POLITECNICO
MILANO 1863

Dipartimento di Elettronica, Informazione e Bioingegneria

Requirement Analysis and Specification Document (RASD)

SafeStreets

- v1.0 -

Authors:

Quacquarelli Sebastiano	945071
Ricchiuti Simone	945613
Sala Nicolò	945898

November 10th , 2019

Contents

1 Introduction

1.1 Purpose

Nowadays road traffic has quickly increased. Consequently, it is more and more common to face traffic violations.
SafeStreets is a system that wants to help solving this critical problem: it allows its users to notify possible traffic violations to authorities, which can evaluate them and eventually generate traffic tickets.
Users are encouraged to report a violation because there is not a direct connection between the reporter and authorities: every report is anonymous.
The only thing a user needs is an electronic device with an internet connection and the SafeStreets app installed. He is only asked to take a picture and select the kind of violation he wants to notify.
When these two simple operations are correctly done, SafeStreets handles the report, publishing it on its platform and forwarding it to authorities, if they are available to collaborate in the reported area.
SafeStreets also provides a feature that allows to see statistics about violations. In this way, for example, it is possible to know the most dangerous area in a city.
SafeStreets can also collect information about incidents from authorities and combine them with the ones about traffic violations to let users do data-mining and to show them more accurate statistics. This advanced feature is only available in areas where authorities collaborate with SafeStreets.
This last feature is especially productive for authorities, because SafeStreets can combine the retrieved information to suggest them some possible solutions for particularly unsafe areas.

1.2 Scope

The SafeStreets environment is basically composed of three parts: the central SafeStreets system, which is the main server of the service, the SafeStreets user interface and the Authority Edition interface.

A user is allowed to use two main functions that are:

- Reporting a traffic violation;
- Analyzing statistics about traffic violations;

Everyone can simply download the SafeStreets app and can immediately use the user Interface to take advantage of these two main functions.
The first one gives the possibility to upload the main details of the traffic violation to report. The only thing that a user knows is that his report will be analysed by an authority if and only if there exists an authority that covers the area where the violation occurred.

The second function helps users to analyse the traffic violation trends in a defined area. These trends are only defined thanks to the reports sent by users to SafeStreets. In addition a user can also apply filters in his analysis such as requesting only confirmed tickets (the ones defined as valid violations by authorities), specifying only some kind of traffic violation, restricting research area in a defined time interval and so forth.

An authority, thanks to the SafeStreets Authority Edition interface, has much more privileges in all SafeStreets system than the ones offered to users.
In fact, an authority is also allowed to use four main functions:

- Evaluating traffic violation reports of its competent area;
- Sending accidents information to SafeStreets system;
- Receiving suggestion from SafeStreets to reduce accidents;
- Analysing advanced statistics about traffic violations;

The first function gives the possibility to validate the reports sent by users. It is assumed that if an authority validates a report, it is consistent and so the offender will be fined.
The second function allows authorities to enrich SafeStreets' database uploading details of accidents in order to enable the system to activate for them the third function defined below.
The third function combines SafeStreets received reports to accidents information to define automatically suggestions in order to reduce accidents. These suggestions are sent, if available, from the SafeStreets system to the Authority Edition interface of the interested authority. Finally, the last function is the advanced version of the user analysing function. It is a more detailed version of that function according to the fact that authorities are allowed to see all the sensible data of SafeStreets system such as all the license plate related to each report.
The detailed description of each function is defined in section 2.

1.2.1 World and shared phenomena

In order to draw up a list of requirements, it is necessary to identify those events that occurs in the world. Some of them are shared with a machine: these phenomena define the interface through which machines interact with the world.

World phenomena

- Some people may not respect road traffic rules.
- Some people may park their own vehicle in a forbidden area.
- Accidents can occur.

Shared phenomena

- Authorities could generate traffic tickets thanks to reports.
- An authority can receive some suggestions, derived from data analysis.
- Traffic violations which are reported can be shown in a map.

1.2.2 Goals

SafeStreets wants to provide to its users a service which aims to reach these goals:

[G1] An individual can report traffic violations to SafeStreets.

[G2] An individual can mine information about violations.

[G3] An authority can share its information about accidents occurred on its operative area with SafeStreets.

[G4] An authority can retrieve suggestions of possible interventions from SafeStreets.

[G5] An authority can evaluate SafeStreets reports related to its area of competence.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Operative area**: the geographic area in which an authority retains control and actually works.
- **Registered authority**: an authority that decided to participate in SafeStreets initiative and that installed in its head office a particular version of the application that provides some features not available to common users (for example, the possibility to rate the traffic violation reports collected by SafeStreets).
- **Report image**: it is the image acquired through the SafeStreets app. It is obligatorily associated with a valid report.
- **Report timeout**: it is the timeout started when SafeStreets app acquires the report image. If the user doesn't complete the completion of the report by the deadline of this timeout, the report is considered invalid.
- **SafeStreets**: it is the crowdfunding app subject of this document.
- **SafeStreets application**: it is the application used by users, installed on their smartphone.
- **SafeStreets Authority Edition**: it is the application used by authorities, installed on computers in their station.

- **SafeStreets Client**: a generic way to point out the SafeStreets application or the SafeStreets Authority Edition.
- **Traffic violation report**: it is the message that SafeStreets collects through its app from users who want to report an alleged violation. It is often abbreviated as "report".
- **Individual**: A generic user or authority.

1.3.2 Acronyms

- **AE** : SafeStreets Authority Edition.
- **OCR** : Optical Character Recognition

1.3.3 Abbreviations

- [Gn]: n^{th} goal.
- [Rn]: n^{th} functional requirement.
- [An]: n^{th} domain assumption.
- [UCn]: n^{th} use case.

1.4 Revision history

Version	Last update	Comments
1.0	10^{th} November, 2019	Initial version
2.0	9^{th} December, 2019	Better organization of assumptions and requirements

Table 1: Revision history

1.5 Document Structure

In this part is shown how the document has been divided. For each chapter, is given a short description:

- *Chapter 1* introduces the problem and describes an analysis of the world and of the shared phenomena. Here are included the purpose of SafeStreets application, its goals and some premises are formalized to provide a better understanding of following chapters.
- *Chapter 2* includes further details on the shared phenomena and domain model, specifying which are the most important requirements and who will be a user of the system. Moreover, a formalization of assumptions and constraints is given.
- *Chapter 3* analyses more deeply all aspects of Chapter 2, providing more details for the development team: in this part, the focus is on requirements.
- *Chapter 4* presents a formal analysis of the problem, using Alloy. Some worlds are generated from the model and assertions are checked.
- *Chapter 5* shows the effort spent by each group member for this project.
- *Chapter 6* simply includes the references.

2 Overall description

2.1 Product perspective

Figure 2.1 gives a visual representation of shared phenomena and domain model.

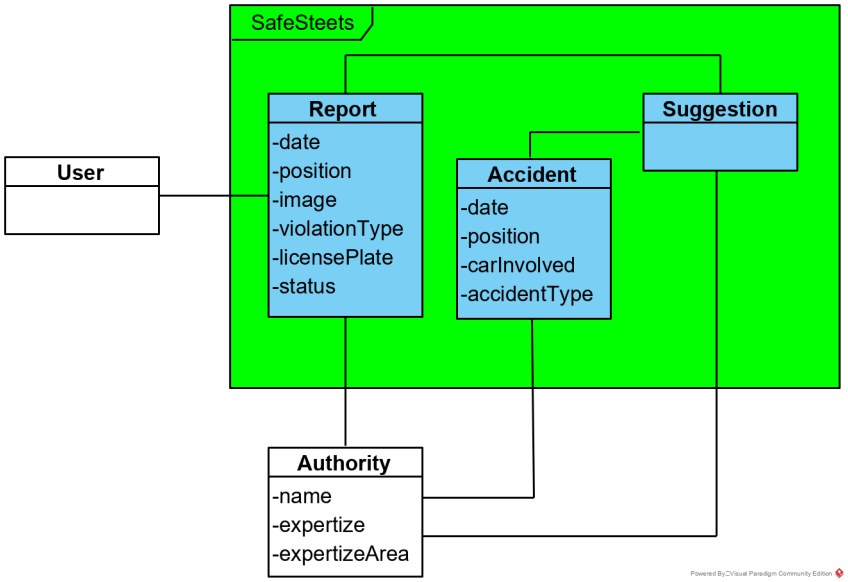


Figure 2.1: Domain Model

The users can report traffic violations through the SafeStreets application. SafeStreets does not collect any data about users, so they are represented as a black box. The interaction between users and SafeStreets is established only through a report, which is published in the SafeStreets system.

A registered authority can be noticed about traffic violations reported in its own operative area and eventually evaluate if that traffic violation report is valid or not. Moreover, an authority can report to SafeStreets any information about accidents in order to identify unsafe areas and suggest possible interventions.

To understand the main events in SafeStreets system, some statechart diagrams are represented in figures below.

Figure 2.2 explains how SafeStreets works when a user try to report a traffic violation.

