to delete that report.

2.2.5 PROVIDE SUGGESTIONS

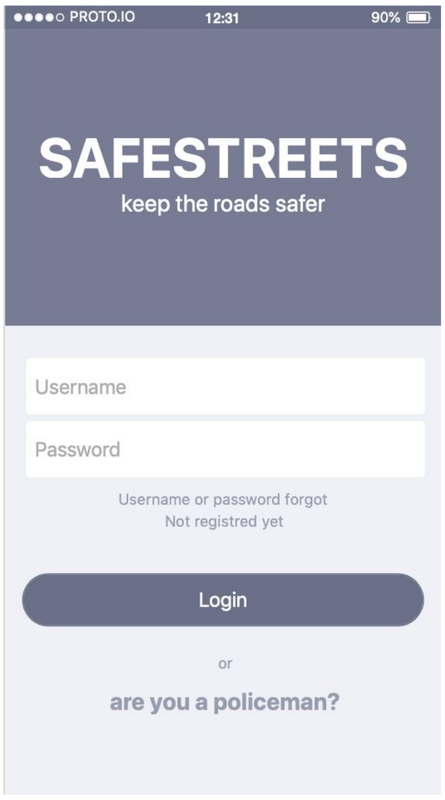
Last, but not least the system analyzing its data and stats will be able to provide municipality some suggestions in order to improve the road conditions of the municipality, for example system could suggest to add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe
parking.

3. SPECIFIC REQUIREMENTS

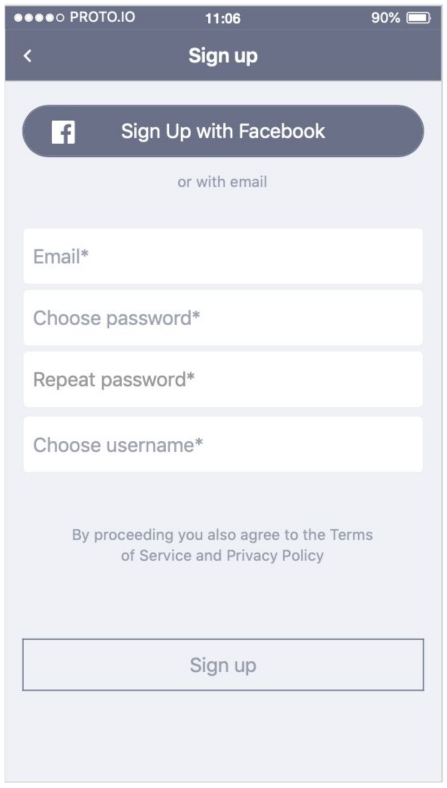
3.1 External interface requirements

3.1.1 User Interfaces

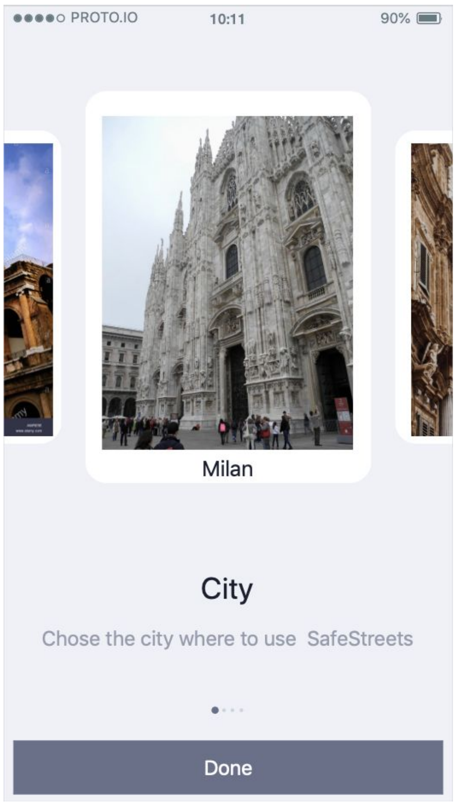
The following mockups give an approximate idea of how the application's interfaces should appear: some of the most important screenshots of the interactions between the system and each of its customers (Citizens and Local police) are represented.



