SafeStreets - RASD

version 1.0

Frangi Alberto, Fucci Tiziano

A.Y. 2019/2020

Table of contents

Chapter 1

Introduction

1.1 Purpose

SafeStreets is a crowd-sourced application that intends to allow users to notify authorities when traffic violations occur, and in particular parking violations, such as vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on. The application allows users to send pictures of violations, including their date, time, and position, to authorities.

SafeStreets stores the information provided by users, completing it with suitable meta-data. In particular, when it receives a picture, it runs an algorithm to read the license plate (one can also think of mechanisms with which the user can help with the recognition), and stores the retrieved information with the violation, including also the type of the violation (input by the user) and the name of the street where the violation occurred (which can be retrieved from the geographical position of the violation).

The application allows both users and authorities to mine the information that has been received, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations.

Moreover, if the municipality offers a service that allows users to retrieve the information about the accidents that occur on the territory of the municipality, SafeStrees can cross this information with its own data to identify potentially unsafe areas, and suggest possible interventions (e.g., add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

1.2 Scope

1.2.1 General description

Safestreet is an application to be used both from civilians (users) and authorities, in order to help the latter and reduce traffic violations. Registered authorities can automatically receive reports made by users, so the service acts as an intermediary. The next paragraph gives a more formal description of which is the purpose of the application.

1.2.2 Goals

The following list describes the goals from the S2B perspective:

User

- **G1**: The application must allow the users to register entering an e-mail address and a password.
- **G2**: The application must allow the users to notify traffic violations, providing the type of violation and a picture of the vehicle.
- **G3**: The application must be able to show to the users the streets and the vehicles with the highest frequency of violations.
- **G4**: The application must allow the users to see all of their reports and their status.
- **G5**: The application must notify the users when one of their reports is evaluated.

Authority

- **G6**: The application must allow the authorities to register providing a valid identification number and a valid password.
- **G7**: The application must allow the authorities to retrieve and evaluate the available reports.
- **G8**: The application must be able to identify potentially unsafe areas and suggest possible interventions.

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- **User**: a civilian customer that can use the application to:
  - notify authorities of some violation;
  - check which are the most dangerous (i.e. with the most violations) streets;

  in this document, "user", "citizen" and "civiian" are completely equivalent, where not specified.
- **Authority**: a member of the local police who has access to reports made by users. The authorities evaluate the reports sent by the user to determine if the violation stands.
- **Report**: a message consisting of:
  - a picture showing the car in order to show the occurring violation;
  - date and time of the picture;
  - GPS position of the place where the violation occurred;
  - the street where the violation occurred (automatically retrieved from the geographical position);
  - the type of the violation (input by the user)
- **Available**: a report is available for an authority if its position is within the municipality assigned to the authority.
- **Violation**: a situation that, according to the user who sent the report, is a violation of the traffic laws.
- **Intervention**: a brief text suggesting a possible solution in order to improve safety and discourage future violations.

1.3.2 Acronyms

- **API**: *Application Programming Interface.*
- **GPS**: *Global Positioning System.*
- **UI**: *User Interface.*
- **S2B**: *Software-to-be.*

1.4 Reference documents

The main reference document is the "SafeStreets Mandatory Project Assignment" specification document. The complete list of references is provided in chapter 6.

1.5 Document structure

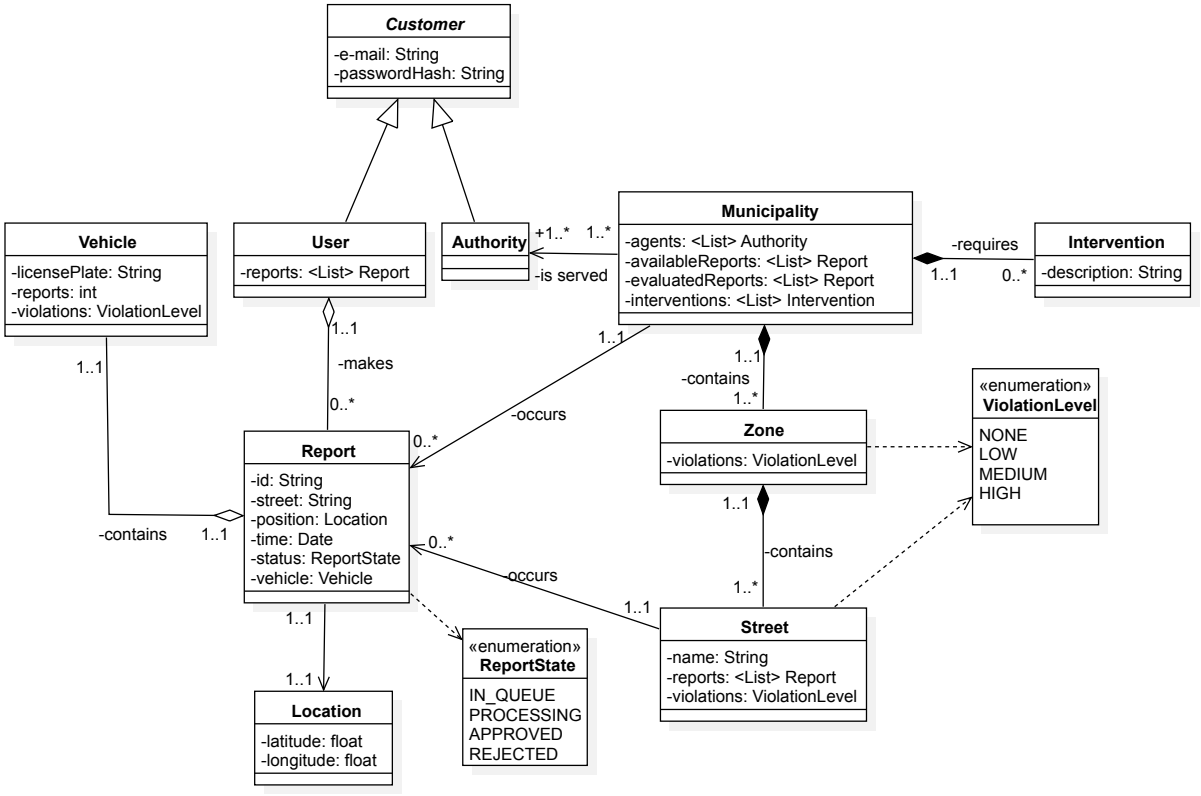
- **Chapter 1** is an introduction: it presents the document and the S2B with its scope and goals. It also helps the reader to understand, by giving the necessary definitions and explaining acronyms and abbreviations;
- **Chapter 2** is a summary description of all the project. It gives a general idea of how the application works and shows the assumpions made in the development;
- **Chapter 3** presents the requirements to be guaranteed in order to achieve every goal of the project, analyzing every use case. It shows the constraints and the attributes of the system in terms of availability, mantainability, secturity etc.;
- **Chapter 4** contains the Alloy model of critical aspects that require special attention. Here it is shown how the project has been modeled and a proof of the model consistency is provided. Moreover, examples of some of the generated world are shown;
- **Chapter 5** is meant to show how work was divided between the two components of the group and how much time was spent;
- **Chapter 6** lists the reference documents and the tools used to develop this document.

Chapter 2

Overall description

2.1 Product perspective

SafeStreets wants to be a mediator between authorities and citizens, allowing the last ones to report traffic violations, see which zone has the highest number of violations, see the recap of all the previous reports and their state (seen, approved, rejected, in queue). In order to be able to report various violations a user has firstly to be in a territory covered by any authority who has signed up to SafeStreets, and has to sign up providing his e-mail address, unique for each user. Any authority has to verify its identity using the institutional email address. To avoid spamming of requests, pending reports will be put in a priority queue based on a reliability index associated to each user and hidden to both side of communication (it is an information used only by the system for the priority queue). If the authorities evaluates a violation, the user who has reported it will receive a notification by SafeStreets, to inform him of the success (or failure) of his report.

