

**SafeStreets Cavadini Molinari
Pederzani**



POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Ivan Cavadini, Nicolò Molinari, Luigi Pederzani
Version:	5.1
Date:	10/11/2019
Download page:	https://github.com/Pederzh/CavadiniMolinariPederzani
Copyright:	Copyright © 2019, Cavadini, Molinari, Pederzani – All rights reserved

Contents

Table of Contents	3
List of Figures	6
1 Introduction	7
1.1 Purpose	7
1.1.1 Goals	7
1.2 Scope	8
1.3 Acronyms and abbreviations	8
2 Overall Description	9
2.1 Product perspective	9
2.1.1 Further details on shared phenomena	9
2.1.2 Domain model	9
2.1.3 Class diagram	10
2.1.4 Statechart	12
2.2 Product functions	13
2.2.1 Notify Traffic Violations RE.1	13
2.2.2 License Plate Recognition RE.2	13
2.2.3 Data Collection RE.3	13
2.2.4 Data Visualization RE.4	14
2.2.5 Data Sets Analysis RE.5	14
2.2.6 Suggest Possible Interventions RE.6	14
2.2.7 Information Integrity RE.7	14
2.2.8 Generating Traffic Tickets RE.8	14
2.3 User characteristics	15
2.3.1 Assumptions, dependencies and constraints	15
3 Specific Requirements	16
3.1 External Interface Requirements	16
3.1.1 User interfaces	16
3.1.2 Hardware interfaces	19
3.1.3 Software interfaces	19
3.1.4 Communication interfaces	19
3.2 Design constraints	20
3.2.1 Standards compliance	20
3.2.2 Hardware limitations	20
3.2.3 Any other constraint	20
3.3 Functional requirements	21

3.3.1	Definition of use case diagram	21
3.3.2	Scenario 1	21
3.3.3	Scenario 2	21
3.3.4	Scenario 3	21
3.3.5	Scenario 4	21
3.3.6	Scenario 5	21
3.3.7	Scenario 6	22
3.3.8	Scenario 7	22
3.3.9	Scenario 8	22
3.4	Description of use case scenarios	22
3.4.1	Description scenario 1	22
3.4.2	Description scenario 2	23
3.4.3	Description scenario 3	24
3.4.4	Description scenario 4	24
3.4.5	Description scenario 5	25
3.4.6	Description scenario 6	25
3.4.7	Description scenario 7	26
3.4.8	Description scenario 8	26
3.5	Activity diagram	27
3.5.1	Scenario 1	27
3.5.2	Scenario 3	28
3.5.3	Scenario 4	29
3.5.4	Scenario 7	30
3.5.5	Scenario 8	31
3.6	Sequence diagram	32
3.6.1	Scenario 2	32
3.6.2	Scenario 5	33
3.6.3	Scenario 6	34
3.7	Software system attributes	34
3.7.1	Reliability	34
3.7.2	Availability	34
3.7.3	Security	34
3.7.4	Maintainability	35
3.7.5	Portability	35
3.8	Mapping on requirements	36
4	Formal Analysis Using Alloy	37
4.1	Alloy code	38
4.2	Predicates testing	45
4.2.1	World 1	45

4.2.2	World 2	47
4.2.3	World 3	49
5	Effort Spent	51
5.0.1	Ivan Cavadini	51
5.0.2	Nicolò Molinari	51
5.0.3	Luigi Pederzani	52
5.1	Versions	53
5.2	Used Tools	53
6	References	54

List of Figures

1	Class Diagram	10
2	Report life cycle	12
3	Street life cycle	12
4	Login form	16
5	Registration form	16
6	User send violation report	17
7	Map showing violations	17
8	Suggestion for possible interventions	18
9	Violations by vehicle	18
10	Scenario 1	27
11	Scenario 3	28
12	Scenario 4	29
13	Scenario 7	30
14	Scenario 8	31
15	Scenario 2	32
16	Scenario 5	33
17	Scenario 6	34
18	Signature 1	38
19	Signature 2	39
20	Signature 3	40
21	Pred and Facts 1	41
22	Pred and Facts 2	42
23	Pred and Facts 3	43
24	Pred, Facts and World	44
25	Pred 1	45
26	Result pred 1	45
27	World 1 generated by pred 1	46
28	Pred 2	47
29	Result pred 2	47
30	World 2	48
31	Predicate 3	49
32	Result pred 3	49
33	World 3	50

1 Introduction

1.1 Purpose

This document aim is to describe Safestreet functionalities and requirements. The main goal of SafeStreets is to provide authorities a tool to control the traffic violations and, in particular, parking violations. The role of citizens is crucial because they send pictures of violations using the system. SafeStreets have to store these information and elaborate it before notify authorities. The elaboration of the information is focused on retrieving some specific data such as:

- the license plate of the cars involved in the violations
- the addresses of the events
- the streets and the cars with the highest number of violations
- the most unsafe areas and the possible solutions to improve the situation

In addition SafeStreets offers to the municipality the data in order to generate traffic tickets. Furthermore the traffic tickets informations are used to build statistics regarding the effective impact of this initiative. An other important aspect is that the application need to ensure the chain of custody of the information coming from the user.

1.1.1 Goals

G1 allow users to send photos and information about a traffic violations

G2 elaborate received data and provide it to the authorities

[G2.1] retrieve the license plates from photos

[G2.2] retrieve addresses [G2.3] ensure the picture reliability

G3 calculate statistics

[G3.1] locate the most unsafe areas

[G3.2] show the impact of the application in terms of number of violations

[G3.3] generate other type of statistics like the most egregious offenders

G4 Suggest possible interventions

G5 Guarantee data integrity

1.2 Scope

The scope where SafeStreets works is typically the urban environment. The main events caused by the world are violations, authorities requiring for statistics and accidents. The shared phenomena are pictures arriving from users, notifications, the data and statistics visualization, and lastly municipality sending information about accidents. The main actors that interact with the application are users and municipality/authorities in general. It is important to consider that a minimum resolution of the pictures taken by the users is one of the biggest constraint the system have to deal with.

1.3 Acronyms and abbreviations

- ANPR: Automatic number-plate recognition, is the portion of the system that is able to recognize the license plate number from the pictures provided by users.

2 Overall Description

2.1 Product perspective

2.1.1 Further details on shared phenomena

Report: is composed by the picture regarding violation and by its metadata: type, road or GPS coordinates, date-time and so forth. The report transfer is done via HTTPS, in order to be encrypted since it contains private data. Once it is received, the algorithm checks the picture trustfulness.

Authority notification: it has to contain the data of the violation that authorities can use to generate traffic tickets.

Data and statistics visualization: crossed data about violations and accidents are used to create the map with colored road according to the number of violations.

PHENOMENA	SHARED
User check his profile	Yes
User sends the photo of a violation	Yes
SafeStreets retrieve licence plate	No
SafeStreets retrieve position	Yes
SafeStreets check reliability of the photo	No
SafeStreets locate the most unsafe area	No
SafeStreets provide suggestions	Yes

2.1.2 Domain model

The platform has to manage different entities. Some of them are linked and they shared data and function. The entities are:

- Users
- Local authorities
- Reports about possible violations (now defined as Report)
- Real violations
- Statistics

When the user see a parking violation, he could take a picture of the involved car (is important that the license plate is visible) and sends it to the system with all the relative information about the type of violation and the road.

All of these information are used to create report entities that are analyzed by the platform's algorithm.

If the report is genuine, these data can be used by authorities to generate traffic tickets, with their systems.

Furthermore, they can analyze which are unsafe areas or the vehicles that commit the most violation.

Results are also available inside SafeStreets. The same operation is done automatically by the platform, crossing reports from users and accidents information provided by authorities.