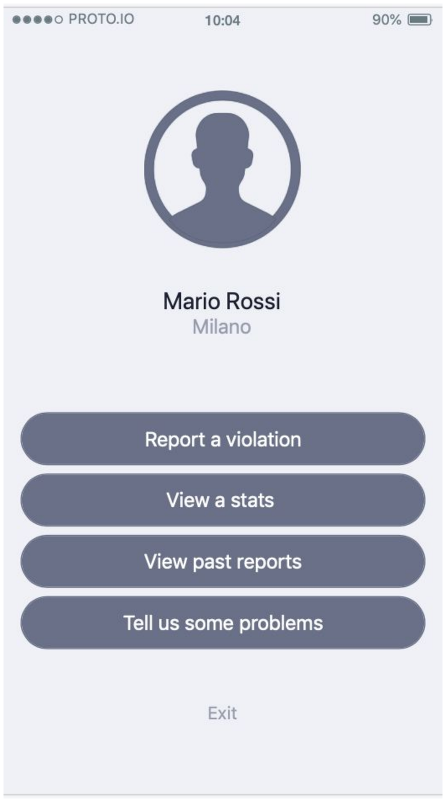
Mockup : SignIn



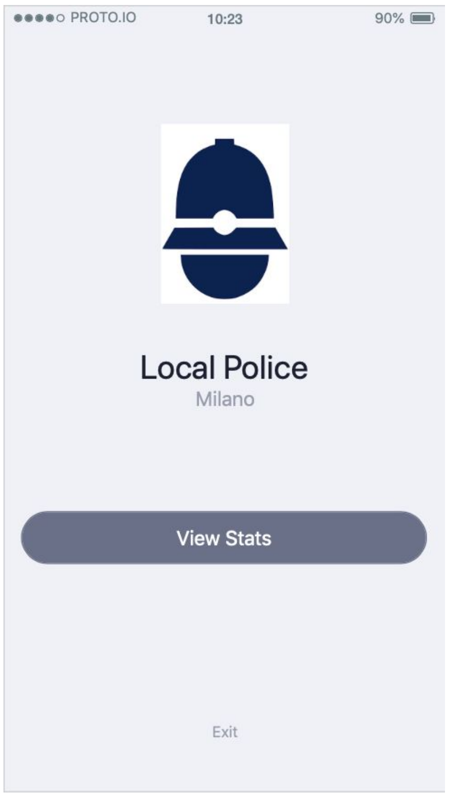
Mockup: SignUp for Citizens



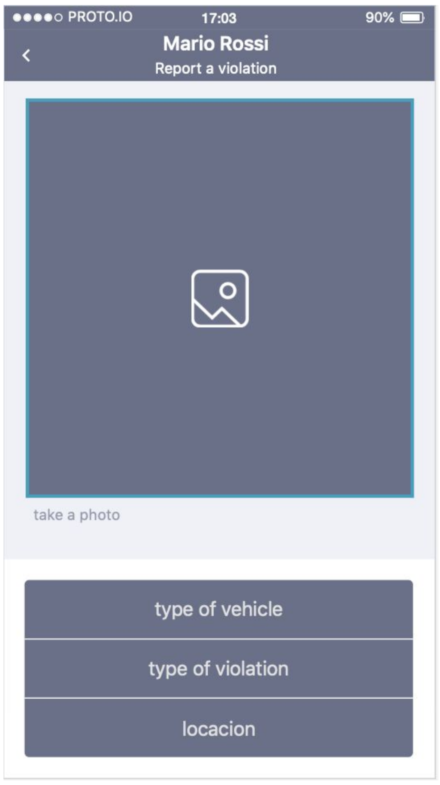
Mockup : City Choice



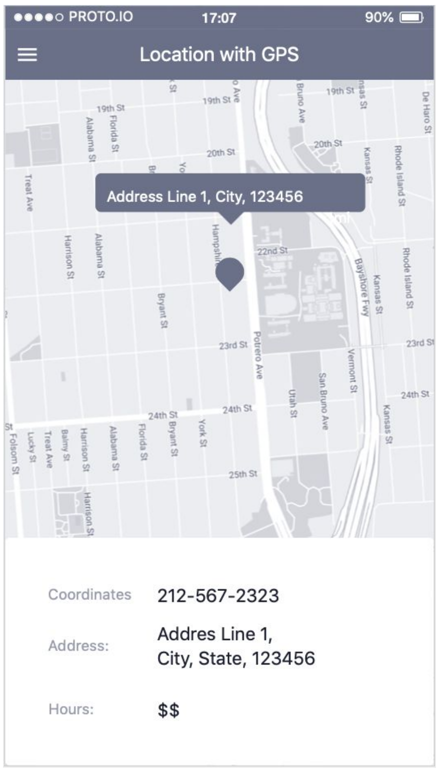
Mockup: citizen's home SafeStreets



Mockup: Local Police's home

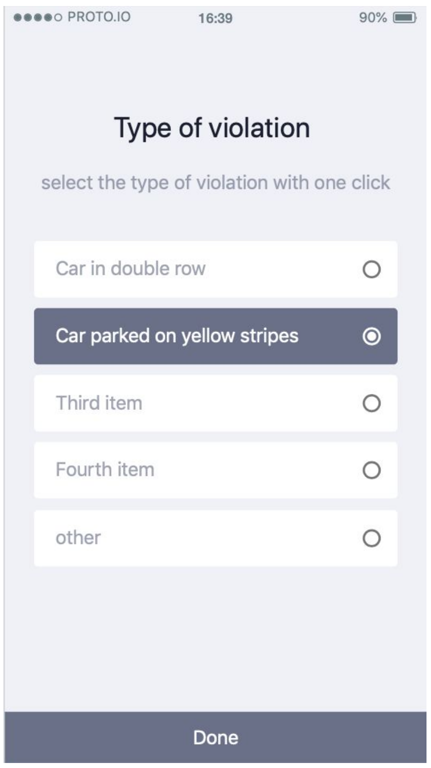


Mockup: make a report



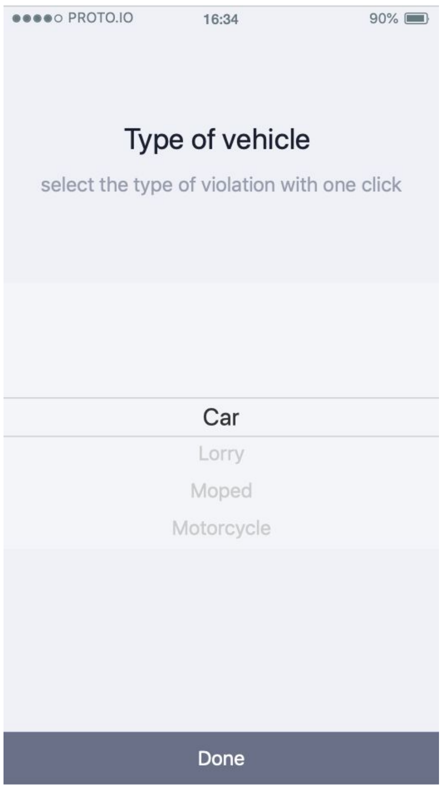
Coordinates	212-567-2323
Address:	Addres Line 1, City, State, 123456
Hours:	$$

Mockup: automatic location search

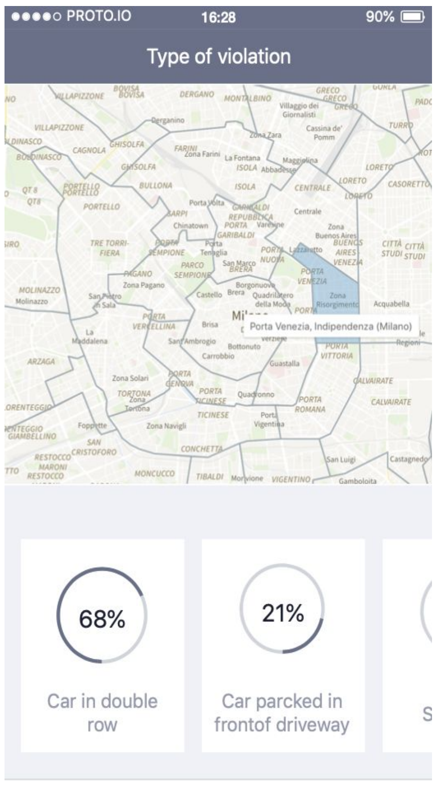


- Car in double row
- [✓ Car parked on yellow stripes]
- Third item
- Fourth item
- other

Mockup : choice of "type of violation"

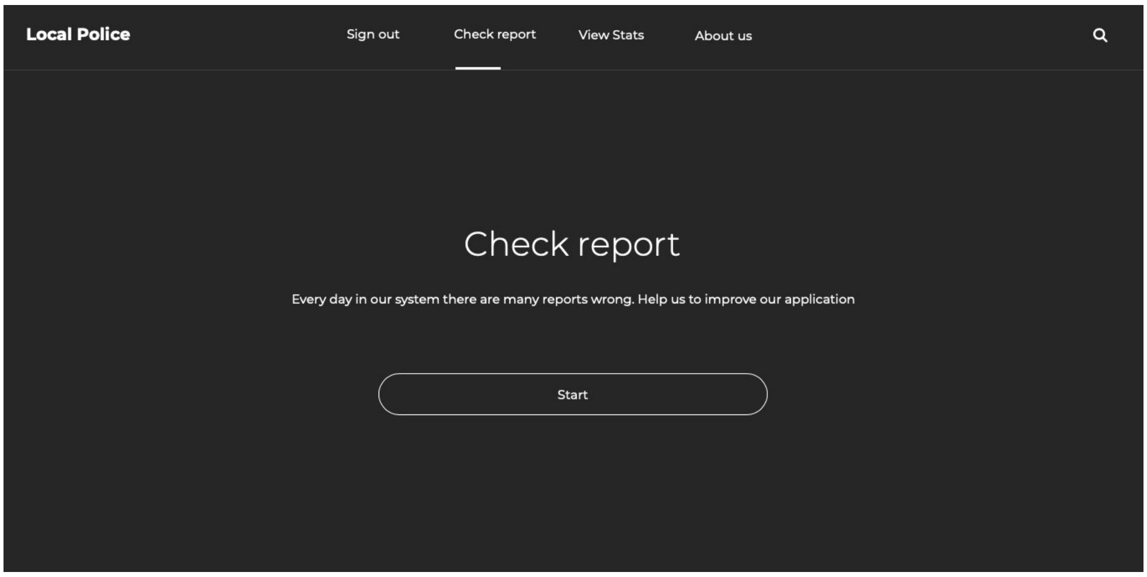


Mockup: choice of "Type of vehicle"



68%	Car in double row
21%	Car parcked in frontof driveway

Mockup: stats' view



Check report

Every day in our system there are many reports wrong. Help us to improve our application

Mockup: Local Police's menu to check report

3.1.2 Hardware interfaces

It is not using any hardware interfaces; however, it requires using a smartphone or a web browser that can use GPS services; furthermore each device used by user has a camera.