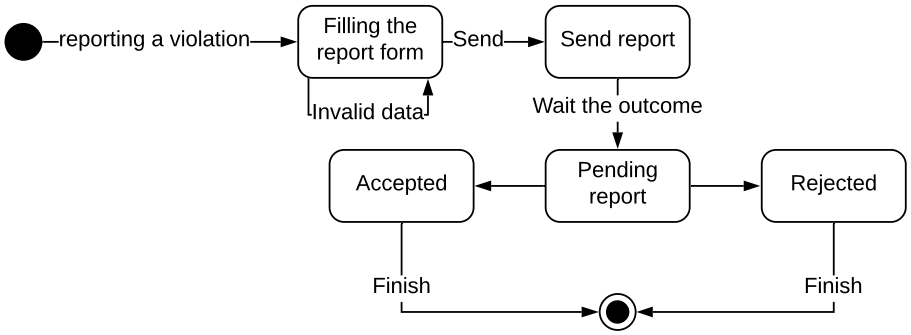


(a) UML model

This figure represents a draft of how the system's UML model will be. Since it is only the model part, the represented classes are only the few essential to save correctly all the data necessary for the analysis. In order to save up space it has been avoided to write every single getter and setter or attribute. A generic Customer has been modeled with an abstract class and both authorities and user inherit it. Report is a combination of various classes, it contains a unique id, the Street (obtained by the GPS position), the time (obtained by the system when it receives the report), the vehicle license's plate and the type of violation, it also has an enum attribute to represent the status of the report. Both "Street" and "Zone" (which is a set of streets) contain a ViolationLevel, that is calculated by the system and based on the number of reports in that particular street/zone. An Intervention is a suggestion made by the system to improve security, it is chosen by a recommender system that applies some simple data mining's algorithms in order to find the best improvement for each street, based on the most frequent violations or accidents. Municipality contains a list of possible interventions, all the authorities that operate in its territory and all reports (both evaluated

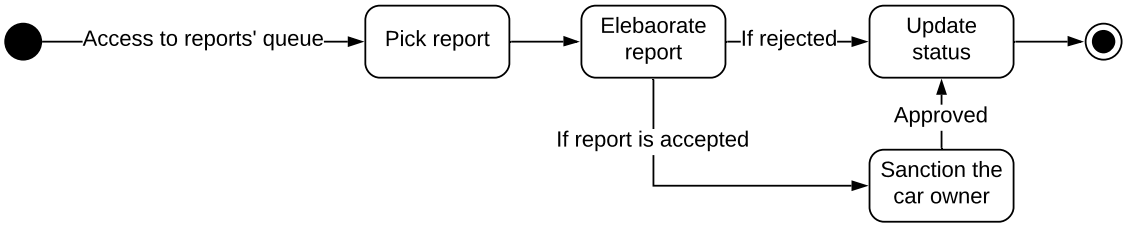
and to evaluate).

Now some critical aspects of the application will be analyzed, modeling their behaviors and showing the evolution over time of their states through appropriate state diagrams, which are reported below.



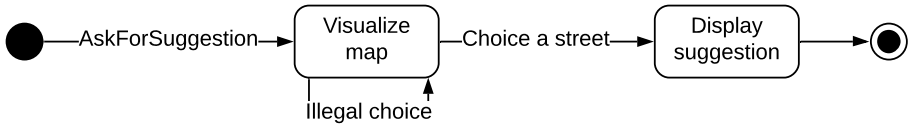
(a) State diagram on the behavior of reports for users

This diagrams represent the behavior of the reporting system (for the citizen). The concept is pretty straight forward: the system does not allow to send reports with missing information or with wrong ones.



(a) State diagram on how reports work for authorities

This diagram is inherent to the reports verification process, it shows the evolution of the states necessary to handle a report, from when the authorities take in charge a user's report to the eventual sanction to the car owner. A sanction will result in the car being noticed, incrementing its reports' number. In the long run, this can take to increasing the violation level of that vehicle.



(a) State diagram on how reports work for authorities

This last diagram represents the last major functionality offered by SafeStreet. It consists in a system that offers suggestions on how improve safety on particular streets, based on the occurrences of car accidents and street violations data that the authorities choose to share with SafeStreets.

2.2 Product functions

In the following section the most important product functions of the system are reported. Safe Streets offers to its users the possibility to report traffic violations, check the status of the previous reports and see a map that indicates the areas with the highest number of reports verified.

2.2.1 Report

To report a violation a user has to take a picture of the possible violation with the plate visible, choose what type of violation it is from various possible categories and send the position.

2.2.2 Status

This functionality allows user to see a recap of all the reports they have made. The information that will be displayed are: date and time, place, type of violation and status. The last one can be:

- **In queue:** in this status the report is waiting to be analyzed by an authority. The report's priority is based on the user who created it. All the users, at the moment of the registration, have a standard value, that can increase or decrease basing on how Safe Street is used. Some important factors are, for example, how many reports a user sent has been rejected, how many reports a user sent can decrease the priority, but a wisely use of the system can increase it, like a streak of approved reports.
- **Processing:** in this status the authorities has seen the report and are considering the violation and if it is a real violation.
- **Approved:** the report has been seen by authorities and the violation has been recognized and sanctioned.
- **Rejected:** the report has been seen by authorities and the violation hasn't been recognized.

2.2.3 Map

This function allows the user to open the map and see the "level of violations" of the different areas. These "levels" are based on how many reports have been approved in that area: higher the level, higher the number of violation. There are 4 levels, which are indicated with different color, from no colour (level 0) to red (level 3).

2.3 User characteristics

The actors of the application are the following:

- **User:** a civilian who has registered to SafeStreets. He doesn't need any particular skill to use the system, he is just supposed to have a smartphone (or tablet) with a functioning camera, an internet connection and a phone that can send its current position. To make a report it is sufficient to take a picutre of the license plate, the violation's type and the system retrieves autonomously the current position of the user.
- **Authority:** An agent of the local police of a certain municipality exploiting SafeStreets, that has access to all the reports made in his municipality. The authority can visualize the available reports and evaluate them, that is telling if the violation reported by the user stands or it is not a proper violation. After seeing a report, the authority changes its status to Approved or Rejected. By activateing an optional service, authorities can check the streets with the highest number of accidents and receive suggestions (chosen by the system) to improve safety.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumption

The following are the assumptions made to simplify and clarify some situation that could uselessly complicate the development of the application. Notice that some trivial assumptions are not included, in order to avoid their repetition in every situation (e.g. "The devices always work without failure").

- **D1**: The users always provide a picture that allows both the algorithm to read the license plate and the authority to identify the violation.

- **D2**: Every user is provided with a device capable to share the exact GPS position at any moment.
- **D3**: The name of the street where a violation occured is retrieved from the GPS position.
- **D4**: Authorities never make mistakes in evaluating a report.
- **D5**: Reports are evaluated only one time.
- **D6**: The user sends the report staying in the same place of the violation.
- **D7**: The picture of the license plate is taken at the moment and shows the right car.

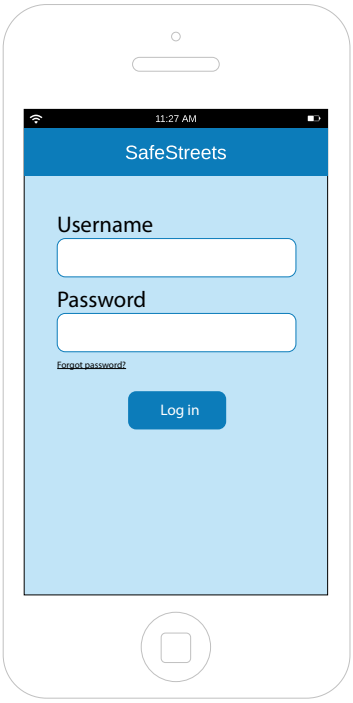
Chapter 3

Specific Requirments

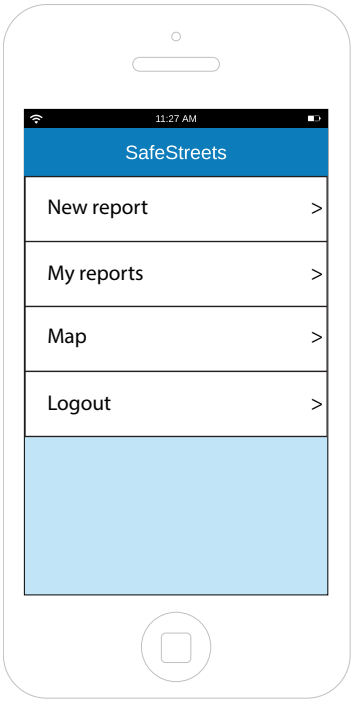
3.1 External interface requirments

3.1.1 User interfaces

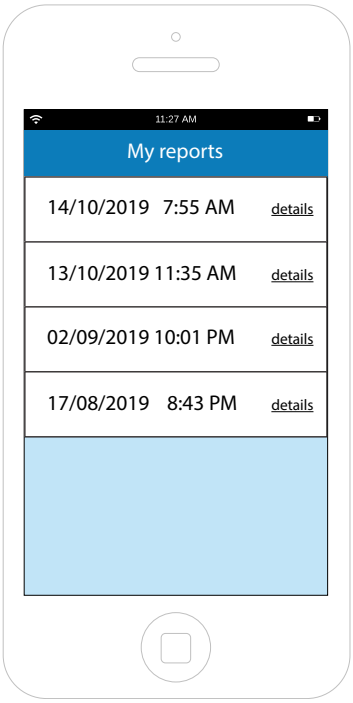
In this section some screenshots from the user interface are shown.



