POLITECNICO DI MILANO

Academic Year 2019/2020

RASD

Requirements Analysis and Specification Document

version 1.0 – 10/11/2019

Computer Science and Engineering

Software Engineering 2

Matteo Falconi 945222 - Davide Galli 944940

Contents

1 Introduction

1.1 Purpose

1.1.1 Project Description

SafeStreets is a cross-platform service available both on app and on web.

App service is focused to develop a software-based service that allows individual basic users to report traffic violations. Those reports consist in pictures of violation, type of violation, date, time and position. When a picture is uploaded, the system runs an algorithm in order to read the license plate. Finally, all those data are stored in *SafeStreets*' databases.
The system allows also authorities registration, who can receive notifications about new violations in a certain area. When a notification occurs, an authority can reserve it taking charge of that violation.
Both basic users and authorities can access to collected data in order to analyze the streets and the relative safeness. However, a basic user can only access to anonymized data clusters, that give an idea of how many violations occur in each area; whereas authorities can also access to specific anonymized data.

Web service, instead, let *SafeStreets* to develop a functionality, in partnership with the municipalities who provide accidents data, that can cross-reference data provided by the users with the accidents one, in order to identify unsafe areas and suggest possible interventions.

1.1.2 Goals

Basic users:

[G.BU1] Basic users can report traffic violations.

[G.BU2] Basic users can view a data clustering about violations that had occurred.

Authorities:

[G.A1] Authorities should choose to receive anonymous notifications in real time about new violations.

[G.A2] Authorities should view and reserve a violation.

[G.A3] Authorities can view both data clustering and specific data about violations that had occurred.

Municipalities:

[G.M1] Municipalities can identify potential unsafe zones.

[G.M2] Municipalities can receive a safety report with suggestions to reduce accidents.

1.2 Scope

1.2.1 World

There are three main types of actors in our world: citizen, authorities and municipalities. Citizen are interested in reporting traffic violations and receiving information about violations in certain areas, authorities and municipalities are interested in exploiting the data gathered from the citizen: the former want to get notified when new violations occur in order to generate traffic tickets, the latter want to identify unsafe zones and to receive possible solutions.
SafeStreets is the service that acts as a bridge between these actors' needs.

1.2.2 Phenomena

Phenomena that occur in the world and that are related to the system application domain are:

- Traffic violations occur in a city;
- Authority patrols an area and makes traffic tickets;
- People are interested in analyzing violation data;
- Municipality wants to reduce the number of accidents.

The system shares also some events with the world in order to communicate with it. Phenomena that occur in the world and are observed by the machine are:

- A user registers and logs in filling the various form;
- A basic user fills the violation data and sends a new report;
- An authority manages notifications, enabling or disabling them;
- An authority researches a violation among those of civilians in which he/she may be interested;
- An authority examines the details of a violation;
- An authority reserves a violation;
- App user views mined data on a map in his/her smartphone;
- Municipality analyzes unsafe zones;

- Municipality views safety report with suggestions for reducing accidents.

On the other hand, aspects generated by the machine and observed by the world are:

- The system tracks the position of users;
- The system uploads, receives and confirms data insert by users through an acknowledgement (login credentials, new violation reports, etc.);
- A connection error occurs, the system notifies the issue to users;
- The system generates notifications about new violations;
- The system loads safety reports for the municipalities, with suggestions to reduce accidents;
- The system loads and renders graphically data to the user (violations list, detail of a violation, etc.)
- The system loads into a map the mined data as highlight zone and shows them to users.

Finally, phenomena inside the machine and not visible directly from the world are:

- The system stores and reads data (users' data, report data, etc.);
- The system deletes data account;
- The system checks authority's personal ID;
- The system retrieves data from municipality database;
- The machine mines data, clustering both violation data and accidents data ;
- The machine compiles safety report;
- The algorithm analyzes photos reading the license plate of a vehicle;

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

User	Any kind of person who uses the system (basic user, authority and municipality).
App User	Any kind of person who uses the system through the App (basic user and authority).
Basic user	Citizen who can report a traffic violation and view a data clustering about violations that had occurred.
Authority	Recognized entity which can empower the law (ex. local police).

Municipality	Authority recognized by the State who holds the government in an area.
Data clustering	Set of anonymous data about violations group by location and type. A cluster is defined only if are reported more than 15 traffic violations of the same type in the same area.
Highlight zone	Area on a map where are shown the data cluster. The information shown consists in location, type of violation and occurrences.
Correlation (between data)	Mutual relationship between accidents and traffic violations. A correlation occurred only if there are more than 5 accidents in the same zone where are reported more than 15 traffic violations of the same type.
Report / Violation report	Organized set of information collected by basic users in order to denounce a traffic violation.
Specific violation / violation data	Information about a traffic violation insert in the report. Contains: three photos, location, type of violation, license plate, date and time.
Traffic violation	Illegal action performs by any vehicle (ex. double parking, stopped on zebra cross).
Accident	Traffic violation result in an injury for at least one person.
Unsafe zone	Area of the city where accidents happen frequently.
PC	Generic system able to navigate through the internet.
Integrated data	*SafeStreets*' violation data unit with municipality's accident data.
Matricula / Personal ID	A code able to identify uniquely an authority, stored in state DB.
Safety report	Document with all information about the safety of a municipality. It consists in three parts: raw data, accidents' analysis and possible solution. *(section 2.2.4)*
Area / zone	Region on a map of 5 Km^2.

1.3.2 Acronyms

API Application Programming Interface
GPS Global Positioning System
S2B Software to Be
UI User Interface
IEEE Institute of Electrical and Electronics Engineers
DB Database
TOS Terms of Service

1.3.3 Abbreviations

[G.BU$_n$] Basic users' nth goal;

[G.A$_n$] Authorities' nth goal;

[G.M$_n$] Municipalities' nth goal;

[D.$_n$] Nth domain assumption;

[R.$_n$] Nth requirement;

[R.M$_n$] Municipalities' nth requirement;

[R.P$_n$] Performance nth requirement;

1.4 Revision history

Date	Version	Log
10/11/2019	v. 1	First RASD release

1.5 Reference Documents

Specification document: "SafeStreets Mandatory Project Assignment"
IEEE 830-1993: IEEE Recommended Practice for Software Requirements Specifications

1.6 Document Structure

This document is composed into 6 sections, organised as follow:

- *Section 1* gives a short introduction to the project; giving a clear idea of who are the actors and what are the goals of the S2B;

- *Section 2* defines the main functions of the project, analysing the constraints and declaring the assumptions;
- *Section 3* is the most important part of the RASD: it analyses functional requirements, shows user interfaces and exemplifies the use of S2B through scenarios and use cases;
- *Section 4* contains the Alloy model that certifies system correctness;
- *Section 5* shows the effort spent in developing the RASD;
- *Section 6* contains the tool used and references.

2 Overall Description

2.1 Product prospective

SafeStreets is a crowd-based service oriented to data acquisition and data analysis. Its software exploits basic user interaction to retrieve data from the world. Once the user registered to the service, he/she can submit a report, allowing the system to collect data. Also, the system is able to analyze the data acquired, sorting it into clusters.

It is mandatory for basic user to have a smartphone able to connect to the internet, take photos and with a built in GPS.

Authorities are interested in data inspection: they can receive notification when a new report is submitted.

Municipalities are interested in data analysis. They provide their accident DBs to *SafeStreets*, which cross-reference the data with their own data collected thanks to the users' report, in order to generate a Safety Report. Municipalities are required a PC with an internet connection.

2.2 Product functions

In the following section we present the major functions that our system will offer.

2.2.1 Profile Management

The system provides a registration form for new users. Each type of user needs to provide different information, as reported in the table below; authorities are required to testify their role providing the personal ID that will be check with the one store in the state database. Creating an account is mandatory in order to exploit the system's functionalities. The system must be able to distinguish accounts for basic users, authorities and municipalities, as it should offer different functionalities between the three of them.

The system will offer the ability to delete an account: the deletion of an account will be permanent.

Account Type	Required Information	Additional Information
Basic User	*Name, surname, email address, password, date of birth*	*None*

Authority	*Name, district's name, personal ID, email, password*	*None*
Municipality	*Name of the municipality, referencing email, referencing name, referencing surname, referencing phone number*	*Comments*

Table 1*: Registration form information*

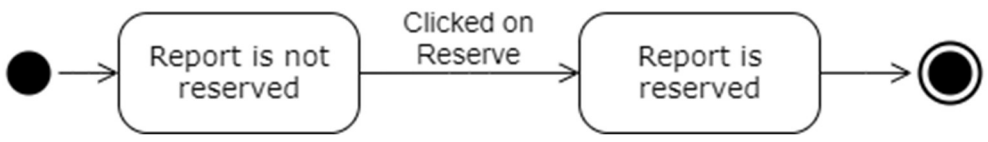
2.2.2 Data Inspection

SafeStreets relies on a database for data storage. Both basic users and authorities can access to the data stored in the database, but with a different granularity level, as explained in this table:

Account Type	Viewable Data
Basic Users	*They can only access to statistics' map, in which the violations are organised by type and by location and shown as highlighted zone (Figure 6).*
Authorities	*They can access both to statistics' map and to a specific violation: the former is as explained above, the latter contains detailed information about a violation in a certain area, but without showing the basic user who reported it. (Figure 14)*

Table 2*: Data Inspection Granularity Level*

If an authority is interested in a violation and wants to take charge of it, he/she can reserve the report. When a report is reserved, it changes its status, so all other authorities can know that it has already been taken over by someone.



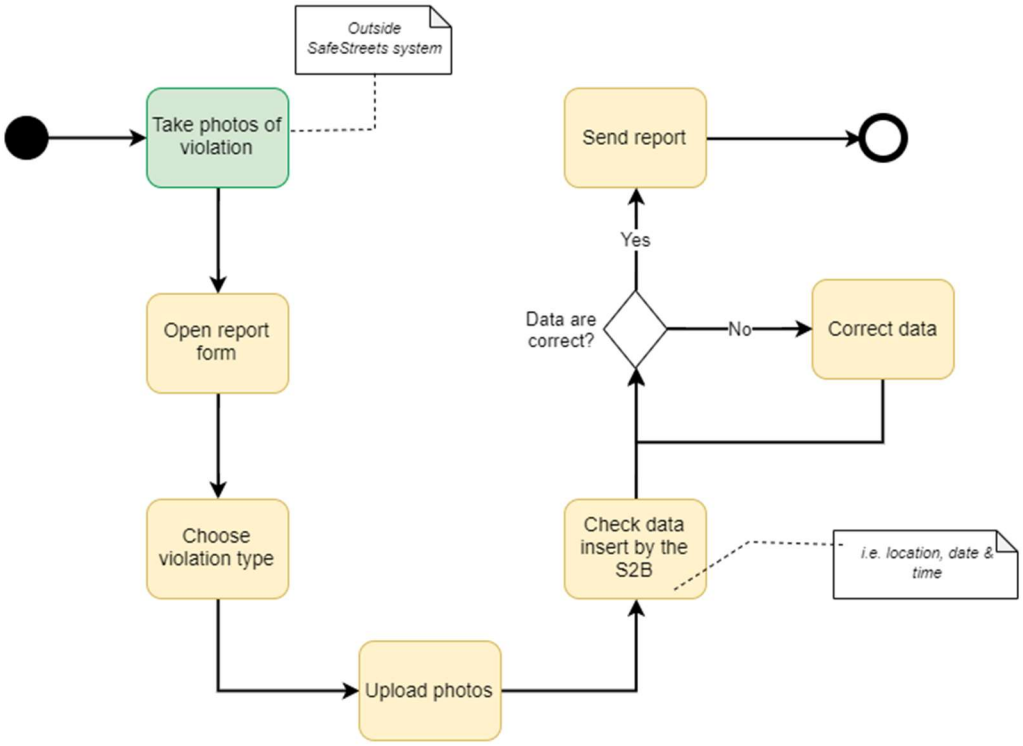
UML 1: *Report State Diagram*

2.2.3 Violation Data Collection

Data collection is exploited through the submission of reports by basic users: collected data includes three photos of the violation taken by the user (one where is shown the violation, and two where is well shown the vehicle both on front and rear), type of the violation chosen among some proposed, a timestamp collected by the device and the location in which the report is submitted.

