POLITECNICO
MILANO 1863

SAFESTREETS

REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT

Sergio Cuzzucoli
Daniele De Dominicis

10 NOVEMBER,2019

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Cuzzucoli Sergio , De Dominicis Daniele
Version:	1.1
Date:	3-December-2019
Download page:	https://github.com/Danieledodo/CuzzucoliDeDominicis.git
Copyright:	Copyright © 2019, Cuzzucoli Sergio, De Dominicis Daniele – All rights reserved

Contents

List of Figures

1 Introduction

1.1 Purpose

Safestreet is the name of an application aiming to offer principally the possibility to signal traffic or parking violation via the internet to public authorities. This document aims to outline the functioning of said app, providing a deep insight of the software and supporting the stakeholders.

Safestreet offers, as a basic functionality, the possibility to fill in a form detailing one of the possible violations (e.g. illegal parking) to every single user. The application gives its users the possibility to compile a form complete with a brief description of the wrongdoing and several additional information. Said information consists of the position of the user at the time of the report (street name) and some pictures, to be attached to the form itself in some way (for example using the smartphone on which the app is installed). The form can then be either approved and investigated or discarded by the public authorities working at the interested area (in both cases the user is warned of the end result of their report). At the same moment Safestreet stores data received by those warnings and computes them to highlight areas in which a violation is most likely to occur, or to show which type of vehicle tends to be associated with different wrongdoings.

In addition to what is written above, the data mined from the warnings can be crossed with that offered by the municipality. If a specific part of the city presents an alarming number of reports it may mean that the infrastructures provided to the population are inadequate or that part of the citizens (especially the weaker ones such as bikers or the elderly) is not preserved enough, thus some suggestions can be made to the authorities in order to avoid further inconveniences.

1.2 Goals

The goals set are expressed from the point of view of the S2B:

G1 The application has to store all the information about violations sent to it, until a ticket is either dropped or accepted by an authority

G2 The system must accepts reports issued from every part of the covered area

G3 The system must allow authorities to access reports in every part of the covered area

G4 The application allows users and authorities to mine information on the system

G5 The application identifies unsafe areas crossing its informations with those offered by the municipality

G6 The application suggests possible solution to problems perceived after the crossing of information

1.3 Scope

Safestreet poses itself as an intermediate between users and authorities.
Users file reports via the mobile app. Authorities (e.g. a police person) examine the report and can discard it or investigate it, leading to a positive result (fine or warning) or negative one (no violation revealed).
Saved reports are analyzed and data extracted by them can be consulted both by users and authorities (at different levels). If the municipality offers data about accidents, they are crossed with those obtained by SafeStreets and suggestion based on them are made. Data is visible to everyone (at different levels).

1.3.1 World Phenomena

- **Broken rules**: Drivers commit violations
- **Observation**: Citizens observe possible violations
- **Inspection**: Authorities inspect possible violations
- **Unfortunate events**: Accidents happen

1.3.2 Machine Phenomena

- **DBMS**: All operations to retrieve or store data
- **Classification**: Statistics are redacted and streets ordered accordingly

1.3.3 Shared Phenomena

Controlled by the world and observed by the machine

- **Login**: Citizens undergo the SMS procedure and authorities login using ID and password
- **File report**: Citizens file a report
- **Inspect report**: Authorities discard or complete a report

Controlled by the machine and observed by the world

- **Show data of violations**: Violations are listed and accessible by everyone
- **Show data of accidents**: Accidents are listed and accessible by everyone
- **Show suggestions**: Suggestion are made according to accidents and violations, for the authorities to see

1.3.4 Table of Phenomena

Phenomenon	Shared	Who controls it
Drivers commit violations	N	W
Citizens observe possible violations	N	W
Athorities inspect possible violations	N	W
Accidents happen	N	W
All operations to retrieve or store data	N	M
Statistics are redacted and streets ordered accordingly	N	M
Citizens undergo the SMS procedure and authorities login using ID and password	Y	W
Citizens file a report	Y	W
Authorities discard or complete a report	Y	W
Violations are listed and accessible by everyone	Y	M
Accidents are listed and accessible by everyone	Y	M
Suggestion are made according to accidents and violations, for the authories to see	Y	M

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Definitions

- Smartphone: Device that allows for calls together with internet connectivity and GPS tracking
- User: Citizen who has installed Safestreet on their smartphone
- Authority: Public official or ent that monitors the territory and mantains the order
- Violation: Act of breaking rules of the road code
- Accident: Violent collision involving vehicles (motorized or not) and/or animate or inanimate objects

1.4.2 Acronyms

Acronym	Meaning
DB	Data Base
DBMS	Data Base Management System
API	Application Program Interface
UI	User Interface
OS	Operating System
RASD	Requirement Analysis and Specification Document
GPS	Global Positioning System
OTP	One Time Password

1.4.3 Abbreviation

G.th : n-th Goal

D.th : n-th Domain Assumption

R.th : n-th Functional Requirement

1.5 Revision History

- **Version 1.0 :**
  -first release
- **Version 1.1 :**
  -removed CNN(Convolutional Neural Network);
  -inserted portability in order to avoid ambiguity;
  -improved readability

1.6 Reference Documents

- Project assignment specifications:[1]
- Alloy Tools: [2]
- UML: [3]

1.7 Document Structure

- **Overall description**: a general description of the service is provided, together with a furter inspection on shared phenomena. Relevant functions of the System are explained. Finally, through the specifications of constraints, dependencies and assumptions, it is unveiled how the System is integrated in the real world.
- **Specific requirements**: aimed to both designers and testers, all the aspects of the previous section are put into perspective. Requirements are fully explained and divided by types and possible interactions with the System are investigated through use cases and sequence diagrams.

2 Overall Description

2.1 Product Perspective

In this paragraph we shall deepen the shared phenomena mentioned in the previous section, providing an oversight of the domain model on different levels of specification, by means of class and state diagrams.

- **Login (controlled by the world and observed by the machine)**: On every opening of the app, users must insert their phone number, then the code generated by the server and sent to them by SMS. Authorities enter using their ID code and password provided to them in a previous moment.
- **File report (controlled by the world and observed by the machine)**: Users must take a photo of the violation and specify the type, position is obtained by the tracker in the phone, date and time are provided by the server, and a brief description can be attached. Nowithstanding that the system exploits convolutional neural network to identify the license plate of the vehicle in the photos, the user must add the number to improve the accuracy of the violation. In case of incomplete identification the user is warned and invited to retake the photo (violations with no photo or incomplete plate identification are not accepted, and the user must start over). If the violation complies with all this rules, it is sent to Safestreets' DBMS.
- **Inspect violation (controlled by the world and observed by the machine)**: Authorities have access to all the pending violations of their city. They can discard the violation (its data is erased by the DBMS) or inspect it. Exploiting Google Maps' API, they are guided to the position of the violation. The result can be either positive (the violation is verified and a fine or a warning is issued) or negative. In the former option the violation is saved on the database, in the latter one, the data is discarded.
- **Show data of violations (controlled by the machine and observed by the world)**: All the successful violations are stored on the DB with statistics regarding every street. Streets are ordered by the number of violations committed during the previous month in decreasing orderd. Users can select a street and acknowledge the number of violation type by type, authorities see also the list of the plates of the offenders.
- **Show data of accidents (controlled by the machine and observed by the world)**: If the municipality offers data about accidents it is crossed with that of Safestreets, street by street. Streets are ordered by the number of accidents, those who have a number of accidents equal to or greater than the treshold (for example 12) are marked unsafe, and the most common type of violation is shown. The number of accidents can be seen by everyone.
- **Show suggestions (controlled by the machine and observed by the world)**: Suggestion are formulated for unsafe streets based on the most common type of violation, they can be seen only by the authorities.