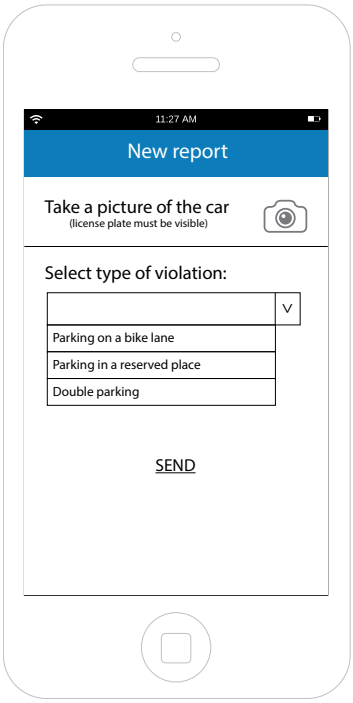
(a) User - login



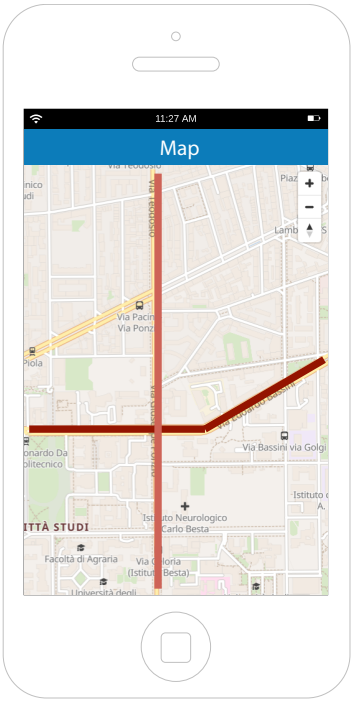
(b) User - home



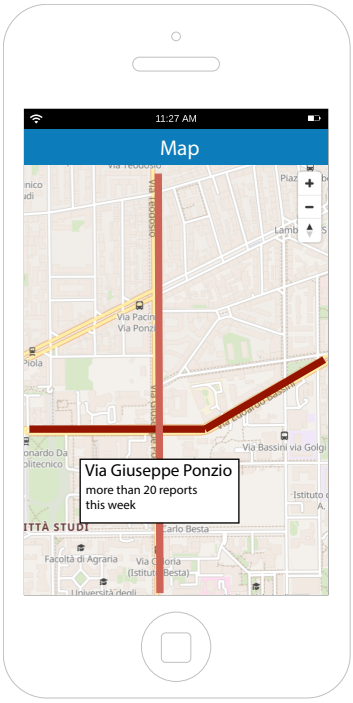
(c) User - list of sent reports



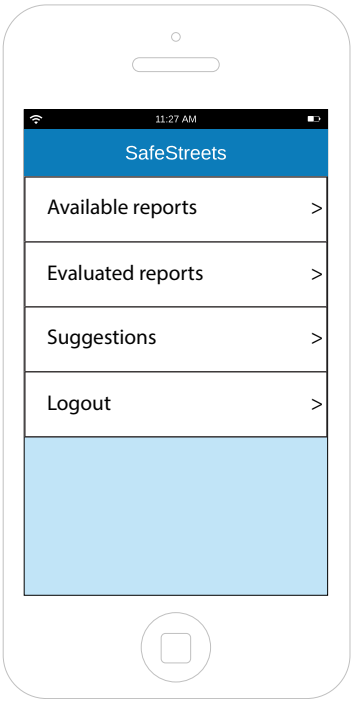
(d) User - new report



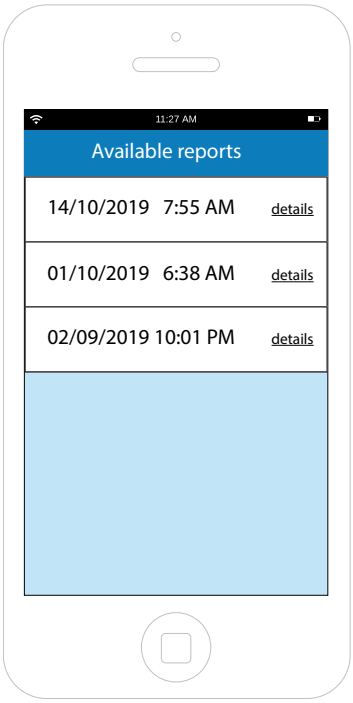
(e) User - map



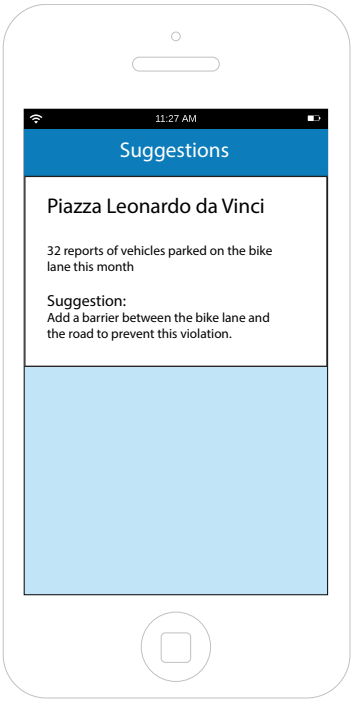
(f) User - map detail



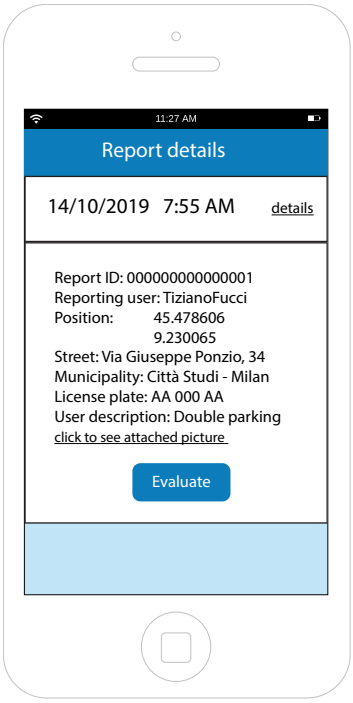
(g) Authority - home



(h) Authority - available reports



(i) Authority - suggestion



(j) Authority - report details

3.1.2 Hardware interfaces

The system has no hardware interfaces.

3.1.3 Software interfaces

The system has no software interfaces.

3.1.4 Communication interfaces

The system has no communication interfaces.

3.2 Functional Requirements

This section presents the requirements to be satisfied to reach each goal, in addition to the domain assumptions that must stand.

3.2.1 User

G1: The application must allow the users to register entering an e-mail address and a password.

- [R1]: Registration must be allowed only entering an e-mail address that is not already associated to an existing SafeStreets account.
- [R2]: Registration must be allowed only entering a password that satisfies the safety conditions.
- [R8]: The system must store the hash of every password, using a safe cryptographic hash function.

G2: The application must allow the users to notify traffic violations, providing the type of violation and a picture of the vehicle.