An image recognition algorithm is run on the uploaded photos in order to read automatically the license plate of the vehicle. For this scope a user must upload two photos (front and rear of the vehicle) in which the license plate is well visible; it's not required to insert the license plate manually.

Every time a new report is submitted by a basic user, the system clusters it according to position and violation type.



UML 2: *Filling violation report Activity Diagram*

2.2.4 Report notifications

Authorities can choose to receive a notification when a new violation is reported. Notifications can be enabled in the S2B's settings and they are set base on a geographical area: when a violation in the area selected is reported, a notification is sent to the

authority. More than one notification can be set in different areas. At any moment, a notification for a certain area can be disabled.
Only authorities can enable notifications.

2.2.5 Accident Data Analysis

This function is only available to municipalities that can provide access to their accidents' DB. The S2B retrieves the data and clusters them according to the location of the accidents, in this way, areas with the most frequency of accidents are found, and they can be shown to the municipality as unsafe zones.

Once the municipality asks for a new safety report, our system extracts the accidents data and compares it with our own data, gathered from the basic users, in order to match the unsafe areas with the most likely triggering violation. It then generates a safety report, compiling the safety form as follows:

Raw Data	Here will be listed all the raw data collected from our DB in the area of interest; listing the number of possible violations report received during the period of time considered.
Accidents' Analysis	This part of the safety report will contain the data extracted from the municipalities' DB, regarding the number and the type of accidents that had happened in the area of interest.
Possible Solution	Here we will list some possible solutions to the most common accidents observed by municipality's data, based on the most common violations report observed by our data.

A new safety report is generated only on a specific request, otherwise, the system loads the latest one generated. The report shall be generated within 48 hours and it shall analyze the whole municipality area.

2.3 Users characteristics

The system interacts with the following actors:

Basic User	Person interested in the reporting traffic violations; he/she is required to create a basic user account in order to exploit *SafeStreets'* functionalities. He/She can also view clustered data about violations in his/her area. Every basic user owns a mobile device able to connect to the internet and able to monitor GPS location.
Authority	Person interested in the viewing violations system; he/she is required to create an authority account in order to exploit *SafeStreets'* functionalities. He/She can also view clustered data about violations in his/her area. Every authority owns a mobile device able to connect to the internet and able to monitor GPS location.
Municipally	Entity interested in safety reports system; it is required to contact *SafeStreets* thought the web-page in order to exploit *SafeStreets*' functionalities. Every municipally owns a digital database about accidents and a PC with internet access.

2.4 Assumption, dependences, constrains

2.4.1 Domain Assumption

[D.1] Geo-location data are correctly encoded.
[D.2] Every license plate is unique and identify uniquely a vehicle.
[D.3] Every authority's Personal ID is unique and identify uniquely an authority.
[D.4] An authority reserving a violation will take care of it.
[D.5] Basic users send only pictures about violations.
[D.6] Municipality partner has a digital database about accidents.
[D.7] Accident data are provided by location.
[D.8] Exist a state database where are stored all the authority's matriculas.

2.4.2 Dependences

SafeStreets relies on:

- Geo-location services, to access users' location.
- Maps provider API, to show maps on application.
- Image scanning algorithm, for reading data from violation's photos.
- Municipality's API, in order to access accident's DB.
- State DB, in order to access authorities' matriculas and verify their identity.

2.4.3 Constrains

Regulation policies

Users must agree to TOS in order to use the service. The TOS should agree about sharing anonymized data with authorities and municipality, collect the location and access to the camera.
Email addresses won't be used for commercial uses.

Hardware limitation

In order to work properly, the system requires:

- EDGE/3G/4G/5G connection;
- iOS or Android smartphone;
- GPS/Glonass/Galileo service.

Municipalities also require:

- Modern browser, we recommend Chrome.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User Interface

We will present the mockups of *SafeStreets*, for all our target users. Forms fields and maps are presented only for illustrative purposes as they may change during development and be different in the final product. These mockups are intended only to give an idea of what the graphical interface of our system will be like.

