POLITECNICO
MILANO 1863

M.Sc. Computer Science and Engineering
Software Engineering 2 Project

Requirements Analysis and Specification Document

Ferrara Fabiana, Formicola Stefano, Guerra Leonardo

10 November 2019

GitHub Repository: https://github.com/ste7en/FerraraFormicolaGuerra

Version 1.0

Contents

Chapter 1

Introduction

1.1 Purpose

This document constitutes the Requirement Analysis and Specification Document (i.e. RASD). Its purpose is to analyze the requirements that will lay the foundations of application services, to specify the application domain, the entities involved and their relationship, to clearly explain the objectives, the constraints and the features that are going to be implemented.
SafeStreets is a crowd-sourced application that intends to provide *Users* with the possibility to notify *Municipality* when *Traffic violations* occur, and in particular parking violations. The application allows *Users* to send pictures of violations, including their date, time, and position to *Municipality*.

1.1.1 Goals

G_1 Collect *User report* in SafeStreets Database;

G_2 Send *User report* to *Municipality* as soon as it is generated;

G_3 Send *User picture* to the *Licence plate recognition service* as soon as it is received;

G_4 Allow *Municipality* to visualize *Detailed statistics*;

G_5 Allow *Users* to visualize *Public statistics*;

G_6 Suggest *Possible interventions* to *Municipality*;

G_7 Collect information about *Ticket feedback*;

G_8 Allow *Users* to visualize their contribution to SafeStreets.

1.2 Scope

Crowd-sourced applications have become more popular nowadays thanks to the massive diffusion of smart devices and the consequent interconnection between people so that they started feeling part of a community, where everyone can contribute concretely.

SafeStreets is a crowd-sourced application whose aim is, indeed, to provide a tool to allow registered citizens to help *Municipality*, reporting *Traffic violations* that occur in their cities. In particular, they can take pictures of parking violations, specifying their type. A *License plate recognition service* recognizes the license plate from the *User picture*. SafeStreets sends the generated *User reports* to *Municipality*, granting they have not been corrupted. Examples of parking which represent a violation are the ones on bike lanes, sidewalks, in front of vehicle entrances or on reserved parking lots.
Users and *Municipality* can also retrieve analytics about data collected by the application, in order to obtain information, for example, about violations in certain areas or to identify vehicles with the highest number of violations, respectively.

Moreover, given that *Municipality* provides access to its data about *Accidents*, the system-to-be can integrate those information with its own data and finally suggest *Municipality Possible interventions* to apply.
SafeStreets can also build statistics from data sent by *Municipality* which concern issued *Traffic tickets* generated by *User reports*, for example about the most egregious offenders, or to show the effectivness of the SafeStreets initiative.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

User: individual registered to SafeStreets who agreed to the acquisition and processing of the data he/she sends;

User identifier: email provided by the *User* during the registration phase; it represents a single *User*;

User report: report created by the *User* including the fields *User identifier*, *User picture*, *Timestamp*, *Recognized license plate* and type of *Traffic violation*;

Terms and conditions: a set of regulations which *Users* must agree to follow in order to use SafeStreets;

Privacy statement: describes why and how SafeStreets collects and uses personal data and provides information about *Users*' rights.

Reference code: unique code which represents a single *Municipality*;

User picture: picture of *Traffic violation* taken by the *User*;

Traffic violation: violation of parking laws;

User position: *Users*' GPS location;

License plate recognition service: third party service which provides the recognition of the license plate from the *User picture*;

Recognized license plate: text representing the license plate recognized by the *License plate recognition service*;

Timestamp: digital record of date and time of the day when it has been registered;

Public statistics: set of *User report* statistics provided by SafeStreets, available for all the registered *Users*;

Detailed statistics: set of *User report* statistics provided by SafeStreets, only available for the *Municipality*;

Traffic ticket: a notice issued to someone who violates a traffic regulation;

Accidents: data of *Municipality* Database, concerning previous *Traffic violations*;

Possible interventions: suggestions automatically generated by SafeStreets and forwarded to the *Municipality*;

Ticket feedback: information on whether a *User report* led to the generation of a *Traffic tickets* or not;

Municipality: district of a town or a city which has local government.

1.4 Revision History

1. Version 0.1 - 20th October 2019 - Start of the RASD;
2. Version 1.0 - 10th November 2019 - First release.

1.5 Reference Documents

- Rumbaugh, Jacobson, Booch. 1999. *The Unified Modeling Language Reference Manual*. Addison-Wesley.
- MIT Software Design Group. *Appendix B: Alloy Language Reference*. http://alloytools.org/documentation.html
- MIT Software Design Group. *Tutorial Materials, Slides*. http://alloytools.org/tutorials/day-course/

1.6 Document Structure

This document is structured as follows:

1. Introduction A general introduction to the goals, the phenomena and the scope of the system-to-be. It aims giving general but exaustive information about what this document is going to explain.

2. Overall Description A general description of the product to be and its requirements. This section provides several information that are explained in detail in Section 3.

3. Specific Requirements All software requirements are explained using scenarios, use-case diagram and activity diagram. Non-functional and functional requirements are also cited.

4. Formal Analysis using Alloy This section includes Alloy code that describes the model and shows its soundness and correctness.

5. Effort spent Effort spent by all team members shown as the list of all the activities done during the realization of this document.

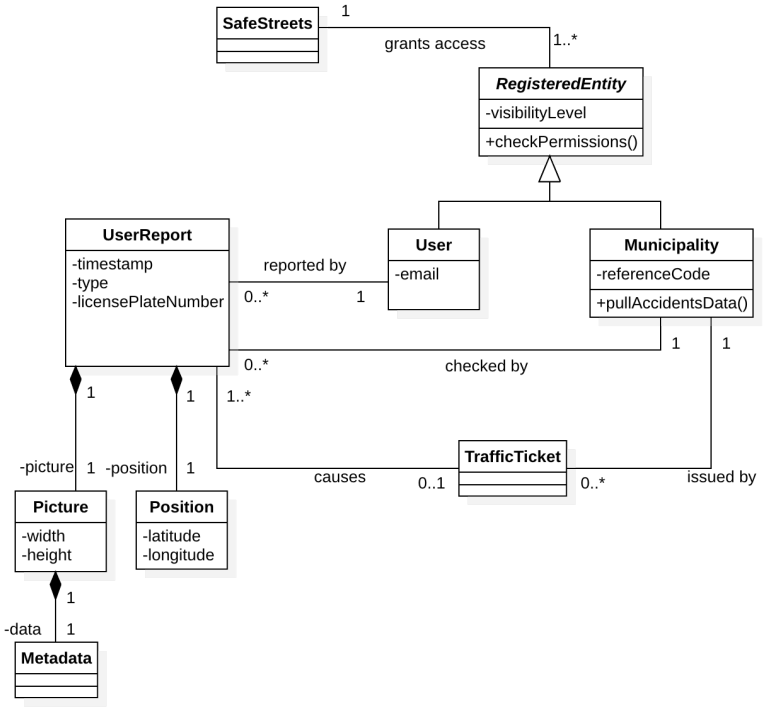
6. References References of documents that this project was developed upon.

Chapter 2

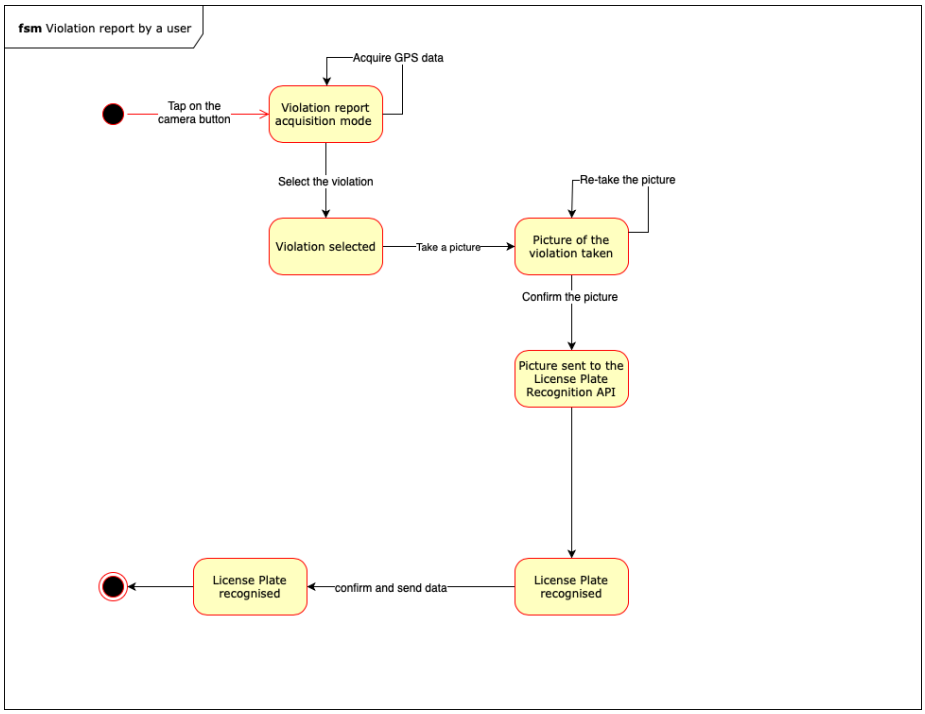
Overall Description

2.1 Product Perspective

2.1.1 SafeStreets



SafeStreets class diagram



SafeStreets state chart referred to *User reports* by *Users*

Shared Phenomena

- *User* reports a *Traffic violation*.
- *Municipality* checks a *Traffic violation*.
- A *Traffic violation* causes a *Traffic ticket* issued by the *Municipality*.
- *User* and *Municipality* have different permission levels to access the S2B data.