Following are the class and state diagrams:

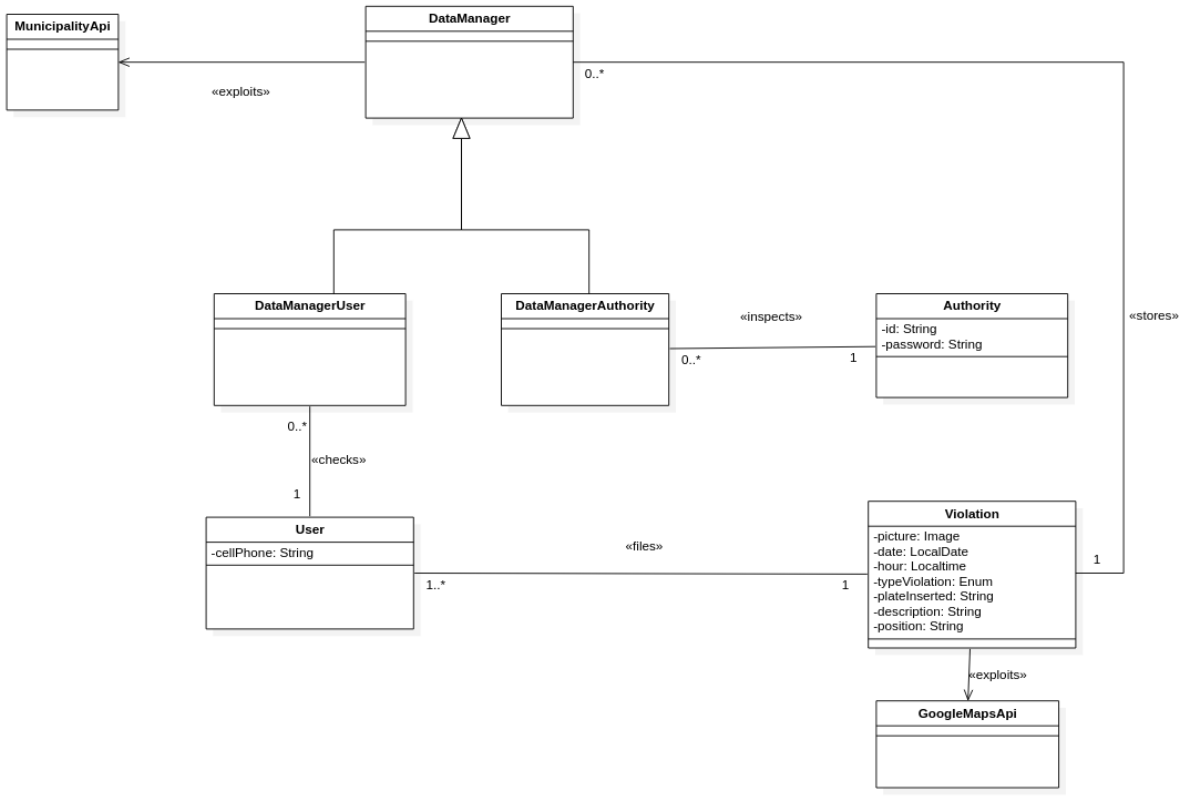


Figure 1: Class Diagram

User and Authority are set as different objects as they undergo different kinds of authentication and have different levels of priviledges

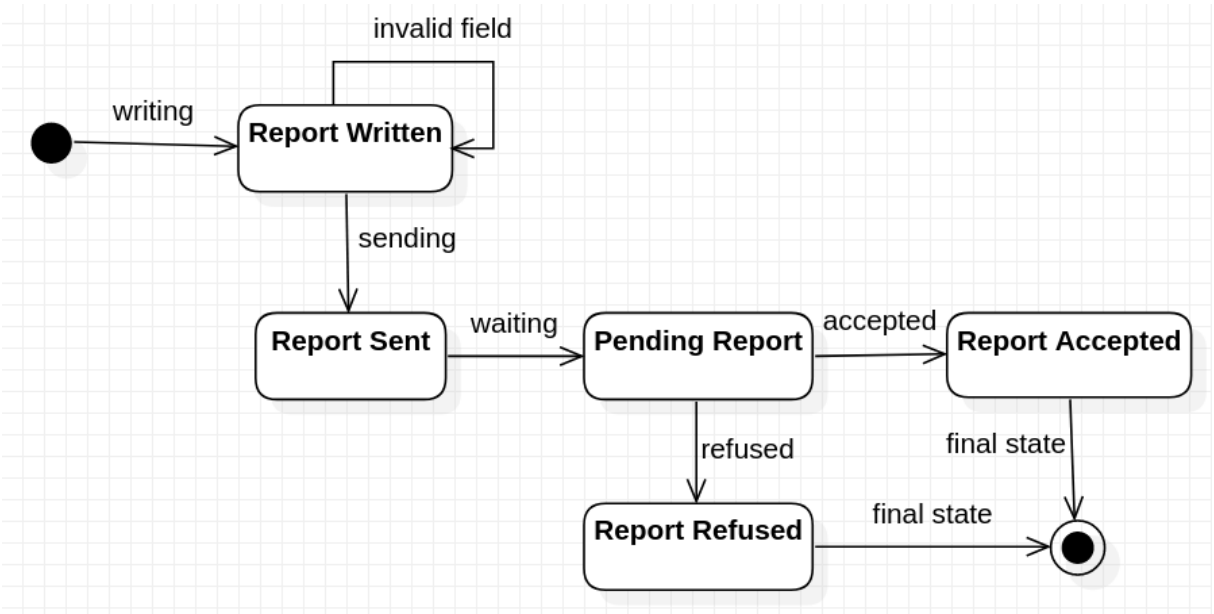


Figure 2: Statechart diagram for the basic function (user)

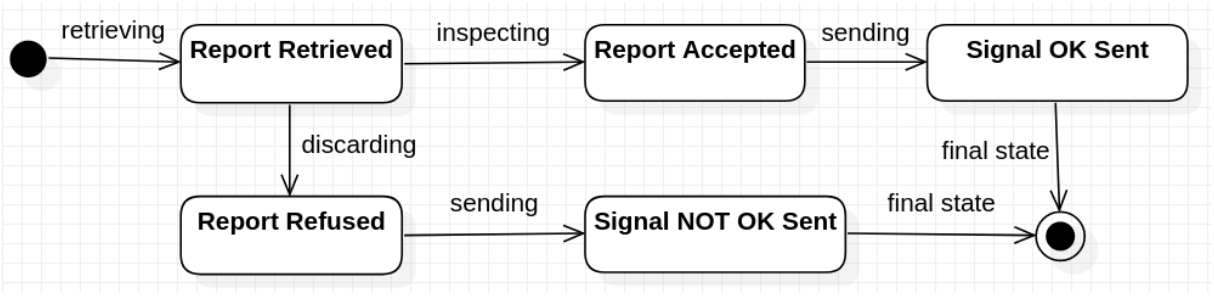


Figure 3: Statechart diagram for the basic function (authority)

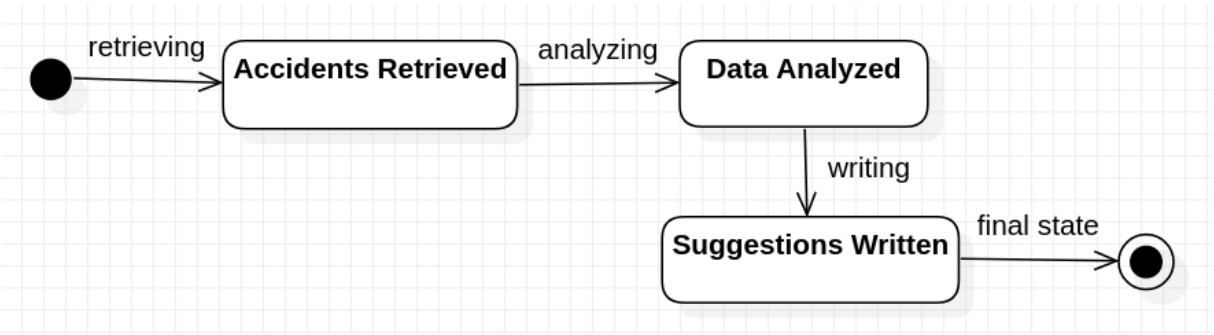


Figure 4: Statechart diagram for the advanced function

2.2 Product Functions

Here the most important features of the product are outlined:

2.2.1 Data management

Data offered by the municipality are replicated and not modified.

Data management by the users

Users are allowed to send information about their violation on the DB but have no power over their conservation.

Data management by the authorities

Authorities have the power to delete (discard) a case, or to inspect it and close it, keeping it on the DB in the end.

2.2.2 Data Requests

Users and authorities can retrieve information stored previously on the DB by the means of a DBMS.

Requests by the users

Users are unable to see their previous report but can retrieve general information about violations under the form of a list of streets, ordered by the one with the greatest number of violation to the one with the smallest. Every street can be highlighted and the types of violations are ordered in a similar fashion. They can also access a list of streets with their respective number of accidents, ordered from the most unsafe street to the least unsafe one.

Requests by the authorities

Authorities can access all the pending violations in their city. They have the same powers as the citizens but they can retrieve more info. When accessing the list of streets by violation, they are allowed to see the plates of repeated offenders of the street, from the most incorrect to the least. Upon inspection of the street, taken by the list which focuses on accidents, they are prompted with suggestions on how to improve the security of the street itself.