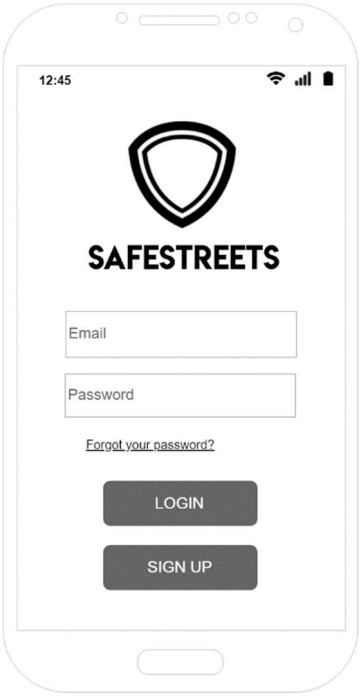


Figure 1: *Basic User and Authority Log In*

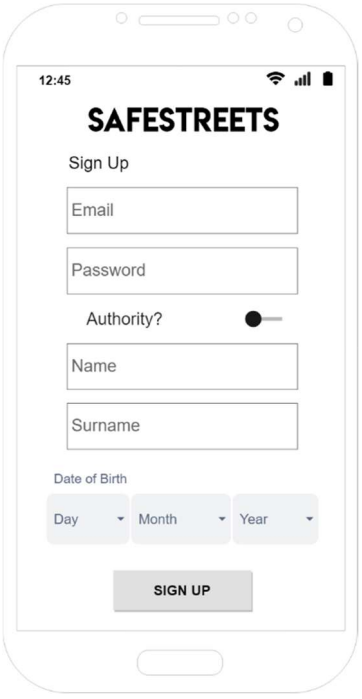


Figure 2: *Basic User Sign Up*

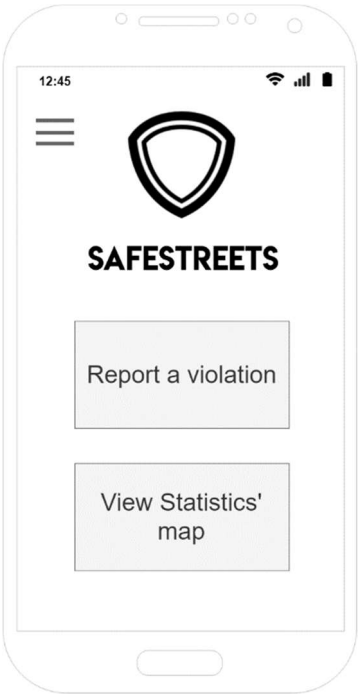


Figure 3: *Basic User Main Page*

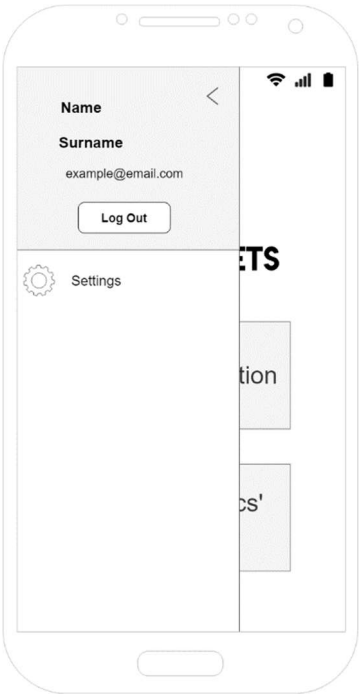


Figure 4: *Basic User Side Manu*

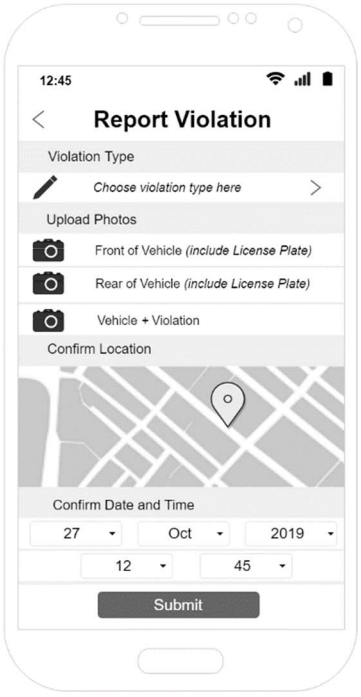


Figure 5: *Basic User Report Screen*

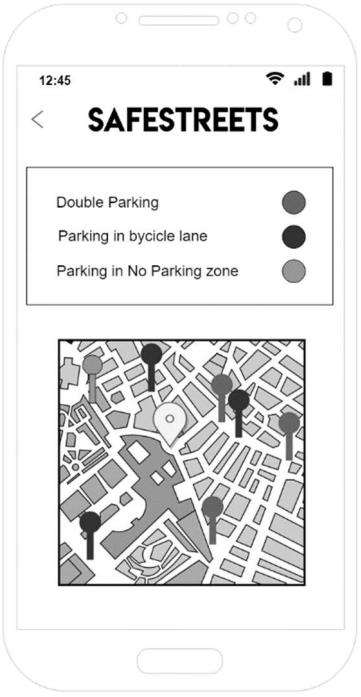


Figure 6: *Basic User and Authority Map Screen*

Figure 7: *Basic User Setting*

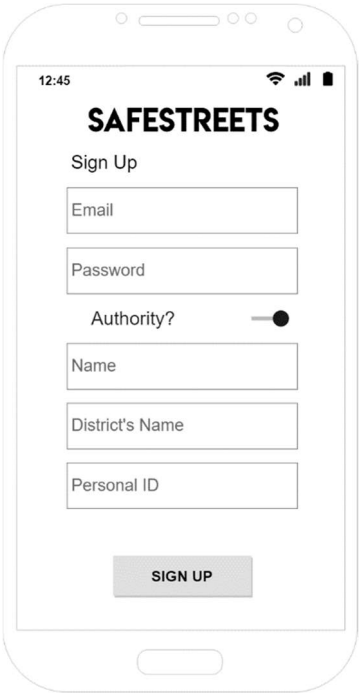


Figure 8*: Authority Sign Up*

Figure 9*: Authority Main Screen*

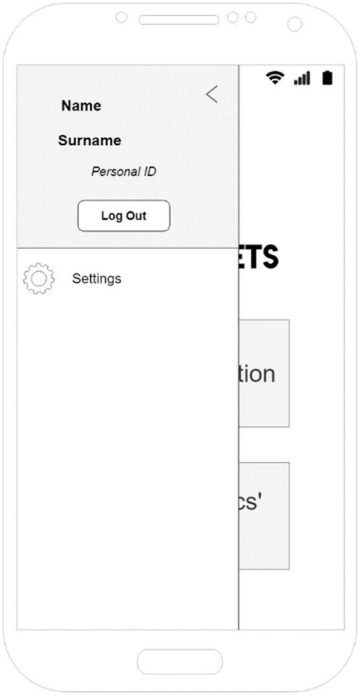


Figure 10*: Authority Side Screen*

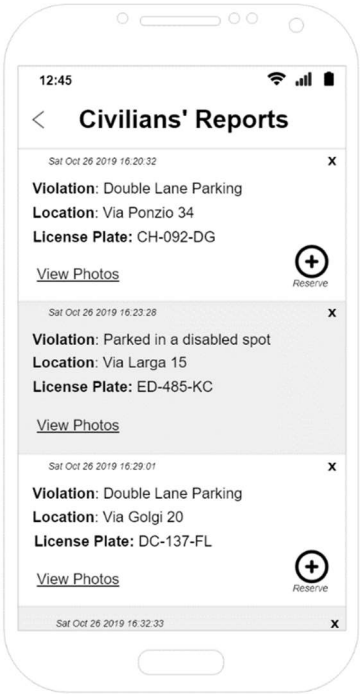


Figure 11: *Authority Report Screen*

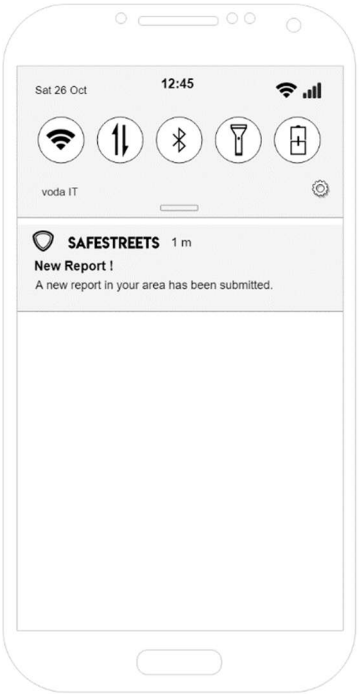


Figure 12: *Authority Report Push Notification*

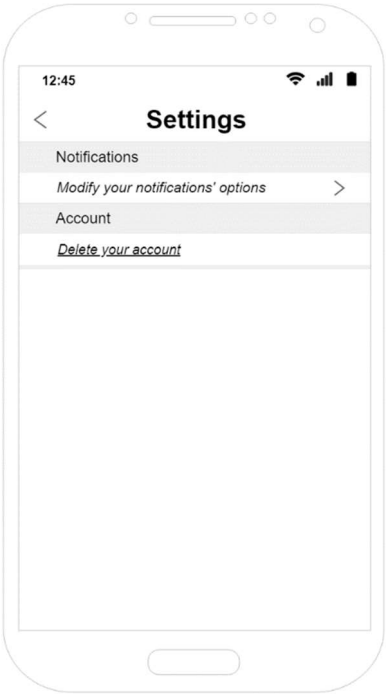


Figure 13: *Authority Settings*

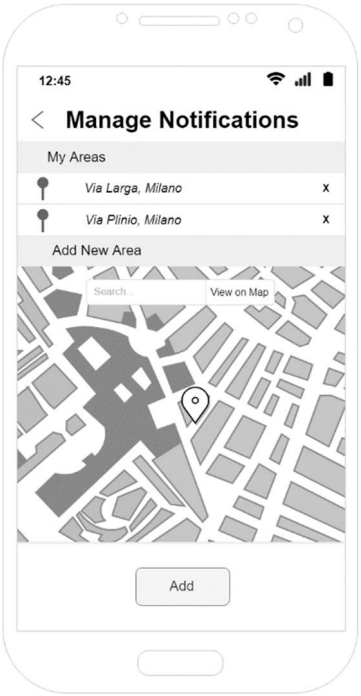


Figure 14: *Authority Manage Notifications*

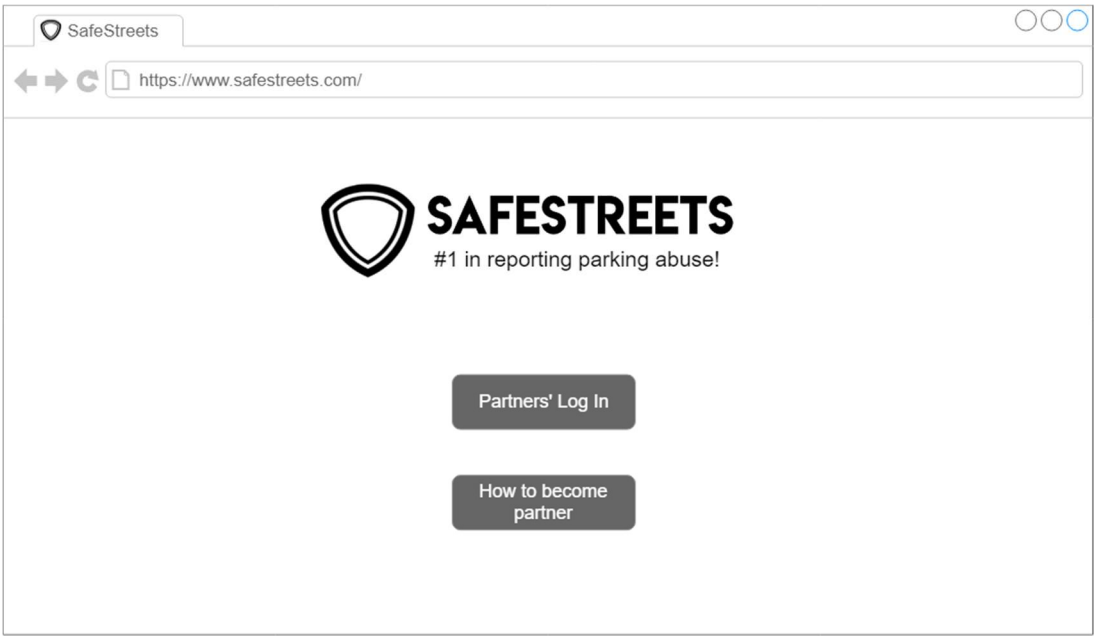


Figure 15: *Municipality Main Page*

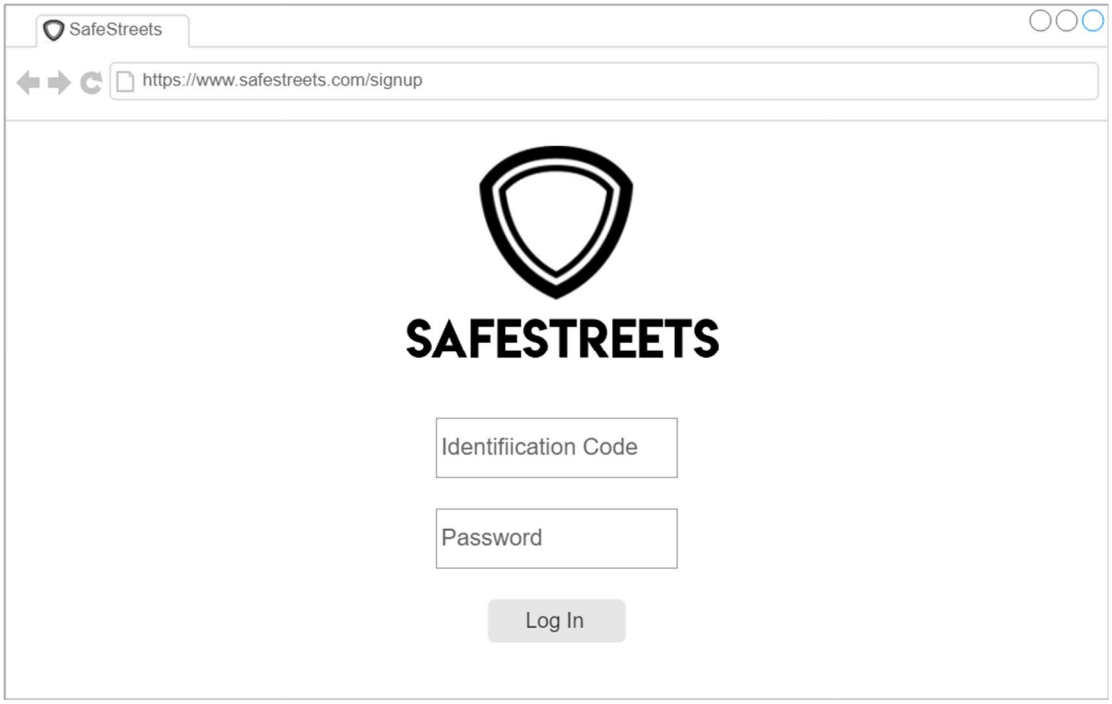


Figure 16: *Municipality Log In*

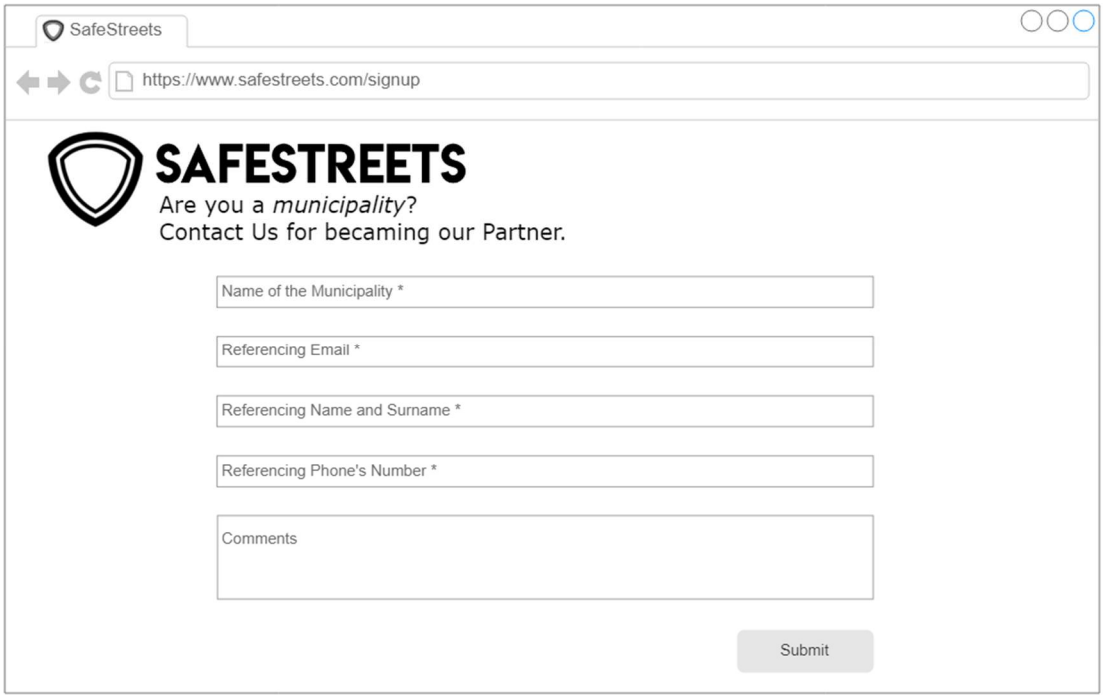


Figure 17: *Municipality Sign Up*

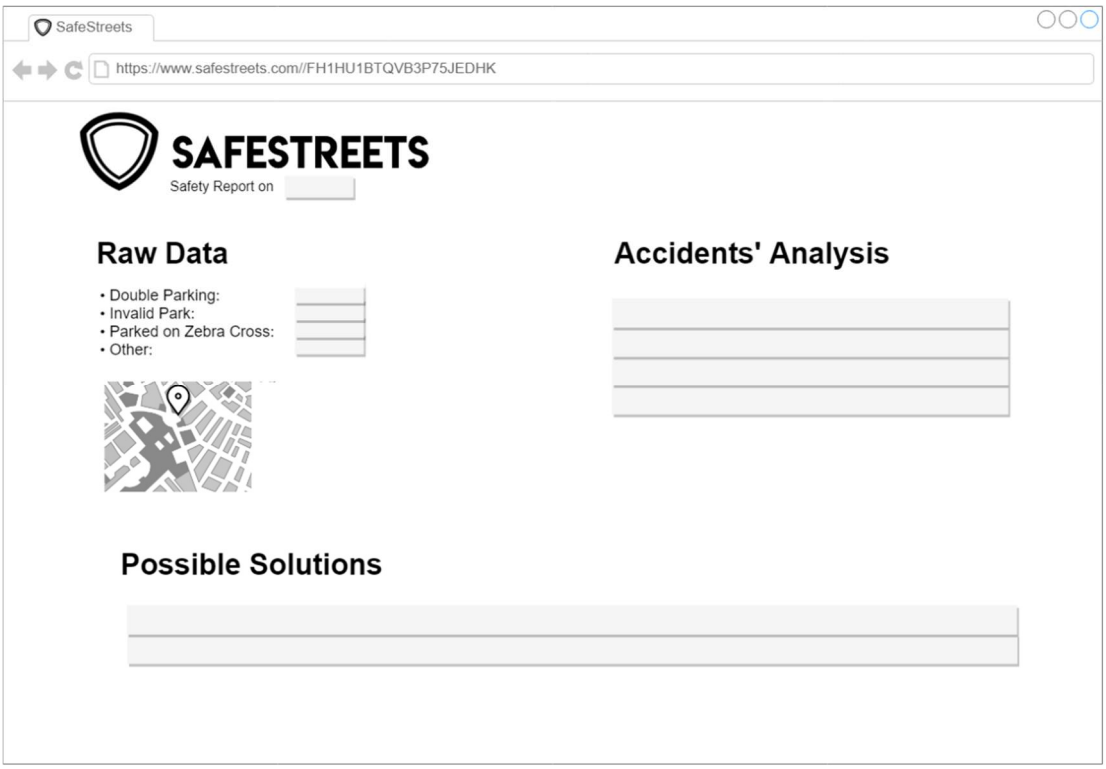


Figure 18: *Safety Report's Sample*

3.1.2 Hardware Interface

The system doesn't provide any hardware interface.