2.2 Product Functions

2.2.1 User registration

SafeStreets will allow individuals to register. These will register by entering all the required information (see R_1- R_2). When registering to SafeStreets, they will first declare to have read the *Privacy statement* and secondly, they will have to accept the *Terms and conditions*, which specifically include their consent to the acquisition and processing of their data.
The *User* registration process will be carried out on the SafeStreets application (see section 3.1.1).

2.2.2 User report acquisition

SafeStreets will collect *User report*. It will allow *User* to take a picture of the *Traffic violation* and to select the *Type of violation*. Once *License plate recognition* has recognized the license plate, SafeStreets will complete and acquire the *User report* and send it to the *Municipality* after *User* confirmation.

2.2.3 User contribution visualization

SafeStreets will show to the *User* his/her contribution in dedicated section where visualize all the previous *User report* and the *Ticket feedback* of each of these, received by the *Municipality*, as well.

2.2.4 Public statistics visualization

SafeStreets will show *Public statistics* required by the *User*. *User* will be able to set the parameters selecting a desired "period of time", a "*Type of violation*" and a "specific zone".

2.2.5 Municipality access

SafeStreets will allow *Municipality* to connect to its dedicated website.

2.2.6 User report evaluation

Municipality will visualize the list of *User report* and generate or not a *Traffic ticket*. If *Municipality* discard a *User report*, it will select a reason among those provided.

2.2.7 Detailed statistics visualization

SafeStreets will show *Detailed Statistics* required by the *Municipality*. *Municipality* will be able to set the parameters selecting the "period of time", a *"Type of violation"*, a "specific zone" and also inserting a "specific license plate".

2.2.8 Possible interventions visualization

SafeStreets will cross *Municipality Accidents* data with its own Database data. Based on the results, it could eventually suggest *Possible interventions* to *Municipality*.

2.2.9 License plate recognition data acquisition

The service will be able to receive the picture taken by *User* and sent through SafeStreets. Once the license plate has been recognized, it is immediately sent back, as a text.

2.2.10 Request Accidents data

SafeStreets, after requesting access to *Municipality Accidents* Database, will be able to cross *Municipality* data with its own data.

2.2.11 Request User report data

User, after requesting access to SafeStreets database for his/her own *User report*, will be able to visualize all the details about his/her previous *User report*.

2.3 User Characteristics

Users: People having a smartphone with Internet connection and willing to report a *Traffic violation* which has occurred.

Municipality: Authorities having the possibility to receive *Traffic violation* reports in order to generate *Traffic tickets*. They can collect reports without being physically present and see all the details about a specific *User report*. This also allow them to motivate any possible *User report* discarded. The *Municipality* visualizes, then, all the *Possible intervention* suggested by SafeStreets (e.g. add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking . . .)

2.4 Assumptions, Dependencies and Constraints

2.4.1 Domain Assumptions

D_1 Data inserted by the *User* at sign up correspond to their real data;

D_2 *User position* collected at a certain instant corresponds to the acual position of the *User* at that exact moment;

D_3 The maps in use accurately represent the world;

D_4 *Users* own a working smartphone which has access to Internet connection;

D_5 *Users* own a working smartphone which has a working GPS antenna;

D_6 SafeStreets and *License plate recognition service* are always online;

D_7 *License plate recognition service* always provides a text representing a correct recognition of the license plate in the *User picture*;

D_8 *Municipality* grants access to its own *Accidents* Database;

D_9 *Municipality* has institutional credentials (*Reference code* and password) which uses to access the system.

2.4.2 Dependencies

- Availability, performance and reliability of services depend respectively on the availability, performance and reliability of SafeStreets.
  SafeStreets, in order to guarantee that the data which constitutes the *User report* are not corrupted, ensures the chain of custody of information.
- Effectiveness of SafeStreets depends on *License plate recognition* response time.

2.4.3 Constraints

- Smartphones must always have a working Internet connection;
- *User* willing to register to SafeStreets must accept *Terms and conditions*;
- User registered must accept the *Privacy policy*;
- SafeStreets must comply with the GDPR regulations for the protection of *Users'* personal data;
- Smartphones must have GPS activated;
- Smartphones must have enough free space for downloading and installing the SafeStreets application;
- SafeStreets must wait for *License plate recognition* response in order to allow *User* to complete the *User report*;
- *Municipality* must have its *Reference code* and password, granted by SafeStreets, to access its *Municipality* area;
- All the collected *User reports* on the *Municipality* area must not be accessible by unauthorized people;
- *Municipality* must grant *Accidents* Database access to SafeStreets;
- *Municipality* must have a modern browser installed on a device connected to the Internet, in order to access the *Municipality* area on its dedicated website.