2.3 Users characteristics

- **User**: A citizen who has SafeStreets installed on their smartphone.
- **Authority**: Social community workers or administrative workers who keep control of the in general,

2.4 Assumptions, dependencies and constraints

2.4.1 Assumptions

Basic Service

- **D1**: The smartphone of the user can provide accurate location
- **D2**: The data provided by the user is correct
- **D3**: Internet connection is reliable
- **D4**: There is no failure in communication
- **D5**: GM is always on and ready to communicate
- **D6**: Data storage is reliable

Advanced Funcionality

- **D7**: Data offered by municipality is correct
- **D8**: Data offered by municipality is up-to-date

2.4.2 Dependencies

- The system needs a DBMS to store and retrieve data
- GM is provided by Google

2.4.3 Constraints

- Users must be equipped with smartphones provided with camera, internet connection and GPS tracker
- Authorities must be equipped with smartphones provided with camera, internet connection and GPS tracker or a computer with a working internet connection and a browser installed upon.
- Users' smartphone mount an OS compatible with the application
- Authorities' smartphone mount an OS compatible with the application
- Authorities' computer has an internet browser installed upon

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

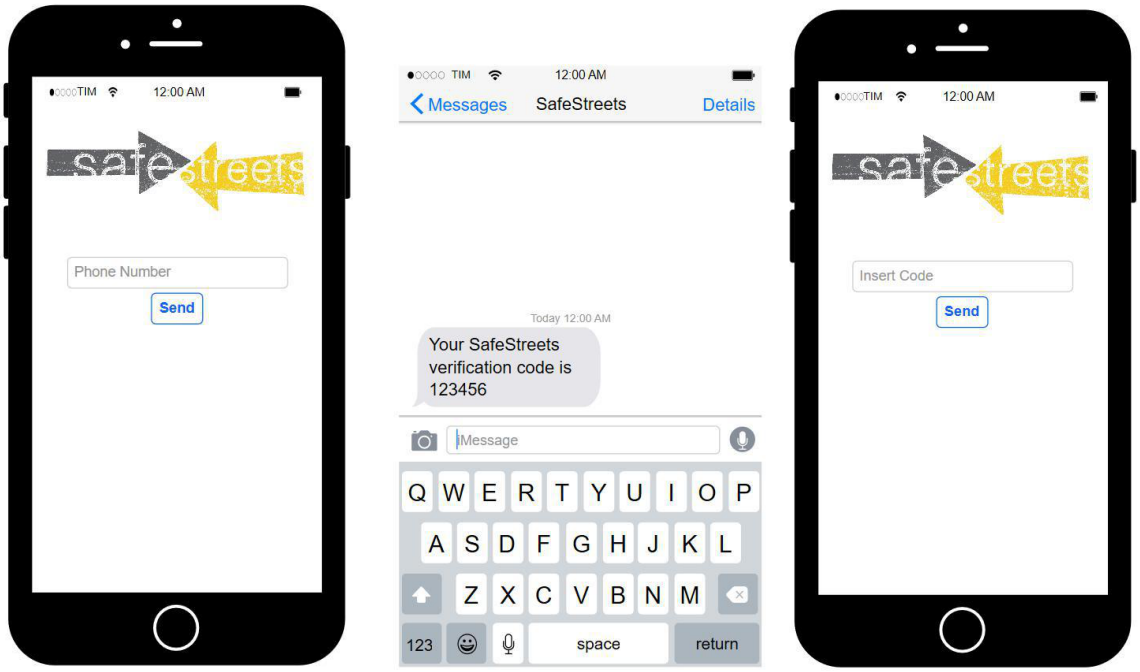


Figure 5: Login

This is the first page the users are shown. To continue they have to input their cellphone number and to insert an OTP sent to them by the system.

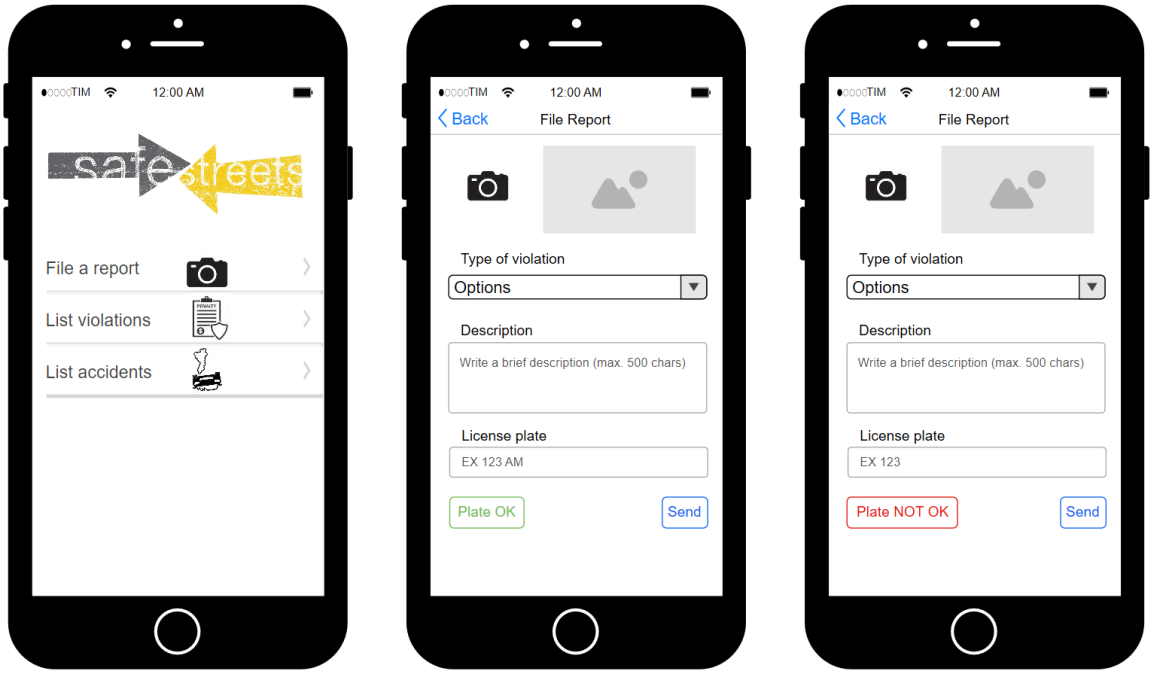


Figure 6: Menu and File Report section

In this case it is shown how the report of violations is handled. The users enters the correct area and fills all the fields.

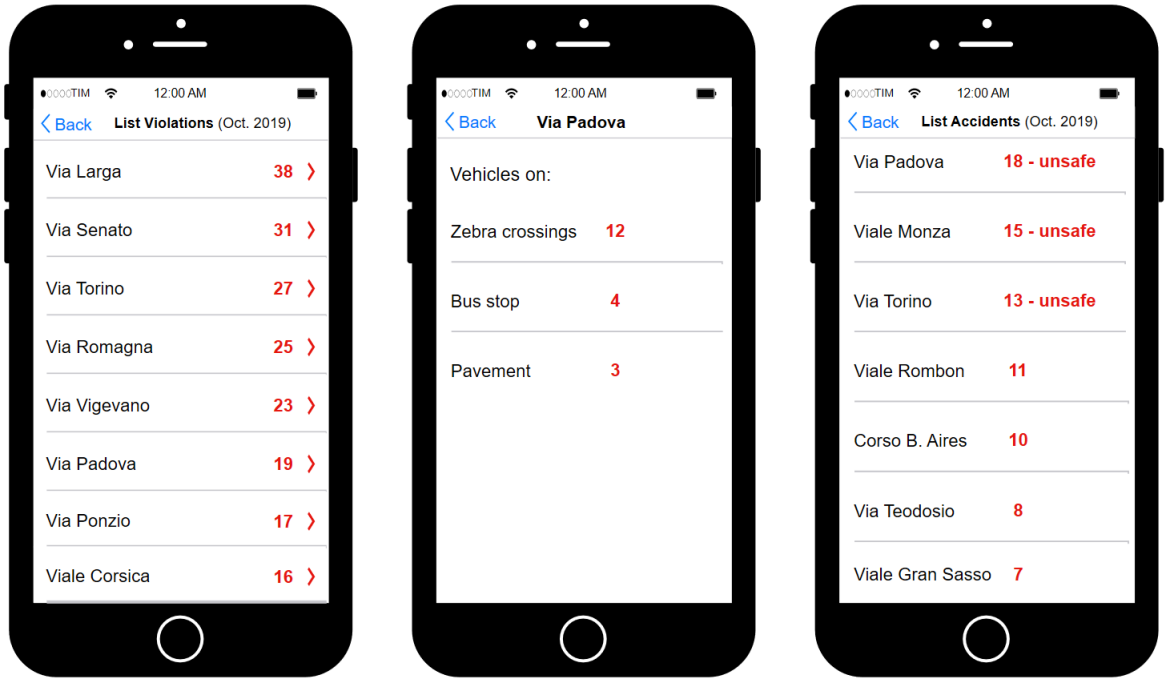


Figure 7: List Violations and List Accidents sections

Upon entering on the apposite areas, the users are shown a list of streets with information about violations. If a precise one is selected, the types are linked to their quantities.