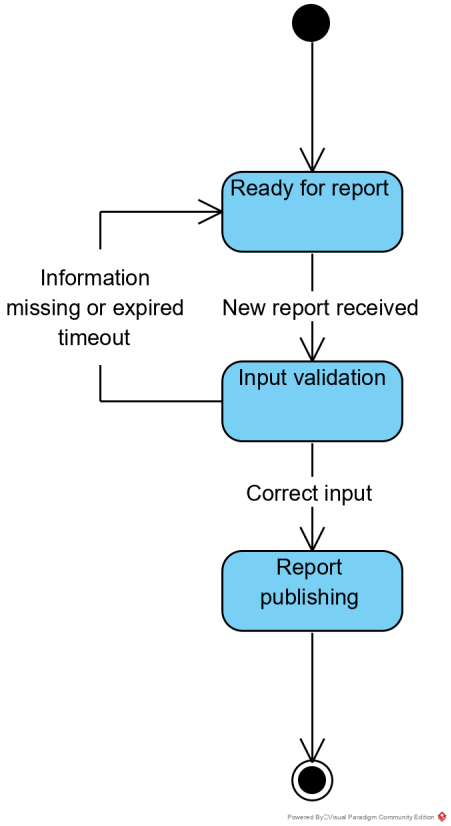


Figure 2.2: State chart Diagram about user report.

The core of the application is based on user reports: the more reports there are, the more the system can provide information about areas in a municipality. If the app considers a report valid, then it is made public.

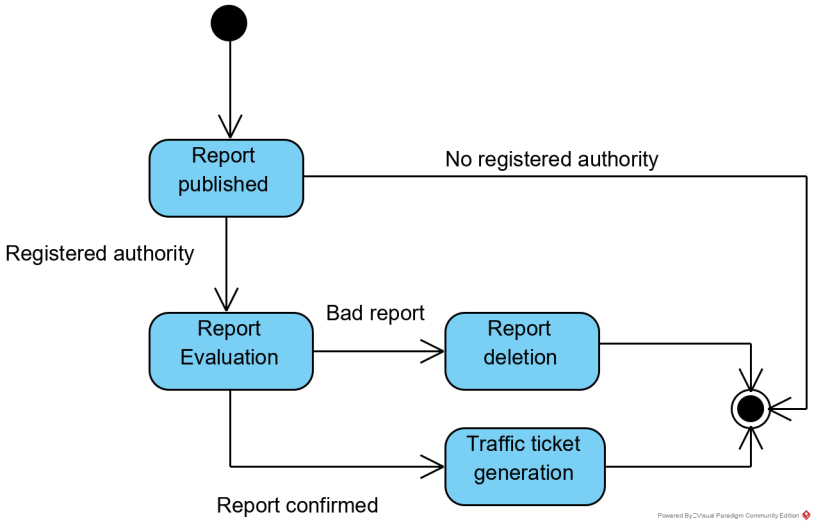


Figure 2.3: State chart Diagram about SafeStreets suggestions for an authority.

Figure 2.3 shows how SafeStreets operates with an authority.
Authorities could eventually have some information about accidents in their operative zone and can decide to share them with SafeStreets.
SafeStreets collects these data and try to cross them with reports about traffic violations: if for an area there are frequent violations or accidents, SafeStreets suggests possible interventions.

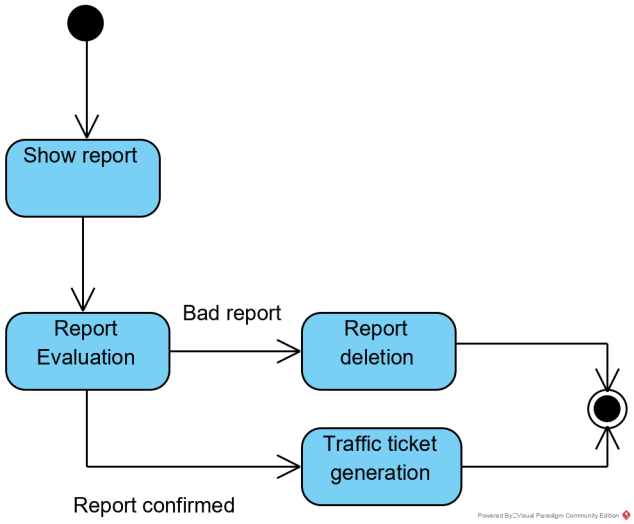


Figure 2.4: State chart Diagram about SafeStreets system for traffic tickets generation.

Finally, Figure 2.4 represents SafeStreets system that allows an authority to show all violations reported in its operative area. Analysing a report, an authority can establish if it is actually valid in order to generate a traffic ticket for the offender. Otherwise, if report is not valid, the authority can delete it.

2.2 Product functions

In the following section all the main product functions are defined. The reporting and data-mining functions are considered the main ones, while the other are advanced ones offered by SafeStreets thanks to the collaboration with authorities. Thus, if authorities, in a defined area, are not sharing information with SafeStreets, these advanced functions are not available in that area.

2.2.1 Violation reporting

This is the core function of SafeStreets. The aim is to give to the user the possibility to report a traffic violation. The system allows the user to use this function without any previous login. The user is asked to complete his reporting action defining the following fields:
- Picture representing the violation;
- Type of violation.

Other important information such as position, date and time of the violation are automatically detected by the application.
In particular, to take the requested picture, the user is allowed only to use the in-app dedicated camera tool in order to prevent users from reporting past violations uploading old pictures and to avoid picture manipulations.
The picture is approved by the app if and only if the license plate is correctly recognized by the dedicated algorithm, otherwise, the app will ask to the user to take it again.
The type of violation is selectable only by the ones offered by SafeStreets.
The user is asked also to complete these two requests in a defined interval of time in order to guarantee the consistency of the report offered to SafeStreets.
When a violation is correctly filled in, it is sent to the system which will publish it and will make it visible to all users.
If an authority that covers the zone in which the violation occured is registered to SafeStreets, then the report is sent to it for evaluation.

2.2.2 Data-mining

This function allows the user to analyse SafeStreets data in order to mine some information, for example, by highlighting the streets (or the areas) with the highest frequency of violations. The kind of information requested can be filtered by user in order to restrict the research area.
Furthermore, SafeStreets can provide to authorities statistics about vehicles, not accessible to common users, for example about the most egregious offenders, or the effectiveness of the SafeStreets initiative by looking for trends in the issuing of tickets.

2.2.3 Accidents collector

This function is available only for authorities that manage accidents. They are allowed to enrich SafeStreets' data inserting information about accidents in the competent area. This function will be useful for the interactive suggestion function defined below.

2.2.4 Interactive suggestions

This function is strictly related to the collaboration between SafeStreets data, offered by users' reports, and authorities, which can enrich the database including information about the accidents that occur.
SafeStreets can cross all these information and it can also identify potentially unsafe areas suggesting possible interventions to authorities.
In a more detailed way, this function, analyzing data and making a comparison between them and some defined limit parameters, can automatically detect possible standard-defined improvements, create suggestions useful to reduce violations and accidents.

2.2.5 Report confirmation

SafeStreets forwards a report received from a user to an authority that covers the area of the report, if such authority is registered to the system.
Then, the authority can evaluate it: if the report is considered valid, then SafeStreets updates the status of that report as "confirmed", assuming the authority has generated a traffic ticket from it.
If the report is considered not valid, SafeStreets deletes it from the system.

2.3 User characteristics

The actors of the application are the following:

- User: a person that is using SafeStreets app. He is a "guest" because he is not logged in, so no specific information about him are given. He can actively use the first two basic functions so he can send a violation reporting or do data-mining.

- Authority: special kind of user. It is allowed to use the user functions but it has also additional privileges on them such as to visualize more detailed information about reports. In particular, it is a registered identity that is responsible to manage reports addressed to its area of competence.
  An authority can send information about accidents of its area of competence to SafeStreets and can use SafeStreets statistics to identify the most egregious offenders.
  A kind of authority is the police, that can use reports to make traffic tickets validating SafeStreets reports.

2.4 Assumptions, dependencies and constraints

This section specifies the assumptions made in the rest of the document, that is those facts that have been considered true a priori and that are not manageable by SafeStreets.

[A1] The device on which the SafeStreets client is installed has internet access.

[A2] The device on which the app is installed has geolocation features.

[A3] The device on which the app is installed has an external camera.

[A4] The device on which the app is installed is able to detect its position with a maximum error of five meters.

[A5] The position detected by the app coincides with the actual position detected by the geolocalizer.

[A6] The algorithm used by SafeStreets for reading a license plate from an image reads the correct number plate, if it is present in the image.

[A7] Each position on earth can be associated with a unique postal code.

[A8] The SafeStreets client must be able to communicate with the SafeStreets system.

[A9] The only license plate shown in the report image is the one of the offender.

[A10] Each violation report must refer to only one offender.

[A11] In an authority confirms a violation report, than the authority generates a traffic tickets from it. So, from the point of view of SafeStreets, every confirmed violation report equals to a traffic ticket.

[A12] The SafeStreets software installed in the offices of the authorities registered with the service can only be used bu the authorities themselves: it is not possible for an unauthorized user to gain access to such software.

[A13] The device on which the SafeStreets Authority Edition is installed is always switched on.