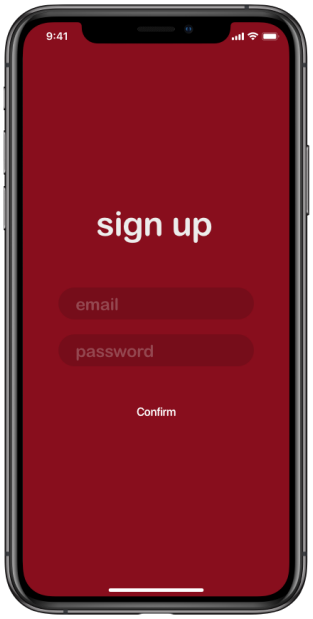
Chapter 3

Specific Requirements

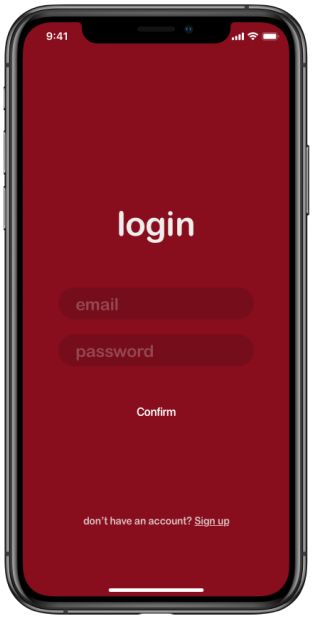
3.1 External Interface Requirements

3.1.1 User Interfaces

SafeStreets Welcome Page



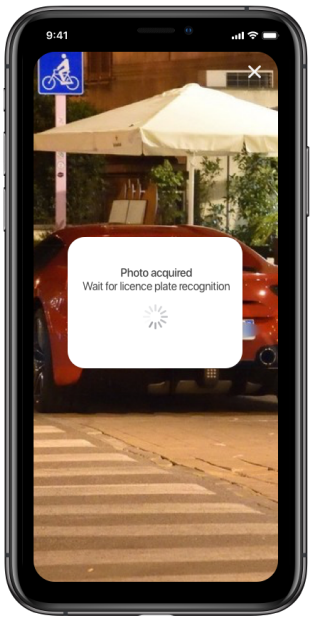
SafeStreets Sign up View



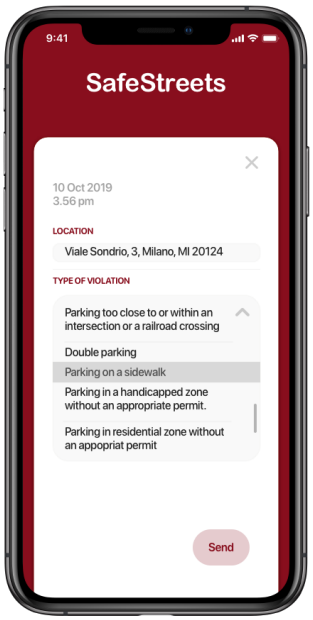
SafeStreets Login View

SafeStreets Camera View

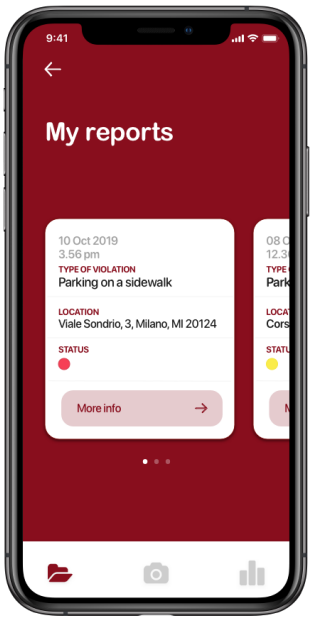
SafeStreets *Traffic violation* photo



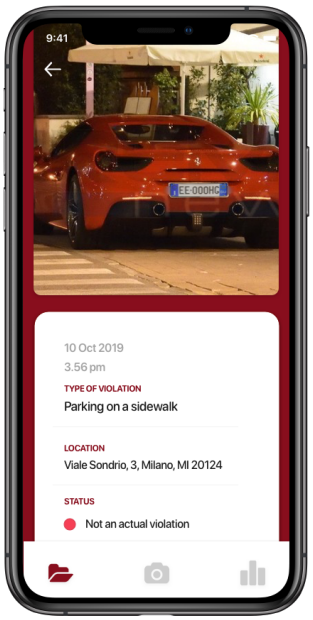
License plate recognition



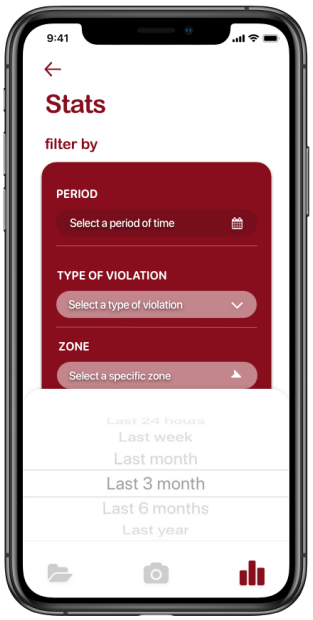
SafeStreets *User report* View



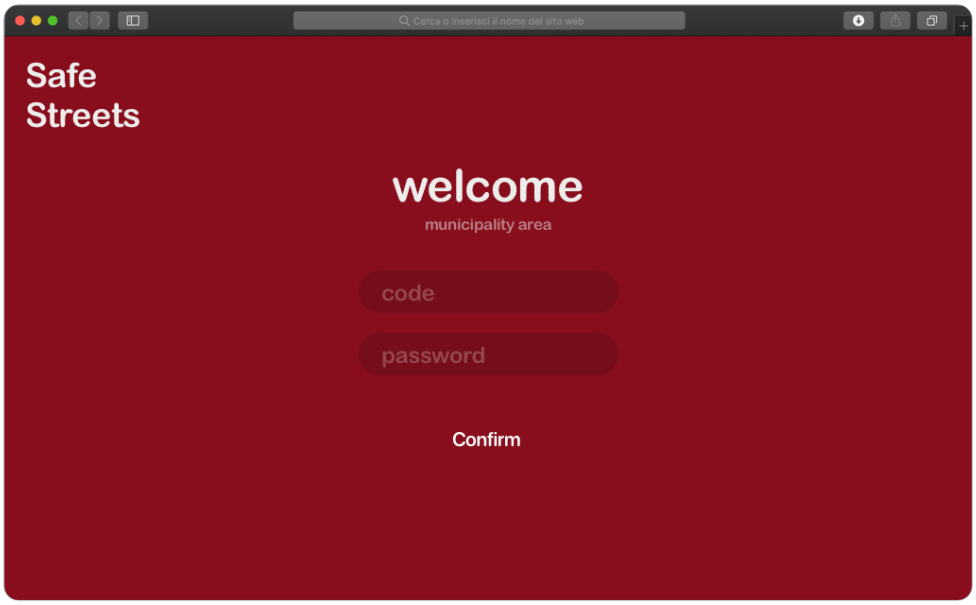
SafeStreets "My reports" View



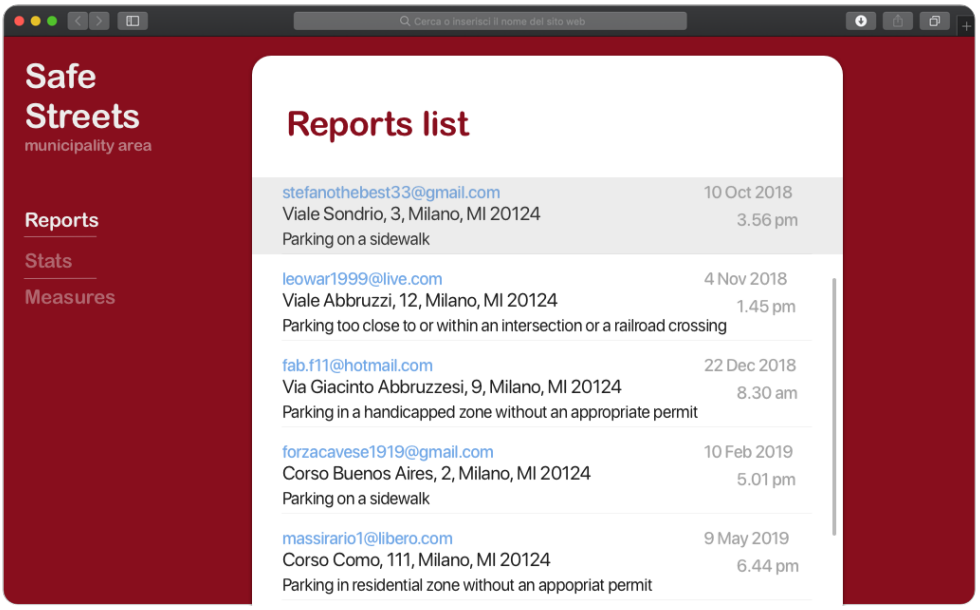
SafeStreets detailed *User report* View



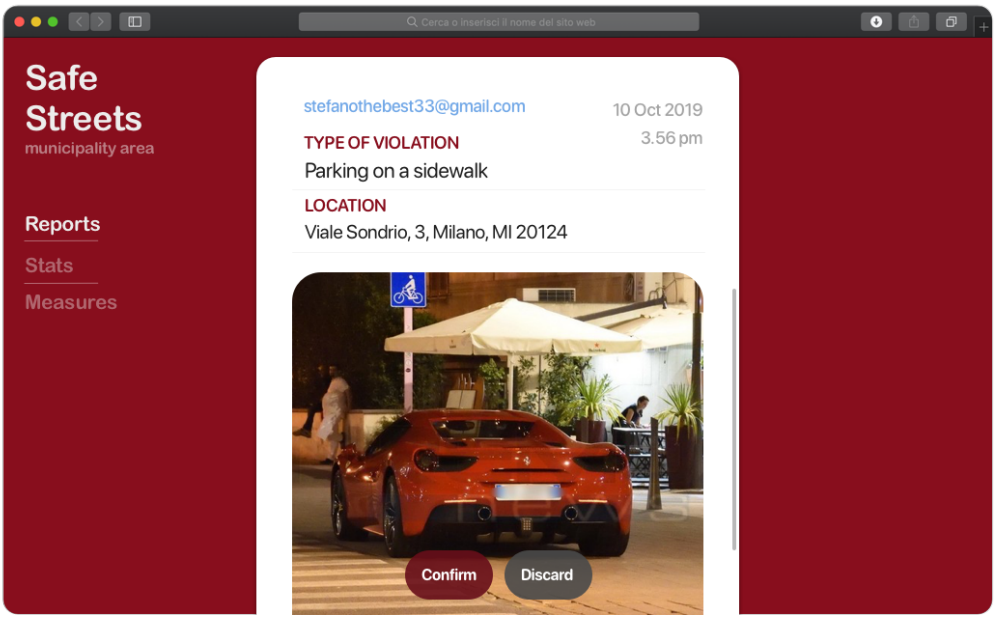
SafeStreets *Public Statistics* View



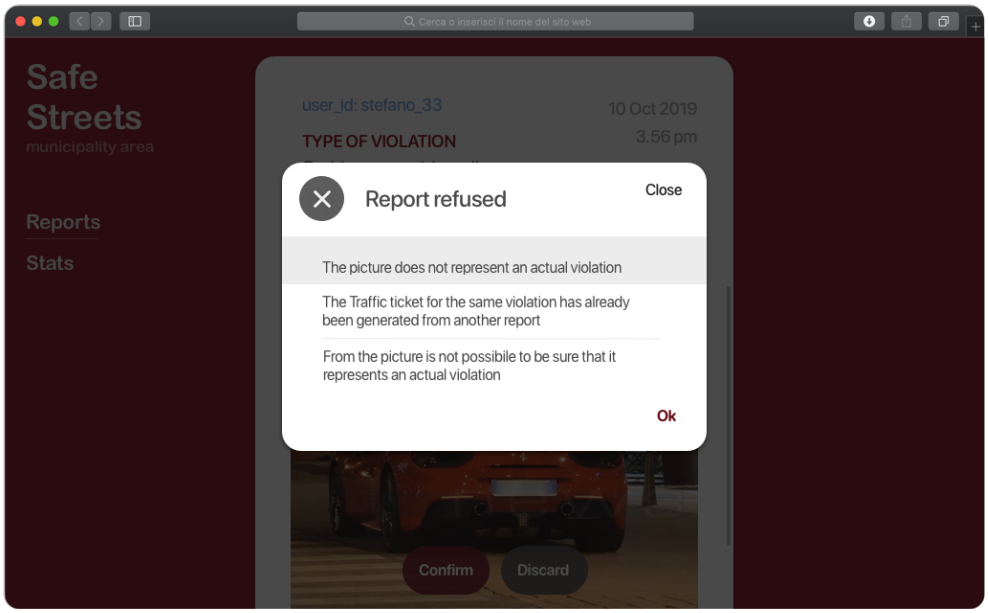
SafeStreets Welcome Page, Municipality area