3.1.3 Software interfaces

SafeStreets uses only one external service:

1. City Maps: they are provided by Google Maps and with GPS make possible and more interactive many functions of SafeStreets like the displaying the most dangerous areas via map

3.2 FUNCTIONAL REQUIREMENTS

3.2.1 USER CHARACTERISTICS

The actors of the application are:

- Citizen: A person that is successfully registered to SafeStreets and is able to send reports and access to stats.
- Local Police: a special account registered to SafeStreets through a special code, that is able to check reports (and eventually delete them) and access to stats.
- Municipality: it participates to the system making available accidents data and taking suggestions.

3.2.2 Domain Assumptions

1. Internet connection works;
2. The usernames used in the system are unique to every user.
3. The devices that acquire users' positions provide a location with an accurate enough current location;
4. Each user have a telephone with a camera;
5. The violation report is considered valid if and only if the traffic plate is well visible
6. The municipality has to keep accident information to make available to SafeStreets.
7. Every picture taken has a resolution at least of 1920x1080 or 1080x1920

3.2.3 Requirements

G1. SafeStreets must allow users to report a violation sending pictures and description of that one.

1. the system must provide to user the form to be filled.
2. The system must control the visibility of traffic plate
3. the system must accept a location both from GPS and through the manual insertion of address from the user.

4. the system must recognize the correctness of user’s information (for example if a user put an address in Milan, that address must exist in Milan)
5. The system must allow to user’s telephone to take a photo or load that one from the gallery.

G2. SafeStreets must create automatic statistics about the kind of violations occurred and the class of vehicles with the most violations; these stats must be updated every 2 weeks;

1. The system must store the information taken from users' report.
2. The system must process each report and classify each one depending on the vehicles which made the violation and the kind of that.

G3. SafeStreets must create statistics providing the most unsafe areas in the municipality territory; these stats must update every 2 weeks;

1.The system must store the information about accidents(occurred in the city) taken from municipality
2.The system must store the information taken from users' report
3.The system must build security’s stats crossing information taken from the municipality and the ones taken from users.

G4. SafeStreets must suggest possible interventions to the municipality (through information sent by municipality and users);

1. the system, analyzing its stats, must provide possible suggestion to the municipality.

G5. SafeStreets must allow both users and local police accessing the available stats;

1. the system has to ask user which parameters use to filter data in order to see the areas in which he is interested in.
2.The system must allow the User to access to all registered data and stats highlighting the areas with the highest frequency of violations or “listing the class of vehicles “with the most violations.

G6. SafeStreets must allow only police to access all reports and in case to delete them if they are not truthful.

1. The system must recognize the user as a policeman.
2. If the user is a policeman the system must provide him a list of each reports with its information.
3. the System must allow policeman to delete a report from the DB if it’s not truthful.

G7. SafeStreets must allow users to access to their past report.

1. the system must provide citizen a list of his reports with their information.

TRACEABILITY MATRIX

REQUIREMENT	**USE CASE**
R1 the system must provide to user the form to be filled.	send violation report
R2 the system must control the visibility of traffic plate	send violation report; check traffic plate
R3 the system must accept a location both from GPS and through the manual insertion of address from the user.	send violation report
R4 the system must recognize the correctness of user’s information (for example if a user put an address in Milan, that address must exist in Milan)	send violation report
R5 The system must allow to user’s telephone to take a photo or load that one from the gallery.	send violation report
R8 The system must store the information taken from users' report.	view past reports made; check reports; get stats
R9 The system must process each report and classify each one depending	get stats

on the vehicles which made the violation and the kind of that.	
R10 The system must store the information about accidents(occurred in the city) taken from municipality	get stats
R11 The system must build security’s stats crossing information taken from the municipality and the ones taken from users.	get stats
R12 the system, analyzing its stats, must provide possible suggestion to the municipality.	
R13 the system has to ask user which parameters use to filter data in order to see the areas in which he is interested in.	get stats; add a filter
R14 The system must allow the User to access to all registered data and stats highlighting the areas with the highest frequency of violations or “listing the class of vehicles “with the most violations.	get stats
R15 The system must recognize the user as a policeman.	Login LocalPolice
R16 if the user is a policeman the system must provide him a list of each reports with its information.	check reports
R17 the System must allow policeman to delete a report from the DB if it’s not truthful	check reports
R18 the system must provide citizen a list of his reports with their information.	view past reports made

3.2.4 SCENARIOS

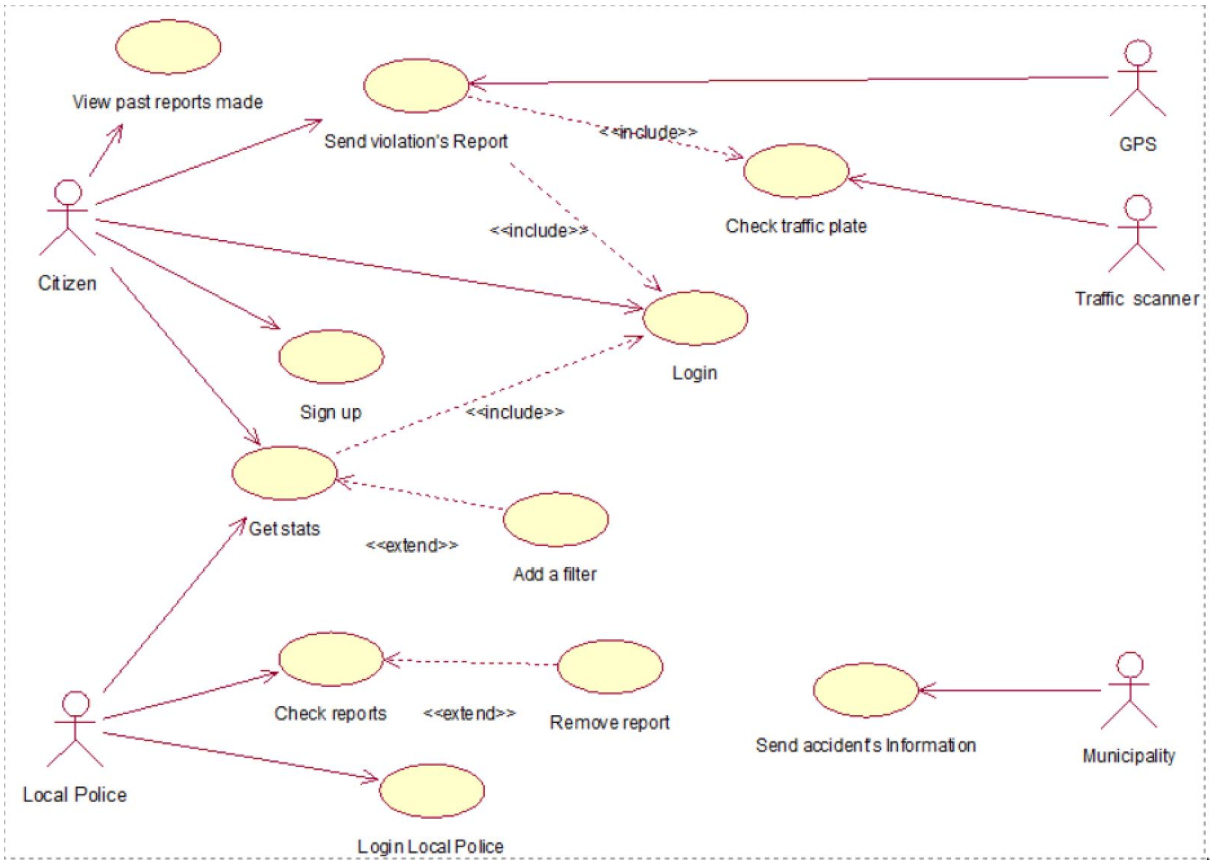
Mario and his disabled son Luigi decide to go to the park together to spend a day outdoors; they head to the Sempione park in Milan by car and once arrived looking for parking with yellow stripes they realize that a car is unjustly parked on a parking lot dedicated to the disabled; the father, rightly angry, decides to use the new SafeStreets app to try to solve the problem: so decide to download the application, to register him as a user, then first login, he takes a picture of the car (making sure you see the license plate), describes the situation and finally before sending the signalization inserts the address Via Giuseppe Toniolo; however the system responds that the location is not accepted because the address is not present in Milan; Mario then uses GPS service and the report is thus correctly recorded.