- [D1]: The users always provide a picture that allows both the algorithm to read the license plate and the authority to identify the violation.
- [D2]: Every user is provided with a device capable to share the exact GPS position at any moment.
- [D3]: The name of the street where a violation occured is retrieved from the GPS position.

- [D5]: Date, time and street of the violation are automatically inferred by the application.
- [D6]: The user sends the report staying in the same place of the violation attached to the report.
- [D7]: The picture of the license plate is taken at the moment and shows the right car.
- [R1]: The system must check if the location of the report belongs to some municipality exploiting SafeStreets.
- [R2]: The system must add the right date, time and street of the violation to the data provided by the user.
- [R3]: The system must correctly read the license plate given a picture attached to a report.

G3: The application must be able to show to the users the streets and the vehicles with the highest frequency of violations.

- [D3]: The name of the street where a violation occured is retrieved from the GPS position.
- [D4]: Authorities never make mistakes in evaluating a report.
- [D5]: Reports are evaluated only one time.
- [R2]: The system must add the right date, time and street of the violation to the data provided by the user.
- [R4]: The system must exploit Google Maps APIs to show to the user the map of the violations.
- [R5]: The system must show the right colors on the map, given the number of approved reports for each street.
- [R6]: When a license plate is recognized, the system must show the number of approved reports for that car.
- [R9]: If the same violation is reported twice, it counts only one time on the map.

G4: The application must allow the user to see all of his reports and their status.

- [R2]: The system must add the right date, time and street of the violation to the data provided by the user.
- [R7]: The system must assign to each report a unique identification number.

G5: The system must notify the user when one of his reports is evaluated.

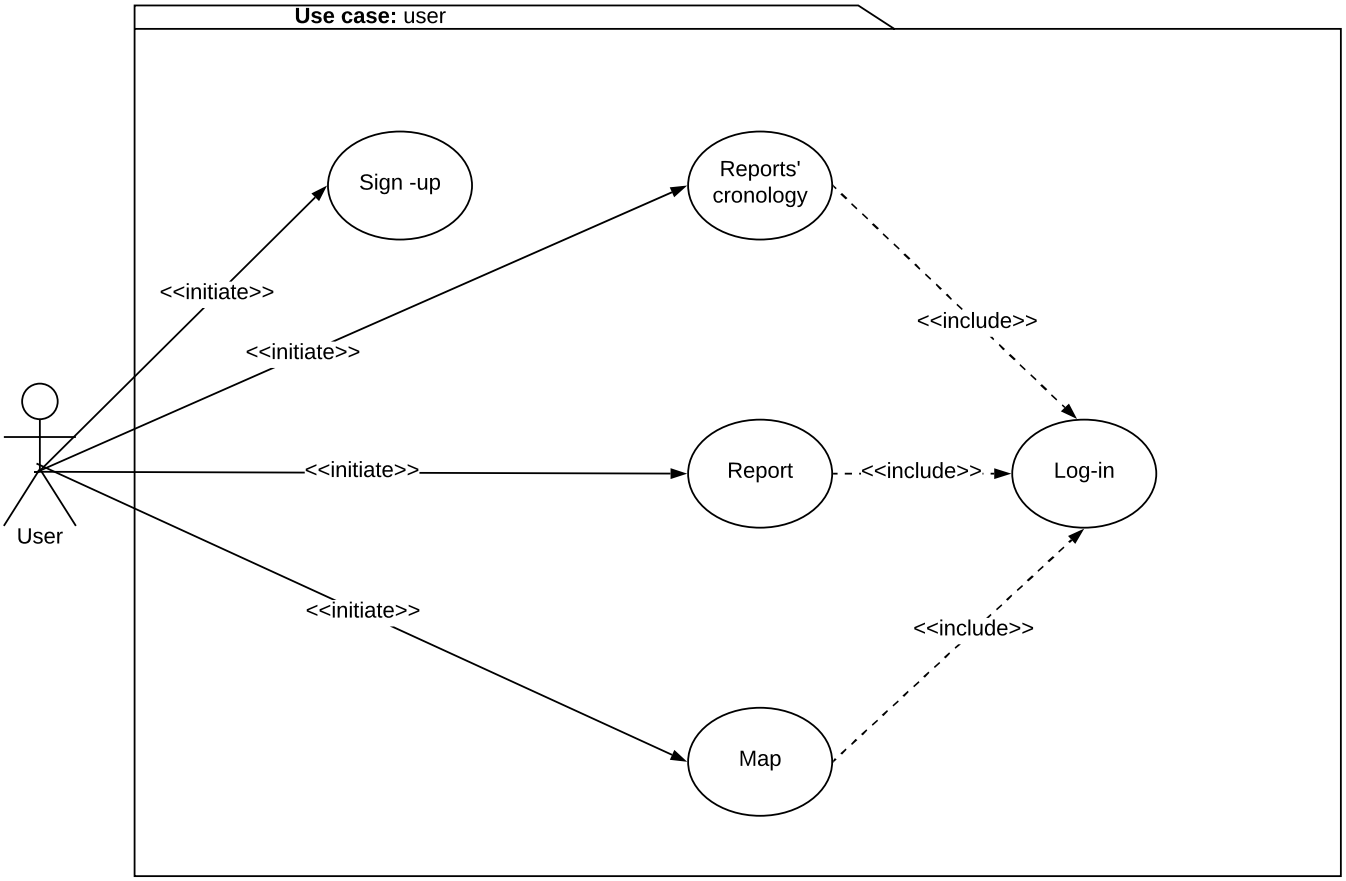
- [D3]: The name of the street where a violation occured is retrieved from the GPS position.
- [D4]: Authorities never make mistakes in evaluating a report.
- [D5]: Reports are evaluated only one time.
- [R2]: The system must add the right date, time and street of the violation to the data provided by the user.
- [R4]: The system must exploit Google Maps APIs to show to the user the map of the violations.
- [R5]: The system must show the right colors on the map, given the number of approved reports for each street.
- [R6]: When a license plate is recognized, the system must show the number of approved reports for that car.
- [R9]: If the same violation is reported twice, it counts only one time on the map.

Scenarios

Scenario 1 Mario, an elderly person who has a reserved parking lot, always has to spend time looking for a free car park, because someone frequently steals the one reserved to him without being sanctioned by the authorities. With the help of SafeStreets, Mario can report the parking thief to authorities and verify that he gets sanctioned.

Scenario 2 Luigi uses very frequently SafeStreets, but he is not sure if what he is doing is considered by authorities. Thanks to SafeStreets' reports history, now he is certain about which reports has been useful and which ones were wrong.

Scenario 3 Giuseppe is collecting some data about traffic and traffic violations for a University project. To retrieve more information he uses the map offered by SafeStreets, in order to learn which zones are more affected by traffic violations.



(a) State diagram on the behavior of reports for citizen

Name	Sign up
Actor	User, Authorities
Entry condition	The User want to use for the first time SafeStreet and open the application on his device
Events Flows	1. The user clicks on the "Sign up" option 2. The user fills all the mandatory fields 3. The user agrees with the SafeStreet's privacy policy 4. The user clicks on the confirmation option 5. The system proceeds to verify user's data and to save them
Exit Conditions	The user is registered and the system has saved his data
Exceptions	• The user has already signed up • Username has been already used. In this case the System notify the user and suggests some free Usernames. • The data were not correct. In this case the system ask to the user to check and correct eventual mistakes. • Not all the mandatory fields were filled. In this case the system warns the user and notify him which fields are empty

Table 3.1: Sing up

Name	Log in
Actor	User, Authorities
Entry condition	The User open the application because he want to access the service he already signed up
Events Flows	1. The user clicks on the “Log in” option 2. The user fills all the Username and password fields 3. The user clicks the confirmation option 4. The system proceeds to verify user’s data 5. The user has free access to the service now
Exit Conditions	The user is logged
Exceptions	• Password wrong • Username wrong, in both case the system will ask to check and correct the log-in information

Table 3.2: Log in

Name	Report
Actor	User
Entry condition	The User wants to report a traffic violation, so he opens the application and logs in, then he chooses the “report” option
Events Flows	1. The User clicks on “Report” option 2. The User takes a photo of the violation and the license plate of vehicle 3. The system tries to read the plate from the photo 4. The user checks that the system recognized correctly the plate 5. The user inserts the type of violation. 6. The user clicks on “send report”. 7. The system retrieves the device street, date and time to fill the last fields of the report
Exit Conditions	The user reported the violation
Exceptions	• The vehicle has been already reported. • The system cannot obtain the user’s position. • Not all the fields has been filled. • There is no internet connection available.

