1. **INTRODUCTION**
   - A. *Purpose*: here we include the goals of the project
   - B. *Scope*: here we include an analysis of the world and of the shared phenomena
   - C. *Definitions, Acronyms, Abbreviations*
   - D. *Revision history*
   - E. *Reference Documents*
   - F. *Document Structure*
2. **OVERALL DESCRIPTION**
   - A. *Product perspective*: here we include further details on the shared phenomena and a domain model (class diagrams and statecharts)
   - B. *Product functions*: here we include the most important requirements
   - C. *User characteristics*: here we include anything that is relevant to clarify their needs
   - D. *Assumptions, dependencies and constraints*: here we include domain assumptions
3. **SPECIFIC REQUIREMENTS:** Here we include more details on all aspects in Section 2 if they can be useful for the development team.
   - A. *External Interface Requirements*
     - A.1 *User Interfaces*
     - A.2 *Hardware Interfaces*
     - A.3 *Software Interfaces*
     - A.4 *Communication Interfaces*
   - B. *Functional Requirements*: Definition of use case diagrams, use cases and associated sequence/activity diagrams, and mapping on requirements
   - C. *Performance Requirements*
   - D. *Design Constraints*
     - D.1 *Standards compliance*
     - D.2 *Hardware limitations*
     - D.3 *Any other constraint*
   - E. *Software System Attributes*
     - E.1 *Reliability*
     - E.2 *Availability*
     - E.3 *Security*
     - E.4 *Maintainability*
     - E.5 *Portability*
4. **FORMAL ANALYSIS USING ALLOY:** This section should include a brief presentation of the main objectives driving the formal modeling activity, as well as a description of the model itself, what can be proved with it, and why what is proved is important given the problem at hand. To show the soundness and correctness of the model, this section can show some worlds obtained by running it, and/or the results of the checks performed on meaningful assertions.
5. **EFFORT SPENT:** In this section you will include information about the number of hours each group member has worked for this document.
6. **REFERENCES**

1. Introduction

1.1. Purpose

1.1.1. General purpose

TODO

1.1.2. Goals

- [G1] A citizen can report a violation.
- [G2] A violation report received by the system must have enough information to be valid, i.e. has at least one picture of the violation, exactly one GPS position, exactly one timestamp, exactly one type of violation and the license plate of the vehicle.
- [G3] Users and municipality can retrieve information about violations, accidents and issued tickets in a certain area, with different levels of visibility.
- [G4] Municipality will be able to retrieve suggestions for possible interventions in order to increase safety.
- [G5] Municipality receives enough information about the violation in order to issue a ticket.
- [G6] The integrity of the violation report is guaranteed.

It is important to mention that all the stakeholders' needs are portrayed in the goals, in fact every function expressed by the stakeholders in the Project Assignment can be fulfilled by some of the goals as shown in the table.

		Goals					
		G1	G2	G3	G4	G5	G6
Functions expressed by the stakeholders in the Project Assignment	Basic service	X	X	X			
	Advanced function 1			X	X		
	Advanced function 2			X		X	X

1.2. Scope

WORLD PHENOMENA	**SHARED PHENOMENA**	**MACHINE PHENOMENA**
A traffic violation occurs	A report of a violation is received	The system processes the information of the violation
	A user signs up	A violation report is saved
	A traffic ticket is generated	A query on data about violations, issued tickets or

		accidents is processed
	A query on data about violations, issued tickets or accidents is received	Information about accidents is retrieved from the municipality
		Information about accidents is crossed with information about violations to suggest possible interventions
		Information about issued tickets is retrieved from municipality

Note to ourselves: Highlighting streets?
Note to ourselves: A normal user can retrieve spatial and temporal data regarding the violations, but not personal information like license plates or photos)

1.3. Definitions, acronyms, abbreviations

1.3.1. Definitions

- User: a citizen registered to the SafeStreets service.
- Municipality: a city or town that has corporate status and local government.
- Timestamp: a representation of date and time.
- Anonymized data: data that don't give any personal information, such as license plate, pictures and names
- Valid violation report: a violation report composed by at least one picture, exactly one location, exactly one timestamp, exactly one type of violation and the license plate of the vehicle.
- Approved violation report: a valid violation report that may represent a correct violation report.
- Correct violation report: an approved violation report that the municipality evaluated as a traffic violation.
- Area: a district or a neighborhood of a country or city, especially one characterized by a particular feature or activity.
- GPS fix: a position derived from measuring in relation to GPS satellites.