Luisa decides to visit Milan; she arrived in Navigli she is undecided if to park in via Mameli or in via Sicilia; not knowing which of two streets is safer on the night she decides to consult the SafeStreets's function which provides a global view on the danger of the streets. She logs in and discovers that via Mameli is safer.

The municipality's administration has received from SafeStreets many suggestions of interventions in some areas of the city because the system noted that on 2018 traffic violations and accidents are really increased; so to solve the problem mayor Beppe Sala decides to allocate funds for improvements of the worst area.

One policeman during his office day decides to logs in on SafeStreets with Local police's code; he gets all the information about all report of the day and checking them one by one suddenly he notices that one signalize isn't truthful so he removes it. Now the violation's stat is more reliable.

3.2.5 USE CASE DIAGRAM



NAME	Sign Up
ACTOR	Citizen
ENTRY CONDITIONS	Citizen opens SafeStreets app by his own device
EVENTS FLOW	1. The user : a. choices a “Sign up” option; b. fills the mandatory fields like Username, email

	and city where the user wants use the system; c. fills the optional fields like personal photo. d. clicks “Confirm” button e. confirms the account by email 2. The system saves all data
EXIT CONDITIONS	The user now is registered
EXCEPTIONS	User: 1. makes a mistake in mandatory fields; in this case SafeStreets notifies the error with a messageBox 2. is already registered; in this case safestrees notifies it with a messageBox 3. uses an Username already used by other users; in this case SafeStreets notifies it with a messageBox 4. The city provided is not enjoy of SafeStreets’ service

NAME	Send violation report
ACTOR	Citizen
ENTRY CONDITIONS	User: 1. has successfully logged in 2. has clicked “Make report” button
EVENTS FLOWS	User : 1. fills the form fields : a) take a photos of violation; b) writes a traffic plate; c) individuates a location(with GPS or writing the address); d) describes the vehicle; e) indicates a kind of violation; 2. sends report.
EXIT CONDITIONS	User completed the report clicking “Done” button
EXCEPTIONS	1. The locations wrote doesn’t exit;

	2. The traffic plate in the photo done is not visible.

NAME	Get stats
ACTOR	Citizen, Local police
ENTRY CONDITIONS	User: 1. has successfully logged in; 2. has clicked “Stats” button.
EVENT FLOWS	1. User clicks the kind of stats that him wants; 2. User can add optional filter; 3. Safestreets provides the possibile stats.
EXIT CONDITIONS	User clicks the button “Return”
EXCEPTIONS	1. There isn’t a statistic available (less then 50 instances)

NAME	Login
ACTOR	Citizen
ENTRY CONDITIONS	User has successfully signed up
EVENT FLOWS	Citizen: 1. write username; 2. write password; 3. clicks “Enter” button.
EXIT CONDITIONS	User clicks the button “Enter”
EXCEPTIONS	1. Password or Username forgot; in this case, SafeStreets provide a way to restore a new password/ username by an email address 2. Password or Username digited are wrong or the account doesn’t exist, in this case the Safestreets provides an error message

NAME	Add a filter
ACTOR	Citizen, local police
ENTRY CONDITIONS	User: 1. has successfully logged in 2. is on the stats' page 3. click the button "Apply filter"
EVENT FLOWS	User select possible filters
EXIT CONDITIONS	User clicks the button "Apply"
EXCEPTIONS	There aren't available stats for the selected filter: 1. Area selected doesn't have enough report 2. For the kind of violation selected there aren't enough report

NAME	Login Local police
ACTOR	Local police
ENTRY CONDITIONS	Local police : 1. opens the app 2. clicks "I'm an authority"
EVENT FLOWS	Local police writes a police code
EXIT CONDITIONS	Local police clicks the button "Login"
EXCEPTIONS	1. The police code is not correct; in this case SafeStreets provides an error message

NAME	Check report

ACTOR	Local police
ENTRY CONDITIONS	Local police : 1. has successfully logged in by Web 2. clicks the “Check report” button
EVENT FLOWS	Local police browses the list of report: could delete or keep the report
EXIT CONDITIONS	A. the report is deleted OR B. the report is kept in memory
EXCEPTIONS	There aren’t report