2.1.3 Class diagram

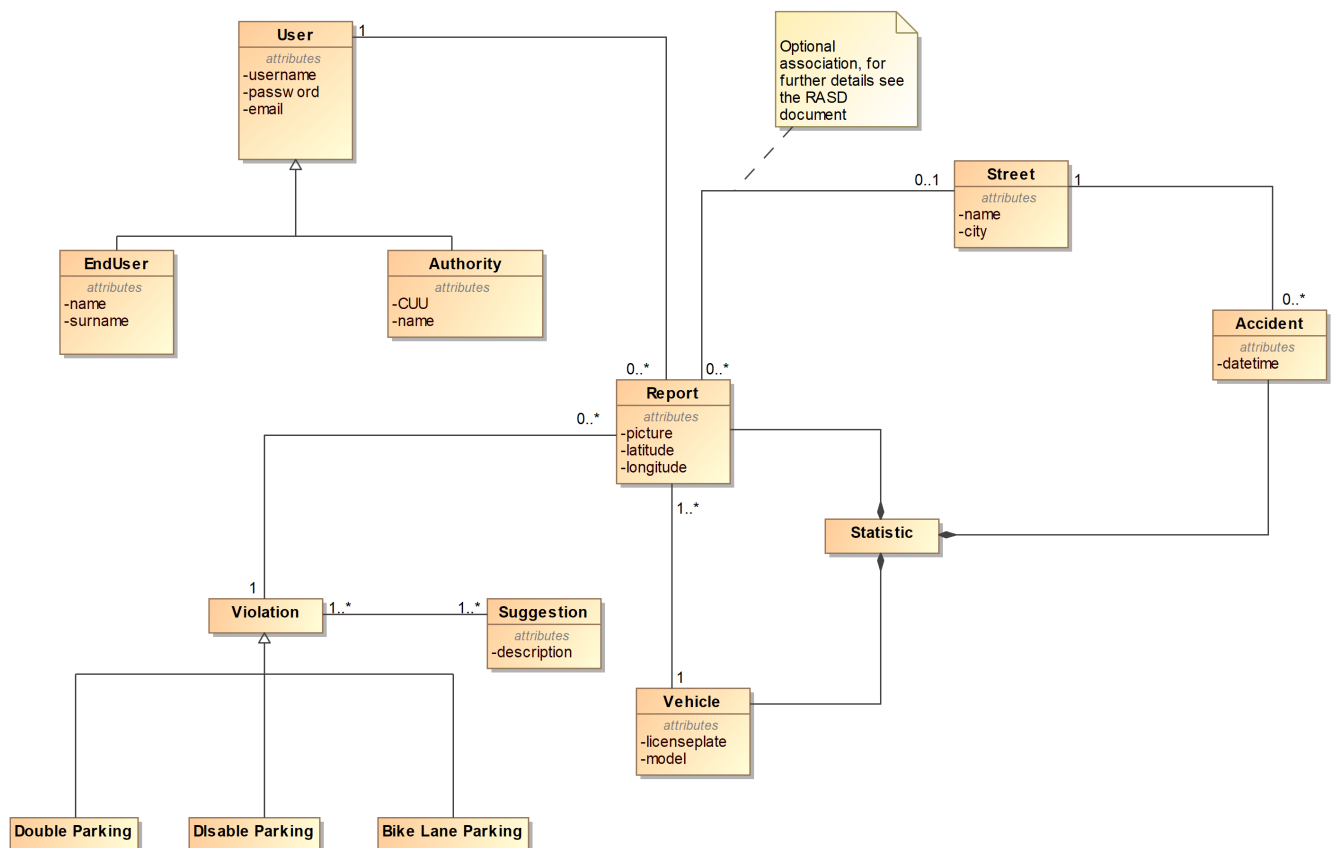


Figure 1: Class Diagram

The association between Report and Street is optional because we retrieve the position of the report automatically using GPS. Only in case of problem with

this operation, we require the address to the user. In this case, SafeStreets calculates the coordinates using the address provided by the user. We use the address also in case of report made in a place different from the one where the violation occurred.

2.1.4 Statechart

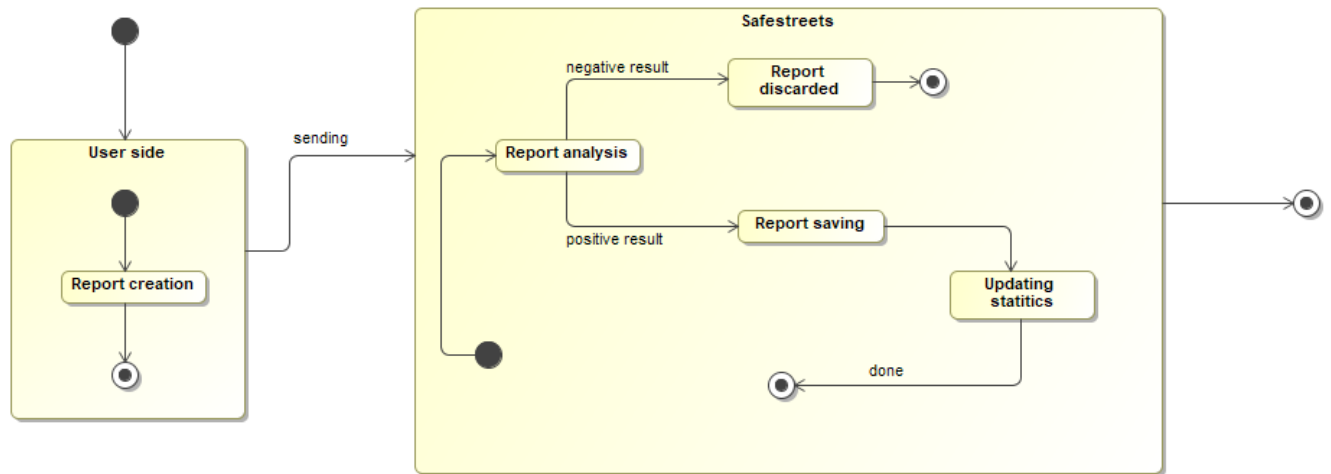


Figure 2: Report life cycle

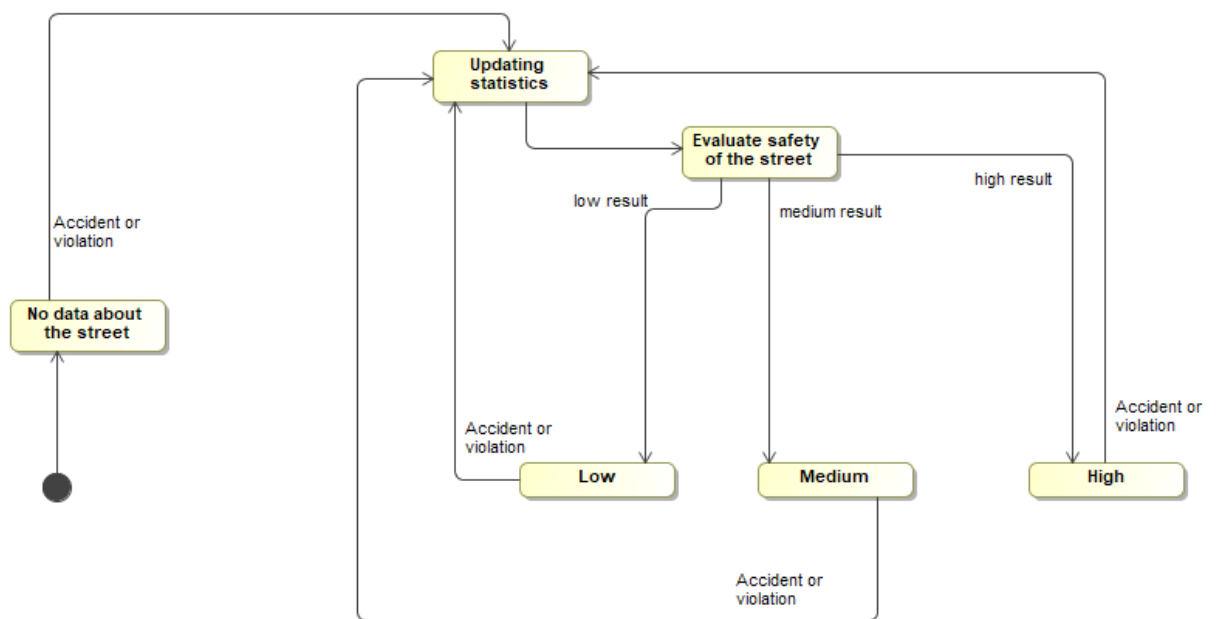


Figure 3: Street life cycle

2.2 Product functions

Considering the objectives requested by SafeStreets, main functions are described below:

2.2.1 Notify Traffic Violations RE.1

The main requirement of the application is to provide users a smart and effortless tool to notify authorities when traffic violations occur.

Users are able to select the type of violation, providing the name of the street, or using their GPS position, and uploading a picture containing the license plate that will be read by an algorithm.

Furthermore, in order to increase the reliability of the system we require the manual insertion of the license plate. When the license will be retrieved from the image, the result will be compared with the number sent by the user. If the comparison will not return equality, the report will be discarded.

2.2.2 License Plate Recognition RE.2

License plate recognition functionality is crucial for SafeStreets, from the license plate we can get information about the owner, the vehicle and its specs.

The fact that there isn't a human being responsible for manually recognizing license plates is important for the scalability, when the application will be used on a large scale.

Automatic number-plate recognition (ANPR) technology consists in seven primary algorithms that the software requires: Plate localization, Plate orientation and sizing, Normalization, Character segmentation, Optical character recognition, Geometrical analysis, Averaging of the recognized values to produce a more reliable or confident result.

Due to the fact that the image will be widely analyzed, it must be in high resolution with no blur and in a good lighting context.

2.2.3 Data Collection RE.3

Due to the fact that data is the most valuable asset of modern industry, data collection is important for all statistics, information and data visualization that SafeStreets provides.

It collects data coming from different sources: users inputs, municipality and police databases. Data collection involve several practices about correctness and security.

2.2.4 Data Visualization RE.4

A portion of the application is dedicated to data visualization by showing a map on which the streets/areas are colored according to the frequency of violations and accidents, compared to the average numbers:

- red = high
- yellow = medium
- green = low
- white = no data

Also there is a part of the UI where authorities can see statistics about vehicles and their violations.

2.2.5 Data Sets Analysis RE.5

The functionality of data analysis is crucial for finding patterns in data sets.

2.2.6 Suggest Possible Interventions RE.6

Thanks to data sets analysis, SafeStreets can suggest possible interventions for already identified "high-violation" streets.

For each type of violation is assigned one or more possible interventions that will help to decrease the occurrence of that specific violation.

2.2.7 Information Integrity RE.7

Data integrity is defined as maintenance and assurance of data consistency over its entire life-cycle.

Ensuring that data are correct and information are never altered is crucial, because local police could take the information about violations coming from SafeStreets.

2.2.8 Generating Traffic Tickets RE.8

This functionality is not an internal feature of the application.

Our scope is to provide the police, data about violations and with this information the municipality could generate traffic tickets with their own systems..

For this reason SafeStreets exposes REST API endpoints to allow authenticated authority users, in this case the police, to get violations information.

2.3 User characteristics

The following actors are the users of this application:

- End users

The end user is a person who, after the sign up, can use SafeStreets to notify traffic violations, consult statistics and see the map with violation information for each street.

- Authorities:

Municipality

Local police

Authority is an organization who, after a different sign up from end users, using CUU code, can use SafeStreets to get information about traffic violations for generating traffic tickets, consult possible interventions suggested, see statistics and the map with violation information for each street. Also they can provide their data to SafeStreets, in this way our platform is able to cross authorities data with its own data.