3.1.3 Software Interface

The system has two interfaces to interconnect with the external DBMS, the former one to acquire accident data from municipality, the latter to access the DB where authorities' matriculas are stored.
Although, there are no functionalities which can be used by an external system, so the S2B will not provide any API.

3.2 Functional Requirement

[R.1] The system shall allow users registration.

[R.2] The system shall distinguish basic users and authorities accounts.

[R.3] The system shall allow authority registration only to real authority.

a. The system shall be able to access the State's DB where authorities' matriculas are store.

[R.4] The system shall guarantee unique account for each user identifier (i.e. email, Identification Code).

[R.5] The system shall allow users to access their account only if they provide correct username and password.

[R.6] The system shall allow users to access *SafeStreets'* functionalities only after the login.

[R.7] The system shall allow users to delete their account.

a. Once an account is deleted, the system shall deny access.

[R.8] The system shall allow a basic user to upload violation data. Moreover:

a. The system shall let the user to choose the type of violation between a predefined list, without allow them to add a custom violation type.
b. The system shall read automatically the license plate from the photos uploaded.
c. The system shall add automatically the location of a new report through the GPS information.
d. The system shall deny reporting a violation with uncompleted data.
e. The system shall recognize syntactical error in data, like inexistent date and time.
f. Once report is correctly stored, the system shall notify the user.

[R.9] The system shall store internally the data.

[R.10] Once data is store, the system shall not erase it.

[R.11] The system shall never show the basic user who reported a particular violation.

[R.12] The system shall allow authority to choose to receive push notifications regarding new violations affecting a given geographical area.

[R.13] When a violation is reported, the system shall be able to generate a real time push notification to all authorities which have enable them in the area where the violation occurred.

[R.14] The system shall allow authority to access and view all data about a violation, without contradicting *[R.11]*.

[R.15] The system shall not allow a basic user to access all the data about a violation.

[R.16] The system shall allow an authority to reserve a violation. Moreover:

a. Once a violation is reserved, the system shall not allow other authorities to reserve it.
b. The system shall allow authorities to know which violations have been reserved.
c. Once report is correctly reserved, the system shall notify the authority

[R.17] Every time new violations data is acquired, the system shall be able to classify it according to violation type and geographical location.

[R.18] The system shall provide to app users an interface able to render mined data graphically, allowing showing geographical area, violation type and number of occurrences on a map.

[R.19] The system shall provide to authorities an interface where are listed all the violation.

[R.20] In case of connection error, the system shall be able to manage it notifying it to the users and without having anomaly behavior.

Municipality requirements:

[R.M1] The system shall allow municipalities to contact *SafeStreets* in order to make a partnership.

[R.M2] The system shall be able to access to accident data of the municipality using the API provided.

[R.M3] The system shall be able to integrate accident data of the municipality with *SafeStreets'* DB matching the location.

[R.M4] The system shall be able to mine the integrated data in order to find unsafe zones.

[R.M5] The system shall provide to municipality an interface able to render mined data graphically in order to highlight the unsafe zones.

[R.M6] The system shall be able to find correlations between accidents and violations.

[R.M7] Once identified correlations between accidents and violations, the system shall be able to generate a safety report with suggestions to reduce them.

[R.M8] The system shall show to each municipality only its own unsafe zone and safety reports.

In the following table the goals are matched with their requirements and their domain assumptions, in order to make the correlations easier to read:

Goal ID	Requirement ID	Domain Assumption ID
G.BU1	*R.1, R.2, R.4, R.5, R.6, R.8, R.9, R.10, R.20*	*D.1, D.2, D.5*
G.BU2	*R.1, R.2, R.4, R.5, R.6, R.11, R.15, R.17, R.18, R.20*	*None*
G.A1	*R.1, R.2, R.3, R.4, R.5, R.6, R.11, R.12, R.13, R.20*	*D.1, D.3, D.8*
G.A2	*R.1, R.2, R.3, R.4, R.5, R.6, R.11, R.14, R.15, R.16, R.19, R.20*	*D.3, D.4, D.8*
G.A3	*R.1, R.2, R.3, R.4, R.5, R.6, R.11, R.15, R.17, R.18, R.20*	*D.3, D.8*
G.M1	*R.4, R.5, R.6, R.M1, R.M2, R.M3, R.M4, R.M5, R.M8*	*D.6, D.7*
G.M2	*R.4, R.5, R.6, R.M1, R.M2, R.M3, R.M6, R.M7, R.M8*	*D.6 D.7*

Table 3: *Traceability Matrix*

3.3 Scenarios

3.3.1 Double Parking Report

Mia is a 35 years old mother of 2 little kids, one of 5 and one of 7. She is currently unemployed and spends her time taking care of her sons. She loves watching them play outside, but unfortunately, she lives in a busy neighborhood, with vehicles always double parked. It is not uncommon for cars to do dangerous maneuvers to move through all the vehicles parked, so to prevent her children from being run over, Mia downloads the *SafeStreets* app. The system allows her to create an account after compiling the sign-up form (UC.1).

The next day, while playing with her kids, she notices a car double parked before a narrow turn: she launches *SafeStreets* app, compiles a report and after checking that all the information are correct, she submits it (UC.5).

3.3.2 Reports Check

Johnny is 62 years old policeman and although his age, he is really into technology. During his career he has always tried to feel useful for the citizen and now that he is near to retirement, he prefers to do easy tasks as parking fines. Recently, thanks to his grandson Jordi, he discovered the *SafeStreets* system and started to use it, creating a new authority account (UC.2).

During his patrols, usually he receives from five to seven notifications, about some violations that have been reported through the system. He opens the notifications and watches the photos: if the vehicle is indeed in violations, and if the report isn't too far from him, he reserves the report and drives to the location indicated in order to fine the car (UC.6).

3.3.3 Violations Statistics Check

Lana, 23 years old Mathematics' university student, is passionate about statistics. She heard from a colleague that *SafeStreets* app allows her to check the areas around her in order to view statistics about violations that had occurred.
Interested about that new app, she downloaded it and created a new account. Then, she fills the log in form with her credentials, and press the "*Log In*" button (UC.4).

Now, every time Lana is in a new city, she opens *SafeStreets* app, clicks on the *"View Statistics' map"* button and has fun checking which violations are the most common (UC.8).

3.3.4 Municipality Suggestion

Sofie, 44 years old, is a municipality's employee in the road safety department. She noticed that, in the last months, the numbers of accidents have been grown dangerously, and she can't understand why. Luckily, her manager forwards her a commercial email about a new system, *SafeStreets*, that is able to cross-reference the violation data that the new system gathers from its users with the already present municipally accidents DB's data, in order to generate a safety report. Sofie, intrigued by the offer, decides to try it. She goes with her PC to *SafeStreets* webpage and contact them through the "*Contact Us*" form (UC.3).

After few days, she receives a confirmation email from *SafeStreets* with her municipality credentials. After setting up the connections between the accidents DB and *SafeStreets* itself, she requests a safety report: in a couple of hours she receives another email from *SafeStreets* with a permalink to the webpage where the safety report is present (UC.9).

3.4 Use Cases

For each use case we don't report the exception about connection error in order to be concise. This exception is common to all use cases and it is managed in the same way: the system notifies the issue to the user and the Flow of Event restarts from the previous point.

ID	UC.1
Name	Register of basic user
Actors	Basic user
Entry Condition	*SafeStreets* app downloaded on basic user's smartphone
Flow of Events	1. Basic user clicks on *SafeStreets* app's icon entering in the Log In page. (Figure 1) 2. Basic user taps on Sign-Up button. 3. Basic user fills the Sign-Up form without checking Authority flag. (Figure 2) 4. Basic user confirms his/her data clicking on Sign-Up button. 5. The system checks the validity of data. 6. The system creates a basic user account. 7. The system returns on the Log In page.
Exit Condition	Basic user's account has been successfully created and added to the system database.
Exceptions	5.* The field email is already taken, or date of birth is an invalid set, or a field is not filled. The system notifies the issue to the user and the Flow of Events returns to 3, erasing invalid fields.
Special Requirements	

ID	UC.2
Name	Register of authority
Actors	Authority
Entry Condition	*SafeStreets* app downloaded on authority's smartphone
Flow of Events	1. Authority clicks on *SafeStreets* app's icon entering in the Log In page. (Figure 1) 2. Authority taps on Sign-Up button. 3. Authority fills the Sign-Up form checking Authority flag. (Figure 8) 4. Authority confirms his/her data clicking on Sign-Up button. 5. The system checks the validity of data, it tries to match the Personal ID and the District Name with the ones on the State's DB. 6. The system creates an authority account. 7. The system returns on the Log In page.
Exit Condition	Authority's account has been successfully created and added to the system database.
Exceptions	5.* The Personal ID and the District Name do not correspond in the state's DB or the Personal ID is already linked to another account, or a field is not filled. The system notifies the issue to the user and the Flow of Events returns to 3, erasing invalid fields. 5.** The access to the State's DB is denied. The system notifies the issue to the user and the Flow of Events goes to 7.
Special Requirements	

ID	UC.3
Name	Contact of municipality
Actors	Municipality employee, *SafeStreets* employee
Entry Condition	Browser open on municipality employee computer
Flow of Events	1. Employee navigates to *SafeStreets*' website. 2. Employee clicks on How to partner with us entering in Sign Up page. (Figure 17) 3. Employee fills the form. 4. Employee clicks on Contact Us. 5. The system forwards the request to a *SafeStreets* employee. 6. A *SafeStreets* employee reviews the request and accepts the partnership. 7. A *SafeStreets* employee sends the credential for logging in and asks the access to their DB. 8. Employee sends the credential for accessing accidents database.
Exit Condition	Municipality's account has been successfully created and added to the system database. *SafeStreets* can view accidents data.
Exceptions	5.* A *SafeStreets* employee reviews the request and refuses the partnership. The system notifies the issue to the user.
Special Requirements	

ID	UC.4
Name	App Log In
Actors	App user
Entry Condition	*SafeStreets* app is open on basic user's or authority's smartphone.
Flow of Events	1. User fills the email and password field. (Figure1) 2. User press the Log In button.
Exit Condition	User is successfully logged in the system and can exploit all the system services; the graphical interfaces moves to the default screen. (Figure 3 / Figure 9)
Exceptions	2.* The system discovers that field email is invalid or that field password doesn't correspond to the one paired with the email. The system notifies the issue to the user and the Flow of Events returns to 1.
Special Requirements	