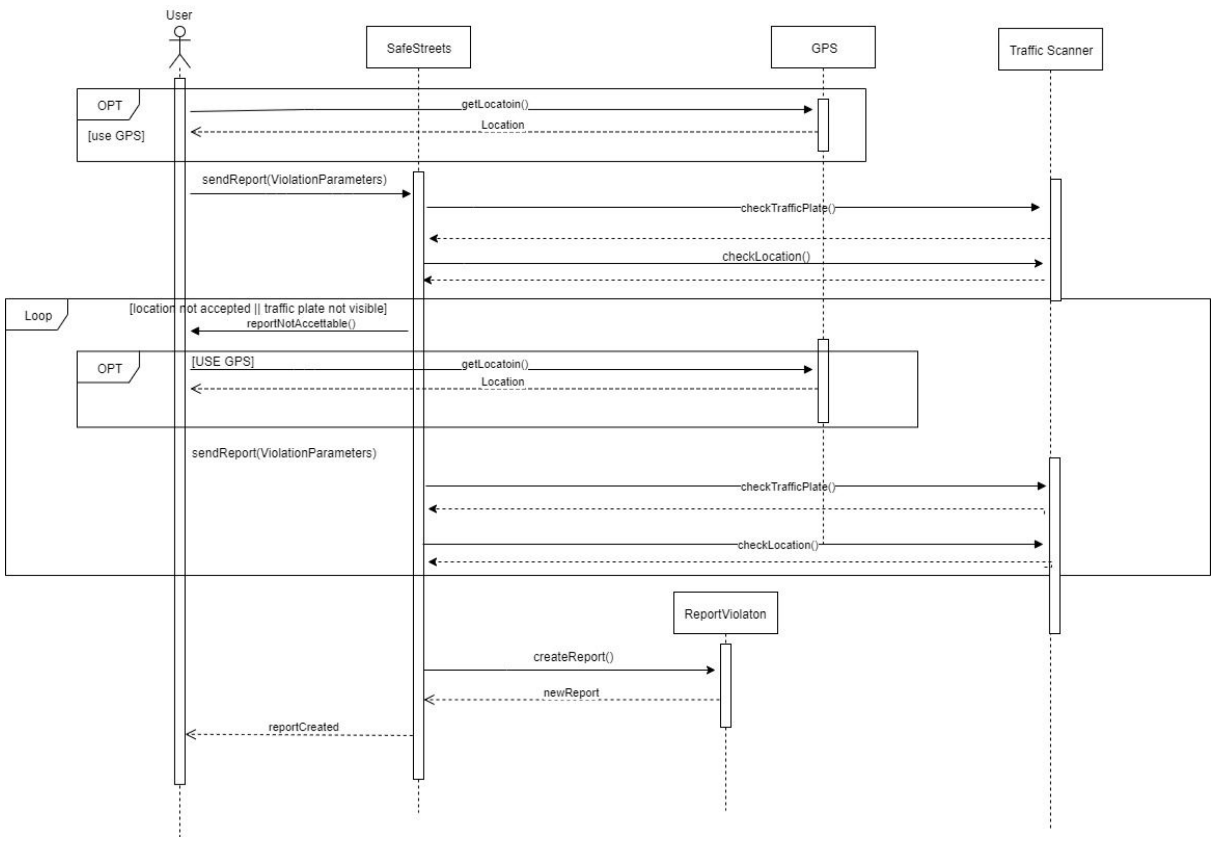
NAME	Remove report
ACTOR	Local police
ENTRY CONDITIONS	Local police : 1. has successfully logged in; 2. clicks the button “Check reports”
EVENT FLOWS	Local police : 1. checks one report 2. see that the report is not truthful 3. clicks button “Remove report”
EXIT CONDITIONS	The report is removed
EXCEPTIONS	/

NAME	Send accident’s infomation
ACTOR	Municipality
ENTRY CONDITIONS	Time out
EVENT FLOWS	Municipality sends all information about accident that occurs
EXIT CONDITIONS	All information are sent

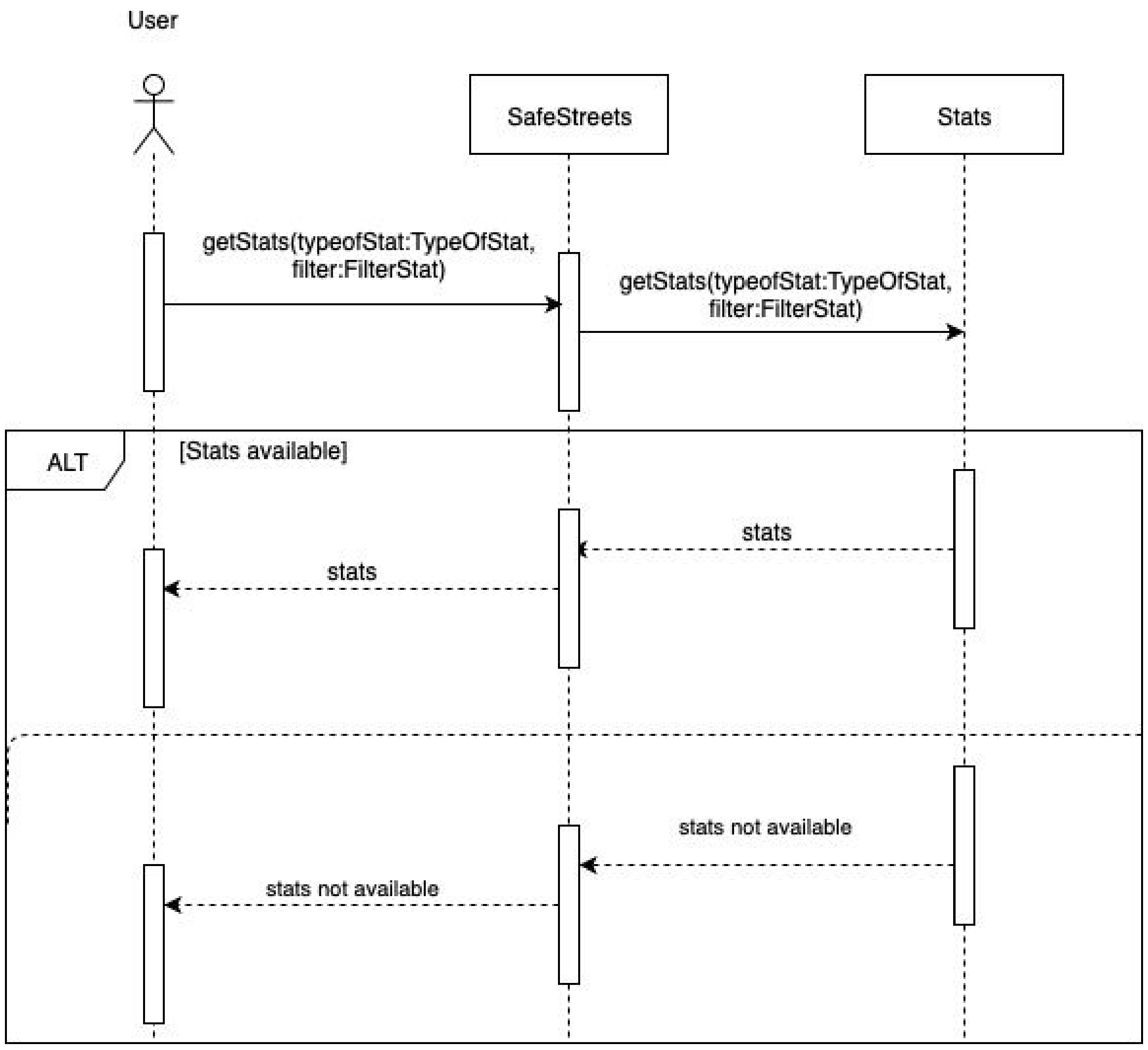
EXCEPTIONS	/

NAME	View past reports made
ACTOR	Citizen
ENTRY CONDITIONS	Citizen : 3. has successfully logged in; 4. clicks the button “View past reports”
EVENT FLOWS	Citizen view all reports which he wants
EXIT CONDITIONS	Citizen clicks the button “Exit”
EXCEPTIONS	Citizen has never done a violation report; in this case the system provides a message that says “You have not yet made a report”