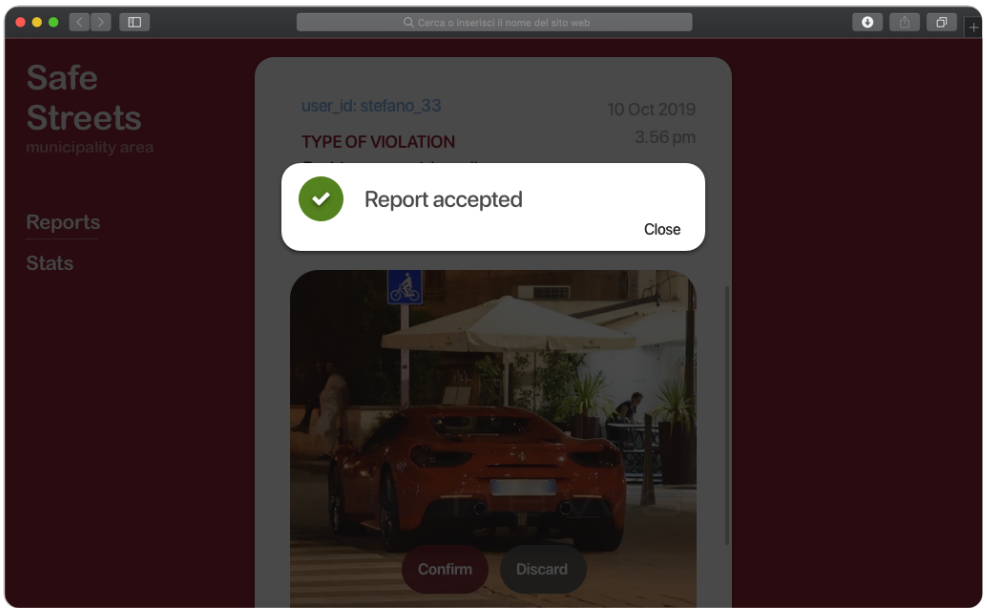
SafeStreets *User reports* list



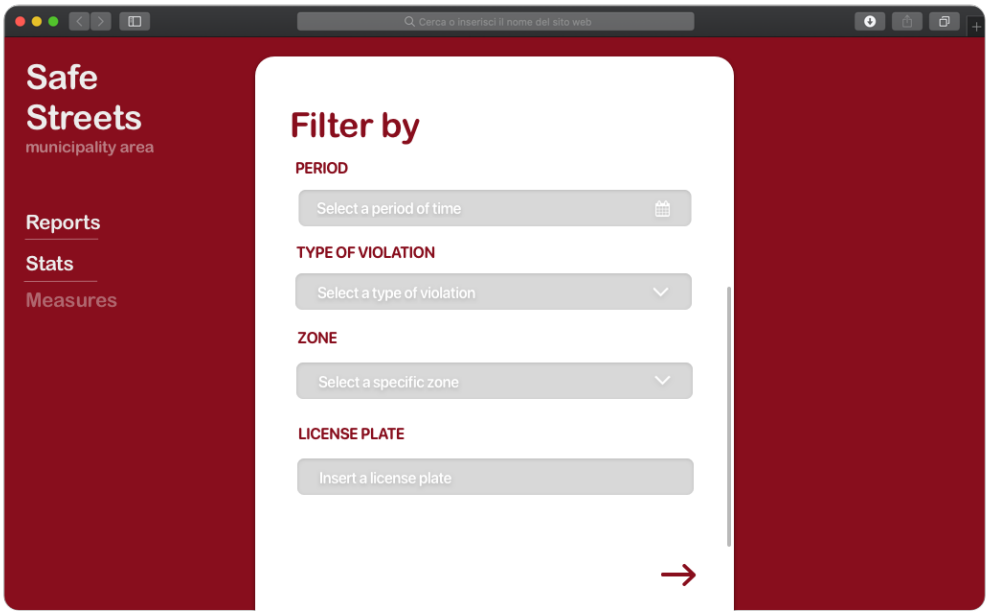
SafeStreets detailed *User report* Page



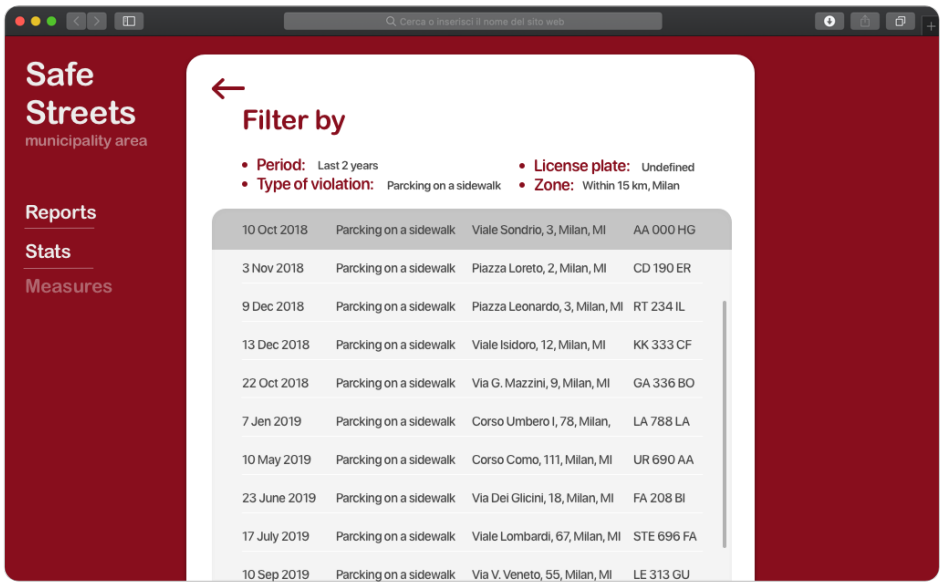
SafeStreets *User reports* refused



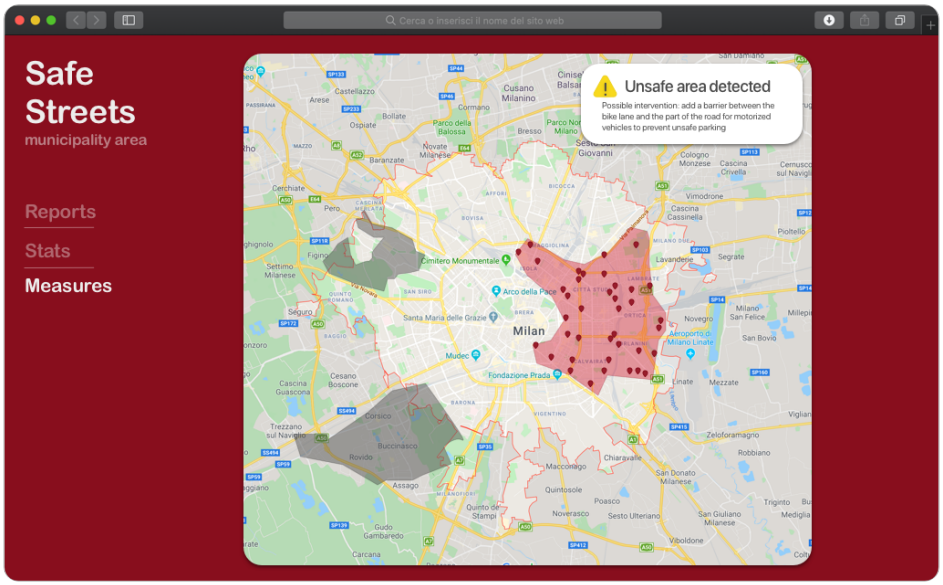
SafeStreets *User report* accepted



SafeStreets "filter by" Page



SafeStreets *Detailed Statistics* Page



SafeStreets *Possible interventions* suggestions

3.1.2 Hardware Interfaces

- Web applications must be accessible by *Municipality* using a computer with characteristics specified in Section 3.4.2.
- Smartphone on which the app will work must provide to the app an Internet connection used to send *User reports* to SafeStreets servers and must have a GPS antenna built in.

3.1.3 Software Interfaces

- Web applications for the *Municipality* must be compatible with the most popular browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari;
- Mobile apps for standard users must be available for both iOS and Android devices;
- Application backend stores collected data in a relational DBMS;
- Web applications show data by accessing the relational DBMS;
- SafeStreets must interface with the smartphone's GPS via the operating system API in order to localize the *User position* reporting a *Traffic violation*;
- Web applications have to interface with SafeStreets in order to receive data about *User reports* and eventually *Possible suggestions*;
- Web applications have to interface also with Maps in order to show the unsafe area detected;
- SafeStreets has to interface with *License plate recognition service*.

3.1.4 Communication Interfaces

The system to be will only make use of the usual communication protocols (TCP/IP) to guarantee the connection between the *User*'s Smartphones and the backend systems.

3.2 Functional Requirements

3.2.1 Scenarios

S_1 Michael is an individual who would like to help the community and the authorities reporting parking violations occurring in his city. He downloads the SafeStreets application on his smartphone and, once opened it, he is asked to insert an email (which will correspond to his *User identifier*) and a password. After inserting them, he accepts the *Terms and conditions* and taps on the arrow below. Later, he tries to login using his email and password; the system accepts his credentials and Michael is now logged in;

S_2 A *Municipality* officier opens SafeStreets web app. He is asked to insert *Reference code* and password in order to access the system, so he inserts his *Municipality* institutional credentials and he is now logged in and can use SafeStreets services;

S_3 Michael, a SafeStreets *User*, is walking down a street and, when he's about to cross the road, he realizes he can't access the pedestrian crossing from the walk side, because of a car parked there. So he opens the app and presses the "Camera" button in the tab bar of the main view, frames the car, making sure that the violation and the license plate are clearly visible, and takes the picture. After pressing the "Confirm" button he is asked to select the "*Type of violation*", choosing from a drop-down list of possible violations, and presses again on the "Send" button, in order to send his *User report*;

S_4 Michael, from scenario S_3, opens the app and taps on "MyReports" in the tab bar of the main view, to see if the last *User report* he has sent has already been evaluated, eventually becoming a *Traffic ticket*, but he sees that the *Ticket feedback* color is red, which means that a *Traffic ticket* has not been generated from his *User report*. To know more, he presses on the "More info" button and understands that the reason why it was not generated is that it was not an actual violation;