3 Specific requirements

This section contains all the requirements of SafeStreets, i.e. all that allows the app to function as desired by the developer. The requirements depend exclusively on the choices of the developer. Aspects not controllable by the developer, but relevant for the application, are the assumptions reported in 2.4.

3.1 External Interfaces Requirements

3.1.1 User Interfaces

The following mockups give an approximate idea of how the interfaces of the application should appear. In the fist block main SafeStreets application interfaces are shown. In the second one main SafeStreets Authority Edition interfaces are shown.

3.1.2 Hardware Interfaces

Since SafeStreets is a crowd-sourced application, each user could take advantage of the system only if it has a smartphone. In particular, the smartphone must have an external camera, GPS and an internet connection. Without the camera or the GPS, it is not possible to report violations. Without an internet connection, every SafeStreets function isn't available.

3.1.3 Software Interfaces

First of all, SafeStreets must be authorized by the operating system of the smartphone on which it is installed to access the camera, the GPS and the internet connection.

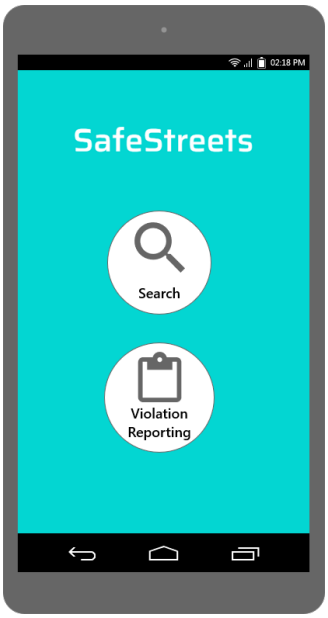
In order to provide information about traffic violations to users, SafeStreets uses external maps to show where reports come from. SafeStreets gets these maps from OpenStreetMap (`https://www.openstreetmap.org`).

Moreover, one of the element to ensure that a report could be formalized is the validation of license plate. SafeStreets takes advantage of an algorithm to automatically recognize characters on license plates (OCR Algorithm).

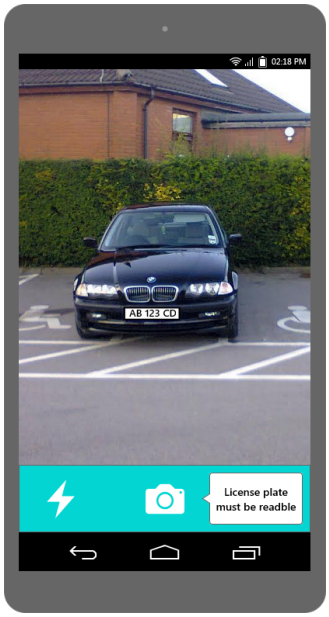
3.1.4 Communication Interfaces

Each function provided by SafeStreets needs an Internet connection.

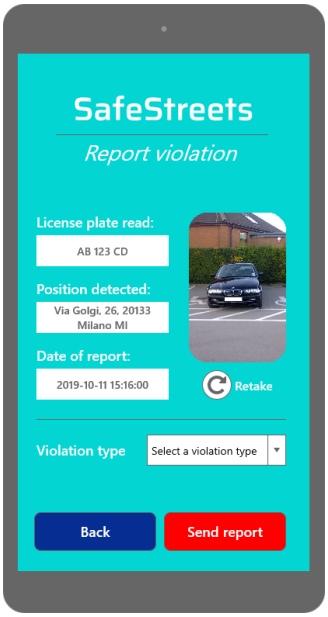
(a) *Welcome page, starting the application*



(b) *Application Home Page. The two main function are shown to the user.*

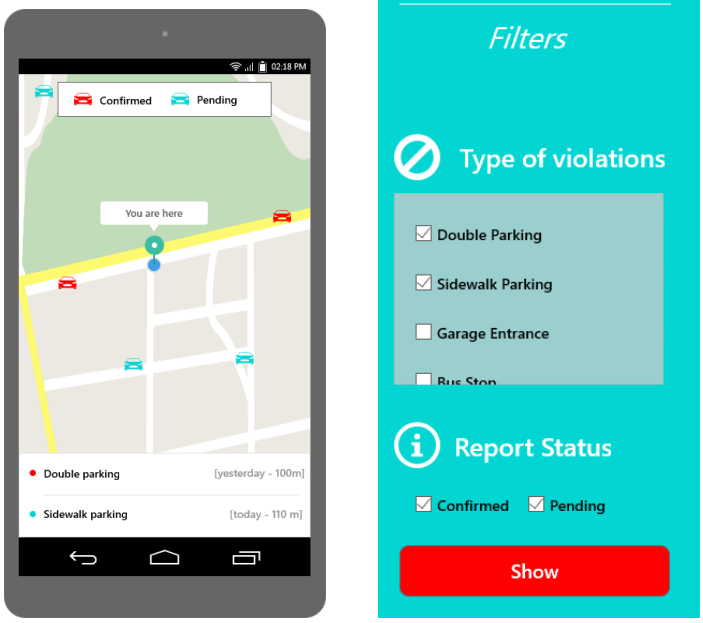


(c) *Violation Reporting: Take Picture*



(d) *Violation Reporting: Metadata acquired from picture + Violation Type selection*

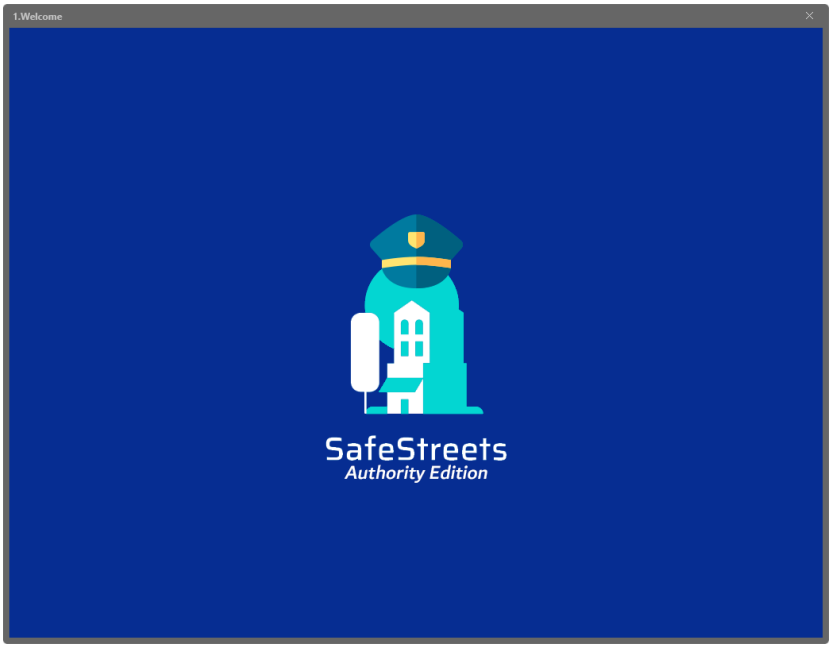
Figure 3.5: SafeStreets application interfaces