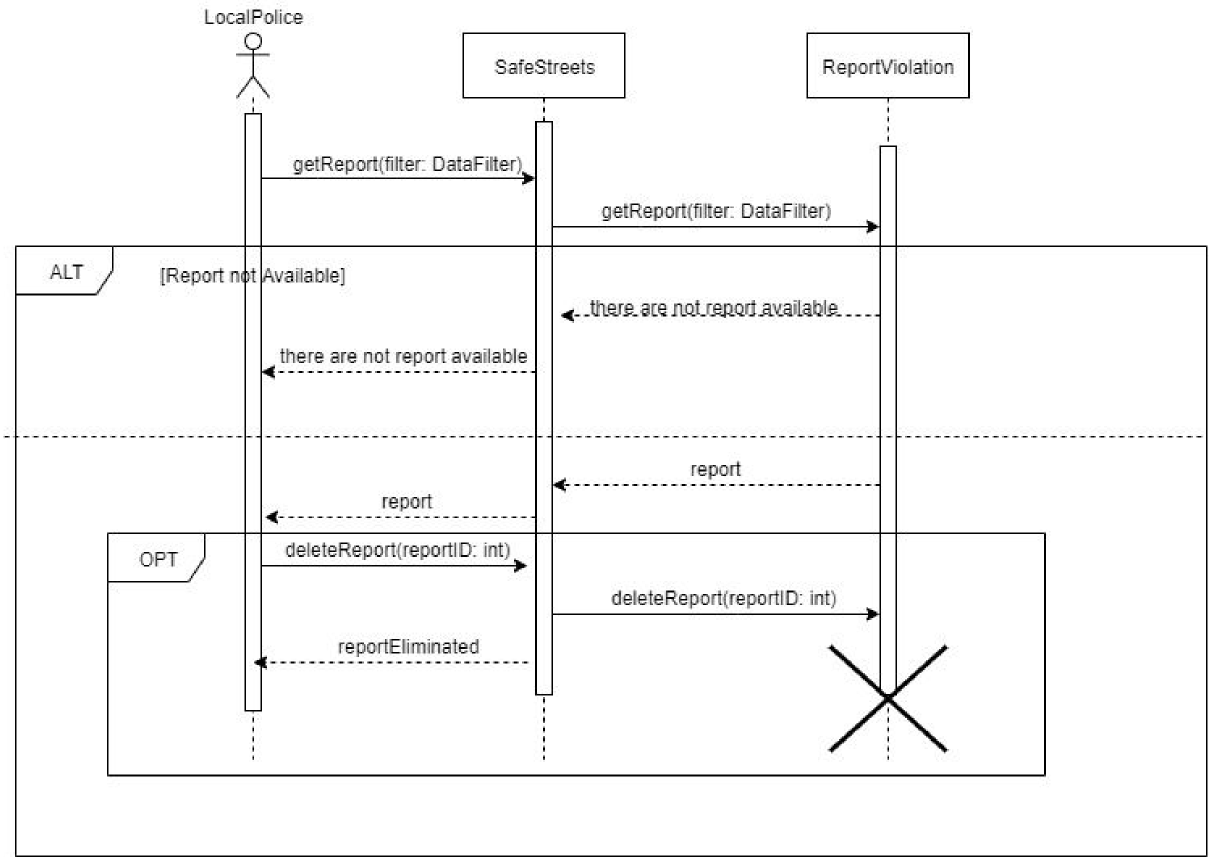
3.2.6 SEQUENCE DIAGRAMS



Make report- sequence diagram


User
SafeStreets
Stats
getStats(typeofStat:TypeOfStat, filter:FilterStat)
getStats(typeofStat:TypeOfStat, filter:FilterStat)
ALT
[Stats available]
stats
stats
stats not available
stats not available

Get stats- sequence diagram


LocalPolice
SafeStreets
ReportViolation
getReport(filter: DataFilter)
getReport(filter: DataFilter)
ALT
[Report not Available]
there are not report available
there are not report available
report
report
OPT
deleteReport(reportID: int)
deleteReport(reportID: int)
reportEliminated

Check report- sequence diagram

3.3 PERFORMANCE REQUIREMENTS

The system has to be able to serve a great number of users simultaneously. It also should be quick and reactive.

3.4 DESIGN REQUIREMENTS

3.4.1 STANDARDS COMPLIANCE

Regarding the privacy of data 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 European Union (EU) and the European Economic Area (EEA).

3.4.2 HARDWARE LIMITATIONS

This software has not strict hardware limitations, the only requirements are:

- iOS or Android smartphone
- 2G/3G/4G/Wi-Fi connection
- 2 megapixel camera
- GPS (preferably)

In case of the software is used by its desktop version it’s enough have a Wi-Fi connection and a Computer with Linux, iOS or Windows.

3.5 SOFTWARE SYSTEM ATTRIBUTES

3.5.1 RELIABILITY

The software must be available 24/7. However small concessions can be tolerated. So the system must be fault tolerant, redundancy is the key technique needed to achieve fault tolerance. One of the first things to have a better reliability is to duplicate the main Server, so that if one server fails, the other will work yet.

3.5.2 AVAILABILITY

Because of the main goal of the system is to guarantee the traffic safety, that is a really important problem, an availability of at least the 99.9% of the time is expected.

3.5.3 SECURITY

The data provided by the user at the sign up contains sensitive information, so security is an aspect of primary importance. All user data, but also traffic plate contained in the reports, must be confidentially stored and encrypted with high-security encryption.

3.5.4 MAINTAINABILITY

The application must be flexible and easy to fix up and modify in future. It is also important that will be easy to add some eventuale feature. In order to do that design pattern will be used and will be well explained in the design document.

3.5.5 COMPATIBILITY

The application offers a service that could be useful to a lots of users, because of that it has to be compatible with most devices and technologies possible in different circumstances.

4. FORMAL ANALYSIS USING ALLOY

In this section the Alloy model is given. Through this tool the most important features of the system are specified in details, we have focused our attention to all the constraints of the system (using mostly facts) specifying the multiplicity -underlined through appropriate comment in the code- of each relation and defining the connection between each signature.
We've used some predicates and assertions to express the main feature of SafeStreets that is the possibility to make reports (on the citizen side), and also the access to stats available also for Loca lPolice.
There will be also the result of the analysis, showing the world generated and the result of the execution of predicates and assertions.