ID	UC.5
Name	Report traffic violation
Actors	Basic user
Entry Condition	Basic user has logged in. Basic user has *SafeStreets* app opened on the default page. (Figure 3)
Flow of Events	1. Basic user taps Report a Violation button. 2. The app responds by presenting a form to the basic user. (Figure 5) 3. Basic user fills the required fields: choses the type of violation in a list, takes some photo of the vehicle and checks if the location, the data and the time are correct (UML 2). 4. Basic user presses the submit button. 5. The system receives the report and checks the correctness of the information. 6. The system reads the vehicle license plate from the uploaded photos.
Exit Condition	The new violation is store and basic user has received an acknowledgment.
Exceptions	3.* Basic user finds out some errors between location, data or time, he/she corrects them selecting new parameters from a list. The Flow of Events continuous to 4. 5.* There are some fields that are incomplete or incorrect, the system notifies the issue. The Flow of Events restarts from 3. 7.* There aren't any authority that has enabled notifications in the area where the violation occurred. This event is skipped.
Special Requirements	

ID	UC.6
Name	Reserve traffic violation
Actors	Authority
Entry Condition	Authority has logged in. Authority has received at least one notification (Figure 12).
Flow of Events	1. Authority taps on notification. 2. The system responds loading the Report Screen page. (Figure 11) 3. Authority chooses one of the available reports. 4. Authority taps on *View Photos.* 5. Authority taps on Reserve button. 6. The system receives the reservation and links the report with the authority.
Variations	Point 4 of Flow of Events is optional, so it can be skipped and go from 3 to 5 directly.
Exit Condition	The report turns into grey and the report is reserved for the authority.
Exceptions	5.* Authority clicks on Reserve but meanwhile the report he/she want to reserve has already been reserved by someone else. The issue is notified to the users and the Flow of Events terminates without verifying the exit condition.
Special Requirements	

ID	UC.7
Name	Add notification
Actors	Authority
Entry Condition	Authority has logged in. Authority in on Setting Page (Figure 13).
Flow of Events	1. Authority taps on "*Manage your notification option*". 2. The system loads Manage Notification page (Figure 14). 3. Authority searches the interested area. 4. The system load on the map the area searched. 5. Authority frames the area in the screen for better selection. 6. Authority taps on "*Add*".
Exit Condition	Notification is added in "*My Area*" section.
Exceptions	
Special Requirements	

ID	UC.8
Name	Exploit mined data
Actors	App user
Entry Condition	App user already has logged in and he/she is on the default page.
Flow of Events	1. App user clicks on View Statistics Map button. 2. The system loads the Map Screen (Figure 6). 3. The system loads the highlight zone on the map.
Exit Condition	Maps are rendered on app user screen and they can view the statistics.
Exceptions	2*. There are not enough data to mine on the current map yet. The map shown is empty and the Flow of Events terminate.
Special Requirements	

ID	UC.9
Name	Request new safety report
Actors	Municipality's employee
Entry Condition	Municipality's employee has *SafeStreets*' site open.
Flow of Events	1. Municipality's employee clicks on "*Partners Log In*" button (Figure 15). 2. Municipality's employee logs in with his/her municipality's credentials. 3. Municipality's employee clicks on "*Request new safety report*". 4. The system creates a new report. 5. The system notifies the municipality's employee with a link to the safety report. 6. Municipality employee opens the link and views the latest safety report (Figure 18).
Exit Condition	Safety Report is open, municipality's employee knows how to improve streets safety.
Exceptions	2.* The system discovers that field Identification Code is invalid or that field password doesn't correspond to the one paired with the Identification Code. The system notifies the issue to the user and the Flow of Events returns to 1.
Special Requirements	4.* The safety report shall be generated in at most 48 hours.

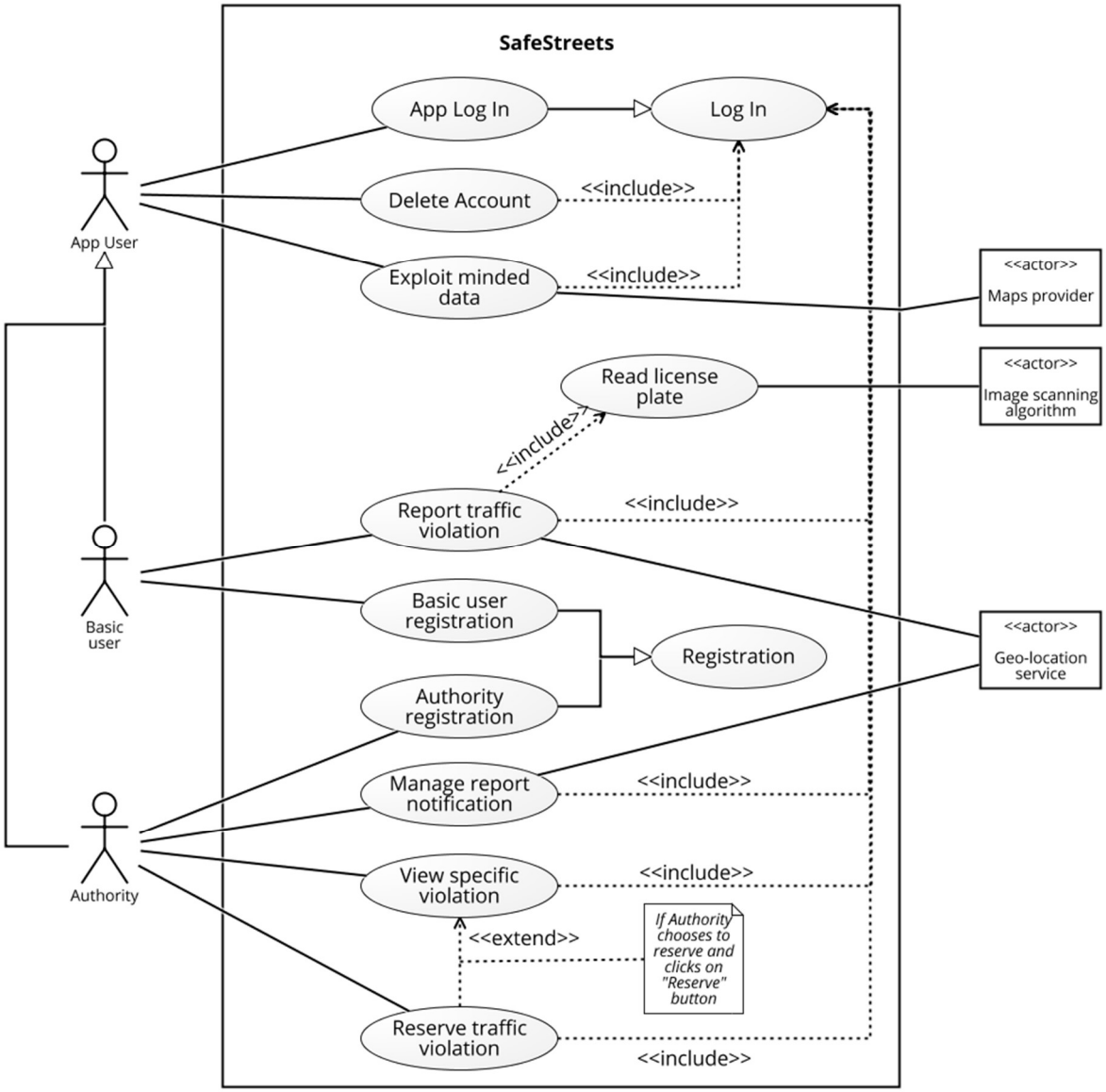
In the following traceability matrix, we are mapping, for each use case, which goals directly illustrate, and which requirement are related to.

Only major requirements are listed for each use case, in order to improve readability and reduce redundancy (i.e. R.6 and R.20 are not reported even though they are necessary for all use cases).

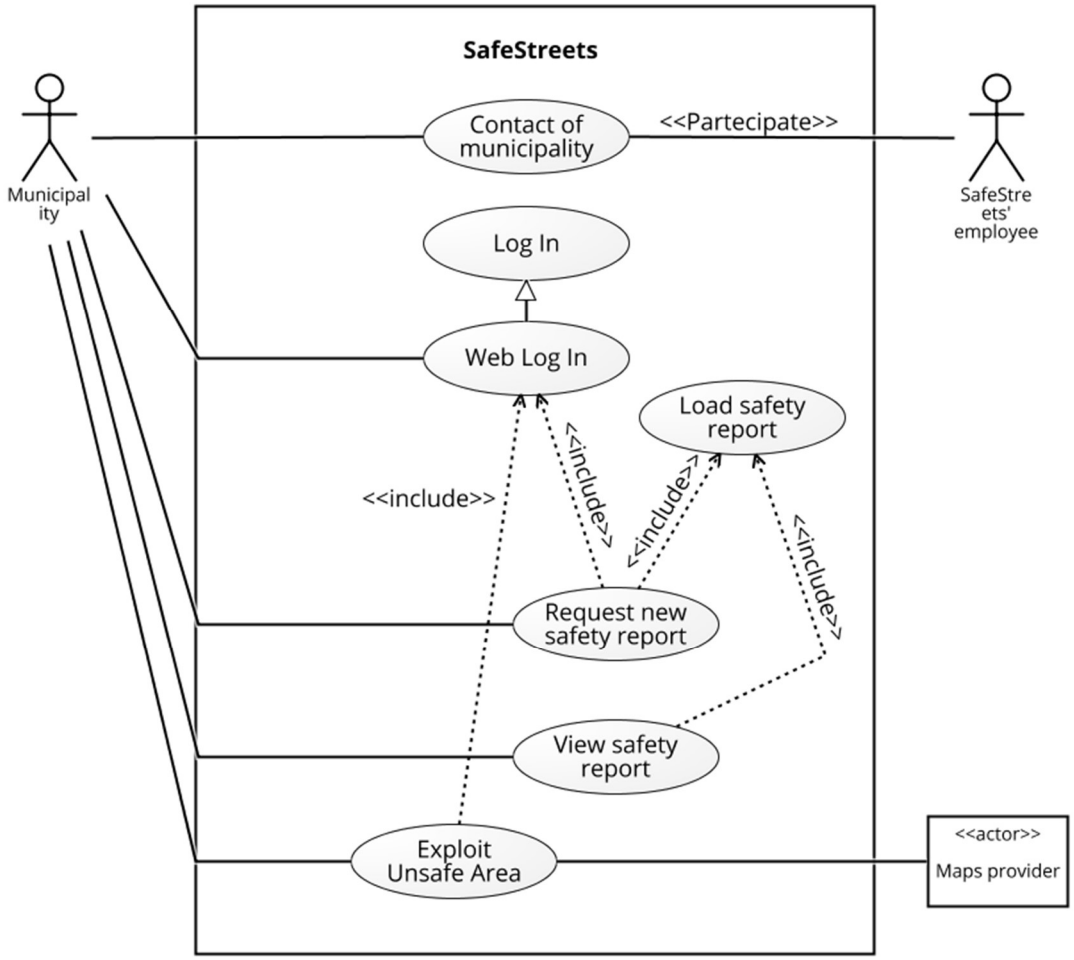
Use Case ID	Goal ID	Requirement ID
UC.1	\	*R.1, R.2, R.4*
UC.2	\	*R.1, R.2, R.3, R4*
UC.3	\	*R.M1, R4*
UC.4	\	*R.5*
UC.5	*G.BU1*	*R.8, R.9, R.10, R.21*
UC.6	*G.A1, G.A2*	*R.11, R.12, R.13, R.14, R.16, R.19*
UC.7	*G.A1*	*R.12*
UC.8	*G.BU2, G.A3*	*R.11, R.17, R.18*
UC.9	*G.M2*	*R.M6, R.M7, R.M8*