S_5 Scarlett, a SafeStreets *User*, wants to visualize *Public statistics* of her neighborhood of the last month, in order to figure out whether her neighbours have been active in reporting *Traffic violation* through SafeStreets or not, and eventually pointing it out during the monthly neighborhood meeting. For that reason, she presses the "*Public statistics*" button and selects an area, a period and a *Type of violation* from the corresponding drop-down lists and consults the statistics she is interested in;

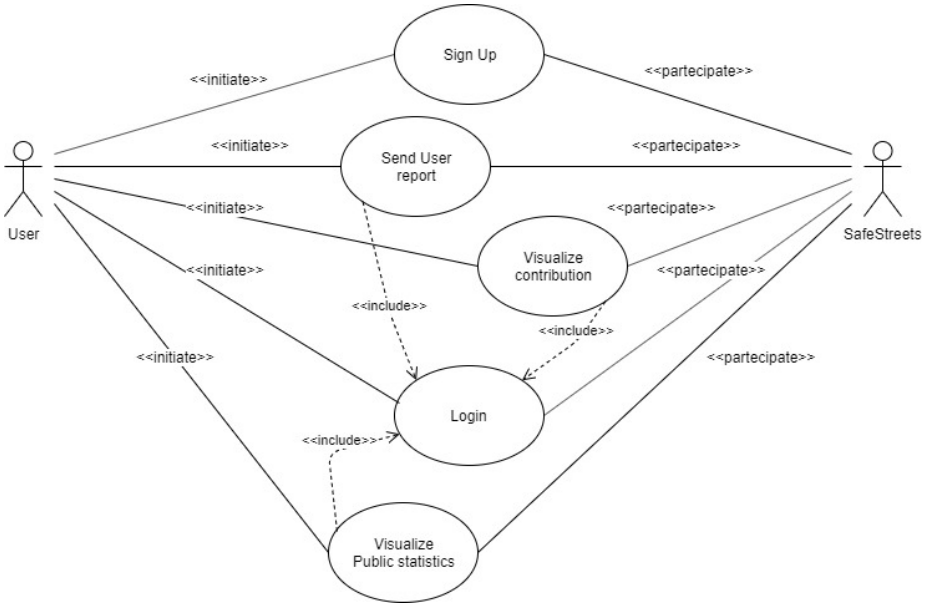
S_6 *Municipality* receives the complaint for a stolen car, so, in order to increase the possibilities to find some useful clues, looks for *User reports* received in the last 24 hours and involving the stolen car license plate in the *Detailed statistics*;

S_7 SafeStreets retrieves previous *Accidents* data from *Municipality* Database and, after crossing them with *User report* data in its own Database, identifies the area where the sum of the number of *Accidents* and the number of *User report*, for a particular *Type of violation*, is the highest, in order to identify and suggest a *Possible intervention* to the *Municipality*;

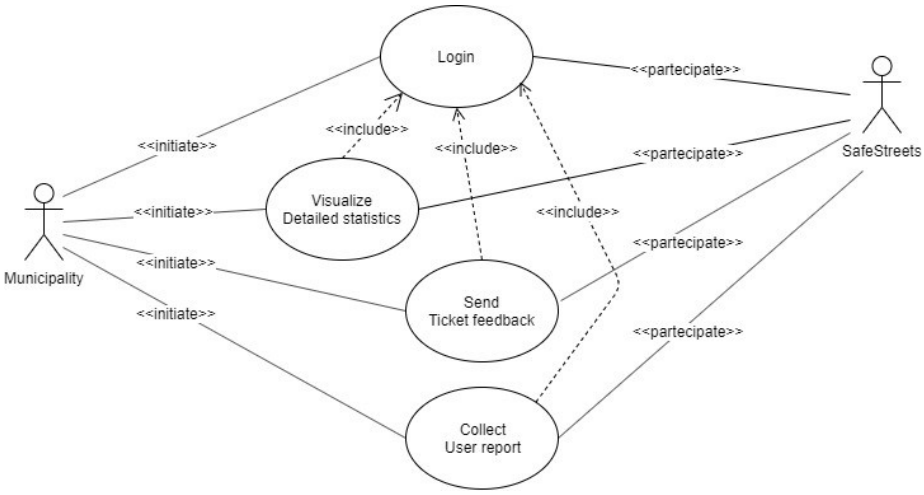
S_8 *Municipality* wants to visualize *Ticket feedback* statistics in the *Detailed statistics* to point out the areas where the service has had the highest incidence (i.e. the areas where the highest number of *User reports* have actually become *Traffic tickets*), in order to intensify the presence of Local Police agents in those areas;

S_9 *Municipality* wants to visualize who have been the most egregious offenders in a certain area of the city, so looks for them in the *Detailed statistics* focusing in particular on the *Ticket feedback* statistics.

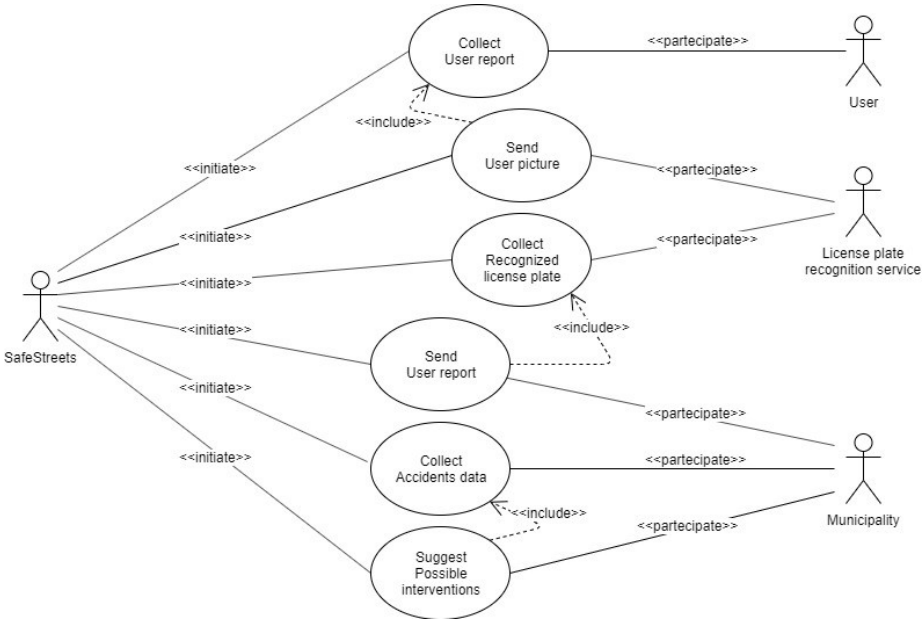
3.2.2 Use Case Diagrams



User Use Case Diagram



Municipality Use Case Diagram



SafeStreets Use Case Diagram

i st

3.2.3 Use Case Analysis

Name	**User Sign Up**
Actors	*User*, SafeStreets
Entry Conditions	*User* successfully installed SafeStreets application on the smartphone
Events Flow	1. *User* taps on "Sign Up" button 2. *User* fills the "email" and "password" fields 3. *User* checks the "Accept terms and conditions" checkbox 4. *User* taps on "Confirm" button 5. SafeStreets saves *User* information
Exit Condition	*User* successfully registered on SafeStreets
Exceptions	1. Inserted email already registered for another *User* 2. Inserted password is not valid 3. Not all required fields are filled in 4. "Accept terms and conditions" checkbox is not checked 5. *User* is already signed up *User* is invited to try again signing up, reporting which error they have committed.

U_1

Name	**User Login**
Actors	*User*, SafeStreets
Entry Conditions	*User* successfully installed SafeStreets application on the smartphone and registered on SafeStreets
Events Flow	1. *User* enters "email" 2. *User* enters "password" 3. *User* presses the "Confirm" button 4. SafeStreets checks *User* credentials
Exit Condition	*User* is successfully logged in
Exceptions	1. Inserted "email" is not valid 2. Inserted "password" is not valid *User* is asked to try again to log into the system.

U_2

Name	***Municipality* Login**
Actors	*Municipality*, SafeStreets
Entry Conditions	*Municipality* is connected to the *Municipality* dedicated web site
Events Flow	1. *Municipality* enters "*Reference code*" 2. *Municipality* enters "password" 3. *Municipality* presses the "Confirm" button 4. SafeStreets checks *Municipality* institutional credentials
Exit Condition	*Municipality* is successfully logged into the system
Exceptions	1. Inserted "*Reference code*" is not valid 2. Inserted "password" is not valid *Municipality* is asked to try again to log into the system.

U_3

Name	**SafeStreets sends *User report* to *Municipality***
Actors	*User*, SafeStreets, *License plate recognition service*, *Municipality*
Entry Conditions	*User* successfully logged into SafeStreets and installed SafeStreets application on the smartphone
Events Flow	1. *User* takes a picture of the *Traffic violation* 2. *User* presses the "Confirm" button 3. SafeStreets receives *User picture* and forward it to the *License plate recognition service* 4. *License plate recognition service* sends back to SafeStreets a text representing the *Recognized license plate* 5. *User* selects a *Type of violation* and presses the "Send" button 6. SafeStreets stores the newly generated *User report* in its own Database 7. SafeStreets sends *Municipality* the newly generated *User report*
Exit Condition	SafeStreets correctly collected *User report*
Exceptions	1. No *Type of violation* has been selected *User* is asked to select a choice for the missing parameter.