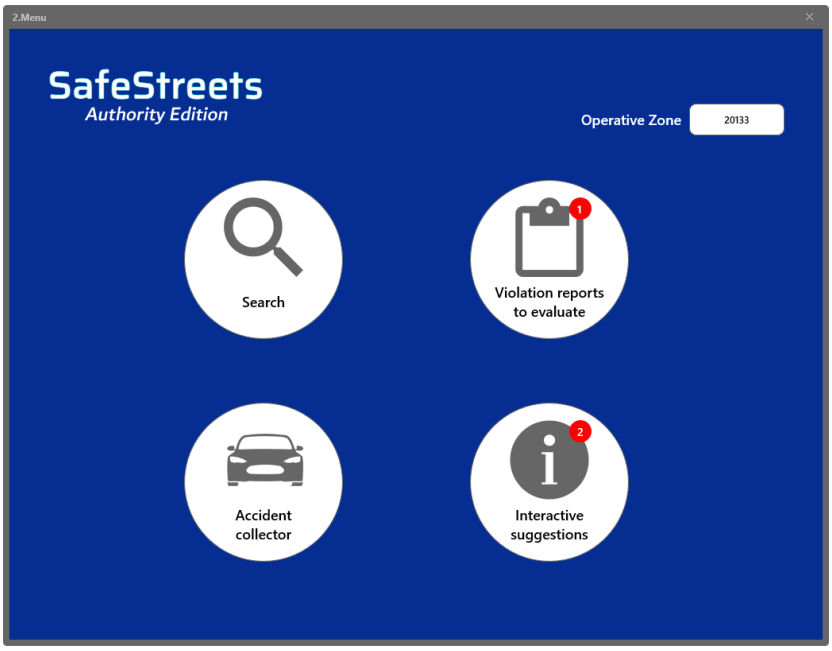
(a) *Data Mining: browse map*

(b) *Data Mining: side menu filters*

Figure 3.6: SafeStreets application interfaces

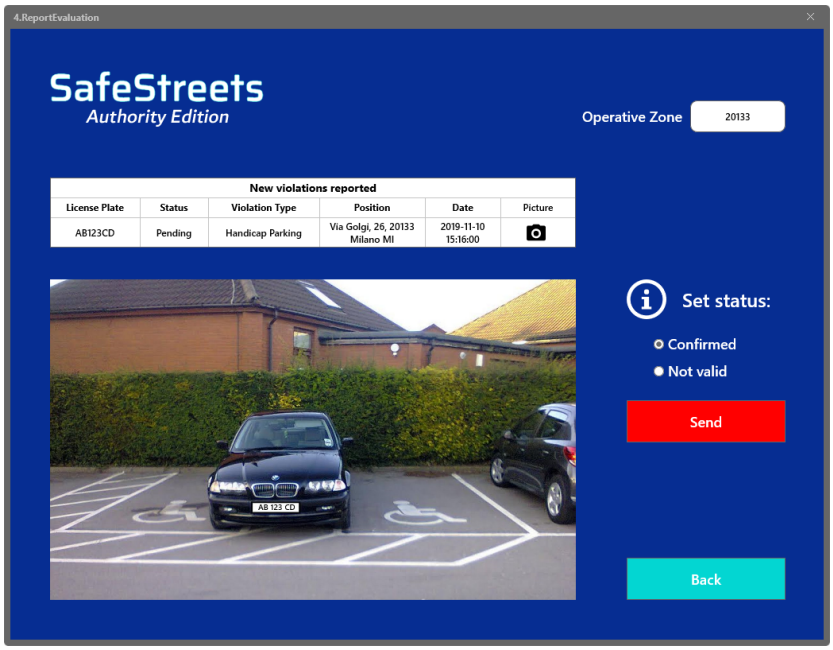


(a) *Welcome page, starting the AE system*

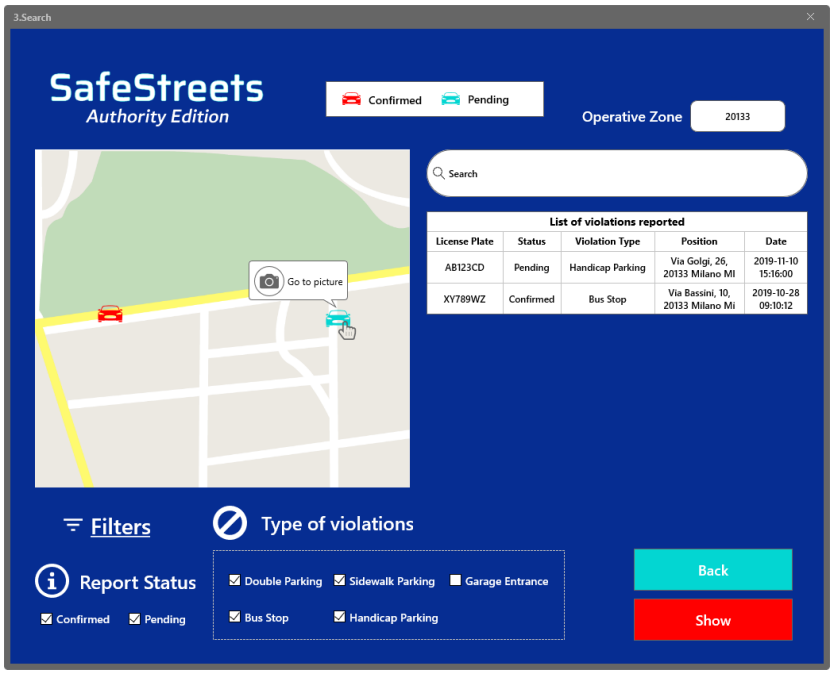


(b) *Home Page*

Figure 3.7: SafeStreets Authority Edition interfaces

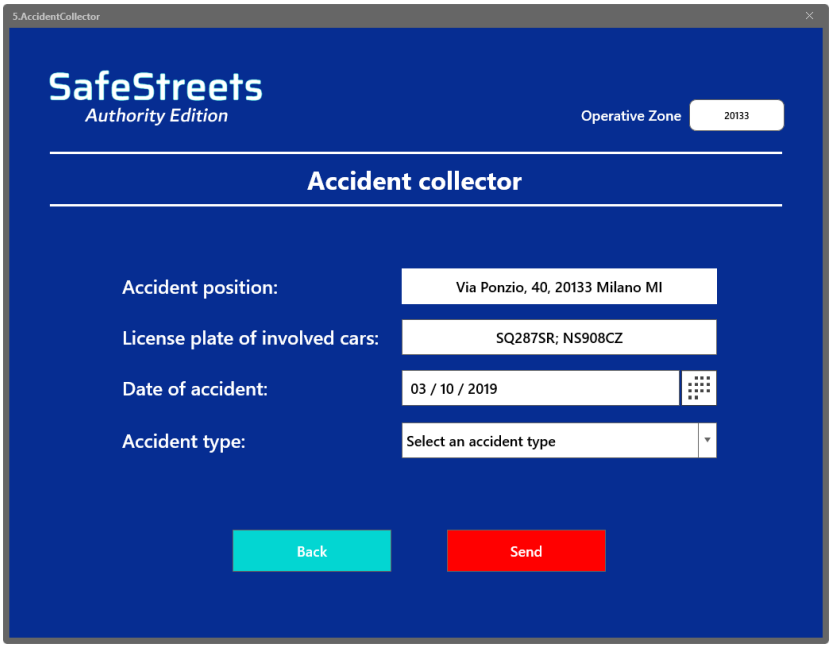


(a) *Report Evaluation*

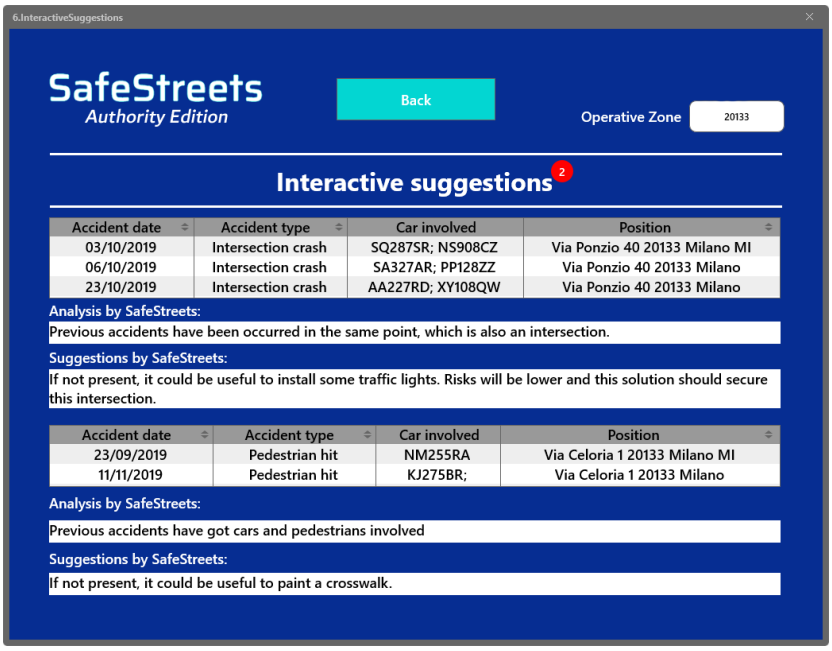


(b) *Data Mining*

Figure 3.8: SafeStreets Authority Edition interfaces



(a) *Accident Collector*



(b) *Interactive Suggestions*

Figure 3.9: SafeStreets Authority Edition interfaces

3.2 Functional Requirements

In the current section will be shown for each goal the list of functional requirements and the assumptions to be respected.

[G1] An individual can report traffic violations to SafeStreets.

- [A1] The device on which the SafeStreets client is installed has internet access.
- [A2] The device on which the app is installed has geolocation features.
- [A3] The device on which the app is installed has an external camera.
- [A4] The device on which the app is installed is able to detect its position with a maximum error of five meters.
- [A5] The position detected by the app coincides with the actual position detected by the geolocalizer.
- [A6] The algorithm used by SafeStreets for reading a license plate from an image reads the correct number plate, if it is present in the image.
- [A7] Each position on earth can be associated with a unique postal code.
- [A8] The SafeStreets client must be able to communicate with the SafeStreets system.
- [A9] The only license plate shown in the report image is the one of the offender.
- [A10] Each violation report must refer to only one offender.
- [R1] The user report must be of one of the type of violation defined by SafeStreets.
- [R2] The report image must show the license plate of the offender.
- [R3] The user must insert the picture of the violation.
- [R4] The user must complete the report before the expiration of the report timeout.
- [R5] The OCR algorithm must detect one license plate into the report image.
- [R6] The APP must detect automatically position and date.
- [R7] Every violation report sent to server must be filled in all its fields.
- [R17] Every violation reported to SafeStreets is consistently saved into the SafeStreets database.

[G2] An individual can mine information about violations.

- [A1] The device on which the SafeStreets client is installed has internet access.
- [A11] If an authority confirms a violation report, than the authority generates a traffic ticket from it. So, from the point of view of SafeStreets, every confirmed violation report equals to a traffic ticket.

[R8] The SafeStreets client must show all and only the information required by the individual.

[R9] Every individual can browse the map.

[R10] Every individual can see statistics in map.

[R11] Every individual can define filters for the research.

[R12] Every authority can see advanced statistics in map (including sensitive information).