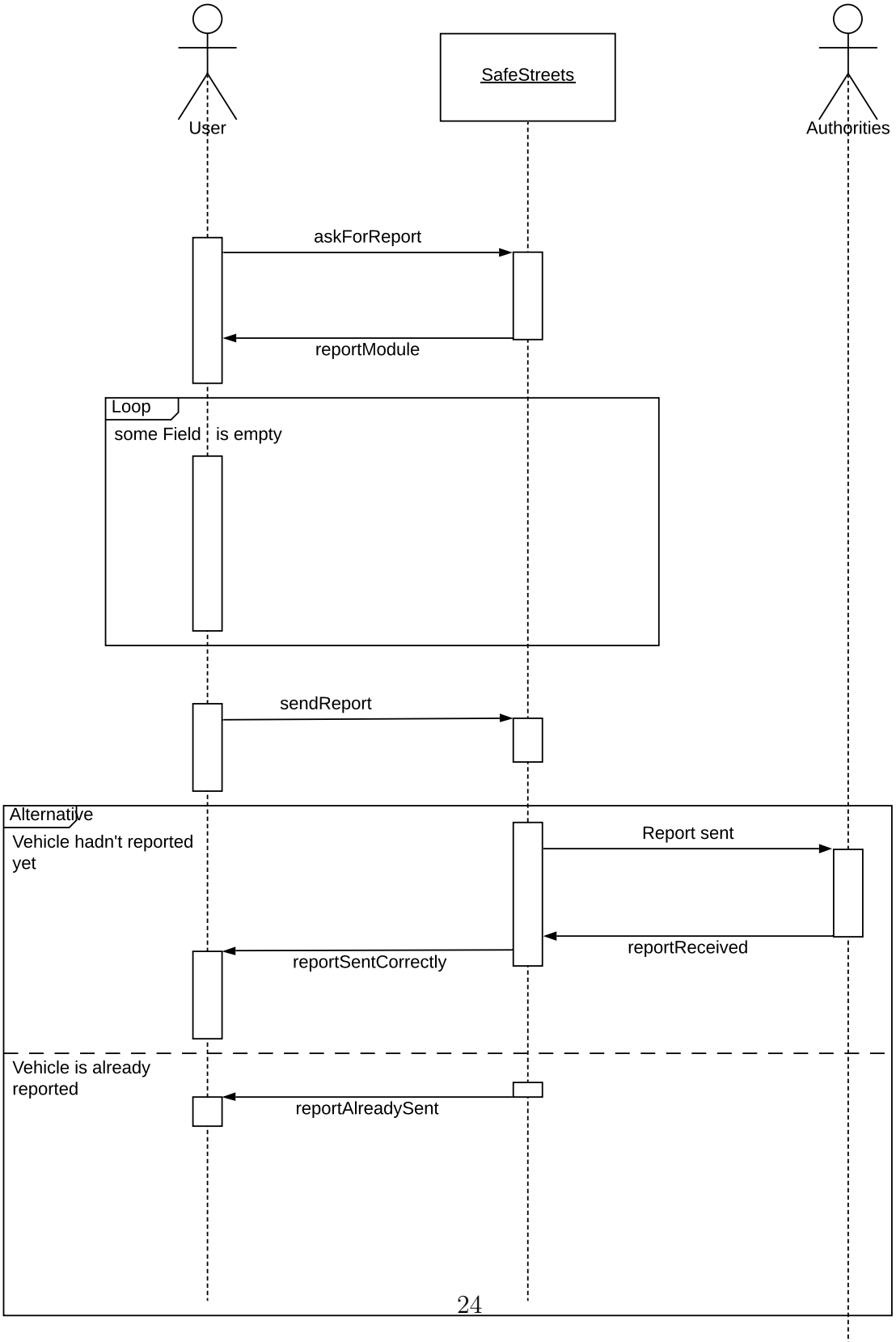
Table 3.3: Report

Name	Chronology
Actor	User
Entry condition	The User open and log in the SafeStreet application
Events Flows	1. The user clicks on “History” option 2. The system searches and sends a recap of all the user’s reports 3. The user can check all his reports
Exit Conditions	The user clicks on “Exit” option
Exceptions	• The user has no reports. • There is no internet connection available.

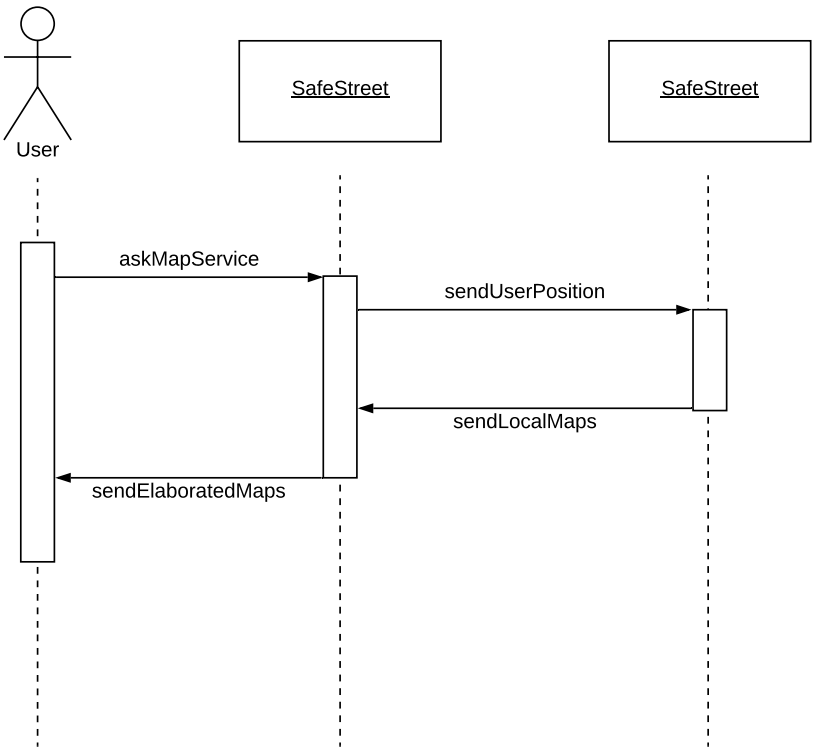
Table 3.4: Use case Chronology

Name	Visualize Map
Actor	User
Entry condition	The User opens SafeStreets, logs in and chooses the “Visualize Map” option
Events Flows	1. The user clicks on “Visualize Map” 2. The system retrieves the device position and obtains information about the near streets’ reports number to calculate their “level” and displays it correctly 3. The user can navigate in the map seeing each street and their level
Exit Conditions	The user clicks on exit button
Exceptions	• There is no internet connection • The GPS system is not functioning • The app doesn’t have permits to access to device’s position

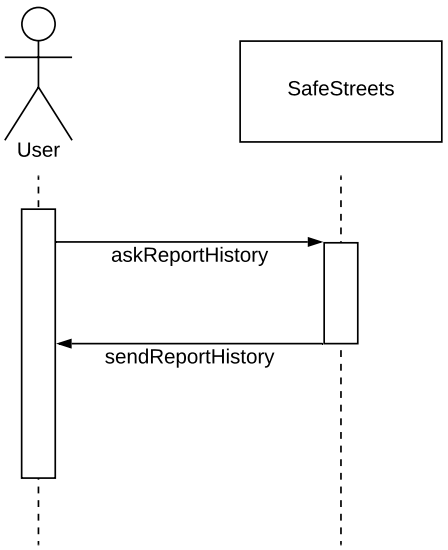
Table 3.5: Use case on Visualize Map



(a) Sequence diagram on the behavior of reports for citizen



(a) Sequence diagram about Map option



(a) Reports History

3.2.2 Authority

G6: The application must allow the authorities to register providing a valid identification number and a valid password.