2.3.1 Assumptions, dependencies and constraints

In this document it is supposed that:

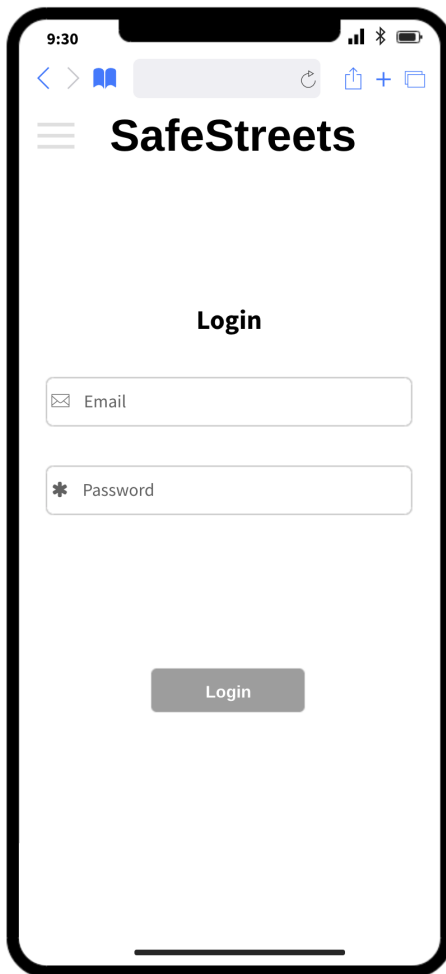
- in the country where SafeStreets will be used, the law admits these type of systems.
- we consider only the following type of violations
 - Double parking
 - Disable parking
 - Bike lane parking

3 Specific Requirements

3.1 External Interface Requirements

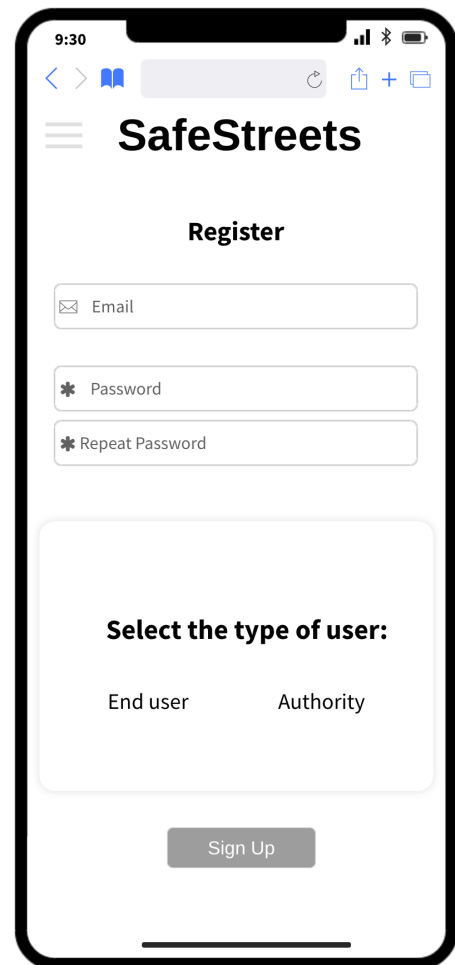
3.1.1 User interfaces

In the design document there will be the specific application UI created after the process (UX) of defining how users interact with SafeStreets. The following mocks describe only the generic application screen design:



The mockup shows a mobile application interface for the 'SafeStreets' app. At the top, there is a status bar with the time '9:30' and various icons. Below the status bar is a navigation bar with a hamburger menu icon, the app name 'SafeStreets', and a search icon. The main content area is titled 'Login'. It contains two input fields: 'Email' with an envelope icon and 'Password' with a star icon. Below these fields is a 'Login' button. The bottom of the screen shows a home indicator bar.

Figure 4: Login form



The mockup shows a mobile application interface for the 'SafeStreets' app. At the top, there is a status bar with the time '9:30' and various icons. Below the status bar is a navigation bar with a hamburger menu icon, the app name 'SafeStreets', and a search icon. The main content area is titled 'Register'. It contains three input fields: 'Email' with an envelope icon, 'Password' with a star icon, and 'Repeat Password' with a star icon. Below these fields is a section titled 'Select the type of user:' with two radio button options: 'End user' and 'Authority'. At the bottom of the form is a 'Sign Up' button. The bottom of the screen shows a home indicator bar.

Figure 5: Registration form

The image shows a mobile app interface for 'SafeStreets'. At the top, there's a status bar with the time 9:30 and various icons. Below that is a navigation bar with a hamburger menu icon, the app name 'SafeStreets', and some utility icons. The main heading is 'Report violation'. Underneath, there's a 'Position:' label followed by a search bar with the placeholder text 'Enter an address'. Below the search bar is a section titled 'Select the violation:' which contains a dropdown menu labeled 'Selected violation' and a list of violations. The list is currently empty, showing only the headers 'List', 'of', and 'violations'. At the bottom of this section is an 'Upload:' label with a 'Choose Image' button and the text 'selected-image'. A 'Send' button is located at the very bottom of the form.

Figure 6: User send violation report

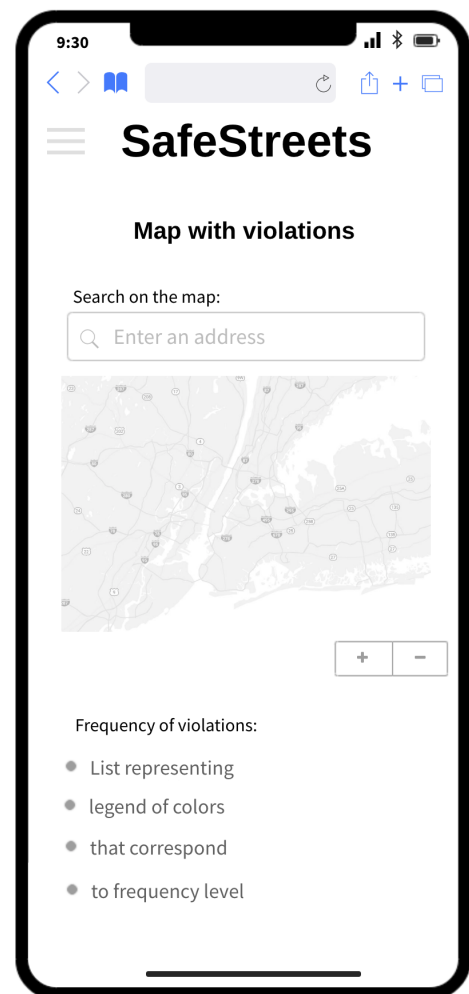


Figure 7: Map showing violations

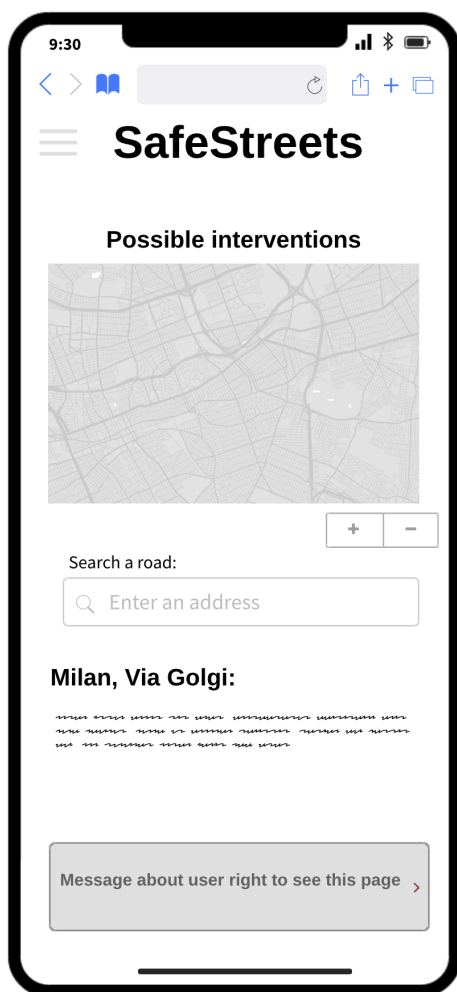


Figure 8: Suggestion for possible interventions

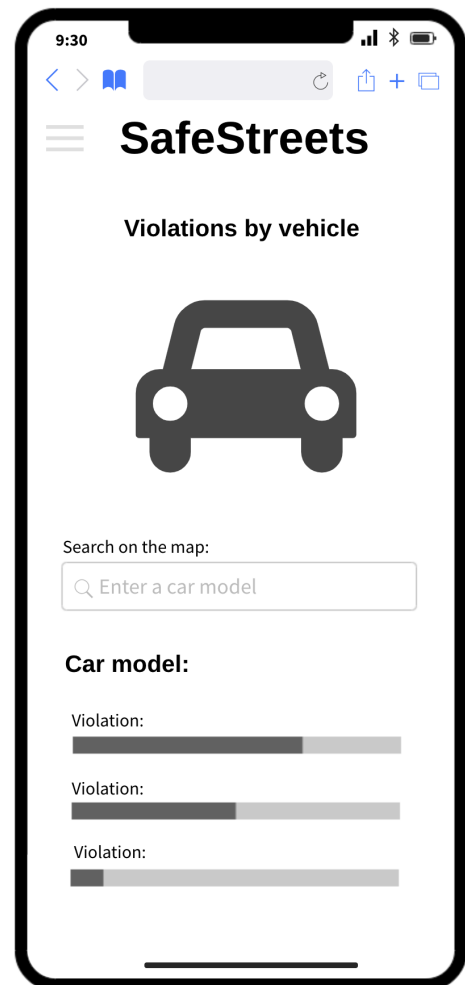


Figure 9: Violations by vehicle

3.1.2 Hardware interfaces

There are no hardware interfaces.