[R13] Every authority can define advanced filters for the research (including sensitive information).

[G3] An authority can share its information about accidents occurred on its operative area with SafeStreets.

[A1] The device on which the SafeStreets client is installed has internet access.

[A7] Each position on earth can be associated with a unique postal code.

[A12] The SafeStreets software installed in the offices of the authorities registered with the service can only be used by the authorities themselves: it is not possible for an unauthorized user to gain access to such software.

[A13] The device on which the SafeStreets Authority Edition is installed is always switched on.

[R14] The information about accidents must be of one of the type defined by SafeStreets.

[R15] Information about accidents occurred must include all the information required by SafeStreets.

[R16] Every accident reported by a registered authority is consistently saved into the SafeStreets database.

[G4] An authority can retrieve suggestions of possible interventions from SafeStreets.

[A1] The device on which SafeStreets client is installed has internet access.

[A12] SafeStreets software installed in the offices of the authorities registered with the service can only be used by the authorities themselves: it is not possible for an unauthorized user to gain access to such software.

[A13] The device on which SafeStreets Authority Edition is installed is always switched on.

[R18] SafeStreets must be able to generate suggestions, based on accidents and confirmed violation reports.

[R19] Every authority must receive suggestions only referred to its operative area.

[G5] An authority can evaluate SafeStreets reports related to its area of competence.

[A1] The device on which the SafeStreets client is installed has internet access.

[A11] If an authority confirms a violation report, than the authority generates a traffic ticket from it. So, from the point of view of SafeStreets, every confirmed violation report equals to a traffic ticket.

[A12] SafeStreets software installed in the offices of the authorities registered with the service can only be used by the authorities themselves: it is not possible for an unauthorized user to gain access to such software.

[A13] The device on which the SafeStreets Authority Edition is installed is always switched on.

[R20] An authority receives a report if and only if that report is related to a position covered by that authority (i.e. the postal code of the position of the violation report is assigned to that authority).

[R21] When a registered authority confirms a violation report, that confirmation is consistently stored into the server.

[R22] A violation report rejected by a registered authority is removed from SafeStreets database.

3.2.1 Use Case Diagrams

In this section the use case of the system is shown.

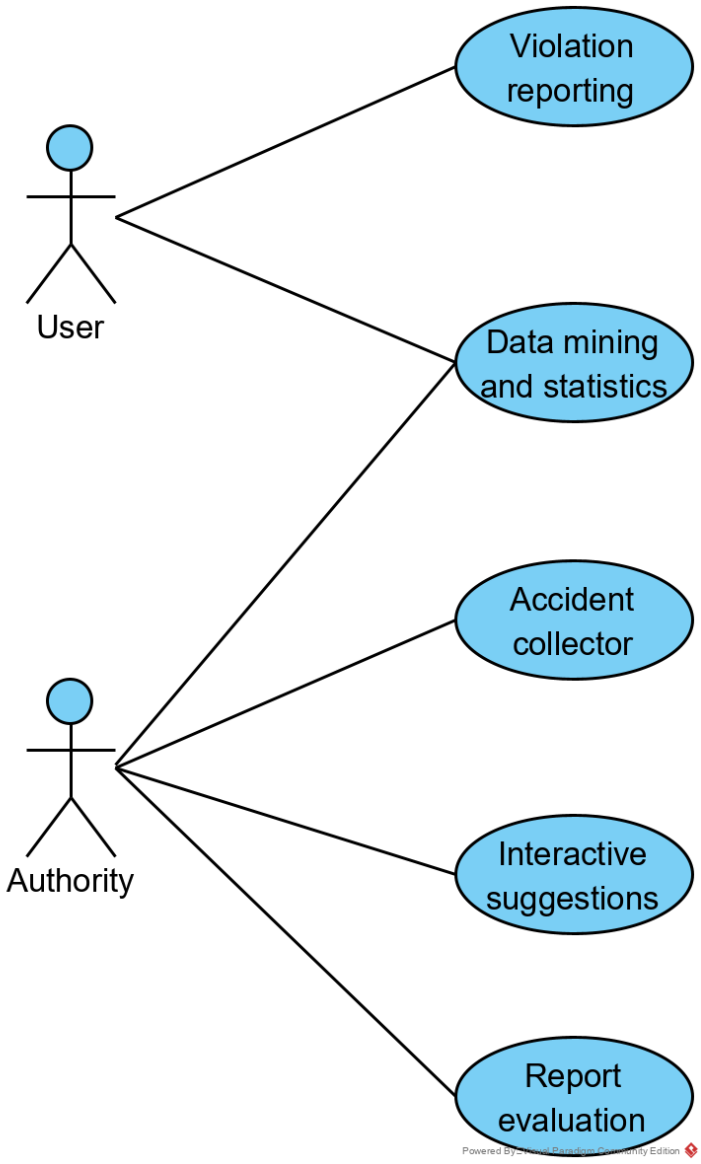


Figure 3.10: SafeStreets Use Case Diagram

3.2.2 Use Case Description

ID: [UC1]
Name: Violation reporting
Actor: User
Entry conditions:

1. Open the SafeStreets application.
2. Open the "Violation reporting" section.

Event flow:

1. The app opens the camera to take the picture of the violation. Take the picture. When the picture is taken, the reporting timeout starts.
2. The app shows the taken picture and the license plate read from it by the license plate recognition algorithm. If the user is not satisfied by the result, he can take a new picture clicking on the button "take picture". The reporting timeout is restarted every time a picture is taken.
3. Choose a "type of violation" from the list of violations recognized by SafeStreets. The user cannot define itself a type of violation.

Exit conditions:

- The user can always leave the violation reporting function clicking on the "Back" button. In this way, he returns to the main page of the app.
- Click "Send" to complete the violation reporting. The app returns to the main page.

Exceptions:

- If the camera or the geolocalization service aren't available, the app shows an error message and returns to the main page: it is not possible to send a violation report.
- If the license plate recognition algorithm fails, the user is forced to take the picture again.
- If the user clicks on "Send" without having filled all the requested fields, the app shows a warning message that tells the user what is wrong. The user can close the warning message and come back to the violation report.
- If the reporting timeout expires before the user has correctly sent the violation report, an error message is shown, than the app discard the violation report and returns to the main page.

ID: [UC2]
Name: Data mining and statistics
Actor: User, Authority
Entry conditions:

1. User opens the SafeStreets application; authority opens SafeStreets AE.
2. Open the "Search" section.

Event flow:

1. The app shows in a new screen the map. Then, report violations are represented in the map with different coloured pins, based on confirmed or unconfirmed reports. This screen also provides a menu through which either a user or an authority can refine its research with data filters. They select the filters they want.
2. Taps on "Show" button.
3. Explores data about result reports. In particular:
   - A user can retrieve information about type of violation, date, position and eventually the label "confirmed" or "pending".
   - An authority can retrieve the same information as before, adding license plates and proof pictures.

Exit conditions:

- Either user or authority can always leave the "Search" function clicking on the "Back" button. In this way, they return to the main page of the app.

Exceptions: -

ID: [UC3]
Name: Accident collector
Actor: Authority
Entry conditions:

1. Open SafeStreets AE.
2. Open the "Accident Collector" section.

Event flow:

1. Compile requested fields: type of accident, offenders' license plate (separated by a semicolon), address of the accident, date of the accident.

Exit conditions:

- The authority can always leave the accident collector function clicking on the "Back" button. In this way, it returns to the main page of the app.
- Click "Send" to complete the accident compilation form. SafeStreets AE returns to the main page.

Exceptions:

- Some of the requested form are not correctly compiled or empty. In this case a warning message is shown. The authority can close the warning message and come back to the accident collector.

ID: [UC4]
Name: Interactive suggestions
Actor: Authority
Entry conditions:

1. Open SafeStreets AE.
2. Open the "Interactive suggestions" section.

Event flow:

1. A list of suggestion are shown.
2. The authority can mark the read suggestion clicking on the "Got it" button in order to hide the already read suggestions.

Exit conditions:

- The authority can always leave the accident collector function clicking on the "Back" button. In this way, it returns to the main page of SafeStreets AE.

Exceptions: -

ID: [UC5]
Name: Report evaluation
Actor: Authority
Entry conditions:

1. Open SafeStreets AE.
2. Click on "Violation reports to evaluate".

Event flow:

1. If there is at least one report to evaluate, the app shows all the information about the report. Choose between the buttons "Confirmed" and "Not valid". Their meaning is straightforward.

Exit conditions:

- Click on the chosen button. If there is another violation report to evaluate, the event flow restarts. Otherwise, SafeStreets AE returns to the main page. Remember that SafeStreets considers every confirmed violation report as a given traffic ticket.
- The authority can always leave the violation report confirmation function clicking on the "Back" button. In this way, it returns to the main page of SafeStreets AE.

Exceptions: -

3.2.3 Activity diagrams

The core of SafeStreets application is based on violation reports by users. Figure 3.11 explains in little steps how a user can do this operation.