Table 4: *Traceability Matrix*

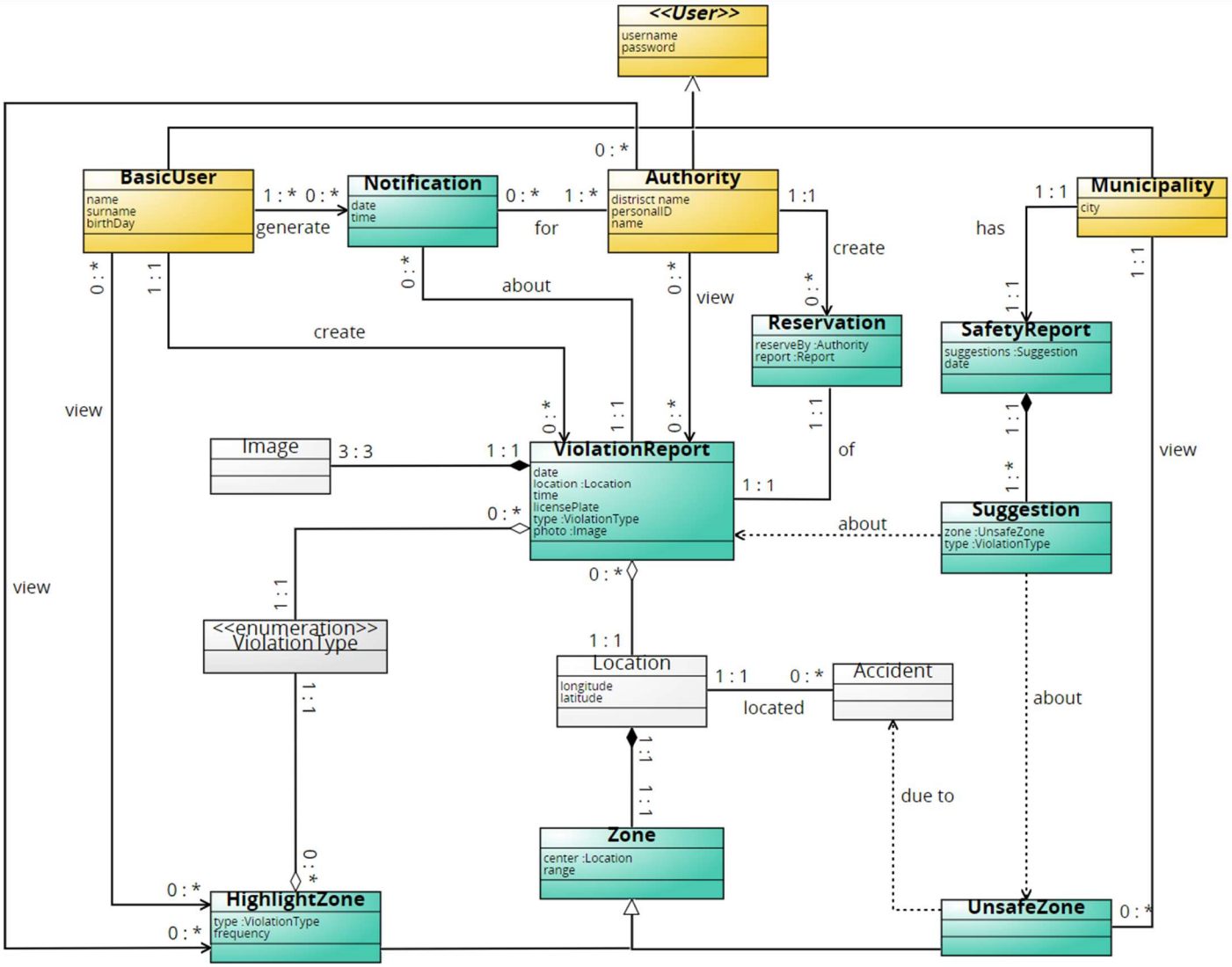
3.4.1 UML Modelling



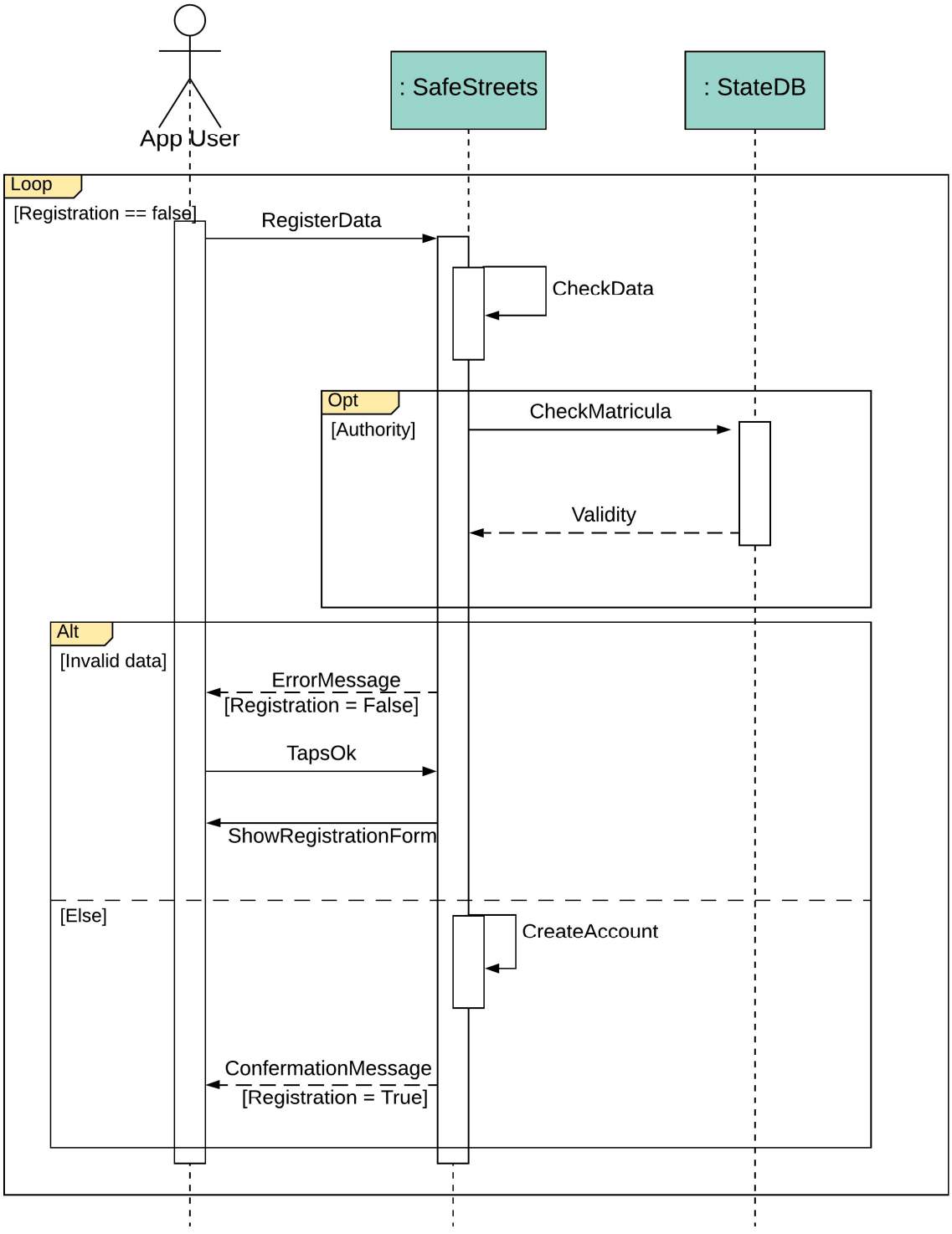
UML 1: *App users Use Case Diagram*



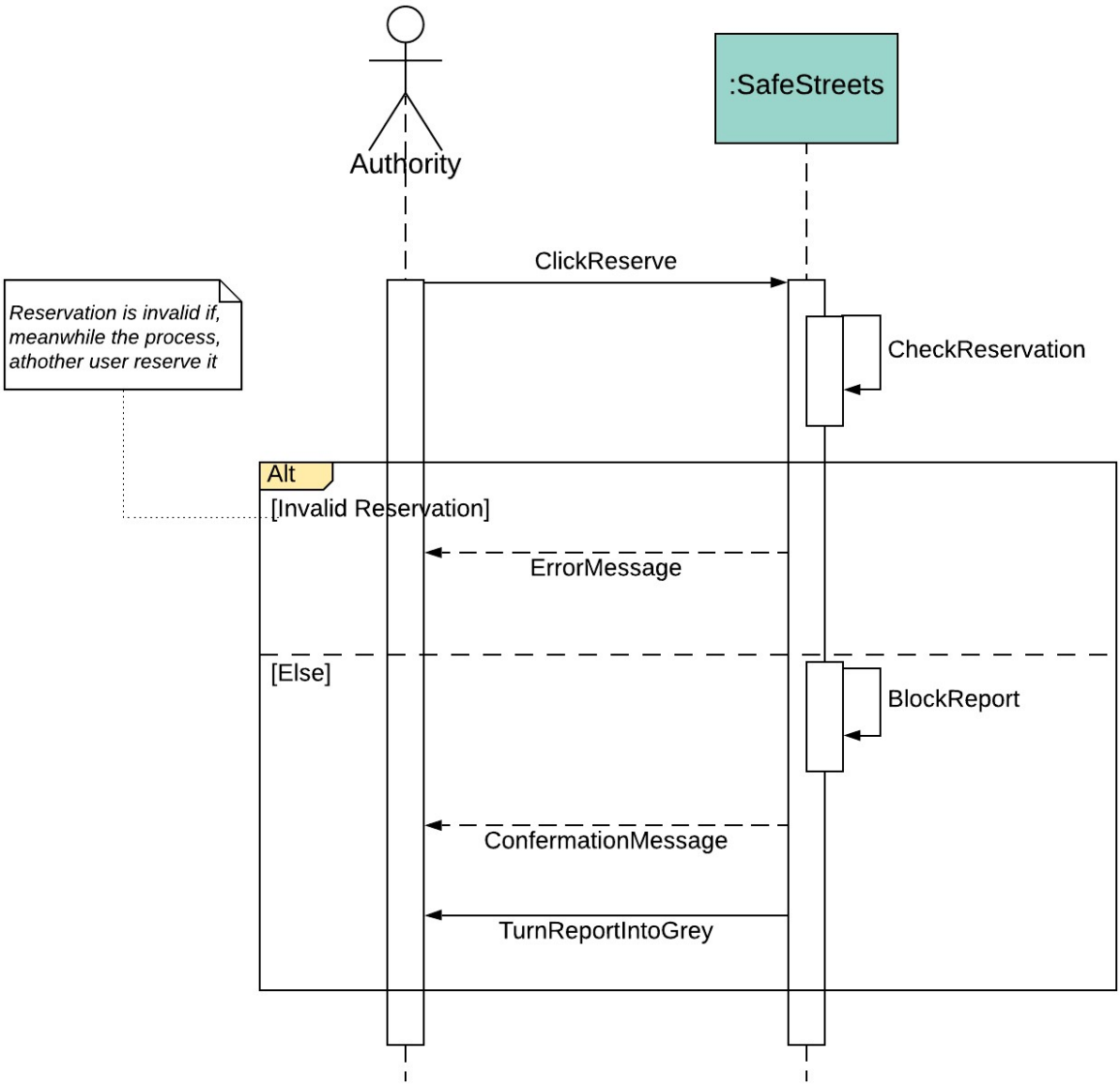
UML 2: *Municipality Use Case Diagram*



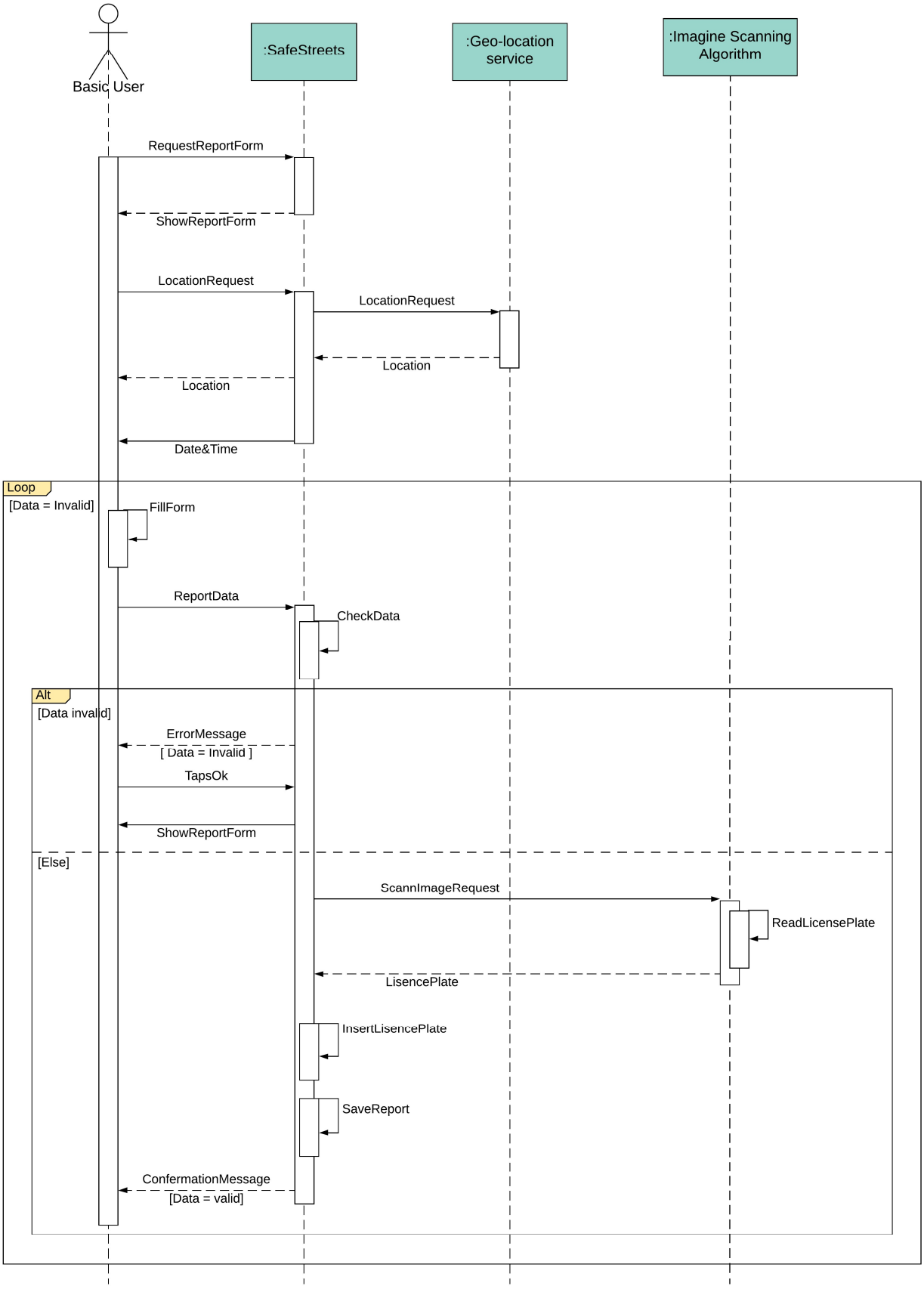
UML 3: *Class Diagram*



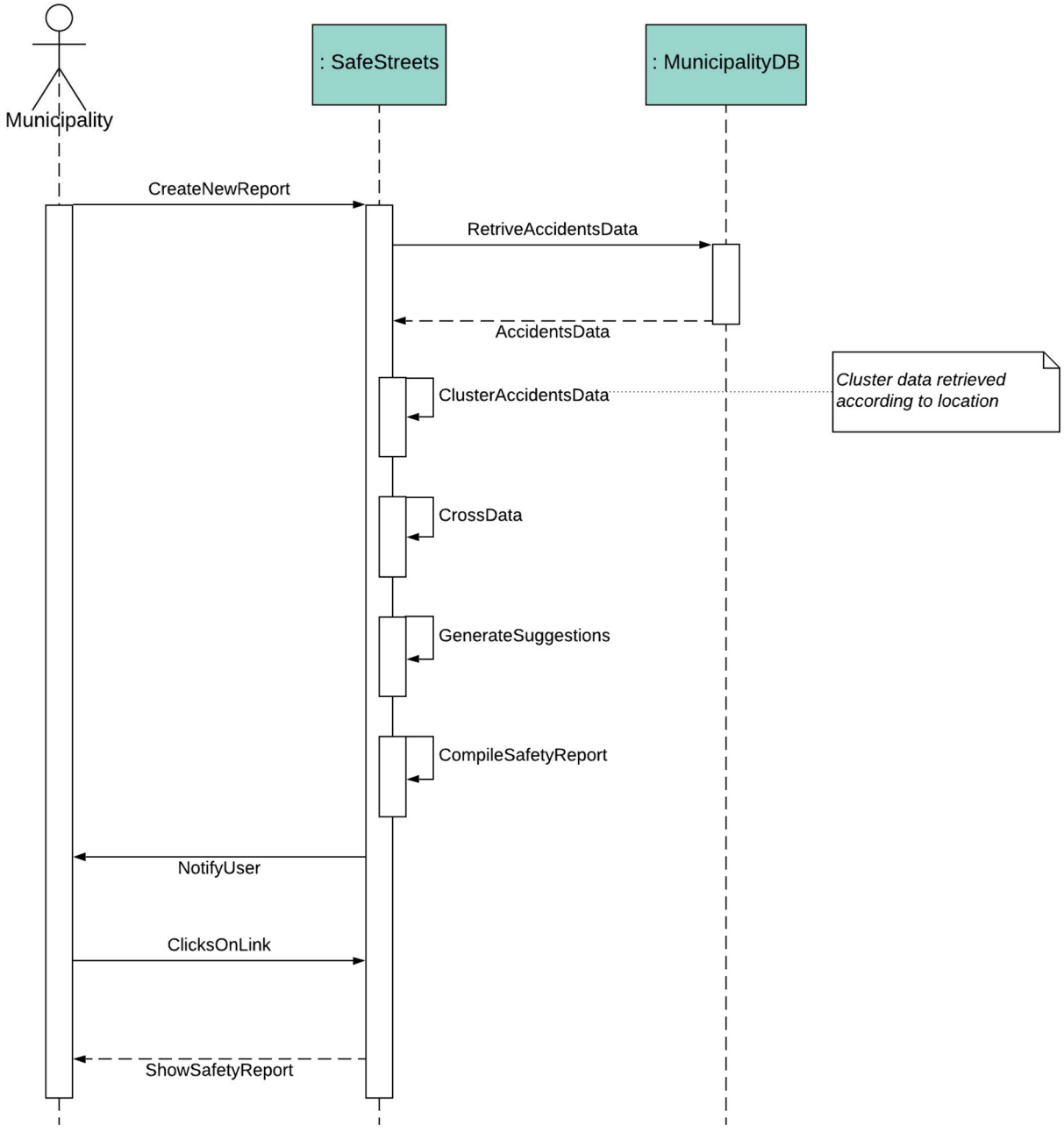
UML 4: *Registration Sequence Diagram*



UML 5: *Reservation Sequence Diagram*



UML 6: *Violation report Sequence Diagram*



UML 7: *Generate safety report Sequence Diagram*

3.5 Performance requirements

[R.P1] The system shall be able to manage a great number of concurrent requests, by any kind of user.

[R.P2] Once a new safety report is required, the system shall compile it in less than 48 hours.

[R.P3] Once a new traffic violation is reported, the system shall notify all authorities in less than 15 seconds.

3.6 Design constrains

3.6.1 Standard compliance

The system shall accept only photos of violation in *.jpeg* format.
The geo-location coordinates shall be encoded as longitude and latitude degree and all measures (i.e. distance and area) shall follow the metric system.

3.6.2 Hardware limitations

The smartphone app version of *SafeStreets* requires Android/iOS operating system, geo-localization system, internet connectivity and a built-in camera to the smartphone on which it is installed, in order to exploit its basic functionalities.

For the municipalities it is also required a modern browser, we recommend Chrome. See section 2.4.3 for a detailed list.

3.6.3 Any other constraints

There aren't any other constraints.

3.7 Software system attributes

3.7.1 Reliability

Given the function *Reliability = 1 – Probability of Failure*, the system shall guarantee a reliability as close to 1 as possible. The main cause of Failure may be an unexpected increase of communication traffic within our system, in the design document are shown architecture and pattern as solution.

3.7.2 Availability

The system is expected to be available 99.99% of the time. In case of failure, a warning must be displayed, and it shall be able to be recovered.

3.7.3 Security