3.1.3 Software interfaces

The system functionalities are provided with the usage of some external web services:

Google Maps Google Maps library is used in the page that shows the map. Furthermore, Google Maps APIs are used to transform addresses in coordinates and vice versa.

Vehicle model identification Using an external web service, SafeStreets is able to retrieve information about the vehicles from the license plate number. This data will be show in statistics about violations by vehicles.

IndicePA The *CUU (Codice Univoco Ufficio)* verification is made by an external web service provided by *IndicePA*. The code is required during the authorities registration to ensure their identity. The external API are exposed only for authorities application and the implementation details will be shown in the design document (DD).

3.1.4 Communication interfaces

Any communication functionality takes place via internet with HTTPS to allow protection and privacy.

This protocol will be used for both access to the web-application and REST communication.

3.2 Design constraints

3.2.1 Standards compliance

Part of the information collected by SafeStreets (e.g. license plate) are sensitive. For this reason the project is subject to the *General Data Protection Regulation (GDPR)*.

One of the technique used by the system to protect these important information is the creation of different type of users.

For example normal account can only send information and see general information. While authorities have a different account that is able to access to the collected and generated data in details.

3.2.2 Hardware limitations

The application is based on retrieving data from the pictures sent by the user.

Obviously the camera of the devices is a crucial aspect to consider during the design process. For this reason a minimum resolution of the sensor is required to install the application.

The identification of this value will be done during design phase and will be written in the design document (DD).

An other constraint is generated by the precision of the GPS sensor of the device, since the impact is smaller than the image resolution, the requirement is not so strict.

3.2.3 Any other constraint

SafeStreets works with violations and traffic tickets so it has to deal with laws and regulations; for this aspect see the specific domain assumption.

Furthermore we have to consider the possible improper use of the platform. To prevent it, SafeStreets does specific controls on images and it is able to detect manipulations or repeated pictures of the same violations.

The implementation of this part will be done with some external algorithm and will be detailed in the design document (DD).

3.3 Functional requirements

3.3.1 Definition of use case diagram

The use case diagrams provide a vision of the uses that can be done with the platform.

They show the relationship between the actors and the use that every actor made with the software to reach the goals.

3.3.2 Scenario 1

Paul wants to sign up to SafeStreets platform. He has to insert his personal data (name, surname, place, mail and so forth). Then he will receive a confirm mail and can use the platform.

3.3.3 Scenario 2

Bob is coming back to his car when he notice that someone double parked and he can't go out from the parking.

He logs in into the SafeStreets and he takes a picture of the car. Then he has to insert the information about the violation: type of violation and route where the violation happened. Once Bob has added the data he can send the report.

3.3.4 Scenario 3

Luke is walking on the sidewalk when he see that a man is parking on a park reserved for disable. Luke logs in into the SafeStreets and he takes a picture of the car. Then he has to insert the information about the violation: type of violation and route where the violation happened. Once Luke has added the data he can send the report.

3.3.5 Scenario 4

Mark has submitted a report to the platform, and he wants to check if it has been accepted or declined. He logs in into the app, he goes to his reserved area and next to his report there is the status of message sent.

3.3.6 Scenario 5

An accidents in via Anzani occurs, the municipality system records the information about it. SafeStreets is able to retrieve a portion of these data and cross it with violations information, in order to generate statistics.

3.3.7 Scenario 6

The local authority wants to check which are the streets with more violations. They log in into the platform and access to their personal area where they can check the streets status.

3.3.8 Scenario 7

The local authority wants to check which are the vehicles with more violations. They log in into the platform and access to their personal area where they can check the cars that have the most reports assigned.

3.3.9 Scenario 8

The system receives pictures from the users and have to check if they are reliable. It runs an algorithm that can detect if the photo is real or it has been manipulated. In the first case, the report will be stored and be used by authorities. In the second case the report will be discarded.

3.4 Description of use case scenarios

3.4.1 Description scenario 1

ACTORS	Visitors
GOALS	Sign up
INPUT CONDITION	Personal data (name, surname, mail, telephone number, place where he lives, car license plate etc.)
EVENTS FLOW	The user from the home page clicks on sign up. He inserts all of his information and if it is all correct he will receive an email of confirm.
OUTPUT CONDITION	The user can now log in and use all the function that the user can do on the platform.
EXCEPTIONS	Data incorrect, user already exists, mail already exists. All of this exceptions will be notified instantly to the user.

3.4.2 Description scenario 2

ACTORS	User
GOALS	Report a violation
INPUT CONDITION	Type of violation (double row park), name of the street,
EVENTS FLOW	The user take a picture of the car that has done the violation. He insert the type of violation and the route where it happened. The system use an algorithm to read the car license plate and then store all the informations once it is established that the photo isn't fake.
OUTPUT CONDITION	The violation has been stored with all the data, it will be generated a traffic ticket, and the local authority can decide to highlight who has made the violation or the street.
EXCEPTIONS	Data incorrect or fake photo. The report will be discarded and no traffic ticket will be generated.

3.4.3 Description scenario 3

ACTORS	User
GOALS	Report a violation
INPUT CONDITION	Type of violation (wrong park), name of the street,
EVENTS FLOW	The user take a picture of the car that has done the violation. He insert the type of violation and the route where it happened. The system use an algorithm to read the car license plate and then store all the informations once it is established that the photo isn't fake.
OUTPUT CONDITION	The violation has been stored with all the data, it will be generated a traffic ticket, and the local authority can decide to highlight who has made the violation or the street.
EXCEPTIONS	Data incorrect or fake photo. The report will be discarded and no traffic ticket will be generated.

3.4.4 Description scenario 4

ACTORS	User
GOALS	Check report status
INPUT CONDITION	User credentials
EVENTS FLOW	The user log in into the platform. He accesses to his personal area and he can check the status of his report
OUTPUT CONDITION	Report status
EXCEPTIONS	The user hasn't made any report. The system show a message that the list of the reports is empty.

3.4.5 Description scenario 5

ACTORS	Authority, SafeStreets
GOALS	Generate statistics
INPUT CONDITION	The authority inserts data about an accident.
EVENTS FLOW	The authority add the data about an accident in their system. SafeStreets retrieve these data generate statistic for the authority.
OUTPUT CONDITION	Accident and violation statistics
EXCEPTIONS	There are no exceptions

3.4.6 Description scenario 6

ACTORS	Authority
GOALS	Check streets status
INPUT CONDITION	Local authority credentials
EVENTS FLOW	The authority log in into the platform. He accesses to his personal area and he can check the streets status
OUTPUT CONDITION	Different colors for the streets based on their status and possibly changes to the color of some streets
EXCEPTIONS	none

3.4.7 Description scenario 7

ACTORS	Authority
GOALS	Check cars status
INPUT CONDITION	Local authority credentials
EVENTS FLOW	The authority log in into the platform. He accesses to his personal area and he can check the cars status based on reports
OUTPUT CONDITION	Different colors for the cars based on how many reports they received and possibly changes to the color of some cars
EXCEPTIONS	none

3.4.8 Description scenario 8

ACTORS	SafeStreets
GOALS	Check the reliability of the photo
INPUT CONDITION	Photo that has been taken by a user.
EVENTS FLOW	The system receives from the user a photo of a possible violation. It runs an algorithm that can establish if it is real or fake.
OUTPUT CONDITION	The photo is real or fake.
EXCEPTIONS	none