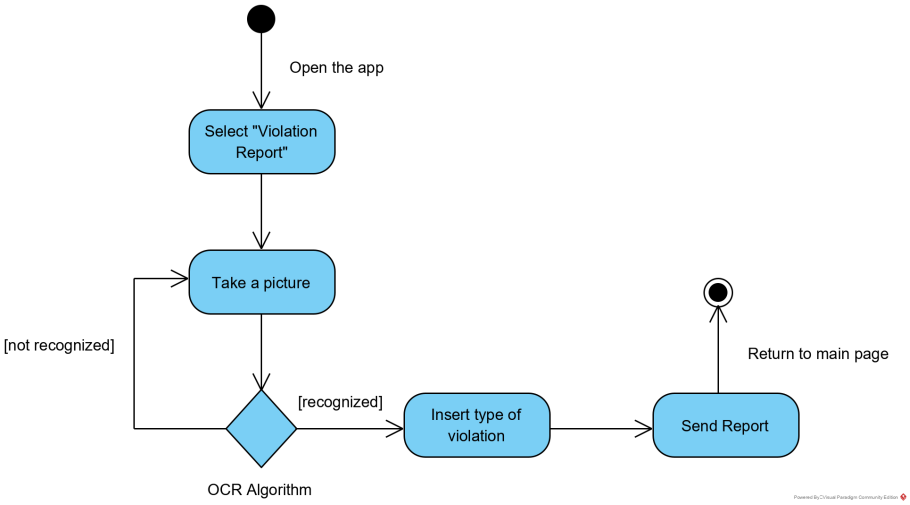


Figure 3.11: SafeStreets activity diagram for violation reporting

Opening the SafeStreets application and selecting the "Violation Report" button, the user has to take a photo of the vehicle he thinks it is in a violation condition. The photo must clearly include the license plate of the vehicle. An OCR algorithm get the picture and shows the read characters: if the user confirms that the license plate is correct, then proceeds in reporting, otherwise he can take a picture again until he's satisfied.
Proceeding in reporting, the user has given the possibility to choose one of the violation types provided by SafeStreets app. After that, report is sent.

Data mining function (Figure 3.12) is executable either by users or by authorities: the main difference is the visibility level of some information.

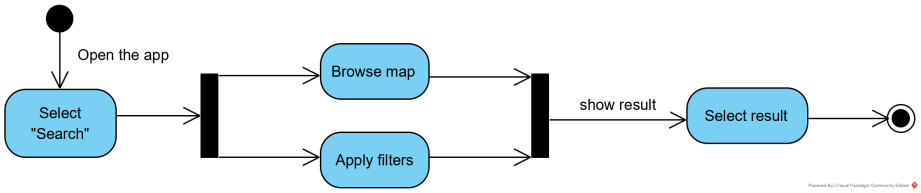


Figure 3.12: SafeStreets activity diagram for data mining

Opening the app (SafeStreets application for users, SafeStreets AE for authority), it is possible to browse information about violation reports, selecting the "Search" button.
Both users and authorities can explore a map of violations position and click to retrieve information about them; moreover, they can specify some additional filter to refine a research.
Finally, results are shown.

3.2.4 Sequence Diagrams

Let's look at details in interactions between a user and SafeStreets, during the violation reporting phase (Figure 3.13).

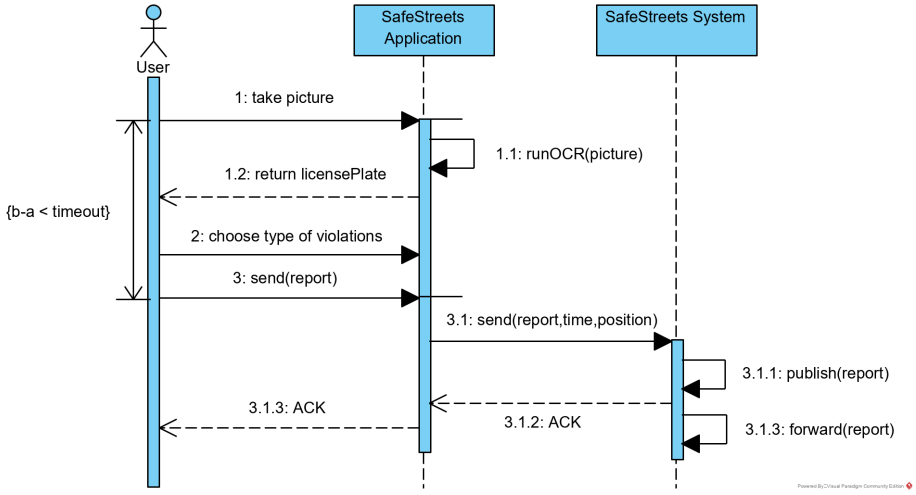


Figure 3.13: SafeStreets sequence diagram for violation reporting

Steps from (1) to (3) have been explained in Figure 3.11: here, an additional information is given representing the time constraint *"b-a"*.
This interval guarantees that a report could be sent about at the same time in which user sees a vehicle in a possible violation condition.
In this way, users can not keep for a long time a report before they send it, so a timeout must be considered.
From step (3.1), the application actually retrieves information about time and position and send the report to SafeStreets system. Then, report will be published and forwarded to the registered authority which operates in the specified position.

The next two sequence diagrams explain detailed steps about authorities' function, provided by SafeStreets AE.

Figure 3.14 represents that service for authorities, able to manage and evaluate reports sent by users.
The authority checks if there are new report to evaluate, so, through the application AE, contact SafeStreets system to retrieve these information.
If there's at least one report to evaluate, SafeStreets AE will show it and permit authorities to establish if the report is correct or not.
If it is correct, the system will update the report status, checking it as "Confirmed"; otherwise SafeStreets system will remove that report.

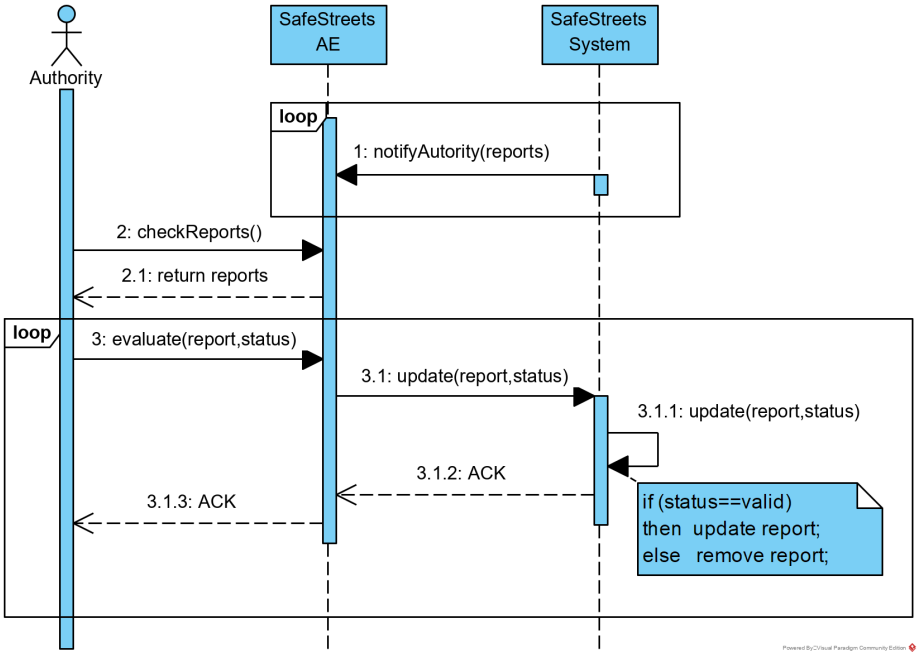


Figure 3.14: SafeStreets sequence diagram for report evaluation

Finally, an authority can eventually insert in SafeStreets system information about accidents occurred in its operative area.
Gradually, SafeStreets system collect these data and cross them with report violations: once it has analyzed enough data, SafeStreets system will generate for the authority a set of suggestions, aimed to secure the area.

Figure 3.15 describes the just mentioned functionality.