1.3.2. Acronyms

- API: Application Programming Interface.
- GPS: Global Positioning System.
- UI: User Interface.
- S2B: Software To Be.
- GDPR: General Data Protection Regulation.
- OS: Operating System.

1.3.3. Abbreviations

- Gn: nth goal.
- Dn: nth domain assumption.
- Rn: nth requirement.

1.4. Revision history

- Version 1.0: Initial release.

1.5. Reference documents

TODO

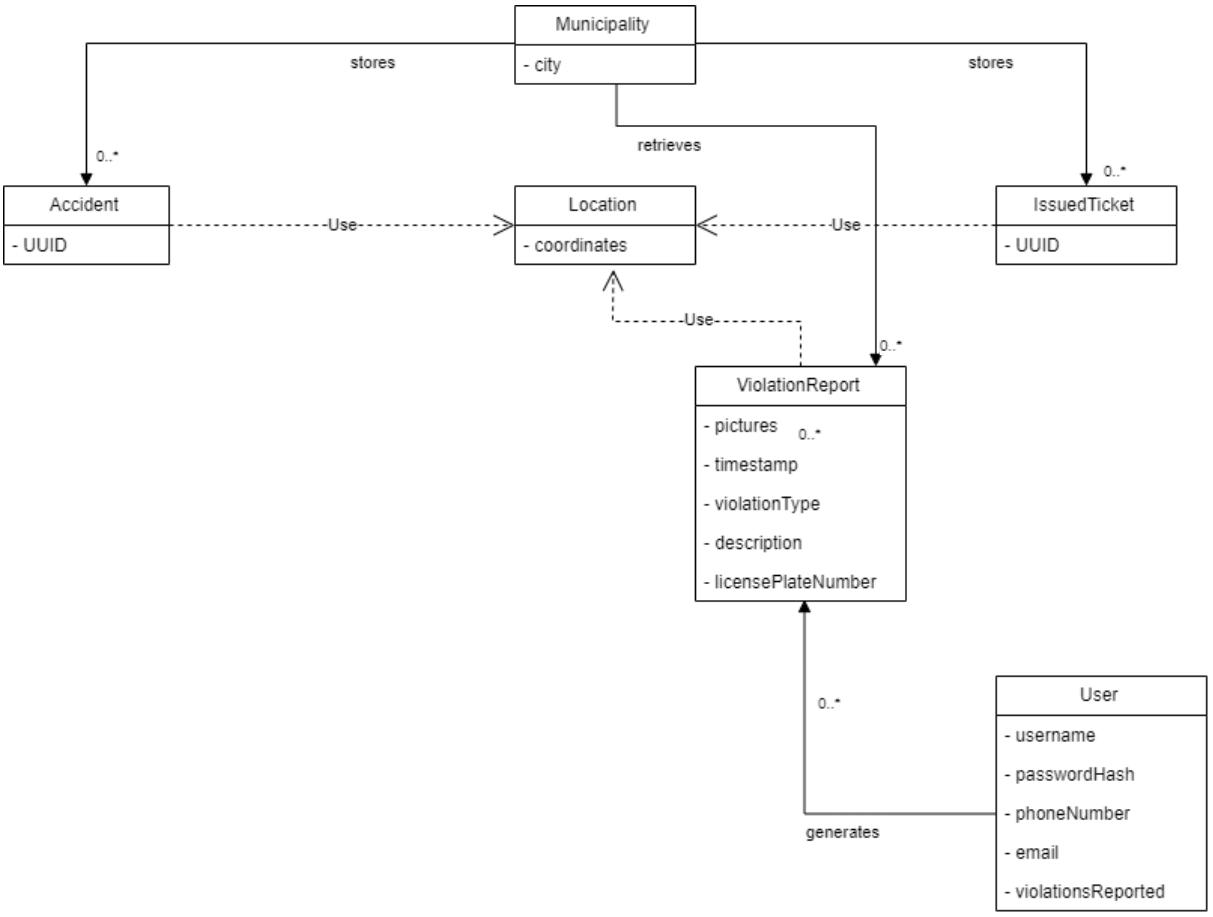
1.6. Document structure

TODO

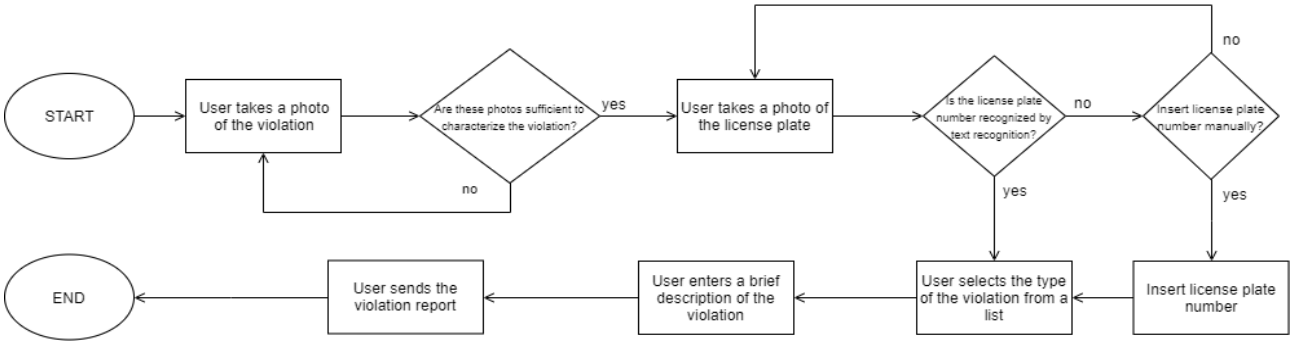
2. Overall description

2.1. Product perspective

SafeStreets is a crowd-sourced application that relies on the contribution of all citizens to gain useful data that can be exploited to make the streets a safer place. Given the widespread use of mobile phones today, SafeStreets will be released as a mobile application. The average user will be a normal citizen, concerned for his well-being and that of others.