- [R1]: Registration must be allowed only entering an e-mail address that is not already associated to an existing SafeStreets account.
- [R2]: Registration must be allowed only entering a password that satisfies the safety conditions.
- [R8]: The system must store the hash of every password, using a safe cryptographic hash function.

G7: The application must allow authorities to retrieve and evaluate the available reports.

- [D3]: The name of the street where a violation occured is retrieved from the GPS position.
- [D5]: Reports are evaluated only one time.
- [R1]: The system must check if the location of the report belongs to some municipality exploiting SafeStreets.
- [R2]: The system must add the right date, time and street of the violation to the data provided by the user.
- [R7]: The system must assign to each report a unique identification number.
- [R10:] The system must guarantee that each municipality is covered by at least one authority.
- [R12]: The system must notify the authorities when a new report is available.

G8: The application must be able to identify potentially unsafe areas and suggest possible interventions.

- [D3]: Every user is provided with a device capable to share the exact GPS position at any moment.
- [D4]: Authorities never make mistakes in evaluating a report.

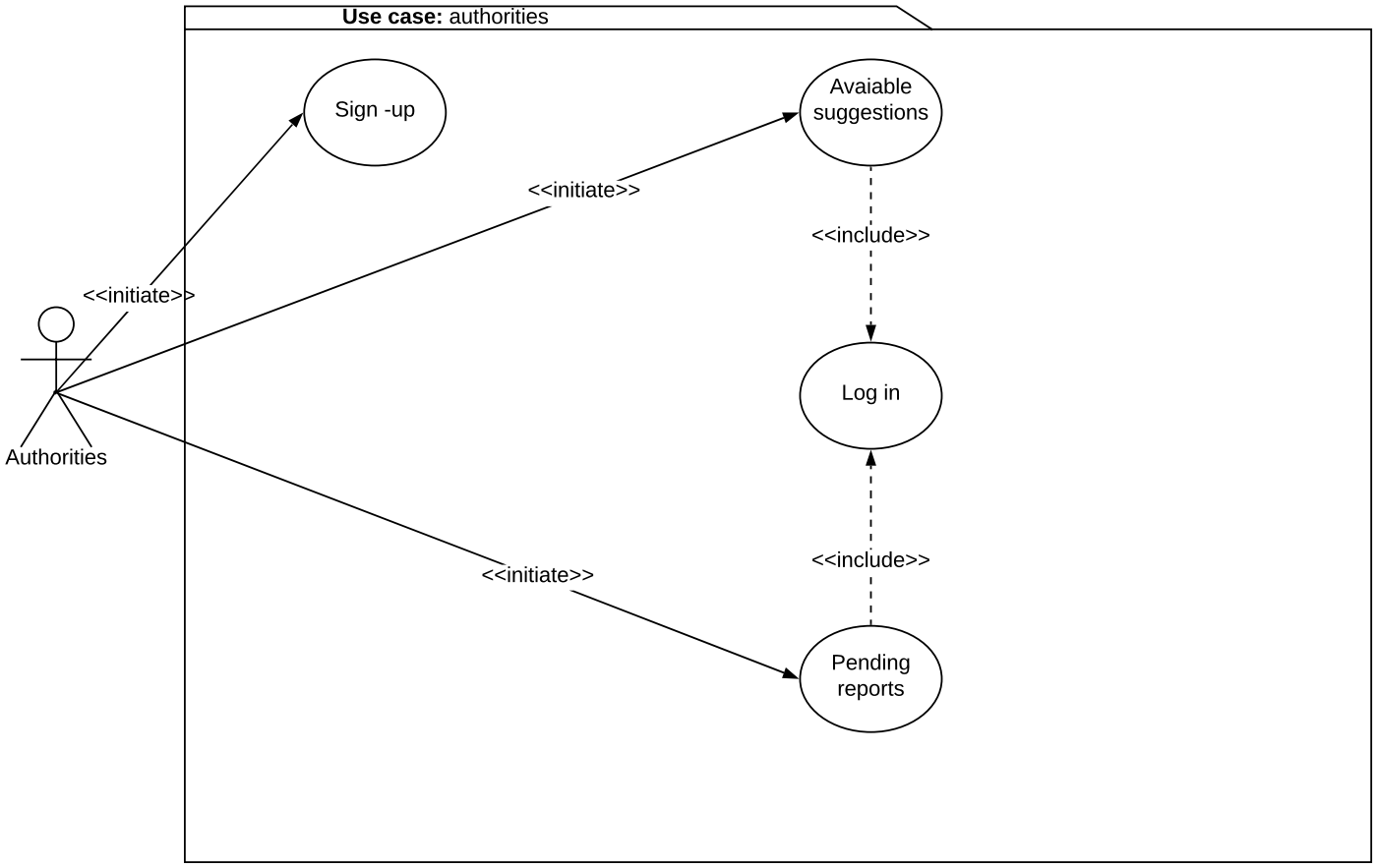
- [D5]: Date, time and street of the violation are automatically inferred by the application.
- [D6]: The user sends the report staying in the same place of the violation.
- [R9]: If the same violation is reported twice, it counts only one time on the map.
- [R10:] The system must guarantee that each municipality is covered by at least one authority.
- [R11]: The system must include a simple recommender system that, according on the number of each type of violation, suggests a possible intervention to the authorities of a municipality.

Scenarios

Scenario 1 Eddie Edison, an officer had noticed a vertiginous increment of traffic violations, so to increase the control on the streets decide to join SafeStreets

Scenario 2 Major Clancy Winchester had noticed that even with various tentative to reduce the car accidents he didn't succeed, so he decide to try to subscribe to and advanced option offered by SafeStreets trying to receive new suggestion to improve the security

In this paragraph will be illustrated all the use cases diagram, use cases and sequence diagrams, the "sign up" and "log in" are not reported because are the same as above.



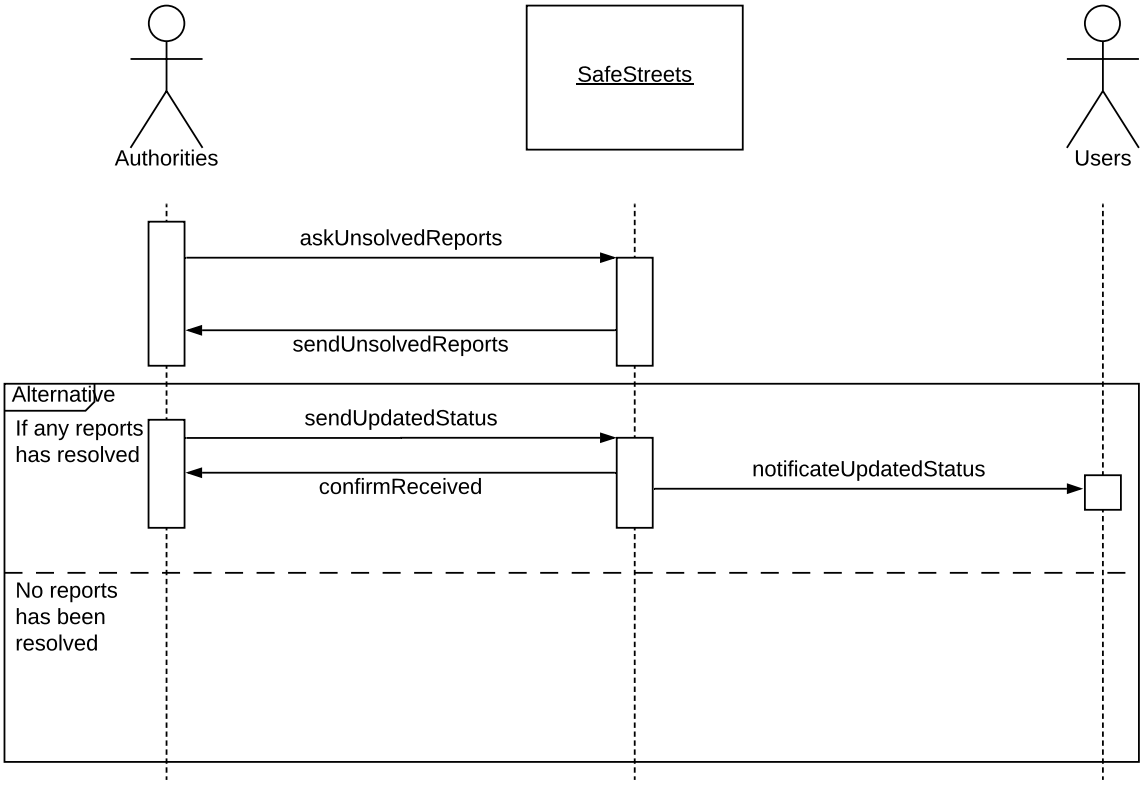
(a) Sequence diagram on the behavior of reports for users