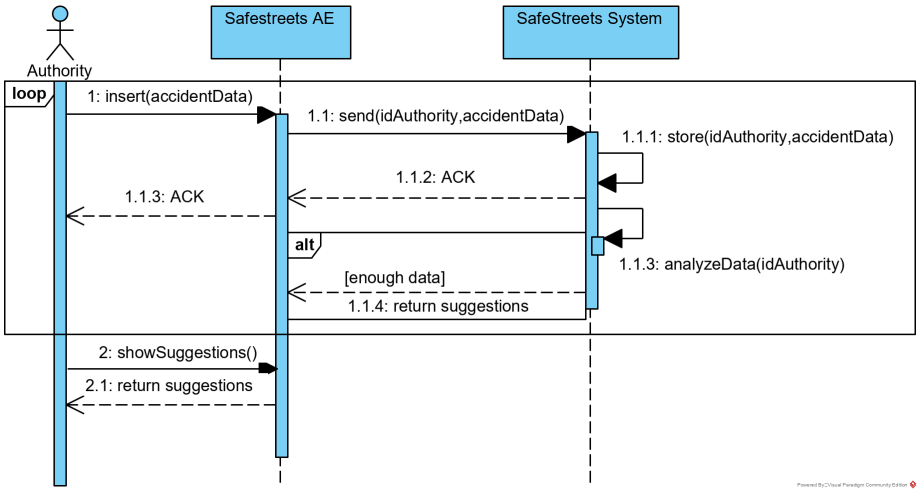


Figure 3.15: SafeStreets sequence diagram for showing suggestions

3.2.5 Requirements traceability matrix

Goal Id	Use case Id	Requirement Id
[G1]	[UC1]	[R1]
		[R2]
		[R3]
		[R4]
		[R5]
		[R6]
		[R7]
		[R17]
[G2]	[UC2]	[R8]
		[R9]
		[R10]
		[R11]
		[R12]
		[R13]
[G3]	[UC3]	[R14]
		[R15]
		[R16]
[G4]	[UC4]	[R18]
		[R19]
[G5]	[UC5]	[R20]
		[R21]
		[R22]

Table 2: Traceability matrix

3.3 Performance Requirements

SafeStreets aims to provide a smooth user experience. For this reason, all the functionalities must be implemented so that the delay between a data request and an answer is unnoticeable, as well as the delay between the sending of a violation report and its publication.
To achieve this goal, the system needs a fast internet connection. Moreover, the data sent between SafeStreets clients and servers must be compressed so to guarantee a good user experience also when the internet connection is pretty slow.
In general, SafeStreets app doesn't need a lot of resources, so it can be installed on a large variety of smartphones, also dated ones.

3.4 Design Constraints

3.4.1 Standards compliance

SafeStreets app must respect the rules of the app stores in which it will be available.
Moreover, it must respect the authorization rules of the operating systems where it will be installed.

3.4.2 Hardware limitations

As mentioned in 3.1.2, the application relies on some hardware components, that, if not present, doesn't allow to use SafeStreets in all its power. An external camera, a GPS service and mechanisms to connect to the internet must be present on the device to use SafeStreets.
SafeStreets AE must have an internet connection of any kind and should be turned on round the clock to receive reports that have to be evaluated.

3.4.3 Any other constraint

SafeStreets doesn't collect personal data from its users. The only sensitive data it has to manage are the license plate associated to violation reports: they are not visible to common users, because it would make possible to associate a license plate to a particular human identity. This is not advisable: SafeStreets is an app that wants to hep authorities in finding infractions, but it doesn't want to have any impact on the reputation of people that, for any kind of reason, had done a certain number of violations.

3.5 Software System Attributes

3.5.1 Reliability

Data collected by SafeStreets are fundamental for its functionalities. Therefore, there must be a redundant server structure to guarantee all the data about

violations and accidents are never lose.

3.5.2 Availability

SafeStreets doesn't provide services that are critical for the life of its users. However, a user should be able to exploit SafeStreets functionalities in every part of the day, so it is interested in keeping his infrastructure active as much as possible.

3.5.3 Security

An information, after being stored on SafeStreets servers, is in "read-only" mode: only the state of a violation report - "confirmed" or "pending"- can be modified by external agents (precisely, by authorities). This guarantees broad security against attacks aimed at changing the meaning of the data held by SafeStreets.
All communications between SafeStreets and its users use security protocols designed to guarantee the confidentiality and integrity of the communication.

3.5.4 Maintainability

SafeStreets source code must be well-documented and well-tested. This means that every part of the program must be commented so that its meaning is "crystal-clear", and there must be at least a 80% coverage of the code.

3.5.5 Portability

SafeStreets must be implemented so that it is as much platform-independent as possible. For this reason, it won't rely on functions that are available only on particular operating systems.

4 Formal analysis using Alloy

4.1 Purpose of the model

In this section some basic functionalities of SafeStreets are proved to be consistent. These are the assertions checked:

1. Every violation report received by the SafeStreets Server is complete in all its fields.
2. Every violation report was sent to the server before the report timeout ended.
3. When a registered authority confirms a violation report, that confirmation is consistently reported into the server.
4. A violation report rejected by a registered authority is removed from SafeStreets database.
5. Every accident reported by a registered authority is consistently saved into the SafeStreets database.

Moreover, there are other fundamental constraints expressed so to get consistent models. For example, it is specified that every SafeStreets App is associated to one and only one user. Anyway, every constraint is straightforward or it is well commented in Alloy code.