SafeStreets is a new, self-contained products and will be built from scratch.
The following class diagram is a model of the application domain: as a high-level representation, it contains only a few essential attributes and does not include every class that will be necessary to the system.

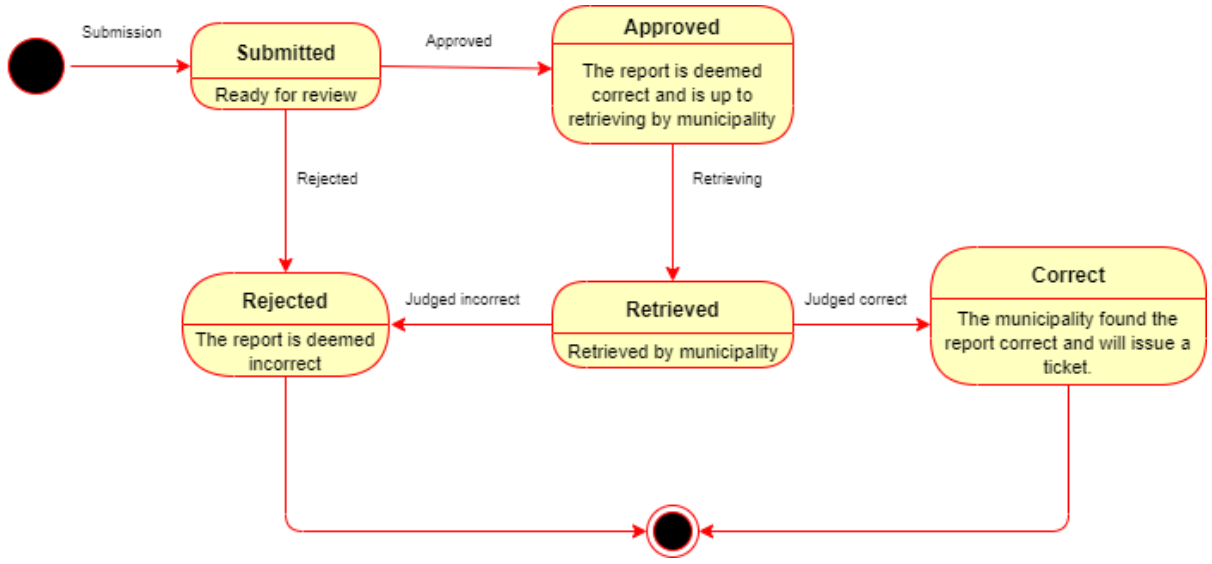


The following flowchart describes the creation of a violation report. The system will be equipped with text recognition in order to recognize the license plate number; however, if the picture taken is not of high enough quality, blurry, or the recognition doesn't work, the user will have the possibility to type the license plate number manually.