The system shall store securely users' credentials and data collected into its database. A cryptographical algorithm is require for storing users' password.

The system shall not allow authorities to retrieve any personal information (including, but not limited to name, surname) about the basic users whose reports are being submitted into *SafeStreets'* database.

During the transfer of data from municipality's and state DBs to *SafeStreets* database, encryption technique shall be used in order to guarantee privacy.

3.7.4 Maintainability

The software shall be maintainable according to the development principles. A future upgrade of the functionalities must be easier and cheaper as possible.

3.7.5 Portability

The system shall run on most Android/iOS devices, and on most modern browser.

4 Alloy modelling

```
open util/boolean

/**
 *    In this Alloy model we describe our world with some
 *    simplification.
 *    - App: we describe the report of a new violation, the
 *           reservation of a violation by an authority and the
 *           notification to authority.
 *           Notification are enabled since the origin of the
 *           world in order to avoid the time component in the
 *           model.
 *           Lastly, we defined the highlight zone where violation
 *           reports are clustered together
 *    - Web: We model the generation of a safety report, describing
 *           it as a set of suggestion
*/

sig Location {}

sig Username {}

abstract sig AppUser {
      username: one Username,
      userPosition: one Location
}

sig BasicUser extends AppUser {}

sig Authority extends AppUser {
      //Notification area are simplified as a set of Locations
      notifArea: set Location,
      //Notification are simplified as a set of Reports
      notifReceived: set Report
}

sig Municipality {
      area: some Location
}

/**
  Only 3 types of violations are provided as example
*/
abstract sig ViolationType{ }

sig DoubleParking extends ViolationType{} {
      #DoubleParking <= 1
}

sig ZebraCrossParking extends ViolationType{} {
      #ZebraCrossParking <= 1
}

sig NoParkingZone extends ViolationType{} {
      #NoParkingZone <= 1
}

/**
  Violation report doesn't provide all the information required,
```