4.2 Alloy model

```
abstract sig Timeout {}
sig Running, Ended extends Timeout {}

sig Date, Time, Position, Image, TypeOfViolation, LicensePlate {}

abstract sig ReportStatus {}
sig Pending, Confirmed extends ReportStatus {}

sig ViolationReport {
        date: one Date,
        time: one Time,
        position: one Position,
        reportImage: one Image,
        type: one TypeOfViolation,
        licensePlate: one LicensePlate,
        status: one ReportStatus
}

fact EqualImagesIsImpossible {
        all ri: Image |
                one vr: ViolationReport |
                        vr.reportImage = ri
}

fact ViolationReportKey {
        all vr1: ViolationReport |
                no vr2: ViolationReport |
                        vr1 ≠ vr2 and
                        vr1.licensePlate = vr2.licensePlate and
                        vr1.date = vr2.date and
                        vr1.time = vr2.time and
                        vr1.position = vr2.position and
                        vr1.type = vr2.type
}

sig ViolationReportRequest {
        violationReport: one ViolationReport,
        timeout: one Timeout
}

fact associatedToAtLeastOneViolation {
        all d: Date, t: Time, p: Position, i: Image, tov:  TypeOfViolation,
            ↪ lp: LicensePlate |
                some vr: ViolationReport |
                        vr.date = d or
                        vr.time = t or
                        vr.position = p or
                        vr.reportImage = i or
                        vr.type = tov or
                        vr.licensePlate = lp or
                        vr.status = Pending or vr.status = Confirmed
}

//Every violation report is associated to one and only one violation report
    ↪ request.
fact ViolationReportAndRequestBijection {
        all vrr1: ViolationReportRequest |
                no vrr2: ViolationReportRequest |
                        vrr1 ≠ vrr2 and vrr1.violationReport = vrr2.
                            ↪ violationReport
}

sig SafeStreetsApp {
        violationReportsSent: set ViolationReport,
        user: one User
```

```
}

sig User {}

// Every SafeStreets App is associated to one and only one User.
fact SafeStreetsAppUserBijection {
        all ssa1: SafeStreetsApp |
                no ssa2: SafeStreetsApp |
                        ssa1 ≠ ssa2 and ssa1.user = ssa2.user
        all u: User |
                one ssa: SafeStreetsApp |
                        ssa.user = u
}

fact onlyRequestWithRunningTimeout {
        all ssa : SafeStreetsApp |
                all vrs: ssa.violationReportsSent |
                        one vrr : ViolationReportRequest |
                                vrr.violationReport = vrs and vrr.timeout =
                                    ↪ Running
}

pred reportViolation [ssa, ssa': SafeStreetsApp, vrr: ViolationReportRequest]
    ↪  {
        (
                vrr.timeout = Running and
                let vr = vrr.violationReport |
                        (
                        vr.date ≠ none and
                        vr.time ≠ none and
                        vr.position ≠ none and
                        vr.reportImage ≠ none and
                        vr.type ≠ none and
                        vr.licensePlate ≠ none and
                        vr.status ≠ none
                        )
        )
        implies
                ssa'.violationReportsSent = ssa.violationReportsSent + vrr.
                    ↪ violationReport
}

// Assertion 1: Every violation report received by the SafeStreets Server is
    ↪ complete in all its fields.
assert NoIncompleteViolationReports {
        all ssa, ssa': SafeStreetsApp, vrr: ViolationReportRequest |
                reportViolation[ssa, ssa', vrr]
                implies
                all vr: ssa.violationReportsSent |
                        (
                                vr.date ≠ none and
                                vr.time ≠ none and
                                vr.position ≠ none and
                                vr.reportImage ≠ none and
                                vr.type ≠ none and
                                vr.licensePlate ≠ none and
                                vr.status ≠ none
                        )
}

// Assertion 2: Every violation report was sent to the server before the
    ↪ report timeout ended.
assert NoTimedOutViolationReports {
        all ssa, ssa': SafeStreetsApp, vrr: ViolationReportRequest |
                vrr.timeout = Running and reportViolation[ssa, ssa', vrr]
                implies
                one vrs: ssa'.violationReportsSent |
                        vrr.violationReport = vrs and vrr.timeout = Running
```

```
}

sig TypeOfAccident {}

sig Accident {
        date: one Date,
        time: one Time,
        position: one Position,
        type: one TypeOfAccident,
        licensePlate: one LicensePlate
}

// The key of an Accident is (date, time, position, type).
pred AccidentKey {
        all a1: Accident |
                no a2: Accident |
                        a1 ≠ a2 and
                        a1.date = a2.date and
                        a1.time = a2.time and
                        a1.position = a2.position and
                        a1.type = a2.type
}

sig AuthorityName, PostalCode {}
abstract sig Competence {}

abstract sig Authority {
        name: one AuthorityName,
        typeOfCompetence: one Competence,
        areaOfCompetence: set PostalCode
}

fact associatedToAtLeastOneAuthority {
        all an: AuthorityName, c: Competence |
                some a:Authority |
                        a.name = an or a.typeOfCompetence = c
}

sig RegisteredAuthority extends Authority {}

// Every authority has a unique name.
fact AuthorityKey {
        all a1: Authority |
                no a2: Authority |
                        a1 ≠ a2 and a1.name = a2.name
}

sig SafeStreetsAE {
        violationsReported: set ViolationReport,
        accidentsToReport: set Accident,
        authority: one RegisteredAuthority
}

fact SafeStreetsAERegisteredAuthorityBijection {
        all ssae1: SafeStreetsAE |
                no ssae2: SafeStreetsAE |
                        ssae1 ≠ ssae2 and ssae1.authority = ssae2.authority
        all ra: RegisteredAuthority |
                one ssae: SafeStreetsAE |
                        ssae.authority = ra
}

sig SafeStreetsServer {
        violationReports: set ViolationReport,
        accidents: set Accident
} { #SafeStreetsServer = 1 }

//Every ViolationReport in a SafeStreetsAE is also present on
```

```
    ↪ SafeStreetsServer.
fact AllReportsInAEAlsoInServer {
        all ssae: SafeStreetsAE |
                all vr: ssae.violationsReported |
                        one vrServer : SafeStreetsServer.violationReports |
                                vr in vrServer
}

pred ConfirmViolationReport[ssae, ssae': SafeStreetsAE, sss:
    ↪ SafeStreetsServer, vr: ViolationReport] {
        vr in ssae.violationsReported //and vr in sss, but this is guaranteed
            ↪  by fact AllReportsInAEAlsoInServer
        implies
        (
                ssae'.violationsReported = ssae.violationsReported - vr and
                (
                        one vrs: SafeStreetsServer.violationReports |
                                vrs = vr and vrs.status = Confirmed
                )
        )
}

// Assertion 3: When a registered authority confirms a violation report, that
    ↪  confirmation is consistently reported into the server.
assert ConsistentConfirmation {
        all ssae, ssae': SafeStreetsAE, sss: SafeStreetsServer, vr:
            ↪ ViolationReport |
                vr in ssae.violationsReported and
                ConfirmViolationReport[ssae, ssae', sss, vr]
                implies
                (
                vr not in ssae'.violationsReported and
                        (
                        one vrs: SafeStreetsServer.violationReports |
                                        vrs = vr and vrs.status = Confirmed
                        )
                )
}

pred RefuseViolationReport[ssae, ssae': SafeStreetsAE, sss, sss':
    ↪ SafeStreetsServer, vr: ViolationReport] {
        vr in ssae.violationsReported//and vr in sss, but this is guaranteed
            ↪ by fact AllReportsInAEAlsoInServer
        implies
        (
                ssae'.violationsReported = ssae.violationsReported - vr and
                sss'.violationReports = sss.violationReports - vr
        )
}

// Assertion 4: A violation report rejected by a registered authority is
    ↪ removed from SafeStreets database.
assert ConsistentRejection {
        all ssae, ssae': SafeStreetsAE, sss, sss': SafeStreetsServer, vr:
            ↪ ViolationReport |
        vr in ssae.violationsReported and
        RefuseViolationReport[ssae, ssae', sss, sss', vr]
        implies
        (
                ssae'.violationsReported = ssae.violationsReported - vr and
                sss'.violationReports = sss.violationReports - vr
        )
}

fact AllAccidentsReportedByAnAuthority {
        all a: Accident |
                one ssae: SafeStreetsAE |
                        a in ssae.accidentsToReport
```

```
}

pred ReportAccident[ssae, ssae': SafeStreetsAE, sss, sss': SafeStreetsServer,
    ↪  a: Accident] {
        a in ssae.accidentsToReport
        implies
        ssae'.accidentsToReport = ssae.accidentsToReport - a and
        sss'.accidents = sss.accidents + a
}

// Assertion 5: Every accident reported by a registered authority is
    ↪ consistently saved into the SafeStreets database.
assert ConsistentAccidentReport {
        all ssae, ssae': SafeStreetsAE, sss, sss': SafeStreetsServer, a:
            ↪ Accident |
                a in ssae.accidentsToReport and
                ReportAccident[ssae, ssae', sss, sss', a]
                implies
                ssae'.accidentsToReport = ssae.accidentsToReport - a and
                sss'.accidents = sss.accidents + a
}


// Checks on assertions.
check NoIncompleteViolationReports for 5
check NoTimedOutViolationReports for 5
check ConsistentRejection for 5
check ConsistentConfirmation for 5
check ConsistentAccidentReport for 5

// Generation of worlds.
pred showWorld1 {}
run showWorld1 {} for 2 but 0 Authority, 0 SafeStreetsAE, 0 SafeStreetsServer
    ↪ , exactly 2 SafeStreetsApp, exactly 2 ViolationReportRequest

pred showWorld2 {}
run showWorld2 for 3 but 0 SafeStreetsApp, 0 User, 0 Accident, 0
    ↪ ViolationReportRequest, exactly 2 SafeStreetsAE, exactly 1
    ↪ SafeStreetsServer, exactly 2 ViolationReport

pred showWorld3 {}
run showWorld3 for 3 but  0 SafeStreetsApp, 0 User, 0 ViolationReport, 0
    ↪ ViolationReportRequest, exactly 2 SafeStreetsAE, exactly 2 Accident,
    ↪ exactly 1 SafeStreetsServer
```

4.3 Checks on assertions with the Alloy Analyser

Executing "Check NoIncompleteViolationReports for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
220770 vars. 730 primary vars. 322312 clauses. 4050ms.
No counterexample found. Assertion may be valid. 214ms.

Executing "Check NoTimedOutViolationReports for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
220539 vars. 725 primary vars. 321974 clauses. 3931ms.
No counterexample found. Assertion may be valid. 61ms.

Executing "Check ConsistentRejection for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
220370 vars. 735 primary vars. 321672 clauses. 3902ms.
No counterexample found. Assertion may be valid. 16ms.

Executing "Check ConsistentConfirmation for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
220292 vars. 730 primary vars. 321595 clauses. 4590ms.
No counterexample found. Assertion may be valid. 39ms.

Executing "Check ConsistentAccidentReport for 5"
Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
220370 vars. 735 primary vars. 321672 clauses. 4077ms.
No counterexample found. Assertion may be valid. 19ms.

Figure 4.16: These are the results of the evaluation of the assertions defined in Alloy.

4.4 Worlds generated in Alloy

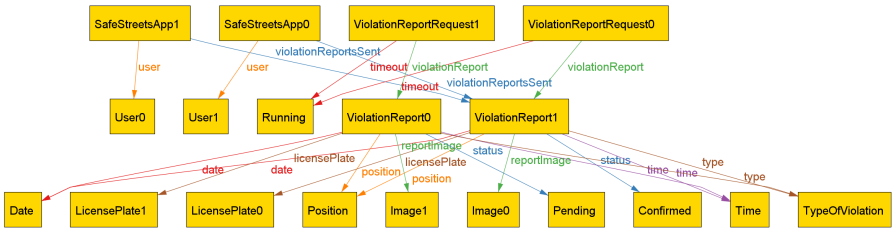


Figure 4.17: This world shows the relation between the report of a violation and SafeStreets App.

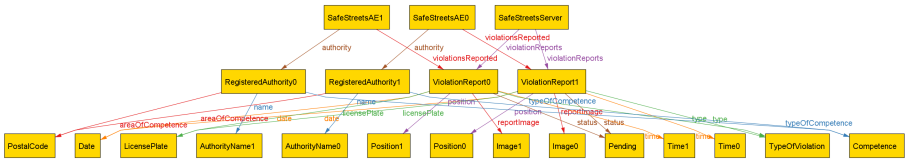


Figure 4.18: This world shows the relation between the SafeStreets server and SafeStreets AE's in violation report evaluation.

Figure 4.19: This world shows the relation between the SafeStreets server and SafeStreets AE's in reporting accidents.

5 Effort spent

Team Work	
Task	**Hours**
Understanding the problem	2
Brainstorming	4
Software system attributes	8
Total	14

Table 3: Time spent by all team members

Individual Work					
Nicolò Sala		**Sebastiano Quacquarelli**		**Simone Ricchiuti**	
Task	**Hours**	**Task**	**Hours**	**Task**	**Hours**
Constraints	2	Definitions	1	Introduction	4
Definitions	1	User interfaces.	5	Product functions	4
UC description	3	Domain model	4	User characteristics	0.5
Performance req.	1	UC descr./diagrams	4	Functional req	2
Design constraints	1	Assumptions	2	UC description	3
Alloy	9	Sequence diagrams	3	Activity Diagrams	2
Total	17	**Total**	19	**Total**	15.5

Table 4: Time spent by each team member

6 References

- "2019-2020 Software Engineering 2 mandatory project: goal, schedules and rules";
- TeXstudio (`https://www.texstudio.org`) to edit the LaTeX document;
- Overleaf (`https://www.overleaf.com`) also to edit the LaTeX document;
- AlloyTools (`https://www.alloytools.org`) to learn how to use Alloy and to download Alloy 5;
- Visual Paradigm CE (`https://www.visual-paradigm.com/`) to create UML diagrams.
- Mockplus Free (`https://www.mockplus.com/`) to design prototypes of user interface.