3.1.2 Authority Interfaces

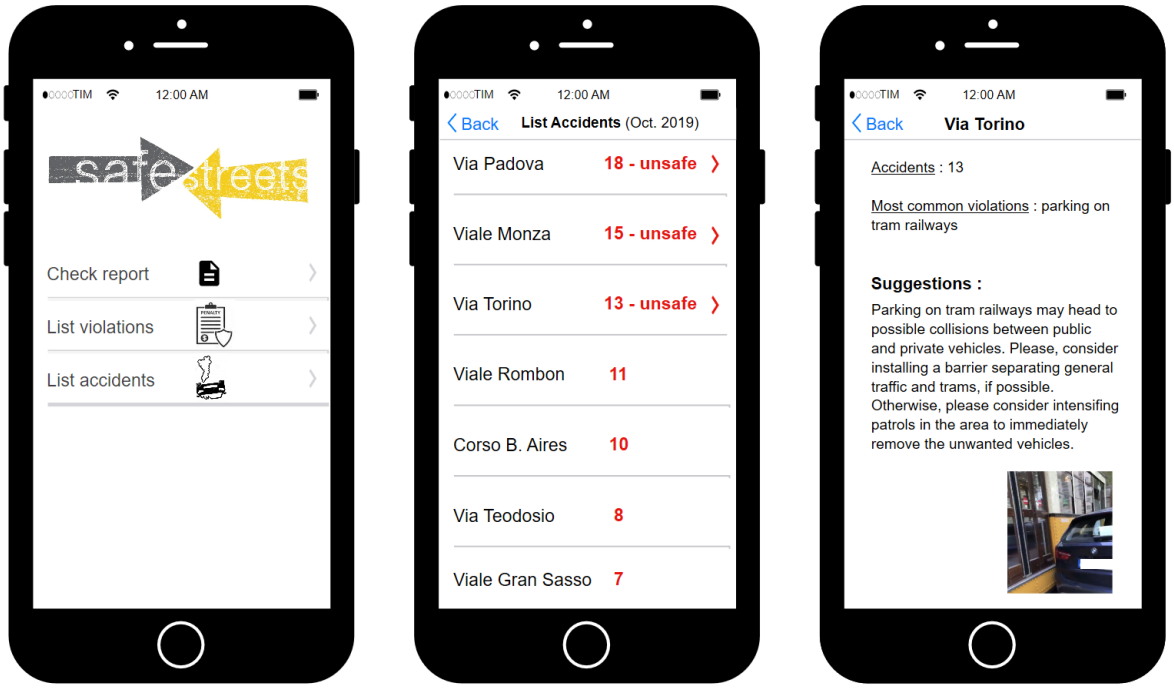


Figure 8: Menu and List Accidents section

Authorities can select unsafe streets from the section regarding accidents. If said street is listed as *unsafe*, the type of most common violations is shown. A suggestion is made accordingly, as it can be argued that if drivers tend to reiterate a specific mistake, it is possible that this is the cause of the accidents.

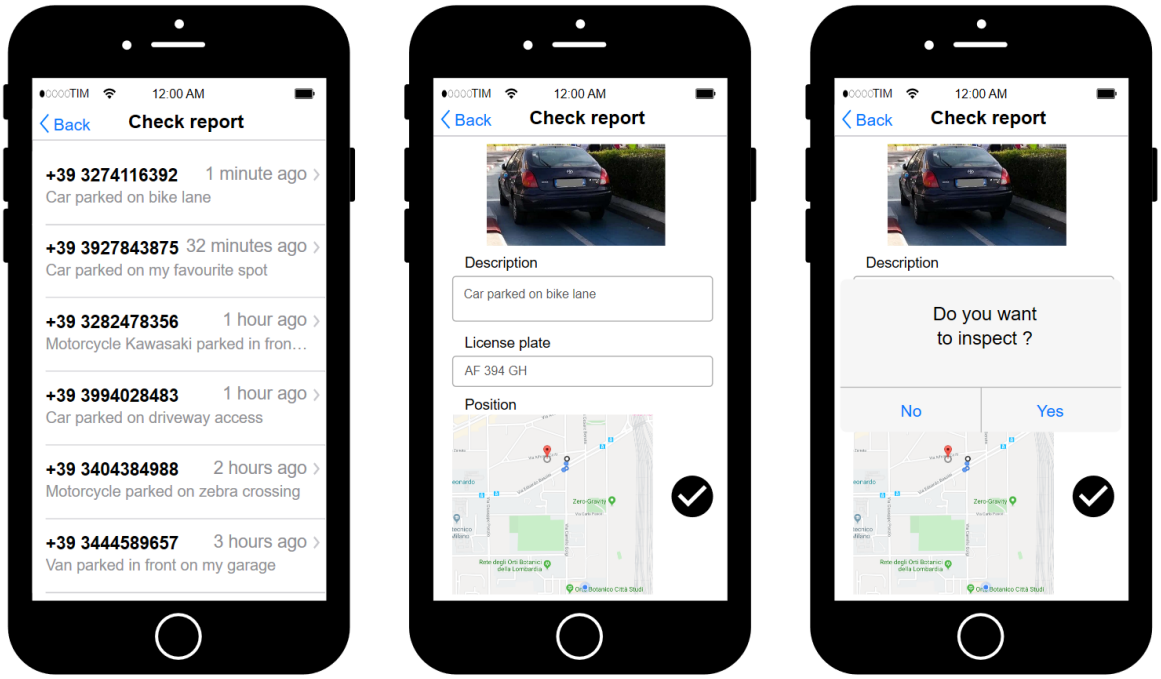


Figure 9: Check Report section

Authorities can discard specific violations or take them in charge. If the latter happens, they are

guided to the site of the violation exploiting GM' API.

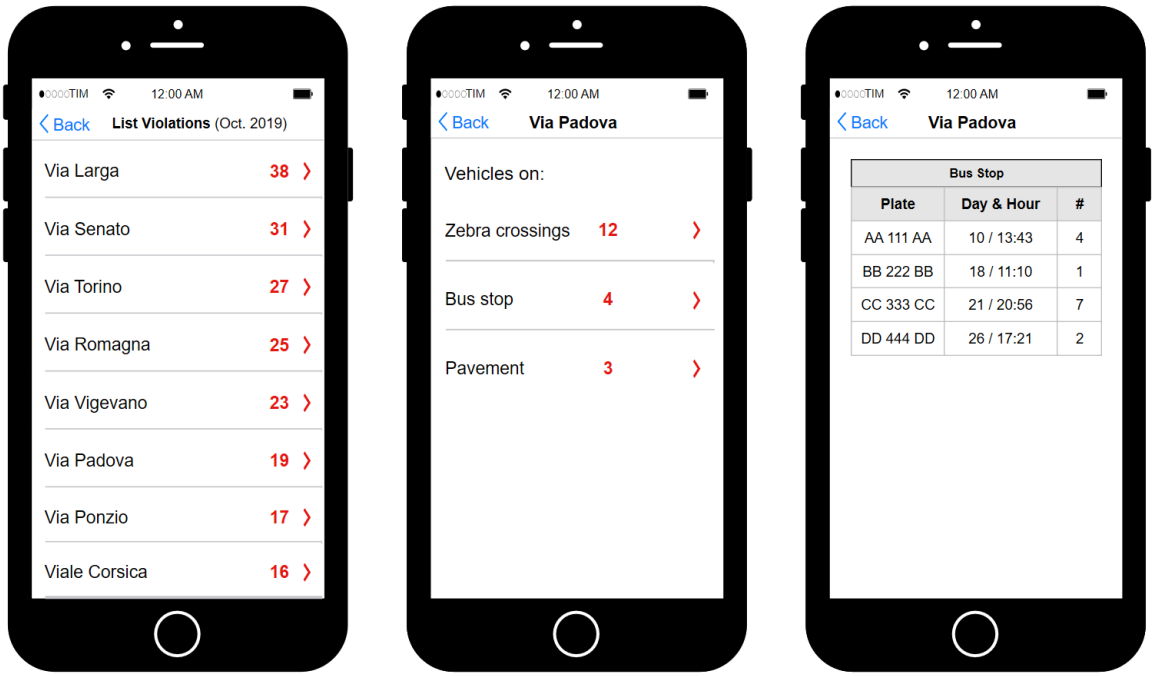


Figure 10: List Violations section

Authorities can see the list of the offenders, street by street, for any kind of violation.