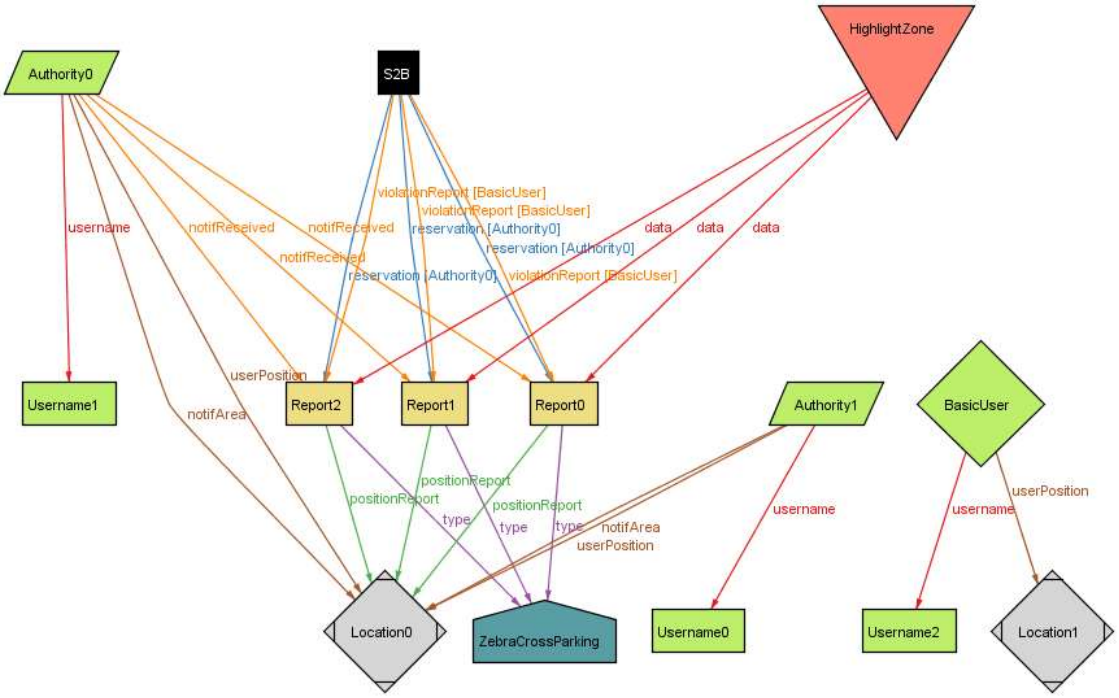
```
  data and time are omitted for simplification
*/
sig Report {
      type: one ViolationType,
      positionReport: one Location,
      //True if this report is reserved by an Authority, false
      //otherway
      reserved: one Bool
} {
      // Report can be reserved if and only if an Authority has
      //reserved it
      reserved = True iff
            this in Authority.(S2B.reservation)
}

/**
      A zone is simplified as unique location instead of an area of
      5Km^2
      To create a cluster are needed only 3 report instead of 15
*/
sig HighlightZone {
      data: some Report
} {
      //At least 3 reports
      #data > 2
      //All report of the same type
      #(Report.type) = 1
      //All report in the same position
      #(Report.positionReport) = 1
}

/**
      A zone for the correlation is simplified as unique location
      instead of an area of 5Km^2
      To create a correlation are necessary only 3 reports and 3
      accidents instead of 15 and 5
*/
sig Suggestion {
      correlation: Accident -> Report
} {
      // At least 3 correlations
      #correlation > 2
      // Mapping 1:1 Accident and Report
      #correlation = #correlation.Report
      #correlation = #Accident.correlation
      // Accidents and report must have the same location
      (Accident.correlation).positionReport =
      (correlation.Report).accidentLocation
      #(Accident.correlation).positionReport = 1
      #(correlation.Report).accidentLocation = 1
      // All reports have to be of the same type
      #(Accident.correlation).type = 1
}

sig Accident {
      accidentLocation: Location
}


/**
      S2B storage with all its data
```

```
*/
sig S2B {
      // Violation report made by a Basic User
      violationReport: BasicUser -> set Report,
      // Reservation on a report by an Authority
      reservation: Authority -> set Report,
      // Safety Report of a Municipality
      safetyReport: Municipality -> set Suggestion
} {
      // No ViolationType outside the system
      Report.type = ViolationType
      // No Report outside the system
      BasicUser.violationReport = Report
      // No Suggestion outside the system
      Municipality.safetyReport = Suggestion
      // No two BasicUser for same Report
      no r: Report |  #violationReport.r != 1
      // No two Authority for same report reservation
      no r: Report |  #reservation.r > 1
      // No two municipality for the same suggestion
      no s: Suggestion | #safetyReport.s != 1
}

/**
   Each suggestion of a municipality is made by data
   from the municipality itself
   (i.e. the location of accidents and report are inside the
   municipality area)
*/
fact suggestionOfMunicipality {
      no sg: Suggestion, m: Municipality |
            sg in m.(S2B.safetyReport) and
            #(m.area & Accident.(sg.correlation).positionReport) !=
            #m.area
}

/**
   Each municipality has a unique area
*/
fact uniqueArea{
      no disj m, m': Municipality |
            some l: Location | l in m.area and l in m'.area
}

fact UniqueUsername {
      no disj u, u' : AppUser |
            u.username = u'.username
}

/**
   Authority a  reserve the report r
   s is the update system
   s' is the old system
*/
pred reserve[s, s': S2B, a: Authority, r: Report] {
      s.violationReport = s'.violationReport
      // r can't have been already reserved
      r not in Authority.(s'.reservation)
      s.reservation = s'.reservation + a->r
      r.reserved = True
}
```

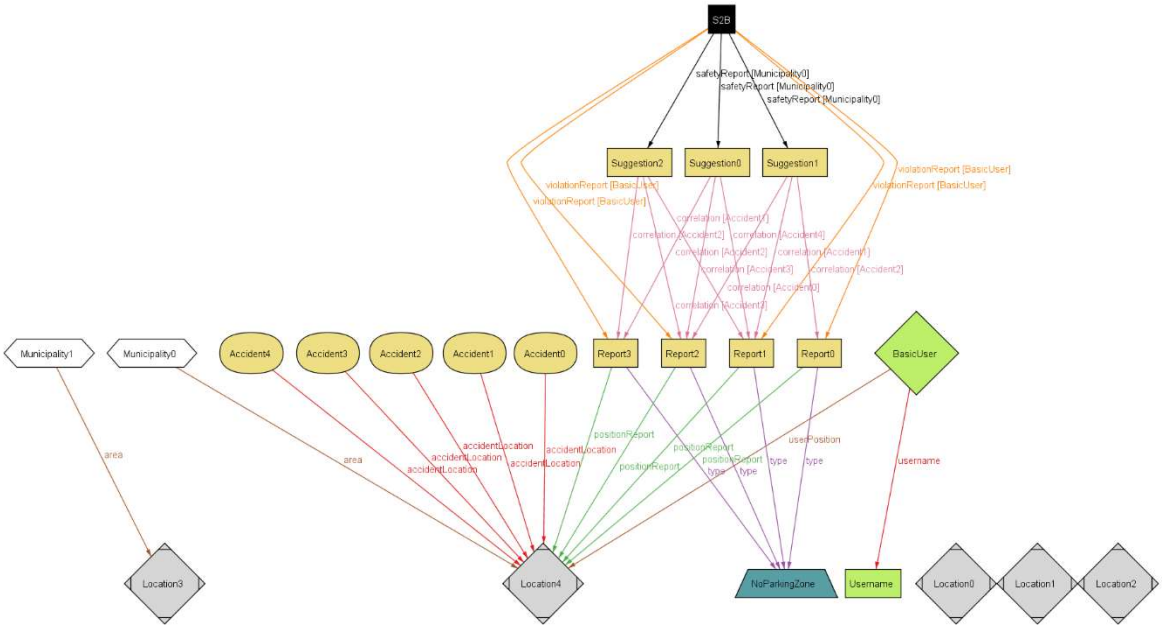
```
/**
   BasicUser u report a new violation of type vt
   s is the update system
   s' is the old system
*/
pred addReport[s, s': S2B, u: BasicUser, r: Report, vt:
ViolationType] {
      s.reservation = s'.reservation
      r.positionReport= u.userPosition
      r.reserved = False
      r.type = vt
      s.violationReport = s'.violationReport + u->r
      notify[r]
}


/**
   Notify each authority who has enabled notification in the
   zone of the report
*/
pred notify[r: Report ] {
           all a, a': Authority |
           a.username = a'.username and
           r.positionReport in a.notifArea =>
                ( a.notifArea = a'.notifArea and
                a.notifReceived = a'.notifReceived + r )
}

/**
   Verify that each time a BasicUser reports a violation,
   all the interested Authorities are notified
*/
assert pushNotification {
     all u: BasicUser, s, s': S2B, r: Report, vt: ViolationType |
           all a: Authority |
                u.userPosition in a.notifArea and
                addReport[s, s', u, r, vt] =>
                     r in a.notifReceived
}

/**
   Verify that all correlations are created
*/
assert allCorrelation {
     all u: BasicUser, s, s': S2B, r: Report, vt: ViolationType,
     m: Municipality |
           // If the report position is inside a municipality area
           u.userPosition in m.area and
           // and if exists at least two reports in the same
           //position with the same type
           #((positionReport :> u.userPosition).Location &
             (type :> vt).ViolationType) > 1  and
           // and if exists at least two accidents in the same
           // position
           #(Accident.accidentLocation :> u.userPosition) > 1 and
           // and if the new report is added to the system
           r not in BasicUser.(s'.violationReport) and
           addReport[s, s', u, r, vt] =>
           // then exists a suggestion with that report
                some sg: Suggestion |
```

```
                    r in Accident.(sg.correlation) and
                    sg in m.(s.safetyReport)
}

/**
     Verify that all HighlightZone are created
*/
assert allHighlightZone {
          all u: BasicUser, s, s': S2B, r: Report,
          vt: ViolationType |
               // and if exists at least two reports in the same
               // position with the same type
               #((positionReport :> u.userPosition).Location &
               (type :> vt).ViolationType) > 1  and
               // and if the new report is added to the system
               r not in BasicUser.(s'.violationReport) and
               addReport[s, s', u, r, vt] =>
               // then exists a HighlightZone with that report
               some hz: HighlightZone |
                    r in hz.data
}

check allCorrelation
check allHighlightZone
check pushNotification
run reserve
run addReport
run {#Municipality=0
     #Accident = 0
     #S2B=1
     #violationReport>0
     #reservation>0
     #HighlightZone>0
     #Location>1} for 3
run {#Authority=0
     #HighlightZone=0
     #S2B=1} for 5
```

```
Executing "Check allCorrelation"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3994 vars. 252 primary vars. 8540 clauses. 27ms.
   No counterexample found. Assertion may be valid. 0ms.

Executing "Check allHighlightZone"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3879 vars. 249 primary vars. 8327 clauses. 29ms.
   No counterexample found. Assertion may be valid. 0ms.

Executing "Check pushNotification"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3844 vars. 252 primary vars. 8189 clauses. 26ms.
   No counterexample found. Assertion may be valid. 15ms.

Executing "Run reserve"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3586 vars. 246 primary vars. 7711 clauses. 30ms.
   Instance  found. Predicate is consistent. 29ms.

Executing "Run addReport"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3788 vars. 249 primary vars. 8100 clauses. 32ms.
   Instance  found. Predicate is consistent. 35ms.

Executing "Run run$6 for 3"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3737 vars. 234 primary vars. 8592 clauses. 36ms.
   Instance  found. Predicate is consistent. 20ms.

Executing "Run run$7 for 5"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   11531 vars. 800 primary vars. 29444 clauses. 33ms.
   Instance  found. Predicate is consistent. 44ms.

7 commands were executed. The results are:
   #1: No counterexample found. allCorrelation may be valid.
   #2: No counterexample found. allHighlightZone may be valid.
   #3: No counterexample found. pushNotification may be valid.
   #4: Instance found.    reserve is consistent.
   #5: Instance found.    addReport is consistent.
   #6: Instance found.    run$6 is consistent.
   #7: Instance found.    run$7 is consistent.
```



World 1: *App World – Run$6*



World 2: *Web World – Run$7*

5 Effort Spent

Date	Falconi	Galli	Theme
23/10	1	2	Problem analysis
24/10	3	1.5	Goals, Definitions
25/10	3	1.5	UI Design, 2.4 paragraph
26/10	3	3	UI Design, Requirement
28/10	1	1	Use Cases
29/10	1.5		UI
30/10	2	3	Use Cases, Scenarios
1/11	2		Section 2
2/11	2	4	Section 3, UMLs, Alloy
3/11	3	4.5	Alloy, Revision
5/11	2	3.5	Sequence diagram, Revision
6/11	5	5	UMLs, Revision
7/11	4	4	Traceability Matrices correction, Revision section 1, 2
10/11	2	2	General revision
TOTAL	34.5	35	

6 References

6.1 Tool used

In this section we will list the tools used to produce this document:

- *Office Microsoft Word* for document writing and building;
- *draw.io* for UI mock-ups;
- *lucidchart.com* for UML sequence diagrams and state diagram;
- *signavio.com* for UML use case diagrams and class diagram;
- *Alloy analyzer* for Alloy modelling and world generation.

6.2 Document References

- Alloy Guide
  http://alloy.lcs.mit.edu/alloy/
- Slides of the course by Prof. Di Nitto
  https://beep.metid.polimi.it/