During its existence, the violation report can assume many different states. This is because we would like to be transparent and show the user what are the consequences of his effort. The following is a UML State Diagram representing the states that the violation report can assume.
A violation that is submitted to us is a valid violation; that means it has the necessary information to be sent from the app.

Then the system will do a preliminary scan of the reports; a report is deemed approved when it potentially represents a violation. That means discarding reports with blurry pictures, photos of anything other that a violation, or with incorrect information. Once the report is approved, it will be up for retrieving by the corresponding municipality. The municipality will then retrieve it and check if the approved violation report is correct; for example, it can check if the license plate really exists, or if the license plate is registered to the same type of car in the picture. The municipality can then decide that the report does not represent a violation, thus rejecting it, or decide that, based on the report, a violation occurred and a ticket will be issued to the driver. In this case, the state of the report will be “correct”.



2.2. Product functions

2.2.1. Violation reporting

The basic function for SafeStreets is Violation Reporting. A user that sees a traffic violation and wants to report it must fill all the information needed for the report. In particular, the user will need to take enough pictures of the incriminated vehicle to discern that it is indeed a traffic violation. A picture of the license plate is needed for text recognition to work but, as said above, if for any reason the text recognition fails the user will have the opportunity to type or correct the license plate number. A short description of the issue is welcomed, although not required. The system will then automatically collect the date, time, and GPS location of the user when the report is sent, so the user must have the GPS functionality enabled.
The report is then reviewed to make sure if it may be correct. If not, then the report is discarded; if it is indeed can be a correct report, it will be published and available for the corresponding municipality to retrieve it.
The municipality can then retrieve the reports and evaluate it. It can decide to consider the report grave enough to issue a ticket to the driver; this will be made publicly known on the report.

2.2.2. Data mining

The data about each violation sent through SafeStreets will be stored in an apposite data structure.
Since the municipalities offer services to retrieve information about accidents and tickets occurred in their area of operation, the intention is to combine these two sources of information.
The system will be able to cross and analyze this data to generate useful statistics that will be used, for example, to search for potentially dangerous roads, or to identify unsafe areas.
These statistics will then be available for both the users and the municipality to retrieve it.
The level of visibility will be different for the two entities: users should only see aggregate data without personal information; municipalities, on the other hand, will be able to see license plates or even query for the history of violations of a certain specified license plate number.
Furthermore, the system will be able to give suggestions about possible intervention in dangerous roads, and make these suggestions publicly available.

2.3. User characteristics

We identify two actors of our system.

- User: a citizen that is registered to the SafeStreets application. The user can report traffic violations, and use the data mining service, although cannot see personal information.
- Municipality: a city or town that has corporate status and local government. The municipality can review reports submitted by citizens, and use the data mining service.

2.4. Assumptions, dependencies and constraints

2.4.1. Domain assumptions

- [D1] A citizen who wishes to report a violation has a mobile phone with the SafeStreets app installed.
- [D2] Municipality offers a service to retrieve information about accidents.
- [D3] Municipality offers a service to retrieve information about tickets.
- [D4] When a device is able to obtain a GPS fix, the location provided has an accuracy of at least 20 meters.
- [D5] The municipality checks if approved violation reports can actually represent a traffic violation (for example they check if the license plate actually corresponds to the car model in the picture).
- [D6] All connections that use modern encryption protocol can not be manipulated.

All these domain assumptions are believed to represent valid properties about the world that can be assumed true in any case.
In particular for:

- D1: it’s trivial to believe that the assumption can be assumed in any case.

- D2 and D3: while there may be cases in which these assumptions may not hold, in those cases it would be practically impossible to reach the goals, so they are effectively needed to be assumed in order to reach the goals.
- D4: it is quite safe to assume that a GPS fix reaches an accuracy of at least 20 meters considering that usually a GPS fix has an accuracy of ≈ 3 meters while driving in an open area.
- D5: it can be assumed since it's something that the municipality must do in order to issue ticket.
- D6: it is known that modern encryption protocols are proved to be safe against cyber attacks.

2.4.2. Dependencies

The SafeStreets application will be dependant on the Operating System hosting it, so it must follow all the requirements of it.
Also the system will have to use a mapping service to perform operations with locations and visualize them.