Figure 11: Web Login interface

Safestreets is provided with a web interface for those authorities who are assigned to administrative section.

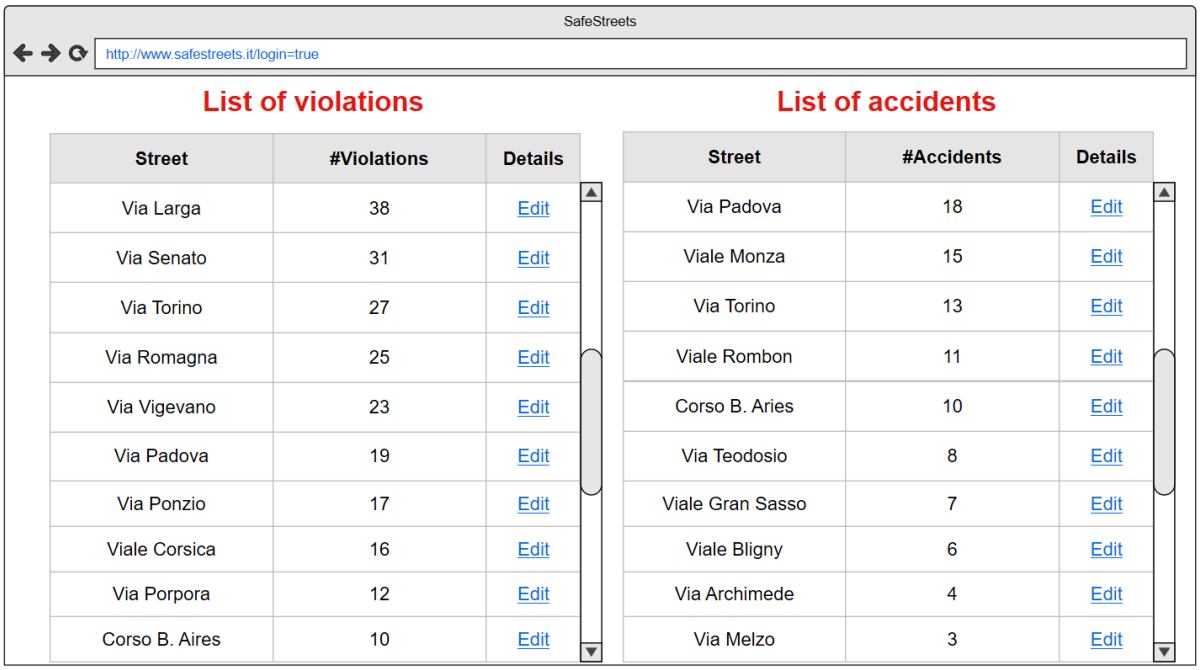


Figure 12: Web interface

This version is used merely for data analysis and inspection.

3.1.3 Hardware Interfaces

The application is available to those who have a smartphone with internet connection (to communicate with SafeStreet), a camera (to take photos of the violations) and a GPS tracker (to provide the position). The web application for authorities can be accessed by any computer with a web browser.

3.1.4 Software Interfaces

- Browser
- Web Server Application
- Operating System: iOS, Android
- DBMS

3.1.5 Communication Interfaces

This application requires an Internet connection for the purpose of transmitting the report information, HTTPS protocol is used to transfer the data securely between the user and the DBMS.

3.2 Functional Requirements

- **G1** The application has to store all the information about violations sent to it, until a ticket is either dropped or accepted by an authority
  - **D5**: There is no failure in communication
  - **D7**: Data storage is reliable
  - **R1**: Users must login to SafeStreets

- **G2** The system must accepts violations issued from every part of the covered area
  - **D1**: The smartphone of the user can provide accurate location
  - **D3**: Internet connection is reliable
  - **D4**: There is no failure in communication
  - **R2**: Users must be connected to the internet with their GPS enabled to use the service
  - **R3**: Users must fill the log to send a report
- **G3** The system must allow authorities to access violations in every part of the covered area
  - **D3**: Internet connection is reliable
  - **D5**: There is no failure in communication
  - **D6**: GM is always on and ready to communicate
  - **D7**: Data storage is reliable
  - **R4**: Authorities must be connected to the internet with their GPS enabled to use the service
  - **R5**: Authorities must login to use Safestreets
  - **R6**: Authorities can close open reports
- **G4** The application allows users and authorities to mine information on the system
  - **D3**: Internet connection is reliable
  - **D5**: There is no failure in communication
  - **D7**: Data storage is reliable
  - **R7**: Users are able to retrieve info about violations for every street
  - **R8**: Users are able to retrieve info about accidents for every street
  - **R9**: Authorities are able to retrieve info about violations for every street
  - **R10**: Authorities are able to retrieve info about accidents for every street
- **G5** The application identifies unsafe areas crossing its informations with those offered by the municipality
  - **D7**: Data storage is reliable
  - **D8**: Data offered by municipality is correct
  - **D9**: Data offered by municipality is up-to-date
  - **R11**: The system can store data permanently
- **G6** The application suggests possible solution to problems perceived after the crossing of information
  - **D2**: The data provided by the user is correct
  - **D8**: Data offered by municipality is correct
  - **D9**: Data offered by municipality is up-to-date
  - **R11**: The system can store data permanently

3.2.1 Scenarios

Scenario A

Alberto, while on on his way to the workplace, notices a car parked on the zebra crossing so he decides to use Safestreets to report it. He inserts his number on the homepage of the app, then he fills the following field with the OTP sent to him by SMS. He takes a photo of the car while he is fare from it,but the "PLATE NOT OK" signal appears on the bottom.He tries again, this time getting nearer and carefully framing the license plate. The number appears correctly on the page (AA 000 AA), so he just chooses the type of the violation ("Parking on zebra crossings") and writes a brief description "Car parked in a critical spot right in front of a school". The GPS on the map points to "Via Goito 4", he clicks on "Send" and the violation is sent to the server, waiting for someone to take it in charge. As he wants to know if that type of violation is common in that area, he selects "List Violations" and then he goes on the section dedicated to Via Goito. He reads that the most common violation there is indeed "Parking on zebra crossing" followed by "Double parking" so he realizes that he should look out for these errors while waling on the street.

Scenario B

Bernardo has just taken the driving license and wants to use his car to reach a friend living on the other side of the city. As he has not much experience yet, he decides to calculate a path to said part before getting behind the wheel. He opens SafeStreets and, after the login procedure, he checks the list of accidents of the city. He sees that the streets on which his friend lives is unsafe and it is subject to a great deal of parking violations. He realizes that he wants to avoid that street as he wants to sharpen his skills before and that parking there is unfavourable, so he calculates a new path and finds a more suitable street to park in.

Scenario C

Chiara is a policewoman patrolling the streets near the Duomo, she decides to open SafeStreets to find any violation to sanction. She logs in using her credentials and she goes on the section "Check Reports". She reads a report of a moped double parked on Via Torino 23 and she goes to check (using the map on the app). She wants to check if that vehicle has already committed another violation, so she goes on the "List Violation" section and highlights Via Torino. She reads that the proprietary of the vehicle has already double parked on the same street. She calls the service for the removal of the moped and she takes in charge the report; it is kept on the DB.

Scenario D

Dario works at the Comando Pulizia Municipale and he is currently revisioning the safety of the streets in the city. To speed the process up he logs on to the webpage of SafeStreets and checks the list of accidents. He finds that a few streets are marked as unsafe, as they have registred a number of accidents over 12. He selects said parts and reads the suggestions listed. Deeming them reasonable, he writes a reports to his boss suggesting these changes and waits for them to be implemented.