U_4

Name	***Municipality* answers to a *User report***
Actors	SafeStreets, *Municipality*
Entry Conditions	*Municipality* successfully logged into SafeStreets
Events Flow	1. *Municipality* presses "Reports" button 2. *Municipality* selects a *User report* to answer 3. *Municipality* taps "Confirm" or "Discard" button (in the second case selects a reason as well) to generate the *Ticket feedback* 4. SafeStreets receives the *Ticket feedback*
Exit Condition	*Municipality* correctly answered to the *User report*
Exceptions	No Exceptions.

U_5

Name	***User* visualizes *Public statistics***
Actors	*User*, SafeStreets
Entry Conditions	*User* successfully logged into SafeStreets
Events Flow	1. *User* presses "Public statistics" button 2. *User* selects a "period of time", a "*Type of violation*" and a "specific zone" 3. SafeStreets shows the *User* the required statistics
Exit Condition	*User* correctly visualizes the required *Public statistics*
Exceptions	1. No value was chosen for the parameters to be set *User* is asked to select at least one value between the filtering parameters.

U_6

Name	***Municipality* visualizes *Detailed statistics***
Actors	*Municipality*, SafeStreets
Entry Conditions	*Municipality* successfully logged into SafeStreets
Events Flow	1. *Municipality* presses the "Stats" button 2. *Municipality* selects a "period of time", a "*Type of violation*" and a "specific zone", then, optionally, a "particular license plate" as well 3. SafeStreets shows *Municipality* the required statistics
Exit Condition	*Municipality* correctly visualizes the required *Detailed statistics*
Exceptions	1. No value was chosen for the parameters to be set *Municipality* is asked to select at least one value between the filtering parameters.

U_7

Name	***SafeStreets* suggests *Possible interventions* to *Municipality***
Actors	*Municipality*, SafeStreets
Entry Conditions	*Municipality* grants SafeStreets access to its own *Accidents* Database
Events Flow	1. A certain number of *User reports* for a particular zone of the city is reached 2. SafeStreets identifies the precise area in which a *Possible intervention* could be carried out 3. SafeStreets sends to *Municipality* the *Possible intervention*
Exit Condition	*Municipality* correctly receives the suggested *Possible intervention*
Exceptions	No Exceptions.

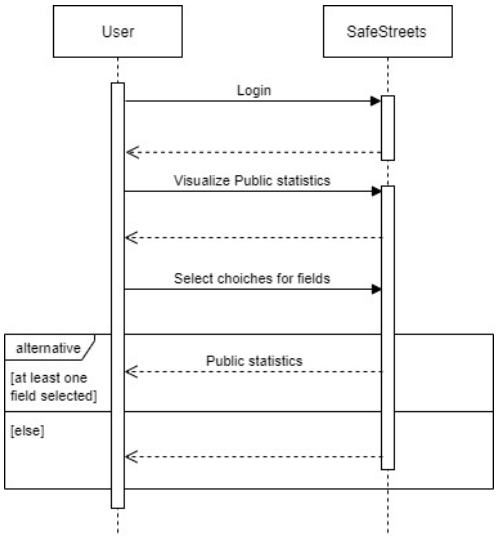
U_8

Name	***User* visualizes his/her contribution to SafeStreets**
Actors	*User*, SafeStreets
Entry Conditions	*User* successfully logged into SafeStreets
Events Flow	1. *User* taps "My reports" button 2. *User* chooses one of his/her previous *User report* to check 3. *User* visualizes all the details about that particular *User report*
Exit Condition	*User* correctly visualizes the details about the chosen *User report*
Exceptions	1. *User* has not reported any *Traffic violation* yet *User* is reminded he/she has not reported any *Traffic violation* yet.

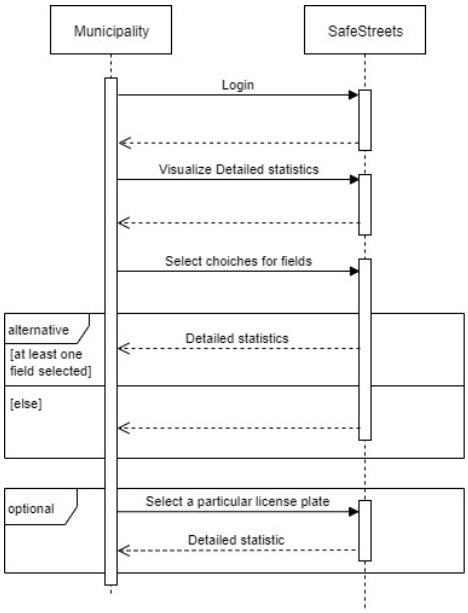
U_9

3.2.4 Sequence Diagrams

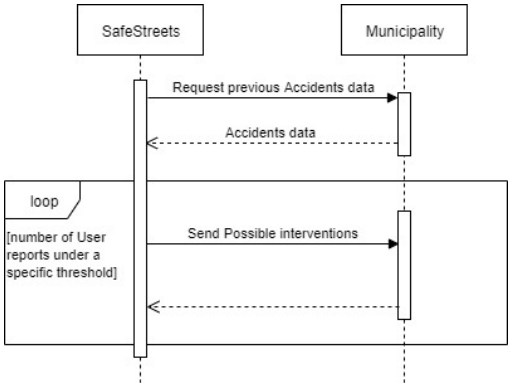
User report Sequence Diagram



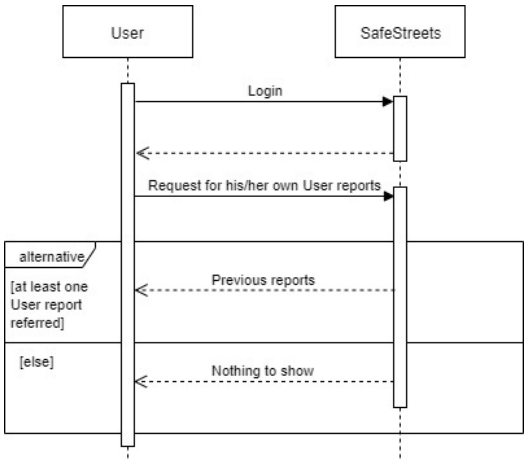
Public statistics visualization Sequence Diagram



Detailed statistics visualization Sequence Diagram



Possible interventions Sequence Diagram



User contribution Sequence Diagram

3.2.5 Requirements

R_1 At sign up, *User* must provide email and password;

R_2 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*;

R_3 Identify a *User* by his/her *User identifier*;

R_4 Receive *User picture*;

R_5 Receive *User* choice for the type of *Traffic violation*;

R_6 Receive *User position*;

R_7 Send *User picture* to the *License plate recognition service*;

R_8 Receive *Recognized license plate* from the *License plate recognition service*;

R_9 Create *User report*;

R_{10} Store *User report* in SafeStreets Database;

R_{11} Send *User report* to *Municipality*;

R_{12} Store *Ticket feedback* in SafeStreets Database;

R_{13} Query SafeStreets Database for a *User* by his/her *User identifier*;

R_{14} Retrieve specific *User reports* by querying SafeStreets Database;

R_{15} Validate the data which constitutes the *User report*;

R_{16} Receive *Municipality* data access request;

R_{17} Validate *Municipality* data access request;

R_{18} Allow *Municipality* to set specific constraints to define *Detailed statistics*;

R_{19} Send *Detailed statistics* to the requesting *Municipality*;

R_{20} Allow *Users* to request all their *User report* and the related *Ticket feedback* stored in SafeStreets Database at any time;

R_{21} Send to a specific *User* his/her *User report* stored;

R_{22} Send to a specific *User* the *Ticket feedback* related to his/her *User report*;

R_{23} Allow *User* to choose among filters which define a group of *User reports*;

R_{24} Send *Public statistics* to the requesting *User*;

R_{25} Send *Accidents* Database access request to *Municipality*;

R_{26} Receive *Accidents* Database access confirmation from *Municipality*;

R_{27} Request *Accidents* data to *Municipality*;

R_{28} Receive *Accidents* data from *Municipality*;

R_{29} Generate *Possible interventions* crossing *Municipality Accidents* data with SafeStreets Database data;

R_{30} Send generated *Possible interventions* to *Municipality*;

R_{31} Identify *Municipality* by its *Reference code*;

R_{32} Receive *Ticket feedback* from *Municipality*.

3.2.6 Satisfying Goals

G_1 Collect *User report* in SafeStreets Database;

- R_1 At sign up, *User* must provide: first name, last name, email and password;
- R_2 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*;
- R_3 Identify a *User* by his/her *User identifier*;
- R_4 Receive *User picture*;
- R_5 Receive *User* choice for the type of *Traffic violation*;
- R_6 Receive *User position*;
- R_7 Send *User picture* to the *License plate recognition service*;
- R_8 Receive *Recognized license plate* from the *License plate recognition service*;

R_9 Create *User report*;

R_{10} Store *User report* in SafeStreets Database;

D_1 Data inserted by the *User* at sign up corresponds to their real data;

D_2 *User position* collected at a certain instant corresponds to the acual position of the *User* at that exact moment;

D_3 The maps in use accurately represent the world;

D_4 *Users* own a working smartphone which has access to Internet connection;

D_5 *Users* own a working smartphone which has a working GPS antenna;

D_6 SafeStreets and *License plate recognition service* is always online;

D_7 *License plate recognition service* always provides a text representing a correct recognition of the license plate in the *User picture*;

G_2 Send *User report* to *Municipality* as soon as it is generated;

R_{11} Send *User report* to *Municipality*;

R_{15} Validate the data which constitutes the *User report*;

R_{31} Identify *Municipality* by its *Reference code*;

D_6 SafeStreets and *License plate recognition service* are always online;

G_3 Send *User picture* to the *Licence plate recognition service* as soon as it is received;

R_4 Receive *User picture*;

R_7 Send *User picture* to the *License plate recognition service*;

D_6 SafeStreets and *License plate recognition service* are always online;

G_4 Allow *Municipality* to visualize *Detailed statistics*;

R_{10} Store *User report* in SafeStreets Database;

R_{13} Query SafeStreets Database for a *User* by his/her *User identifier*;

R_{14} Retrieve specific *User reports* by querying SafeStreets Database;

R_{16} Receive *Municipality* data access request;

R_{17} Validate *Municipality* data access request.