Name	Evaluate report
Actor	Authority
Entry condition	The authority is logged in
Events Flows	1. The authority clicks on the "Evaluate reports" options 2. The system retrieves all the pending reports 3. The authority chooses if accept or reject each single report based on the information contained in the report
Exit Conditions	The user clicks on exit button
Exceptions	/

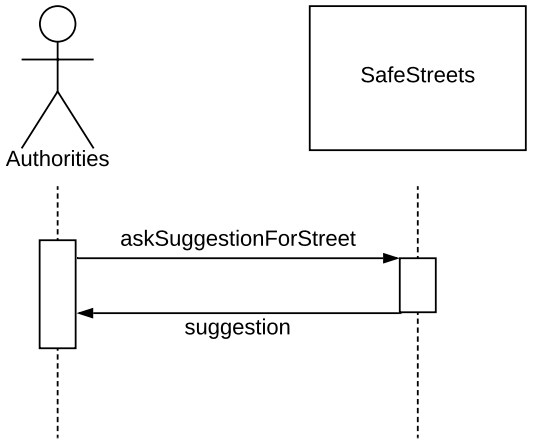
Table 3.6: Evaluate report

Name	Suggestion
Actor	Authority
Entry condition	The authority is logged in
Events Flows	1. The authority clicks on the "Suggestion" option 2. He chooses one Street under his control 3. The system based on the street and the accidents gives a recommendation
Exit Conditions	The authority clicks on exit button
Exceptions	The authority hasn't activated this optional functionality

Table 3.7: Evaluate report



(a) Sequence diagram on the behavior of reports for authorities



(a) Sequence diagram on the behavior of reports for authorities

3.2.3 World and Machine

In this subsection the World and Machine matrix is presented.

Phenomenon	Shared	Who controls it
Log-in	N	M
Sign-up	N	M
Someone made traffic violation	N	W
Report	Y	M
Visualize Map	N	M
Color on the map	N	M
Violations number	Y	W
Report Evaluated	Y	W
Notification received	N	M
Accident Occurs	N	W
A suggestion is made	N	M

Table 3.8: World and Machine matrix

3.2.4 Traceability Matrix

The following figure shows all the various requirements and use cases linking to each other.

Requirement	Use Case
R1	Sign up & Log in Report
R2	Sign up & Log in Report
R3	Report
R4	Visualize map
R5	Visualize map
R6	Report
R7	Report
R8	Sign up &Log in
R9	Reports Visualize Map
R10	Reports Evaluate report
R11	Make suggestion
R12	Report

Table 3.9: Traceability matrix

3.3 Performance Requirments

The System has to be able to serve a large number of users and authorities, simultaneously guaranteeing quick, reactive and correct responses. It's important to underline that the accuracy of the suggestions received by the system are based on the data shared with SafeStreets and that the system does not guarantee that the proposed solution will surely solve the problem.

3.4 Design constraints

3.4.1 Standards compliance

The error of the user's position is expressed in meters or feet (it depends on the unity preferred by the authorities) and has to be less than 10 meters.

The system is designed to protect the privacy of users' data, as a matter of fact the entire project is subject to the General Data Protection Regulation (GDPR), a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA).

3.4.2 Hardware limitations

As specified in the "dependencies and constraints" section, hardware requirements are present only in relation to specific functionalities. For example, to make a report it is necessary a device with a functioning internet connection, a camera and a GPS sensor.
So all the hardware requirements are:

- any type of Internet connection;
- a working gps sensor;
- a working camera.

3.4.3 Any other costraint

The system does not have any other type of particular constraints. In order to protect users' privacy is important to remind that the data will be maintained secret (the authorities will not be able to see which user made a report) and that SafeStreets will respect the GDPR regulation.

3.5 Software System Attributes

3.5.1 Reliability

The system must be able to run continuously without any interruptions. In order to do that, it must be ensured that the system is fault tolerant, this means that some precautions will be taken to avoid the system to be offline or unavailable. One of the many will be a daily backup of all the data on another server to avoid major data losses.

3.5.2 Availability

The system must have a fault tolerant architecture, the most important availability is about the reports and is supposed to be 99.999% of the time available (otherwise a huge number of reports could be lost and the system would

never recover them). The other functionalities are supposed to be available 99.99% of the time.

3.5.3 Security

The privacy is granted to both communication sides: the user doesn't know who evaluated his report and the authorities only see a number representing the user, so they cannot see any information like name or e mail. Client-server communication is granted using a PKC. The public key is used at the beginning of the session to exchange a secret symmetric key. The communication continues encrypting the messages with the symmetric key (e.g. an AES-256 key).

3.5.4 Maintainability

The system is realised with a modular structure, so it will be very easy to improve or substitute any part of it. For example, if one day a better recommender system is designed, the introduction of the new one will not impact the rest of the system.

3.5.5 Portablity

The application can be used on any mobile device respecting the hardware constraints.

Chapter 4

Formal Analysis

4.1 Purpose

This chapter shows the analysis of the Alloy model of the proposed system. Obviously, here the focus is only on some critical aspects of the application that require a special attention. In particular, the purpose of this model is to guarantee that:

- The provided e-mail address is different for each user or authority;
- Every report has a unique identification number;
- Every municipality has at least one authority assigned;
- The authorities correctly receive notifications when a report is available;
- The users correctly receive notifications when one of their report is evaluated.

4.2 Model

```
open util/boolean

sig Email {}

sig Password {}

sig Municipality {
        agents: set Authority
} {
        #agents > 0
}

sig Id {}

abstract sig ViolationType {}
-- showing all subclasses is not the purpose of this model

abstract sig Notification {}

sig AvailableNotification extends Notification {
        report: one Report
}

sig EvaluatedNotification extends Notification {
        report: one Report,
        approved: Bool
}

sig Report {
        id: one Id,
        author: one User,
        municipality: one Municipality
}

abstract sig Customer {
        email: one Email,
        password: one Password
}

sig User extends Customer {
        -- the reports the user has sent
        reports: set Report,
        -- list of new evaluated reports
        notifications: set EvaluatedNotification
}

sig Authority extends Customer {
        -- the working place of the authority
        municipalities: set Municipality,
        -- list of new available reports
        notifications: set AvailableNotification
} {
        #municipalities > 0
}

-- every e-mail address must be associated to a user or an authority
fact {
        all mailaddr : Email | some c : Customer | mailaddr in c.email
}
```

```
-- just to simplify the alloy model
fact {
        no disj c1,c2: Customer | c1.password = c2.password
}

-- every password must be associated to a user or an authority
fact {
        all pw : Password | some c : Customer | pw in c.password
}

-- every customer must have a unique e-mail address
fact {
        no disj c1,c2: Customer | c1.email = c2.email
}

-- every report must generate an available notification
fact {
        all r : Report | one an : AvailableNotification | an.report = r
}

-- each authority must work in at least one municipality
fact {
        all a : Authority | some m : Municipality | a in m.agents
}

-- each report must have a unique identification number
fact {
        no disj r1,r2: Report | r1.id = r2.id
}

-- every id belongs to a report
fact {
        all i : Id | some r : Report | i = r.id
}

-- authorities must receive notifications of reports in their working areas
fact {
        all r : Report, a : Authority | one an : AvailableNotification | r.
            ↪ municipality in a.municipalities implies
                an in a.notifications and r = an.report
}

-- municipalities must have as agents only authorities that work there
fact {
        all m : Municipality, a : Authority | m in a.municipalities iff a in
            ↪  m.agents
}

-- every authority works in a municipality if and only if that municipality
    ↪ is in his working places
fact {
        all a : Authority, m : Municipality | a in m.agents iff m in a.
            ↪ municipalities
}

-- every user is the author of a report if and only if that report is in his
    ↪  reports list
fact {
        all u : User, r : Report | u = r.author iff r in u.reports
}
```

```
-- the user must be notified of the evaluation of his reports
fact {
        all r : Report, en : EvaluatedNotification, u: User | r in u.reports
            ↪  and en.report = r implies en in u.notifications
}

-- the other users must not be notified of the evaluation of reports from
    ↪ another user
fact {
        all r : Report, en : EvaluatedNotification, u: User | (r not in u.
            ↪ reports and en.report = r)
                implies (en not in u.notifications)
}

-- every report must not produce more than one evaluated notification
fact {
        all r : Report | lone en : EvaluatedNotification | en.report = r
}

assert EveryReportIsReceived {
        all r : Report | some a : Authority | r in a.notifications.report
}

assert EveryEvaluationIsNotified {
        all en : EvaluatedNotification | one u : User | u = en.report.author
            ↪  and en in u.notifications
}

assert EveryMunicipalityIsCovered {
        all m : Municipality | some a : Authority | a in m.agents
}

pred world1 {
        #Report = 2
        #User = 2
}

check EveryReportIsReceived for 6

check EveryMunicipalityIsCovered for 6

check EveryEvaluationIsNotified for 6

run world1 for 5
```