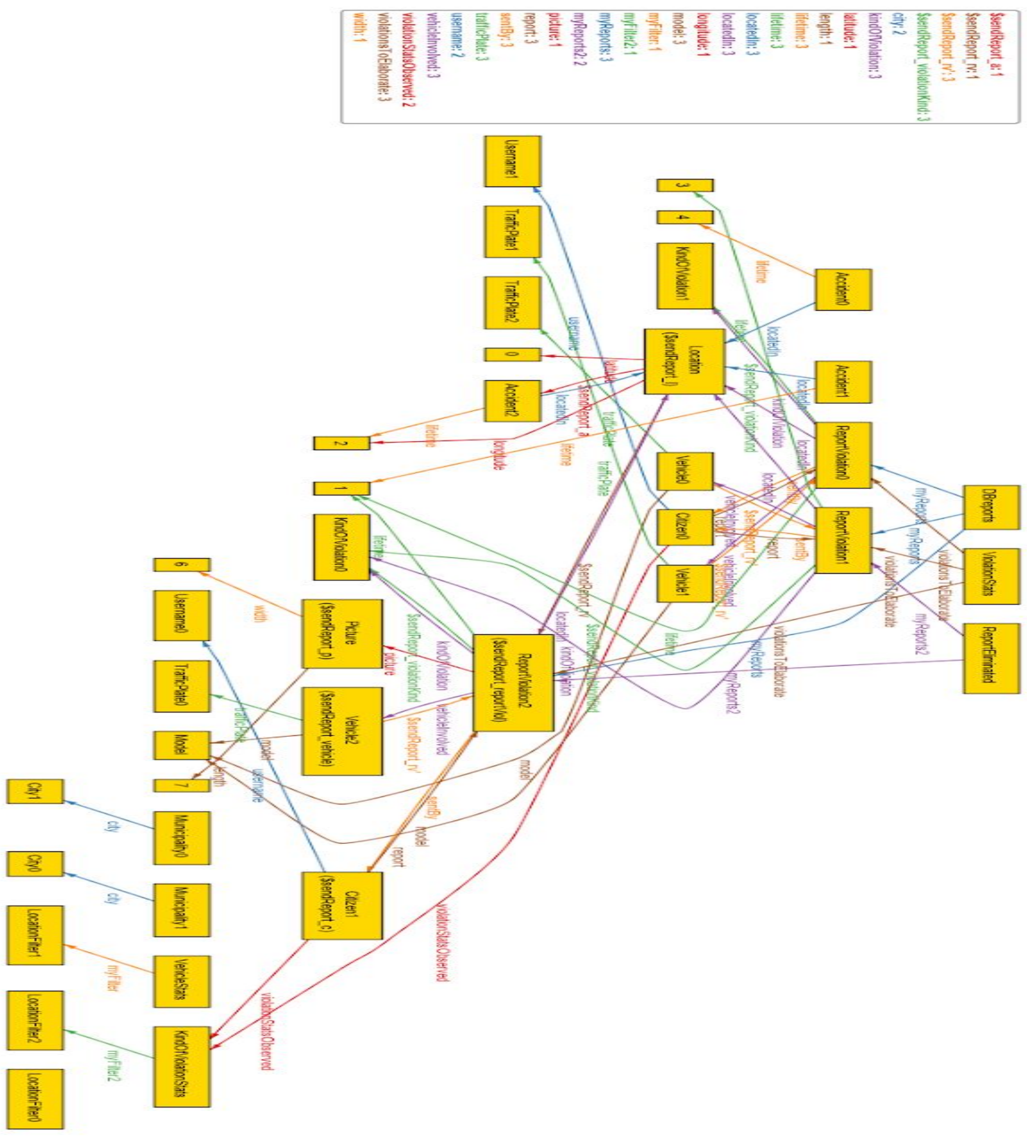
```
open util/integer


sig  Username{}

sig User{

 violationStatsObserved: one KindOfViolationStats
}

sig SpecialCode{}

sig Citizen extends User {
  username: one Username,
 report: set ReportViolation
}
{
        #report > 0
}


//fact {
 //#report >0
//}
one sig ReportEliminated{
 myReports2: set ReportViolation
}

sig LocalPolice extends User{
  specialCode: one SpecialCode,
  monitors: set ReportViolation ,
 eliminates: set ReportViolation      //sistemare relazione police-report-elimina
}

fact allReportEliminated{
  all  policeman:  LocalPolice  |  one  re:  ReportEliminated  |  policeman.eliminates  in
re.myReports2
}
//fact to connect each username d to an unique Citizen
fact account{
  all u:Username |  one c:Citizen |  c.username=u
}

//fact to connect each  specialCode  to an unique LocalPolice
fact lpaccount{
  all sc:SpecialCode | one lp:LocalPolice | lp.specialCode=sc
}

// through this fact we ensure that a User can only be a LocalPolice or a Citizen
 fact {
  LocalPolice+Citizen=User
}


sig KindOfViolation{}

sig Model{}

sig Picture{
  length: one Int,
  width: one Int
}{
(length>=1  and  width>=2)  or(length>=2  and  width>=1)      //  (length>=1920  and
width>=1080) or(length>=1080 and width>=1920)
}
```

```
sig Location{
  latitude: one Int,
  longitude: one Int
}{
 latitude>=-3 and latitude<= 3 and longitude>=-6 and longitude<=6 /*atitude>=-90 and latitude<= 90 and longitude>=-180 and longitude<=18*/  //latitude>=-10 and latitude<= 10 and longitude>=-20 and longitude<=20 /*atitude>=-90  and latitude<= 90 and longitude>=-180 and longitude<=18
}

sig TrafficPlate{}

sig Vehicle{
  trafficPlate: one TrafficPlate,
  model: one Model,
  violation: set ReportViolation
}

//fact to connect each  traffic plate to an unique vehicle
fact vehicleTrafficPlate{
  all tp:TrafficPlate | one v:Vehicle | v.trafficPlate=tp
}

sig ReportViolation{
  kindOfViolation: one KindOfViolation,
  picture: set Picture,
  locatedIn: one Location,
  sentBy : one Citizen,
  vehicleInvolved: one Vehicle,
  lifetime: one Int
}{
  lifetime>= 1 and lifetime <=5 //lifetime <=500
 }

fact {
        no disj c1,c2: Citizen | some r:ReportViolation | ((r in c1.report)  and  (r  in c2.report))
}

fact {
        all r:ReportViolation | r in (r.sentBy.report)
}


sig Accident{
  locatedIn: one Location,
  lifetime: one Int
}{
  lifetime>= 1 and lifetime <=5
}


//fact to connect each  picture to a unique reportViolation
fact Pictureconnection{
  all p: Picture | one rv:ReportViolation | p in rv.picture
}

//SISTEMARE
//fact to connect each  location to a ReportViolatoin or an Accident
fact locationconnection{
   all l:   Location | some a: Accident|some rv:ReportViolation | a.locatedIn = l or
rv.locatedIn=l
}

//through this fact we ensure that each vehicle is associated to at least one
ReportViolation
fact associationVehicle{
  all v:Vehicle | some rv:ReportViolation | rv.vehicleInvolved= v
}

//through this fact we ensure that doesn't exist report older than 500 days
fact notOlder{
  no rV:  ReportViolation | rV.lifetime >= 5 //500
}



//through this fact we ensure that doesn't exist accidents  older than 500 days
fact notOlderAccident{
  no a:  Accident | a.lifetime>= 5 //500
}

sig City {}
sig Municipality{
  city: one City
}


fact cityMunicipalityConnection {
  all c: City | one m:Municipality | m.city = c
}


sig Suggestion{
  locatedIn: one Location
}

//this fact constraints each suggestion to referr to some accident or violation location
fact locationconnectionSuggestion{
    all s: Suggestion | some a: Accident|some rv:ReportViolation | a.locatedIn =
s.locatedIn or rv.locatedIn=s.locatedIn
}

sig Stats{
  violationsToElaborate: set ReportViolation,

}


sig LocationFilter{}



sig ViolationStats extends Stats{}

sig VehicleStats extends ViolationStats {
	myFilter: one LocationFilter
}

sig KindOfViolationStats  extends ViolationStats{
	myFilter2: one LocationFilter
}

sig UnsafeZoneStats extends Stats{
  accidentsToElaborate: set Accident,
 }

one sig DBreports{
 myReports: set ReportViolation
}



fact allReportsSaved{
 all cittadino:Citizen | one db: DBreports | cittadino.report in db.myReports}

//through this assertion we want to demonstrate that Stats doesn't work with report older than 300 days
//assert noMoreThan300 {
//
//}


pred sendReport [violationKind: kindOfViolation, p:Picture, l:Location, c:Citizen, vehicle:Vehicle, reportViol:ReportViolation]{
  //reportViol.kindOfViolation = violationKind
  reportViol.picture=p
  reportViol.locatedIn=l
  reportViol.sentBy=c
  reportViol in c.report
  reportViol.vehicleInvolved= vehicle
  reportViol.lifetime=1
  c.report=c.report + reportViol
  }

run sendReport

pred showStats [u:User, f: LocationFilter]{
	u.violationStatsObserved.myFilter2=f
}

run showStats

assert sentReport{
   all violationKind: kindOfViolation, p:Picture, l:Location, c:Citizen, vehicle:Vehicle, reportViol:ReportViolation | one db: DBreports | sendReport[violationKind, p, l, c, vehicle, reportViol] implies reportViol in db.myReports
```