R_{18} Allow *Municipality* to set specific constraints to define *Detailed statistics*;

R_{19} Send *Detailed statistics* to the requesting *Municipality*;

R_{31} Identify *Municipality* by its *Reference code*;

D_9 *Municipality* has institutional credentials (*Reference code* and password) which uses to access the system;

G_5 Allow *Users* to visualize *Public statistics*;

R_{10} Store *User report* in SafeStreets Database;

R_{23} Allow *User* to choose among filters which define *Public statistics*;

R_{24} Send *Public statistics* to the requesting *User*;

G_6 Suggest *Possible interventions* to *Municipality*;

R_{10} Store *User report* in SafeStreets Database;

R_{13} Query SafeStreets Database for a *User* by his/her *User identifier*;

R_{25} Send *Accidents* Database access request to *Municipality*;

R_{26} Receive *Accidents* Database access confirmation from *Municipality*;

R_{27} Request *Accidents* data to *Municipality*;

R_{28} Receive *Accidents* data from *Municipality*;

R_{29} Generate *Possible interventions* crossing *Municipality Accidents* data with SafeStreets Database data.

R_{30} Send generated *Possible interventions* to *Municipality*;

R_{31} Identify *Municipality* by its *Reference code*;

D_8 *Municipality* grants access to its own *Accidents* Database;

G_7 Collect information about *Ticket feedback*;

R_{12} Store *Ticket feedback* in SafeStreets Database;

R_{31} Identify *Municipality* by its *Reference code*;

R_{32} Receive *Ticket feedback* from *Municipality*;

G_8 Allow *Users* to visualize their contribution to SafeStreets;

R_3 Identify a *User* by his/her *User identifier*;

R_{10} Store *User report* in SafeStreets Database;

R_{12} Store *Ticket feedback* in SafeStreets Database;

R_{13} Query SafeStreets Database for a *User* by his/her *User identifier*;

R_{20} Allow *Users* to request all their *User report* and the related *Ticket feedback* stored in SafeStreets Database at any time;

R_{21} Send to a specific *User* his/her *User report* stored;

R_{22} Send to a specific *User* the *Ticket feedback* related to his/her *User report*;

R_{32} Receive *Ticket feedback* from *Municipality*.

3.3 Performance Requirements

The system to be does not have specific requirements on performances such as response time or algorithm complexity.

3.4 Design Constraints

3.4.1 Standard compliance

- GDPR 2016/679 (EU) - General Data Protection Regulation
- ISO/IEC/IEEE 29148:2011 - Standard on requirement engineering

3.4.2 Hardware limitations

In order to use the service, *user*'s hardware should comply to these minimum requirements:

Mobile application

- Smartphone:
  - iOS or Android operative system;
  - UMTS/4G Internet connection with a minimum speed of 1Mb/s;
  - GPS antenna;
  - 300 Mb available memory;
  - Phone camera of at least 720p resolution;
  - Dual-core processor;
  - 1 Gb RAM.

Web application

- Computer:
  - Internet connection with a minimum speed of 1Mb/s;
  - Browser application;
  - 720p monitor resolution.

3.4.3 Any other constraint

SafeStreets has no other constraints.

3.5 Design Constraints

3.5.1 Reliability

The reliability of the system mainly depends on that of the Internet connection when it comes to sending data to SafeStreets and to the *License plate recognition service*. Fault detection time must be kept minimum so as to promptly identify a fault and repair it as soon as possible.

3.5.2 Availability

The system must offer the maximum availability, granting its service every day at any time. The lack of service must be minimal.

3.5.3 Security

A high security level is a mandatory constraint for a system like this. This constraint was implicitly filled choosing for a hardened database technology to face with the data storing management problem. Furthermore, talking about data transmission mechanism, every communication between mobile or web interface and the server will be ciphered with AES protocols, and secured with HTTPS/SSL protocols.
The login of *Users* and especially must be very safe (using state of the art login techniques is recommended) to avoid unauthorized individuals to access private information of *Users*. Moreover, the means of communication must be encrypted to save the confidentiality of information sent to SafeStreets.
In addition, integrity about every information, coming from the *User*, related to the *User report*, must be granted by SafeStreets before it is sent to the *Municipality*. Information must not be altered or corrupted.

3.5.4 Maintainability

A modular approach is one of the several key values which we had chosen to identify for the system.

3.5.5 Portability

Portability of *User* data from a device to another is possible by entering personal login data, also for devices with different operating systems. Personal data and *User reports* are stored in a database and they are downloaded when a new device is connected.