4.3 Results

The following is the result of the Alloy Analizer with the input shown in the previous section.

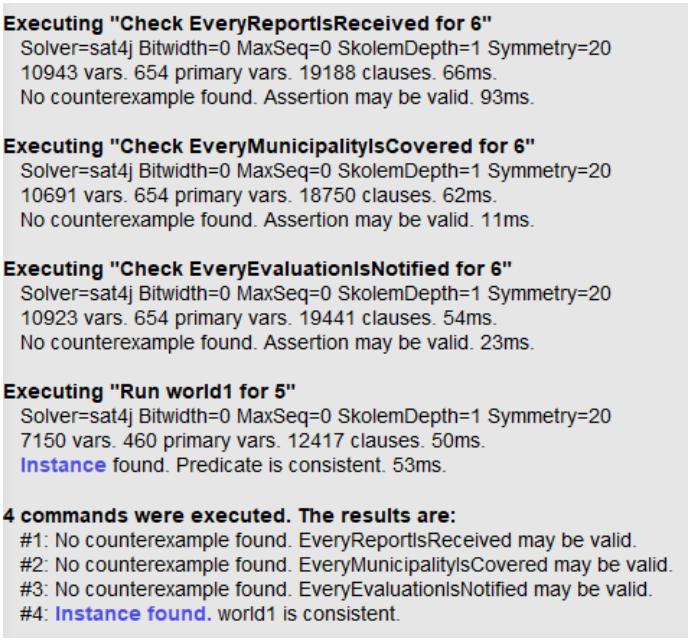


Figure 4.1: Results

The *world1* predicate is used to generate a world suitable to be read, which will be shown in the next section.

4.4 Generated world

Here is shown one of the possible worlds obtained by running *world1*.

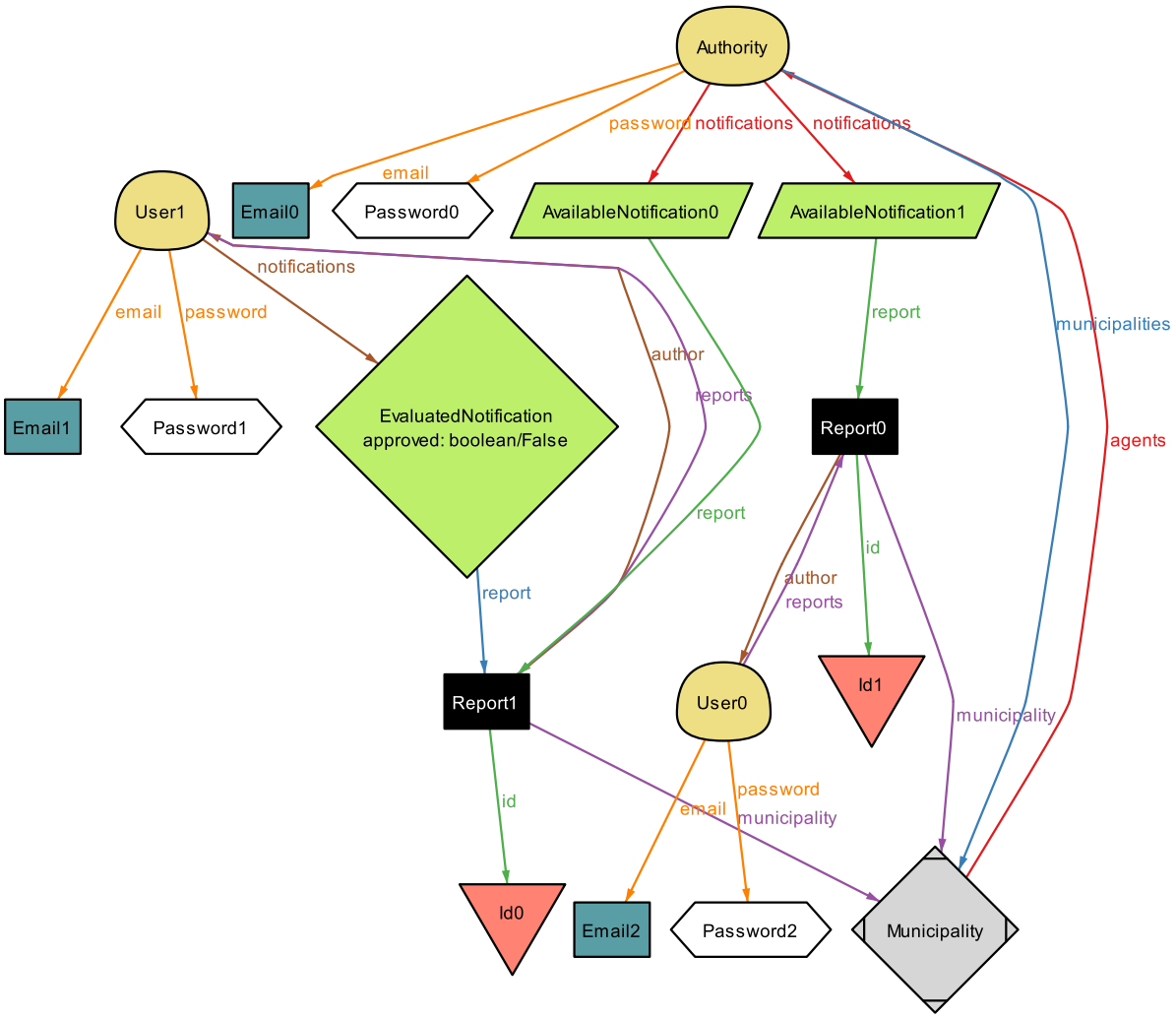


Figure 4.2: Generated world with two users, two reports

Of the two reports, only one has been evaluated and produced a notification that is correctly shown to the user. As said before, in the Alloy model every user has a different password, just to avoid arrows going all across the graph.

Chapter 5

Effort Spent

Chapter	Frangi (hours)	Fucci (hours)
Chapter 1	1	2
Chapter 2	7	4
Chapter 3	8	5
Chapter 4	2	8
Total hours:	18	19

Chapter 6

References

6.1 Bibliography

- Course slides from Software Engineering 2 - *Professor Elisabetta Di Nitto*;
- 830-1998, IEEE Recommended Practice for Software Requirements Specifications - *IEEE*;
- 29148-2011, ISO/IEC/IEEE International Standard - Systems and software engineering - Life cycle processes-Requirements engineering - *IEEE*.

6.2 Tools

- StarUML 3.1.0 - to draw and export UML diagrams;
- `lucidchart.com` - to draw, share and export state diagrams, sequence diagrams;
- Alloy Analizer 4.2 - to write and test the Alloy model of the application;
- Adobe Illustrator CC 2019 - to create mockups of the user interfeace;
- TeX Live 2019 - to write and organize this document;
- GitHub 2.23.0 - version control.