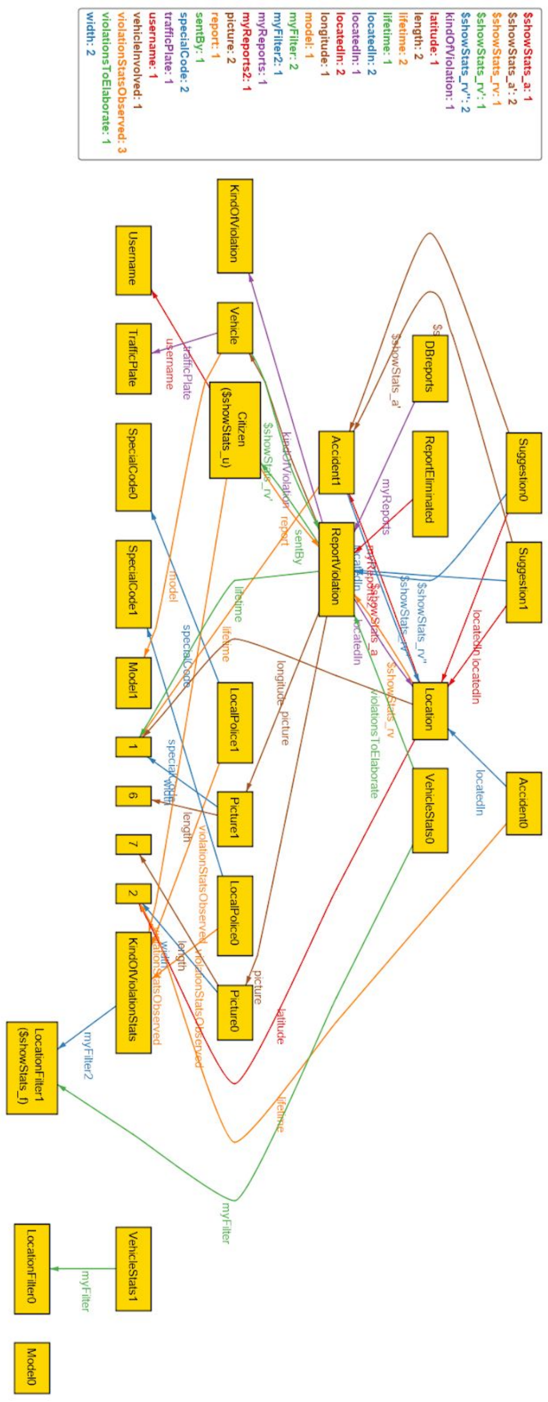
}

check sentReport

Here there are the worlds generated through the execution of the predicates:

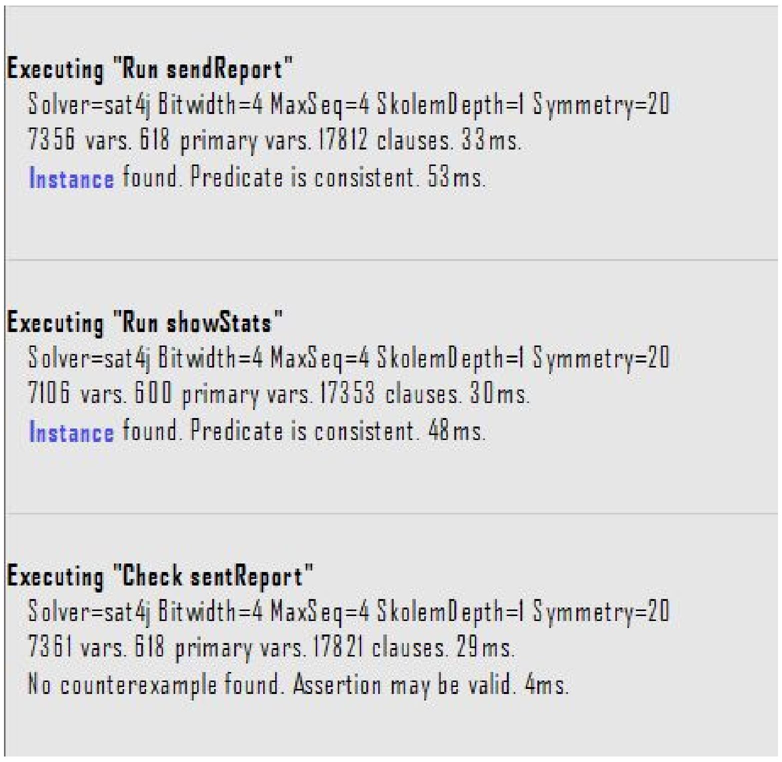


generated by the predicate sendReport.



generated by the predicate showStats..

ALLOY analysis results:



```
Executing "Run sendReport"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7356 vars. 618 primary vars. 17812 clauses. 33ms.
   Instance found. Predicate is consistent. 53ms.

Executing "Run showStats"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7106 vars. 600 primary vars. 17353 clauses. 30ms.
   Instance found. Predicate is consistent. 48ms.

Executing "Check sentReport"
   Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
   7361 vars. 618 primary vars. 17821 clauses. 29ms.
   No counterexample found. Assertion may be valid. 4ms.
```

5. EFFORT SPENT

Student GIUSEPPE ASARO

DESCRIPTION OF THE TASK	HOURS
Purpose, scope, definitions	5
Product perspective	4
Product functions	3
Domain assumptions	4
External interface requirements	1
Functional requirements	11
Non - functional requirements	3
Formal analysis using alloy	12

Student GIUSEPPE ITALIA

DESCRIPTION OF THE TASK	HOURS
Purpose, scope, definitions	5
Product perspective	4
Product functions	3
Domain assumptions	4
External interface requirements	1
Functional requirements	11
Non - functional requirements	3
Formal analysis using alloy	12

6. REFERENCE DOCUMENTS

- Specification document: “Mandatory Project assignment AY 2019-2020”
- IEEE Std 830-1998 IEEE RecoMmended Practice for Software Requirements Specifications
- Alloy: http://alloytools.org/tutorials/online/
- UML: https://www.uml.org/