3.2.2 Use Case Diagrams

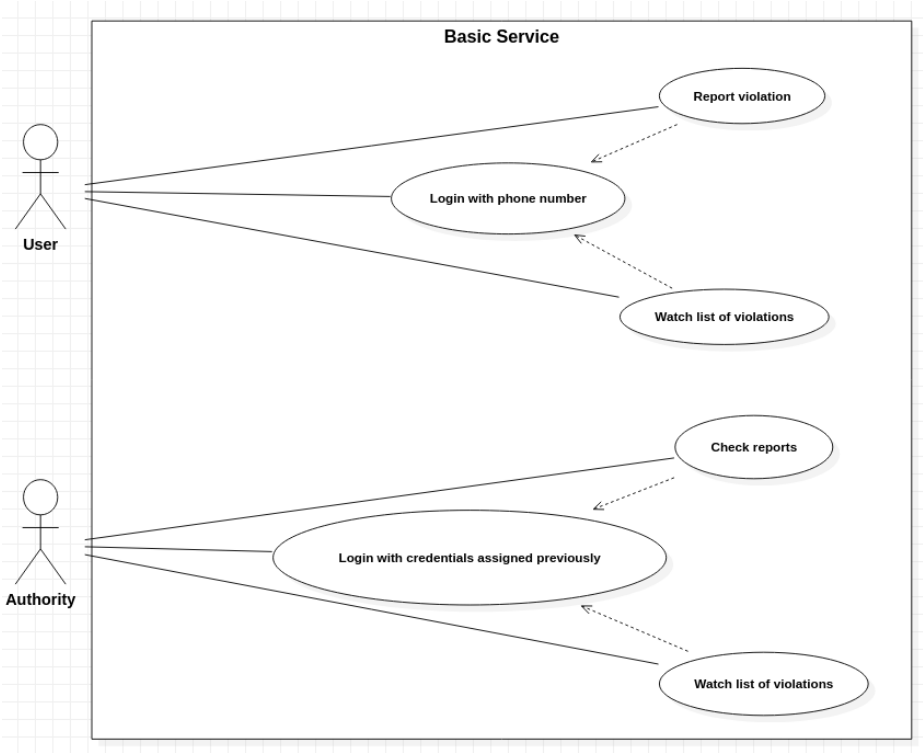


Figure 13: SafeStreets basic service use case diagram

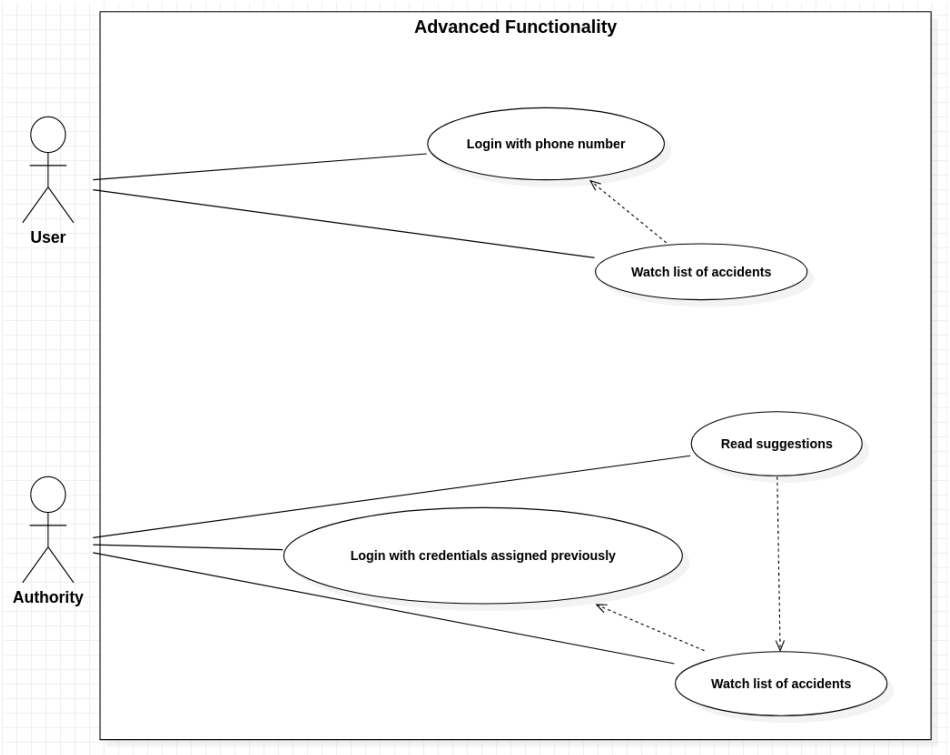


Figure 14: SafeStreets advanced functionality use case diagram

3.2.3 Use Case

Name	Login with phone number
Actor	User
Entry conditions	The user has installed the application on their device
Events flow	1. The user inserts their phone number 2. The user taps on 'send' 3. The user receives an SMS with an OTP 4.The user inputs the OTP 5. The user taps on 'send'
Exit conditions	The user logins and is prompted with the interface of the App
Exceptions	1. The user uses an invalid phone number 2. The user inputs an invalid OTP

Name	Login with credentials assigned previously
Actor	Authority
Entry conditions	The authority was provided with credentials
Events flow	1. The authority inputs their username 2. The authority inputs their password 3. The authority taps on 'login'
Exit conditions	The authority logins and is prompted with the interface of the App
Exceptions	1. The authority inputs a wrong username 2.The authority inputs a wrong password

Name	Report violation
Actor	User
Entry conditions	The user is logged in
Events flow	1. The user selects 'file a report' on the menu 2. The user taps on the camera icon 3. The user takes a photo of the violation 4. The user selects the type of violation 5. The user writes a brief description 6. The user corrects the license plate(if incorrect) 7. The user taps on 'send'
Exit conditions	The user is notified the success of the operation and is brought back to the main menu
Exceptions	1. The user doesn't fill all the fields 2. The user has already filed a report and it has not been closed yet

Name	Check reports
Actor	Authority
Entry conditions	The authority is logged in
Events flow	1. The authority selects 'check report' on the menu 2. The authority selects a report 3. The authority accepts or discards the report
Exit conditions	The authority stays on the page and can choose later to close or eliminate the report
Exceptions	

Name	Watch list of violations
Actor	User
Entry conditions	The user is logged in
Events flow	1. The user selects 'list violations' on the menu 2. The user taps on a specific street
Exit conditions	The user can see the types of violation with the respective values
Exceptions	No violation occured the previous month

Name	Watch list of violations
Actor	Authority
Entry conditions	The authority is logged in
Events flow	1. The authority selects 'list violations' on the menu 2. The authority taps on a specific street 3. The authority taps on a specific type of violation
Exit conditions	The authority can see the list of the repeated offenders
Exceptions	No violation occured the previous month

Name	Watch list of accidents
Actor	User
Entry conditions	The user is logged in
Events flow	The user selects 'list accidents' on the menu
Exit conditions	The user can see the list of the streets with the respective number of accidents, marked the most unsafe
Exceptions	No accident occured the previous month

Name	Watch list of accidents
Actor	Authority
Entry conditions	The authority is logged in
Events flow	1. The authority selects 'list accidents' on the menu 2. The authority selects one of the unsafe streets listed
Exit conditions	The authority can see the number of accidents, the most common type of violation and suggestions for the street
Exceptions	1. No accident occured the previous month 2. No street registered at least 12 accidents

3.2.4 Sequence Diagram

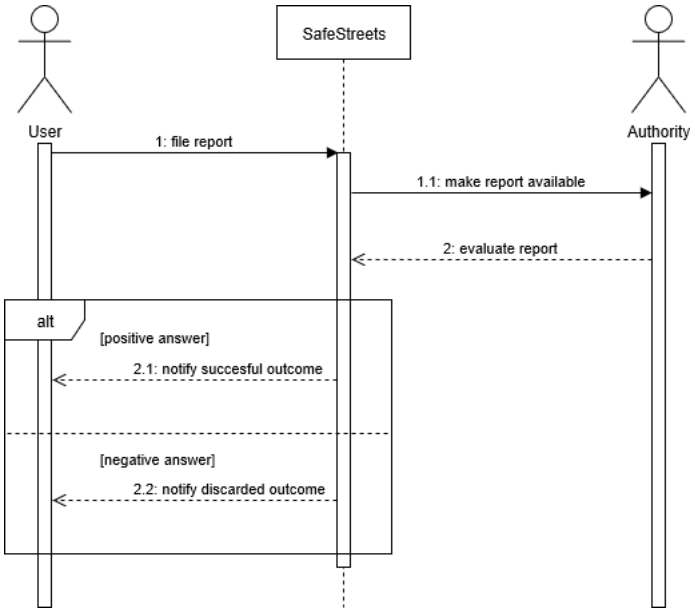


Figure 15: Lifetime of a report

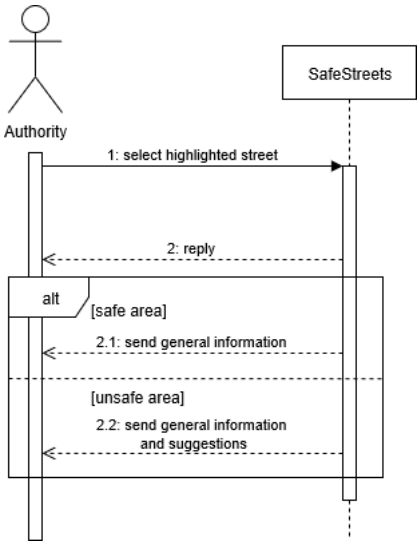


Figure 16: Examination of accidents

3.3 Performance Requirements

The system must be able to handle large quantities of data. On the other hand the importance of speed is limited as violations and their report does not have to be handled at the same moment they are put into the system.