Chapter 4

Formal Analysis using Alloy

4.1 Alloy model

```
----- SIGNATURES

abstract sig RegisteredEntity {}

sig Individual { }

sig Email { }

sig User extends RegisteredEntity {
        individual: one Individual,
        email: one Email,
        pStatistic: UserReport -> PublicStatistic
}

sig Municipality extends RegisteredEntity {
        referenceCode: one Int,
        dStatistic: UserReport -> DetailedStatistic
} { referenceCode > 0 }

sig Metadata { }

sig Picture {
        width: one Int,
        height: one Int,
        data: one Metadata
} { width > 0 and height > 0}

sig Position {
        latitude: one Int,
        longitude: one Int
} { latitude > 0 and longitude > 0 }

// Fields describing the type of violation and
// the license plate number have been, for simplicity
// reasons, modeled as integers instead of strings
sig UserReport {
        user: one User,
        timestamp: one Int,
        typeOfViolation: one Int,
        licensePlateNumber: one Int,
        picture: one Picture,
        position: one Position
```

```
} { licensePlateNumber > 0 and timestamp > 0 and
        (typeOfViolation=0 or typeOfViolation=1 or typeOfViolation=2 or
            ↪ typeOfViolation=3) --for simplicity 4 types of violation
}

one sig SafeStreets {
        registeredUsers: set User,
        registeredMunicipalities: set Municipality,
        pStatistic: UserReport -> PublicStatistic,
        dStatistic: UserReport -> DetailedStatistic
}

abstract sig Filter {
        typeOfViolation: lone Int,
        timestamp: set Int,
        position: set Position
} { (#typeOfViolation>0 or #timestamp>0 or #position>0) }

sig PublicFilter extends Filter {}

sig DetailedFilter extends Filter {
        licensePlateNumber: lone Int,
} { licensePlateNumber > 0}

sig PublicStatistic {
        user: one User,
        pFilter: one PublicFilter
}

sig DetailedStatistic {
        municipality: one Municipality,
        dFilter: one DetailedFilter
}


----- FUNCTIONS

fun getReportsOfDetailedStatistic [ss:SafeStreets, ds: DetailedStatistic]:
    ↪ set UserReport {
        (ss.dStatistic).ds
}

fun getReportsOfPublicStatistic [ss: SafeStreets, ps: PublicStatistic]: set
    ↪ UserReport {
        (ss.pStatistic).ps
}

fun retrieveReportsToDetailedStatistic [ss: SafeStreets, ds:
    ↪ DetailedStatistic]: UserReport -> DetailedStatistic {
        ss.dStatistic :> ds
}

fun retrieveReportsToPublicStatistic [ss:SafeStreets, ps: PublicStatistic]:
    ↪ UserReport -> PublicStatistic {
        ss.pStatistic :> ps
}


----- FACTS

fact uniqueEmailForUsers {
        all e: Email | one u: User | u.email = e
}
fact uniqueRefCodeForMunicipalities {
        all m1, m2: Municipality | (m1 ≠ m2 iff m1.referenceCode ≠ m2.
            ↪ referenceCode)
```

```
}
fact allUsersBelongToSafeStreets {
        all u: User | u in SafeStreets.registeredUsers
}
fact allMunicipalitiesBelongToSafeStreets {
        all m: Municipality | m in SafeStreets.registeredMunicipalities
}
fact userIndividualIsUnique {
        all i: Individual | one u: User | u.individual = i
}
fact pictureBelongsToOneReport {
        all p: Picture | one r: UserReport | r.picture = p
}
fact positionBelongsToOneReport {
        all p: Position | one r: UserReport | r.position = p
}
fact metadataBelongsToOnePicture {
        all m: Metadata | one p: Picture | p.data = m
}

fact UserReportBelongsToOneUser {
        all r: UserReport | one u: User | u in r.user
}

fact publicFilterCorrespondsToOnePublicStatistic {
        all pf: PublicFilter | one stat: PublicStatistic | stat.pFilter = pf
}

fact publicStatisticUserIsUnique{
        all ps: PublicStatistic | one u: User | ps.user = u
}

fact detailedStatisticMunicipalityIsUnique{
        all ds: DetailedStatistic | one m: Municipality | ds.municipality = m
}

fact detailedFilterCorrespondsToOneDetailedStatistic {
        all df: DetailedFilter | one stat: DetailedStatistic | stat.dFilter =
            ↪  df
}

fact userReportInMunicipalityImpliesUserReportInSafeStreetsAndViceversa {
        all r: UserReport, ds: DetailedStatistic, ss: SafeStreets, m:
            ↪ Municipality |
        r -> ds in m.dStatistic iff r -> ds in ss.dStatistic
}
fact userReportInUserImpliesUserReportInSafeStreetsAndViceversa {
        all r: UserReport, ps: PublicStatistic, ss: SafeStreets, u: User |
            ↪
        r -> ps in u.pStatistic iff r -> ps in ss.pStatistic
}

fact detailedStatisticMadeOfReportsRespectingDetailedFilter {
        all ds: DetailedStatistic, ss: SafeStreets |
        ( ( (#(ds.dFilter.timestamp)>0 and ((getReportsOfDetailedStatistic[ss
            ↪ , ds]).timestamp in ds.dFilter.timestamp)) or
                (#(ds.dFilter.timestamp)=0) )
        and
        ( (#(ds.dFilter.typeOfViolation)>0 and ((
            ↪ getReportsOfDetailedStatistic[ss, ds]).typeOfViolation = ds.
            ↪ dFilter.typeOfViolation)) or
                (#(ds.dFilter.typeOfViolation)=0) )
        and
        ( (#(ds.dFilter.position)>0 and ((getReportsOfDetailedStatistic[ss,
            ↪ ds]).position in ds.dFilter.position)) or
                (#(ds.dFilter.position)=0) )
        and
        ( (#(ds.dFilter.licensePlateNumber)>0 and ((
```

```
            ↪ getReportsOfDetailedStatistic[ss, ds]).licensePlateNumber =
            ↪ ds.dFilter.licensePlateNumber)) or
                (#(ds.dFilter.licensePlateNumber)=0) ) )
}

fact publicStatisticMadeOfReportsRespectingPublicFilter {
        all ps: PublicStatistic, ss: SafeStreets |
        ( ( (#(ps.pFilter.timestamp)>0 and ((getReportsOfPublicStatistic[ss,
            ↪ ps]).timestamp in ps.pFilter.timestamp)) or
                (#(ps.pFilter.timestamp)=0) )
        and
        ( (#(ps.pFilter.typeOfViolation)>0 and ((getReportsOfPublicStatistic[
            ↪ ss, ps]).typeOfViolation = ps.pFilter.typeOfViolation)) or
                (#(ps.pFilter.typeOfViolation)=0) )
        and
        ( (#(ps.pFilter.position)>0 and ((getReportsOfPublicStatistic[ss, ps
            ↪ ]).position in ps.pFilter.position)) or
                (#(ps.pFilter.position)=0) ) )
}


----- PREDICATES

pred sendDetailedStatistic[f: DetailedFilter, m: Municipality, m':
    ↪ Municipality, ss: SafeStreets, ds: DetailedStatistic] {
        ds in UserReport.(ss.dStatistic)
        f in ds.dFilter
        m'.referenceCode = m.referenceCode
        m'.dStatistic = m.dStatistic + retrieveReportsToDetailedStatistic[ss,
            ↪  ds]
}

pred sendPublicStatistic[f: PublicFilter, u: User, u': User, ss: SafeStreets,
    ↪  ps: PublicStatistic] {
        ps in UserReport.(ss.pStatistic)
        f in ps.pFilter
        u'.individual = u.individual
        u'.email = u.email
        u'.pStatistic = u.pStatistic + retrieveReportsToPublicStatistic[ss,
            ↪ ps]
}

pred show{}


----- ASSERTIONS

assert sendDetailedStatisticOK {
        all m,m': Municipality, f: DetailedFilter, ss: SafeStreets, ds:
            ↪ DetailedStatistic |
        sendDetailedStatistic[f,m,m',ss,ds] implies
        (
                ds in UserReport.(m'.dStatistic) and
                getReportsOfDetailedStatistic[ss,ds] in m'.dStatistic.
                    ↪ DetailedStatistic
        )
}

assert sendPublicStatisticOK {
        all u,u': User, f: PublicFilter, ss: SafeStreets, ps: PublicStatistic
            ↪  |
        sendPublicStatistic[f,u,u',ss,ps] implies
        (
                ps in UserReport.(u'.pStatistic) and
                getReportsOfPublicStatistic[ss,ps] in u'.pStatistic.
                    ↪ PublicStatistic
```

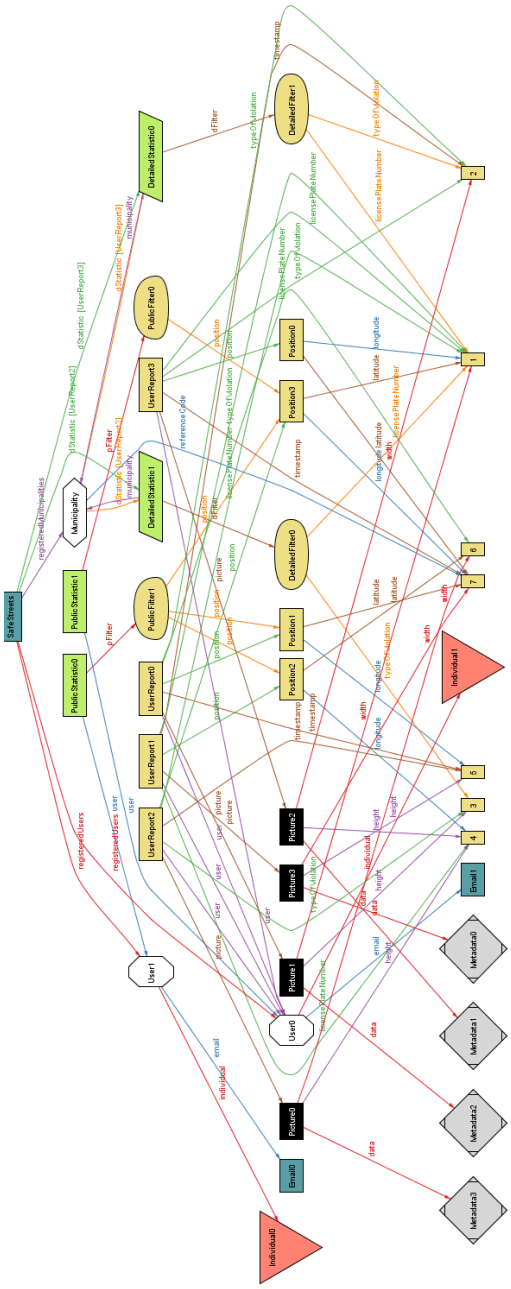
```
        )
}


----- CHECKS

check sendDetailedStatisticOK for 10

check sendPublicStatisticOK for 10

----- RUN

run show for 10 but exactly 2 User, exactly 1 Municipality, exactly 4
    ↪ UserReport, exactly 2 DetailedStatistic, exactly 2 PublicStatistic
```

4.1.1 Analysis results

```
Executing "Check sendDetailedStatisticOK for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  82344 vars. 5241 primary vars. 235603 clauses. 802ms.
  No counterexample found. Assertion may be valid. 161ms.

Executing "Check sendPublicStatisticOK for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  82444 vars. 5241 primary vars. 235951 clauses. 598ms.
  No counterexample found. Assertion may be valid. 115ms.

Executing "Run show for 10 but exactly 2 User, exactly 1 Municipality, exactly 4 UserReport, exactly 2 DetailedStatistic, exactly 2 PublicStatistic"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  25789 vars. 1815 primary vars. 77108 clauses. 125ms.
  Instance found. Predicate is consistent. 376ms.

3 commands were executed. The results are:
  #1: No counterexample found. sendDetailedStatisticOK may be valid.
  #2: No counterexample found. sendPublicStatisticOK may be valid.
  #3: Instance found. show is consistent.
```

Alloy Analyzer summary

4.2 Graph



Chapter 5

Effort Spent

Ferrara Fabiana Total hours of work: 50 h

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 4h RASD Review homework
- 4h Purpose, Scope, Goals
- 3h Assumptions, Constraints
- 2h Product perspective
- 2h Product functions
- 2h User characteristics
- 4h Goal revision, Domain Assumptions
- 10h Mockup
- 2h Design constraints
- 3h Specific Requirements
- 4h Functional Requirements, Performance Requirements
- 2h Alloy revision
- 6h General revision

Formicola Stefano Total hours of work: 48 h

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 4h RASD Review homework
- 4h Purpose, Scope, Goals

- 3h Assumptions, Constraints
- 4h Product Perspective, Class Diagram and State Chart Diagram
- 1.5h Product Functions revision
- 0.5h Mockup revision
- 2h Specific Requirements
- 2h Functional Requirements revision
- 2h Requirements and Satisfying goals revision
- 5h Introduction to the Alloy specification language and its tool
- 8h Alloy modelling of the S2B
- 3h Alloy revision
- 6h General revision

Guerra Leonardo Total hours of work:

- 2h Reading of Project Delivery Document, General LaTeX setting.
- 3h RASD Review homework

Chapter 6

References

1 E. Di Nitto. *Lecture Slides*. Politecnico di Milano.

2 E. Di Nitto. *Mandatory Project Assignment AY 2019-2020*. Politecnico di Milano.

3 ISO/IEC/IEEE 29148:2011. *Standard on requirement engineering*. https://standards.ieee.org/standard/29148-2011.html.