3.5 Activity diagram

3.5.1 Scenario 1

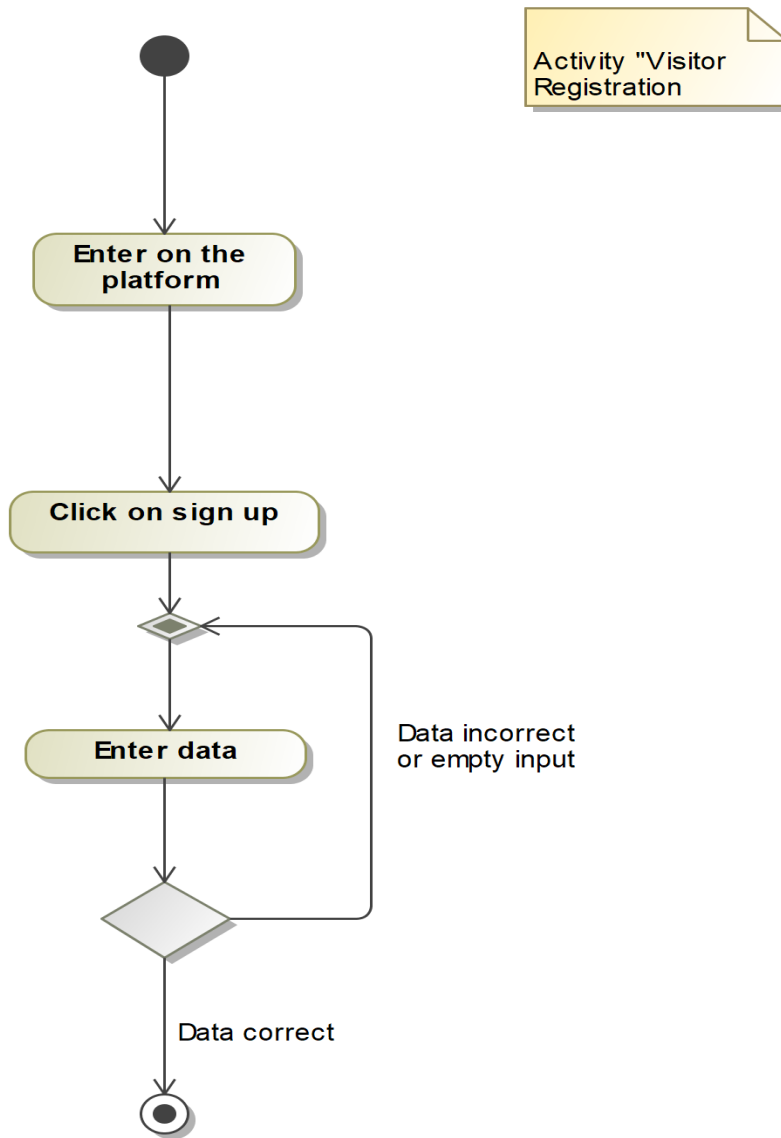


Figure 10: Scenario 1

3.5.2 Scenario 3

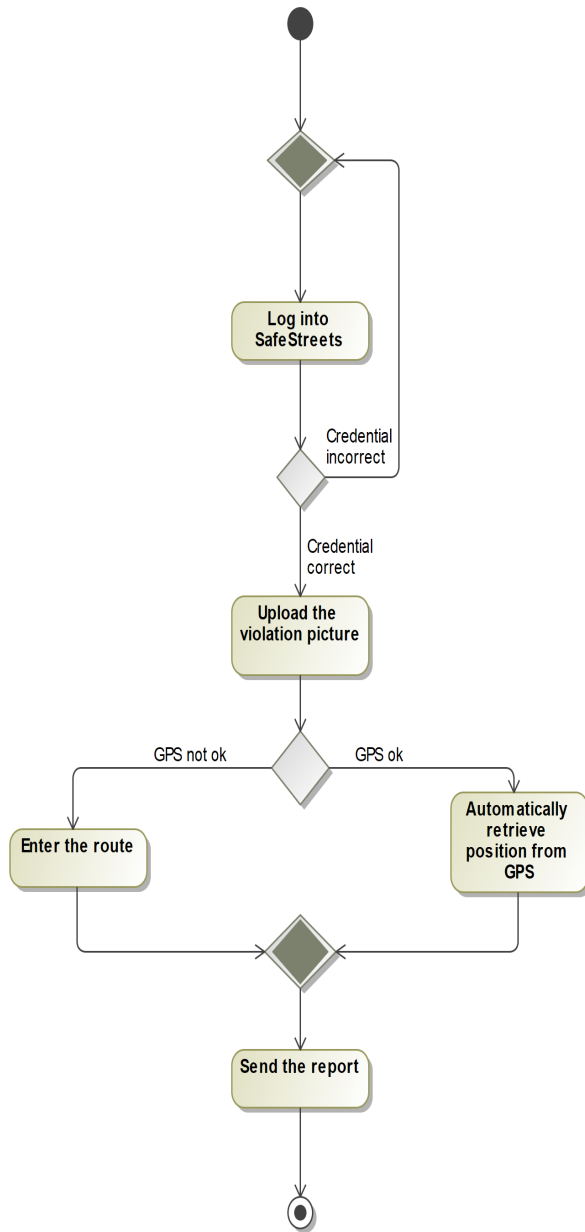


Figure 11: Scenario 3

3.5.3 Scenario 4

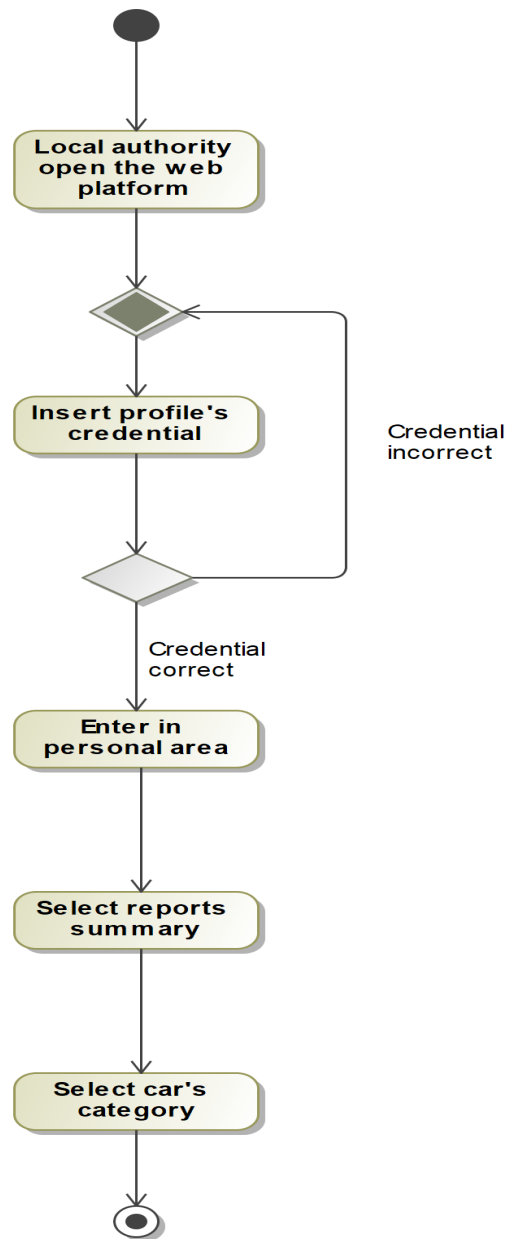


Figure 12: Scenario 4

3.5.4 Scenario 7

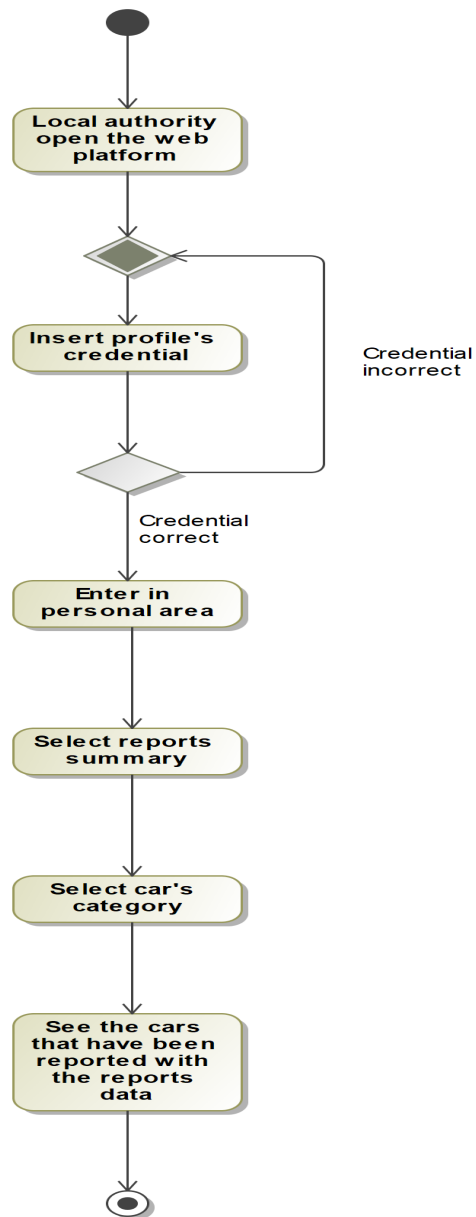


Figure 13: Scenario 7

3.5.5 Scenario 8

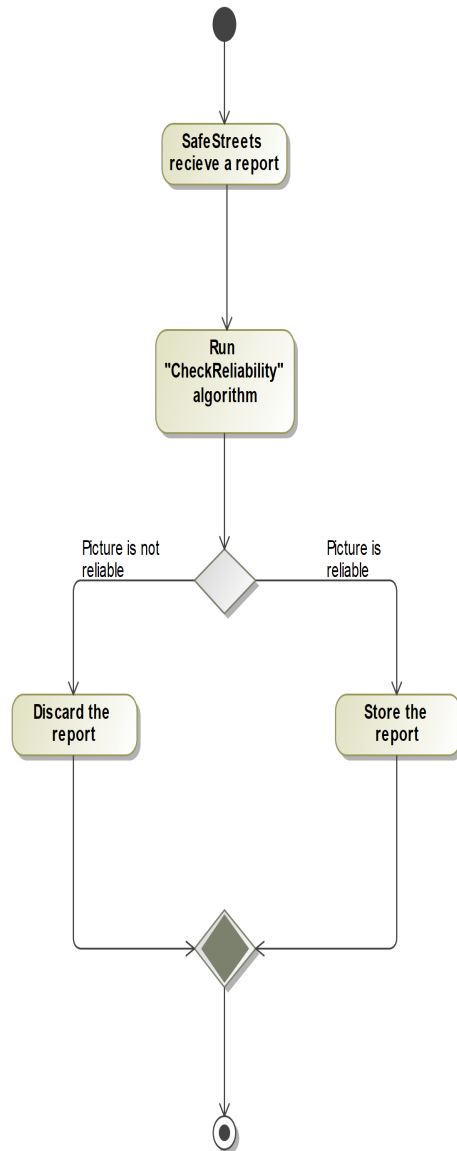


Figure 14: Scenario 8

3.6 Sequence diagram

3.6.1 Scenario 2

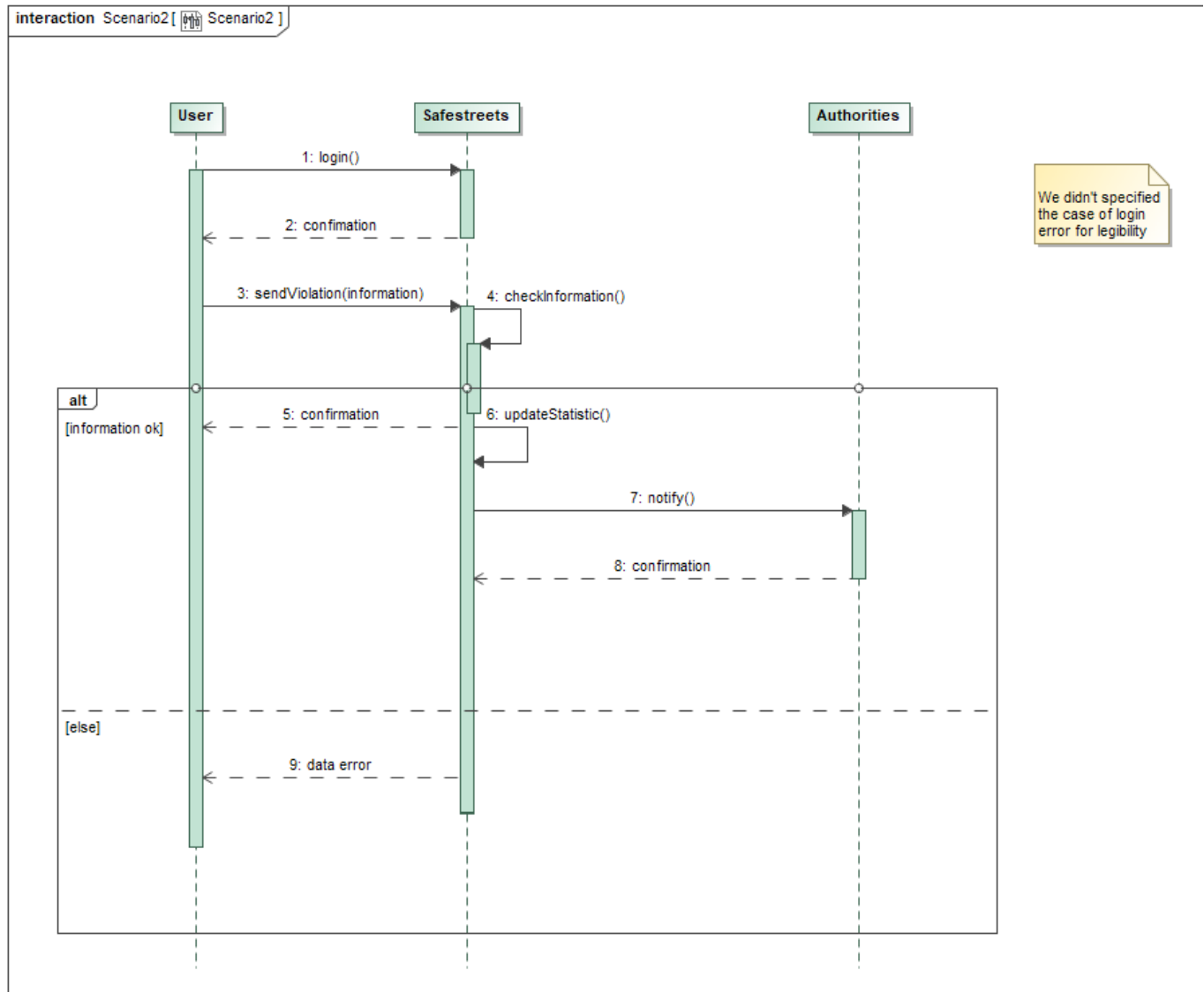


Figure 15: Scenario 2

3.6.2 Scenario 5

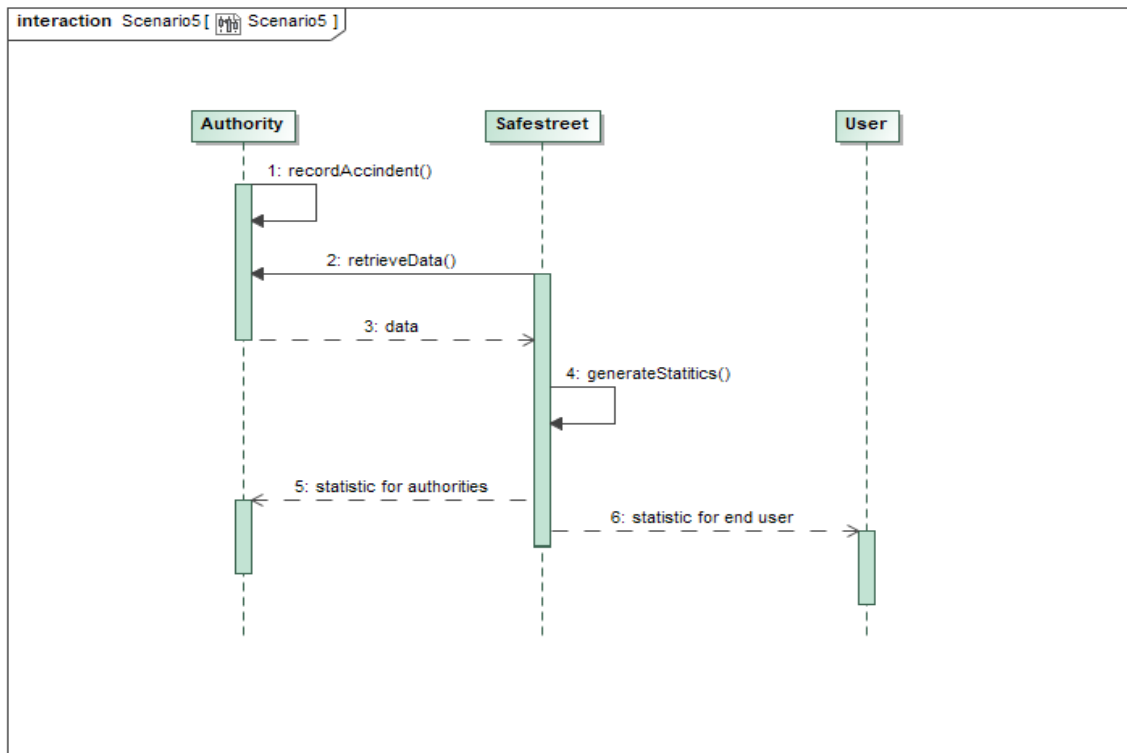


Figure 16: Scenario 5

3.6.3 Scenario 6

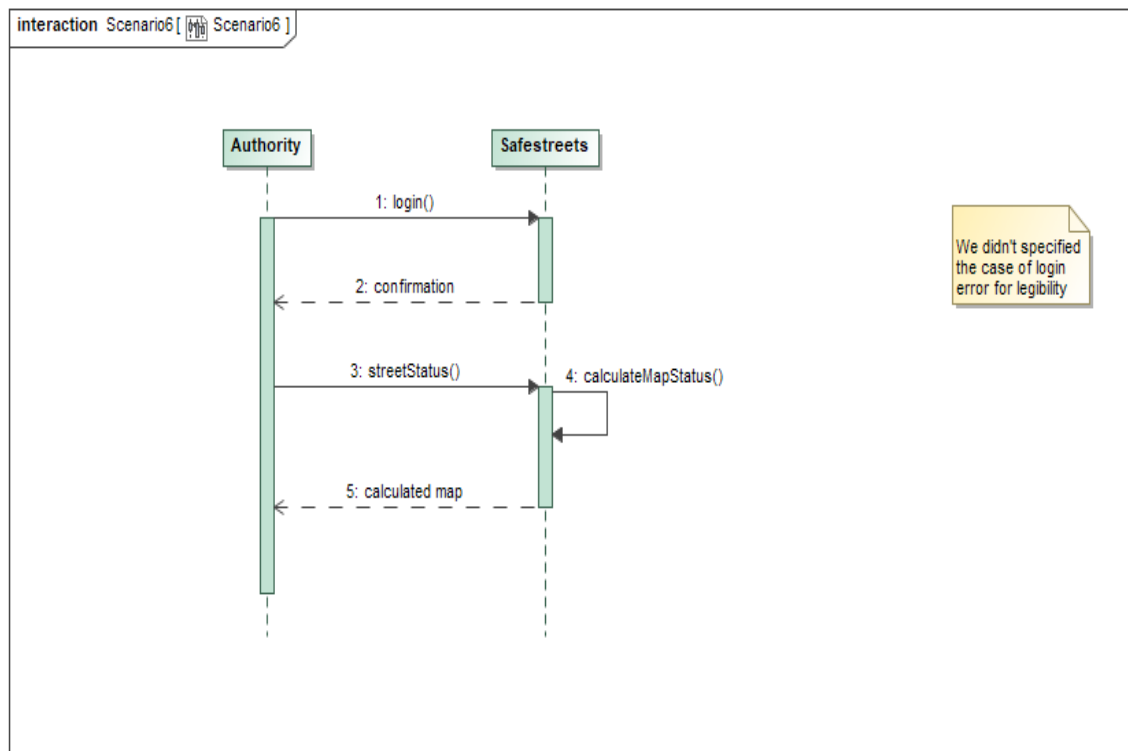


Figure 17: Scenario 6

3.7 Software system attributes

3.7.1 Reliability

The system has to ensure reliability, for this reason we have decided to keep a backup server, updated everyday. Consistency is guaranteed by the algorithm we have already described. Furthermore SafeStreets guarantees reliability in terms of plate recognition because the results are compared with the information coming from users.

3.7.2 Availability

The platform has to be available every day, especially in the rush hours because are the ones where there traffic is higher. SafeStreets must have an availability of 99% (3.65 days/year downtime).

3.7.3 Security

The data have to be encrypted, to grant the privacy (e.g. user's position, car license plate and so forth).

For this reason is used HTTPS protocol to transmit data from user to SafeStreets.

Every account must have a strong password with the combination of upper-case, lower-case letters and numbers.