3.4 Design Constraints

3.4.1 Hardware Limitation

The application does not have strict hardware limitation. Nevertheless it requires:

- 2G/3G/4G connection
- Working camera
- GPS

3.5 Software System Attributes

3.5.1 Reliability

The system should guarantee 99% of reliability: its component must have a failure rate that guarantees this goal.

3.5.2 Availability

The service should ideally be available 24/7.

3.5.3 Security

The system uses HTTPS protocol for communication. Plate numbers are stored and encrypted.

3.5.4 Portability

The Application is developed to be used by as many people as possible, thus it is developed for devices that run Android or IOS. In addition a web server interface is provided.

4 Formal Analysis Using Alloy

```
sig CellPhone{}
sig User{
        cellphone: one CellPhone
}
sig Id{}
sig Authority{
        id: one Id
}
sig Picture{}
sig Date{}
sig Hour{}
sig TypeViolation{}
sig Plate{}
sig Description{}
sig Position{}
sig Violation{
        sender: one User,
        checker: lone Authority,
        picture: one Picture,
        date: one Date,
        hour: one Hour,
        typeviolation: one TypeViolation,
        plate: one Plate,
        description: one Description,
        position: one Position,
}
abstract sig ViewListViolation{
        violations: set Violation
}
lone sig ViewListViolationUser extends ViewListViolation{
        observer: some User
}
lone sig ViewListViolationAuthority extends ViewListViolation{
        observer: some Authority
}
abstract sig ViewListAccident{
        accidents: Int
} {accidents ≥ 0}
lone sig ViewListAccidentUser extends ViewListAccident{
        observer: some User
}
lone sig ViewListAccidentAuthority extends ViewListAccident{
        observer: some Authority,
        suggestion: lone GiveSuggestion
}{#suggestion=1 iff accidents≥3 }
sig GiveSuggestion{}

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

--Two lists of accidents can't have two different quantities, as both users and
    ↪ authorities can inspect them
fact EqualNumberAccidents{
        no disjoint t1,t2: ViewListAccident | t1.accidents≠t2.accidents
}
--If a list of violation exists, it can see every violation
fact ListViolationAll{
        all v : Violation | all vl: ViewListViolation | v in vl.violations
}
--A picture is considered only if a related violation exists
fact ExistancePicture{
        all p : Picture | one v : Violation | p in v.picture
}
--A plate is considered only if a related violation exists
fact ExistancePlate{
        all p : Plate  | some v : Violation | p in v.plate
}
--A date is considered only if a related violation exists
```



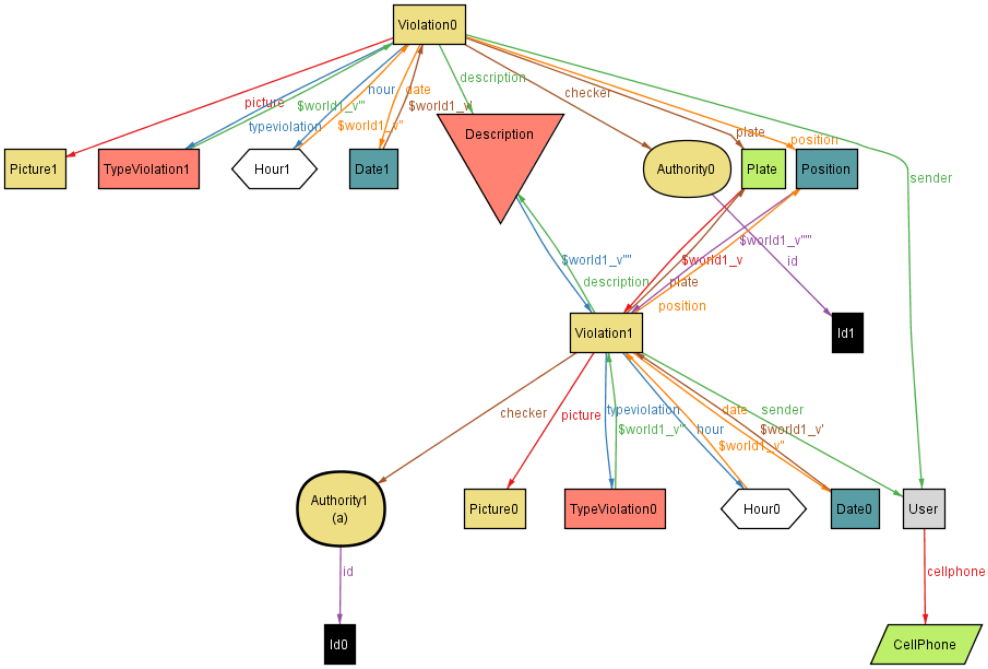
```
fact ExistanceDate{
        all d : Date | some v : Violation | d in v.date
}
--An hour is considered only if a related violation exists
fact ExistanceHour{
        all h : Hour | some v : Violation | h in v.hour
}
--A type is considered if a related violation exists
fact ExistanceTypeViolation{
        all t : TypeViolation | some v : Violation | t in v.typeviolation
}
--A description is considered only if a related violation exists
fact ExistanceDescription{
        all d : Description | some v : Violation | d in v.description
}
-- only if a related violation exists
fact ExistancePosition{
        all p : Position | some v : Violation | p in v.position
}
--All CellPhones have to be associated to a User
fact CellPhoneUserConnection{
        all cp: CellPhone | one u: User | cp in u.cellphone
}
fact NoIdWithoutAuthority{
        all i: Id | one a : Authority | i in a.id
}
--each Authority can check only one violation
fact ViolationAuthorityConnection{
        some a: Authority | lone v: Violation | a in v.checker
}
-- each Violation must be checked by one Authority
fact EachViolationIsCheckedByAnAuthority{
        all v: Violation | one a: Authority | a in v.checker
}
--no vehicles on the same spot at the same time
fact NoBusySpots{
        no disjoint t1,t2: Violation | t1.plate ≠ t2.plate and t1.position = t2.position
            ↪ and t1.date = t2.date and t1.hour = t2.hour
}
--different violations with same plates can't be done at the same time
fact NoRepetition{
        no disjoint t1,t2: Violation | t1.plate = t2.plate and t1.date = t2.date and t1.
            ↪ hour = t2.hour
}
--unique picture for violation
fact UniquePicture{
        no disjoint t1,t2: Violation | t1.picture = t2.picture
}
--exists only if a related Accident exists
fact ExistanceSuggestion{
        all s : GiveSuggestion | one v : ViewListAccidentAuthority |s in v.suggestion
}


--user's cellphone is unique
assert UserCellPhoneisUnique{
        no disjoint t1,t2 : User | t1≠t2 and t1.cellphone = t2.cellphone
}
--authority's id is unique
assert AuthorityIdIsUnique{
        no disjoint t1,t2 : Authority | t1≠t2 and t1.id = t2.id
}

assert sugg{
        all a: ViewListAccidentAuthority | #a.suggestion=1 iff a.accidents ≥3
}

check sugg

check UserCellPhoneisUnique

check AuthorityIdIsUnique
```



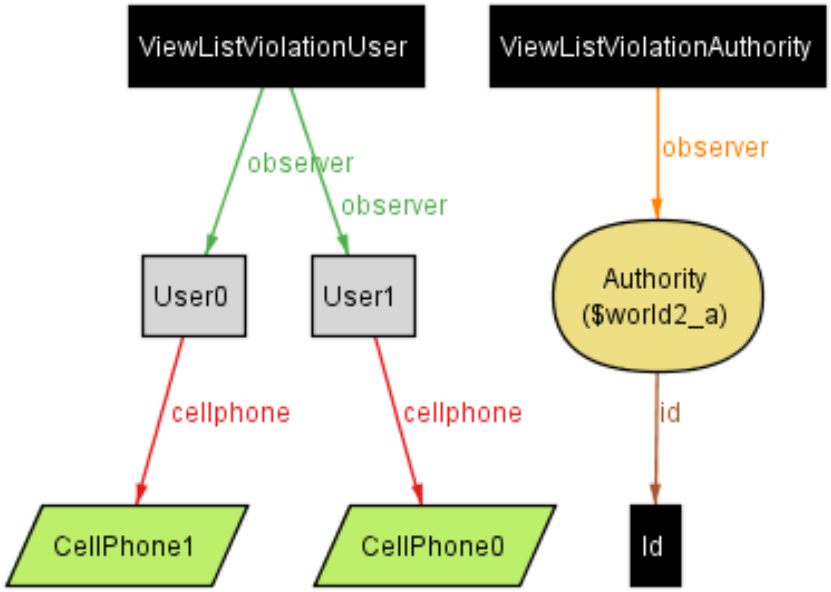
```
pred world1{}

pred world2{}

pred world3{}

pred world4{
        #Violation=3
        #ViewListViolationUser=1
        #ViewListViolationAuthority=1
        #ViewListAccidentAuthority=1
        #ViewListAccidentUser=1
}

run world1 for 2 but 2 Violation, 0 ViewListViolationUser, 0 ViewListViolationAuthority,
    ↪ 0 ViewListAccidentAuthority, 0 ViewListAccidentUser

run world2 for 2 but 0 Violation, 0 ViewListAccidentAuthority, 0 ViewListAccidentUser, 3
    ↪ User, 2 Authority

run world3 for 3 but 0 ViewListViolationUser, 0 ViewListViolationAuthority

run world4 for 4
```


4.1 Generated Worlds

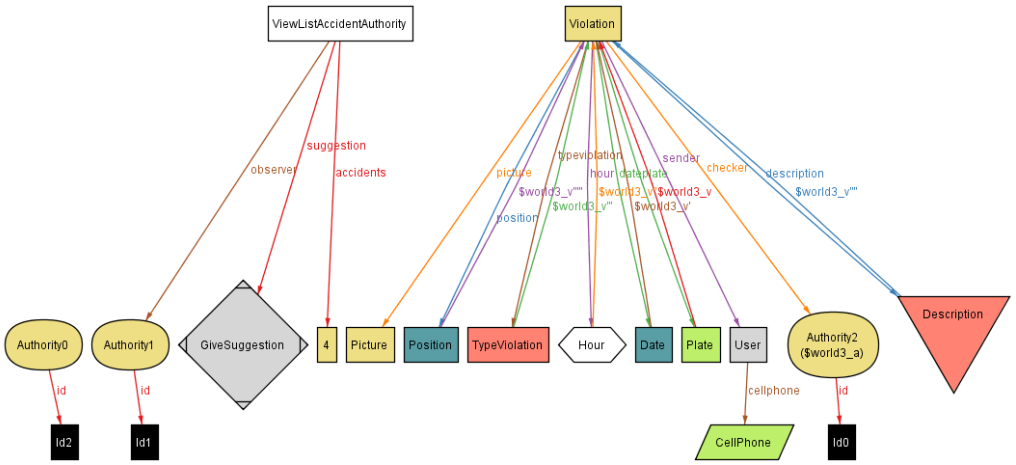
```
run world1 for 2 but 2 Violation, 0 ViewListViolationUser, 0 ViewListViolationAuthority,
    ↪ 0 ViewListAccidentAuthority, 0 ViewListAccidentUser
```



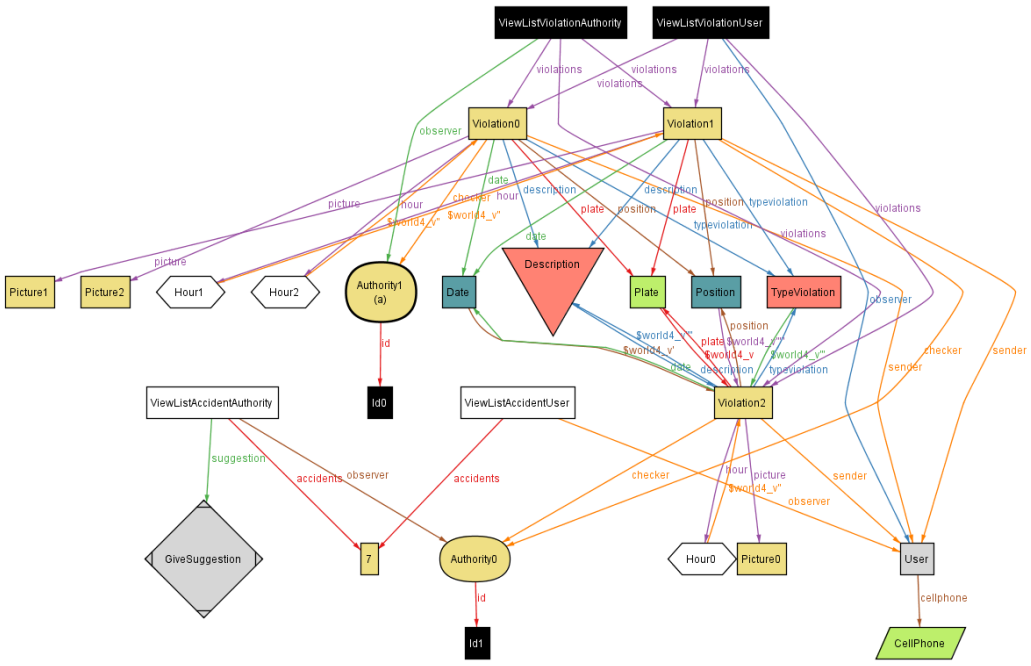

```
run world2 for 2 but 0 Violation, 0 ViewListAccidentAuthority, 0 ViewListAccidentUser, 3
    ↪ User, 2 Authority
```




```
run world3 for 3 but 0 ViewListViolationUser, 0 ViewListViolationAuthority
```




```
run world4 for 4
```



4.2 Alloy results

```
Executing "Check sugg"
  Solver=minisat(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3169 vars. 273 primary vars. 6068 clauses. 20ms.
  No counterexample found. Assertion may be valid. 4ms.

Executing "Check UserCellPhoneisUnique"
  Solver=minisat(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3041 vars. 277 primary vars. 5457 clauses. 25ms.
  No counterexample found. Assertion may be valid. 2ms.

Executing "Check AuthorityIdIsUnique"
  Solver=minisat(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  3041 vars. 277 primary vars. 5457 clauses. 23ms.
  No counterexample found. Assertion may be valid. 2ms.
```

Executing "Run world1 for 2 but 2 Violation, 0 ViewListViolationUser, 0 ViewListViolation
Solver=minisat(jni) Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
925 vars. 96 primary vars. 1345 clauses. 11ms.
Instance found. Predicate is consistent. 19ms.

Executing "Run world2 for 2 but 0 Violation, 0 ViewListAccidentAuthority, 0 ViewListAc
Solver=minisat(jni) Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
323 vars. 51 primary vars. 454 clauses. 9ms.
Instance found. Predicate is consistent. 25ms.

Executing "Run world3 for 3 but 0 ViewListViolationUser, 0 ViewListViolationAuthority"
Solver=minisat(jni) Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
2787 vars. 249 primary vars. 5029 clauses. 20ms.
Instance found. Predicate is consistent. 17ms.

Executing "Run world4 for 4"
Solver=minisat(jni) Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
4788 vars. 416 primary vars. 8157 clauses. 30ms.
Instance found. Predicate is consistent. 18ms.

5 Effort Spent

The effort of the components of the group is here summarized, please note that the project was developed side-by-side so every section was developed and supervised by the whole group.

DATE	TASK	HOURS
25/10/2019	First meeting	2
27/10/2019	Layout and introduction	3
30/10/2019	Finished introduction	2
1/11/2019	Class diagram and state	1
2/11/2019	Finished overall description	4
3/11/2019	Mockups	3
4/11/2019	Functional requirements	4
5/11/2019	Use case diagram	2
6/11/2019	Finished specific requirements	2
7/11/2019	General fixes	3
8/11/2019	Alloy	4
9/11/2019	Scenarios	3
10/11/2019	Review	2

Cuzzucoli Sergio

DATE	TASK	HOURS
25/10/2019	First meeting	2
28/10/2019	Revision of the layout	2
30/10/2019	Finished introduction	2
1/11/2019	Class diagram and state	3
2/11/2019	Finished overall description	2
3/11/2019	Mockups	5
4/11/2019	Functional requirements	2
5/11/2019	Use case diagrams	3
6/11/2019	Sequence diagrams	2
7/11/2019	Alloy	4
8/11/2019	Alloy and worlds	2
9/11/2019	Alloy results	2
10/11/2019	Review	2

De Dominicis Daniele

References

[1] Di Nitto Elisabetta. Mandatory project: goal, schedule, and rules, 2019.

[2] Software Design Group. about alloy. Technical report, MIT, 1997. URL: https://alloytools.org/about.html.

[3] The Object Management Group (OMG). Unified modelling language: Infrastructure, 2011. version 2.4.1. omg document: formal/2011-08-05. Technical report, , 2011.