Chain of custody

The system have to guarantee the chain of custody, thus SafeStreets needs to ensure that the data received from the users are genuine.

In order to implement this feature the system checks the authenticity of the pictures recived before creating and storing new reports.

In this way we have the certainty that the data stored are reliable and all the other portions of the system can work without deal with this aspect (for them it is trasparent).

3.7.4 Maintainability

The system is developed to be compatible with other platform, like the system of the local authority. There could be maintenance interventions, when it's possible there will be in the hours with the less use of the platform (e.g. midnight). There will be released updates, both for web-application and system.

3.7.5 Portability

The entire system is portable, every user (citizen or local authority) can access to their profile, see data, statistics and make reports of violation from their mobile, their tablet or PCs.

3.8 Mapping on requirements

RawID	GoalID	ReqID	Use Case ID
1	G1	RE.1	Scenario 2/3
2	G2.1	RE.2	
3	G2	RE.3	
4	G3.1	RE.4	Scenario 6
5	G3	RE.5	
6	G4	RE.6	
7	G5	RE.7	Scenario 8
8	G2	RE.8	

4 Formal Analysis Using Alloy

During the draft of the alloy, we omitted the aspects regarding the chain of custody because, as mentioned in Security description (3.7.3), these controls are made before storing information in the system, thus the other portions of the application are able to ignore it.

For this reason there is no constraint in Alloy that checks the fact that the report and the image can only exist if the information are genuine.

4.1 Alloy code

```

open util/time
open util/integer

sig Str{}

--Account
abstract sig User{
  username: one Str,
  password: one Str,
  email: one Str,
  seeGroupedStatistics: lone GroupedStatistics
}

sig EndUser extends User{
  name: one Str,
  surname: one Str
}

sig Authority extends User{
  CUU: one Str,
  name: one Str,
  seeCityStatistic: lone CityStatistic
}

sig Image{
  capturedLicenseplate: lone Licenseplate
}

sig Report{
  user: one EndUser,
  position: one Street,
  violation: one ViolationEnum,
  picture: one Image,
  licenseplateHelper: lone Licenseplate,
  datetime: one Time
}

```

Figure 18: Signature 1

```
sig Vehicle{
  vehicleplate: one Licenseplate,
  type: one Str
}

sig Licenseplate{}

sig Street{
  name: one Str,
  city: one City
}

sig City{
  cityName: one Str
}

sig CityStatistic{
  relatedCity: one City,
  fromCityReports: set Report
}

lone sig GroupedStatistics{
  fromReports: set Report,
  fromAccidents: set Accident
}

sig Accident{
  position: lone Street,
  datetime: one Time
}

sig Suggestion{
  description: one Str,
  relatedviolation: one ViolationEnum
}
```

Figure 19: Signature 2

```
sig Licenseplate{}

sig Street{
  name: one Str,
  city: one City
}

sig City{
  cityName: one Str
}

sig CityStatistic{
  relatedCity: one City,
  fromCityReports: set Report
}

lone sig GroupedStatistics{
  fromReports: set Report,
  fromAccidents: set Accident
}

sig Accident{
  position: lone Street,
  datetime: one Time
}

sig Suggestion{
  description: one Str,
  relatedviolation: one ViolationEnum
}

sig Violation{}

abstract sig ViolationEnum{}

one sig DoubleParking, DisableParking, BikeLaneParking extends ViolationEnum{}
```

Figure 20: Signature 3


```

fact usernameIsUnique{
  no disjoint u1, u2: User | u1.username = u2.username
}

fact emailIsUnique{
  no disjoint u1, u2: User | u1.email = u2.email
}

fact authorityIsUnique{
  no disjoint a1, a2: Authority | a1.CUU = a2.CUU
}

fact ImageRelatedOnlyToOneReport {
  all i: Image | #picture.i = 1
}

fact LicenseplateRelatedOnlyToOneVehicle {
  all l: Licenseplate | #vehicleplate.l = 1
}

fact noReportWithoutCityStatistic{
  all r: Report | #fromCityReports.r > 0
}

fact noSameStreetInTheSameCity{
  all s1, s2: Street |
  s1.name = s2.name and s1 != s2
  implies s1.city != s2.city
}

```

Figure 21: Pred and Facts 1

```

pred reportFromSameCityInCityStatistic{

  all r: Report, c: City, cs: CityStatistic |
  cs.relatedCity = c and r.position.city = c
  implies r in cs.fromCityReports

  all r: Report, cs: CityStatistic |
  r in cs.fromCityReports
  implies r.position.city = cs.relatedCity
}

fact userForStat{
  all u: User | #GroupedStatistics = 1
  implies #u.seeGroupedStatistics = 1
}

pred groupedStatisticsHasAllReportsAndAccindets{

  all gs: GroupedStatistics | #gs.fromReports > 0 or #gs.fromAccidents > 0

  all r: Report, gs: GroupedStatistics | r in gs.fromReports

  all a: Accident, gs: GroupedStatistics | a in gs.fromAccidents
}

fact reportCanOnlyExistIfHasLicenseplateInfo {

  all r: Report | #r.picture.capturedLicenseplate = 1 or #r.licensePlateHelper = 1
}

pred licenseplateHaveSameValueOfLicenseplateHelper{

  all r: Report | #r.picture.capturedLicenseplate = 1 and #r.licensePlateHelper = 1
  implies r.picture.capturedLicenseplate = r.licensePlateHelper
}

```

Figure 22: Pred and Facts 2

```
fact reportCanOnlyExistIfHasLicenseplateInfo {  
    all r: Report | #r.picture.capturedLicenseplate = 1 or #r.licensePlateHelper = 1  
}  
  
pred licenseplateHaveSameValueOfLicenseplateHelper{  
    all r: Report | #r.picture.capturedLicenseplate = 1 and #r.licensePlateHelper = 1  
    implies r.picture.capturedLicenseplate = r.licensePlateHelper  
}
```

Figure 23: Pred and Facts 3

```

pred groupedStatisticsHasAllReportsAndAccidents{

  all gs: GroupedStatistics | #gs.fromReports > 0 or #gs.fromAccidents > 0

  all r: Report, gs: GroupedStatistics | r in gs.fromReports

  all a: Accident, gs: GroupedStatistics | a in gs.fromAccidents
}

fact reportCanOnlyExistIfHasLicenseplateInfo {

  all r: Report | #r.picture.capturedLicenseplate = 1 or #r.licenseplateHelper = 1
}

pred licenseplateHaveSameValueOfLicenseplateHelper{

  all r: Report | #r.picture.capturedLicenseplate = 1 and #r.licenseplateHelper = 1
  implies r.picture.capturedLicenseplate = r.licenseplateHelper
}

fact reportCanOnlyExistIfHasLicenseplateInfo {

  all r: Report | #r.picture.capturedLicenseplate = 1 or #r.licenseplateHelper = 1
}

//world1
run groupedStatisticsHasAllReportsAndAccidents for 5 but exactly 1 GroupedStatistics, exactly 5 Report, exactly 2 Accident

//world2
run reportFromSameCityInCityStatistic for 4 but exactly 2 CityStatistic, exactly 2 Report, 2 City

//world3
run licenseplateHaveSameValueOfLicenseplateHelper for 5 but exactly 3 Report

```

Figure 24: Pred, Facts and World

4.2 Predicates testing

4.2.1 World 1

Predicate about the aggregation of data coming from reports with the traffic information.

This predicate describes the constraint that the signature that represent statistics can only exist if and only if there is at least one report or accident. Furthermore all reports and all accidents must be related to statistics.

```
pred groupedStatisticsHasAllReportsAndAccidents{
  all gs: GroupedStatistics | #gs.fromReports > 0 or #gs.fromAccidents > 0
  all r: Report, gs: GroupedStatistics | r in gs.fromReports
  all a: Accident, gs: GroupedStatistics | a in gs.fromAccidents
}

run groupedStatisticsHasAllReportsAndAccidents for 5 but exactly 1 GroupedStatistics, exactly 0 Report, exactly 0 Accident
```

Figure 25: Pred 1

To show that the predicate works as expected, we have run the predicate with with inconsistent number of sig entities: exactly of 0 reports and 0 accidents but 1 GroupedStatistic

```
Executing "Run groupedStatisticsHasAllReportsAndAccidents for 5 but exactly 1 GroupedStatistics, exactly 0 Report, exactly 0 Accident"
Solver=minisat(jni) Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
0 vars. 0 primary vars. 0 clauses. 12ms.
No instance found. Predicate may be inconsistent. 0ms.
```

Figure 26: Result pred 1

The following represented world is generated from the following run: run groupedStatisticsHasAllReportsAndAccidents for 5 but exactly 1 GroupedStatistics, exactly 5 Report, exactly 2 Accident

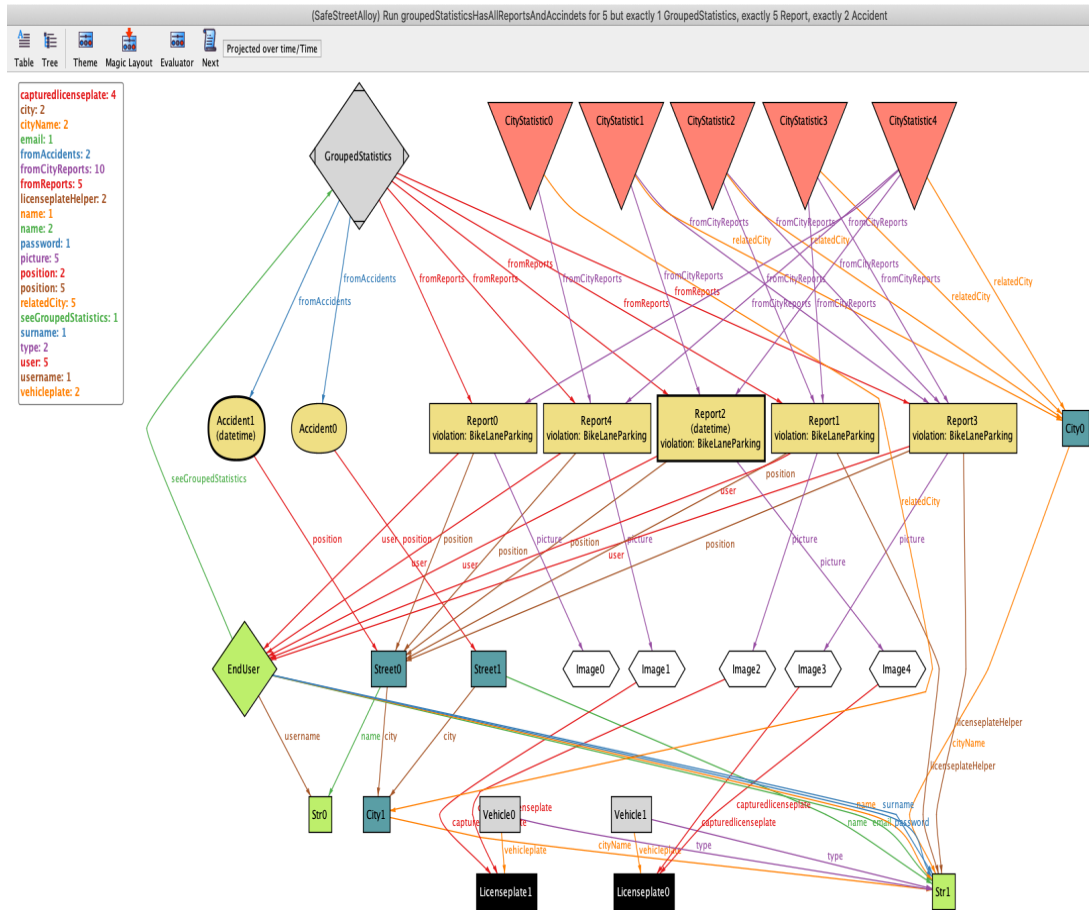


Figure 27: World 1 generated by pred 1

4.2.2 World 2

Consistency between city and statistics.

This predicate describes the constraint that the signature that represent statistics specific about a city and all reports about all streets of the city must be related.

```

pred reportFromSameCityInCityStatistic{
    all r: Report, c: City, cs: CityStatistic |
    cs.relatedCity = c and r.position.city = c
    implies r in cs.fromCityReports

    all r: Report, cs: CityStatistic |
    r in cs.fromCityReports
    implies r.position.city = cs.relatedCity
}

run reportFromSameCityInCityStatistic for 5 but exactly 0 CityStatistic, exactly 3 Report

```

Figure 28: Pred 2

To show that the predicate works as expected, we have run the predicate with inconsistent number of sig entities: exactly 0 CityStatistic, exactly 3 Report.

```

Executing "Run reportFromSameCityInCityStatistic for 5 but exactly 0 CityStatistic, exactly 3 Report"
Solver=minisat(jni) Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
0 vars. 0 primary vars. 0 clauses. 12ms.
No instance found. Predicate may be inconsistent. 0ms.

```

Figure 29: Result pred 2

The following represented world is generated from the following run: run reportFromSameCityInCityStatistic for 4 but exactly 2 CityStatistic, exactly 2 Report, 2 City

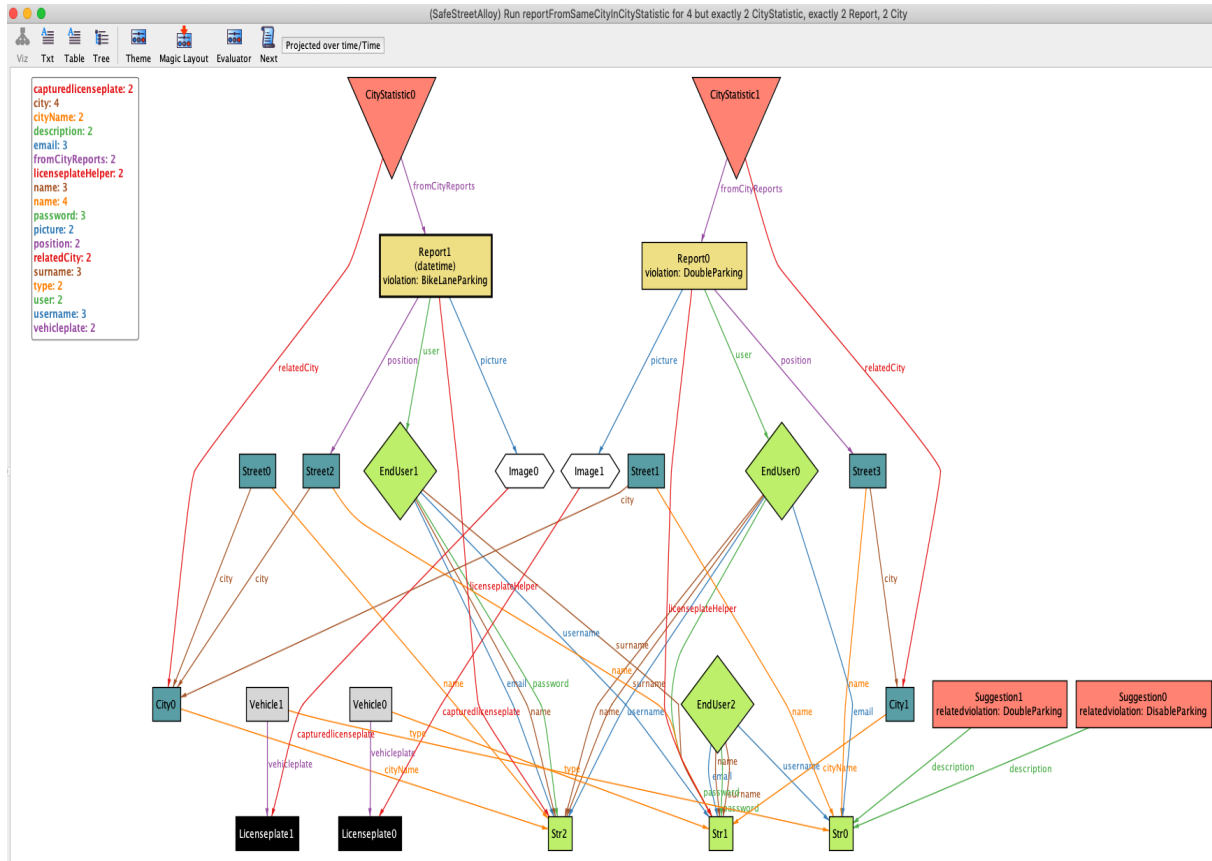


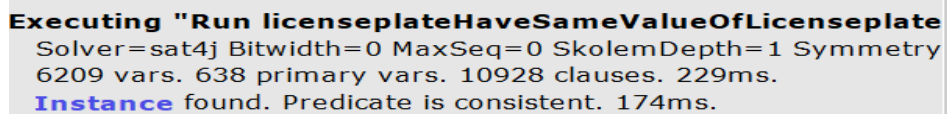
Figure 30: World 2

4.2.3 World 3

Predicate that test if the licenseplate retrieve from SafeStreets is the same suggested by the user.

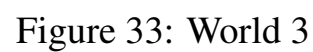
```
pred licenseplateHaveSameValueOfLicenseplateHelper{  
  
  all r: Report | #r.picture.capturedLicenseplate = 1 and #r.licenseplateHelper = 1  
  implies r.picture.capturedLicenseplate = r.licenseplateHelper  
  
}
```

Figure 31: Predicate 3



```
Executing "Run licenseplateHaveSameValueOfLicenseplate"  
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry  
6209 vars. 638 primary vars. 10928 clauses. 229ms.  
Instance found. Predicate is consistent. 174ms.
```

Figure 32: Result pred 3



5 Effort Spent

5.0.1 Ivan Cavadini

TASK	TIME
Domain model	1h
Class diagram	3h
Scenarios	3h
Use case diagram	2h
Activity diagrams	2h
Software system attributes	3h
Alloy	6h
Various	3h
TOTAL	23h

5.0.2 Nicolò Molinari

TASK	TIME
Purpose	1h
Class diagram	3h
Statechart diagrams	2h
Software interfaces	1h
Sequence diagrams	2h
Mapping on requirements	1h
Design constraints	3h
Alloy	6h
Various	4h
TOTAL	23h

5.0.3 Luigi Pederzani

TASK	TIME
Mockups	5h
Class diagram	3h
Product functions	2h
User characteristics	1h
User Interfaces	2h
Communication interfaces	1h
Alloy	6h
Various	3.5h
TOTAL	23.5h

5.1 Versions

VERSIONS	DESCRIPTION
1.0	Introduction creation
1.1	Introduction review
2.0	Requirements draft
2.1	Goals review
2.2	Correction of scenarios
2.3	Added statechart diagrams
2.4	Added activity and sequence diagrams
3.0	Creation of overall description
3.1	Added class diagram
3.2	Correction of class diagram
4.0	First Alloy draft
4.1	Last Alloy draft
4.2	Alloy review
5.0	Document review
5.1	Final version

5.2 Used Tools

- TeXstudio 2.12.14
- Magic Draw 19.0
- Alloy Analyzer 5.1.0
- MockFlow
- Dropbox Paper

6 References

- Specification document: "SafeStreets mandatory project assignment"
- IEEE 29148 - 2011: "IEEE standard on requirement engineering"
- RASD sample from previous year
- Slides from the course Software Engineering 2
- Alloy material