3. Specific requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

TODO (mockups?)

3.1.2. Hardware Interfaces

The system doesn’t have any hardware interfaces.

3.1.3. Software Interfaces

The system will provide an API that will be used by municipalities to retrieve violation reports. The system will also use APIs provided by the municipalities to retrieve issued ticket and accidents happened in the territory of the municipality.

3.1.4. Communication Interfaces

All the communications inside (server - application) and outside the system (system - municipality) will use standard TCP/IP network interfaces.

3.2. Functional Requirements

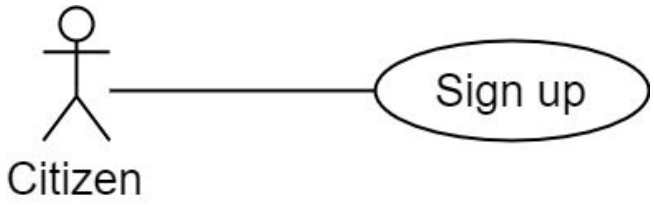
3.2.1. Citizen

3.2.1.1. Scenarios

3.2.1.1.1. Scenario: citizen registration

Phoebe's mom is getting old and has difficulty walking on her own, so she needs to use a wheelchair to move around. Phoebe, as a good daughter should do, every saturday takes her to the supermarket and helps her with the shopping. But most of the times the parking lot reserved for people with disabilities is occupied by cars that do not have the handicap badge. Phoebe argued with the supermarket owner, but he said that this type of problems do not border him. One day, while she was searching for a new mobile game, Phoebe's attention is drawn towards the suggested app section exactly on the SafeStreets app. She thinks that this application can help her with her problem. The registration is very easy and straightforward, she only needs to fill the mandatory fields: her name, her surname, her phone number, a username of her choice and a password. Now she is ready to report anyone that could prevent her to spend some time with her mother.

3.2.1.2. Use cases



3.2.1.2.1. Sign up

Name	Sign up
Actor	Citizen
Entry condition	The citizen opens the application.
Event flow	1. The citizen on the first screen of the application presses the "Sign up" button. 2. The citizen provides his information by filling in all the mandatory fields. 3. The citizen confirms the registration by pressing on the "Confirm" button. 4. The system saves the data of the new user. 5. The system sends a validation SMS to the new user.
Exit condition	The citizen has become a user, from now on he/she can log in to the system using the username and password provided during his/her sign up process.

Exceptions	1. The phone number is already associated with another account: is suggested to the citizen to insert another one. 2. The username is already associated with another account: is suggested to the citizen to insert another one. 3. Some information is not valid: the not valid fields are highlighted and is suggested to the citizen to insert valid information. 4. The citizen does not fill all the mandatory fields: the missing fields are highlighted and is suggested to the citizen to fill them.

3.2.2. User

3.2.2.1. Scenarios

3.2.2.1.1. Scenario 1: safe areas

Rachel and Ross are expecting a baby, but their current apartment is too small for them all, so they decided to buy a new one in the center of their town. Since they opt for the best for their child when he/she will grow up, they are not only searching for a comfortable apartment, but also for one situated in a safe area of the city. Fortunately they have the SafeStreets' app installed on their devices and can easily check the safest streets in town. They can easily visualize a map that specifies in which areas there is a higher rate of accidents or in which people usually park where they shouldn't.

3.2.2.1.2. Scenario 2: reporting a violation, no bike parking lot

Joey cares about himself and about the environment, so he has decided to use the bicycle to go to work. Unfortunately today he turned the alarm clock off and fell asleep again. The sound of the traffic passing by his apartment wakes him up, it is 8 o'clock and his turn at the museum starts at 8:30. So he dresses up, skips his usual coffee and runs to take his bike. On his way he sees that the cycle lane is blocked by a parked yellow car. It is not the first time that this happens, but calling the police would take too much time. Fortunately Joey downloaded and signed up to SafeStreets. Now for Joey is very easy to report the violation, all it takes is logging in, taking a photo of the car with the licence plate well visible and fill the field that specifies the type of violation. Joey arrives in time to the museum and he knows that the owner of the car will receive a ticket and hopefully will learn the lesson.

3.2.2.1.3. Scenario 1

Mr. Heckles is a lonely old man

3.2.2.2. Use cases

3.2.1.2.1. Log in of a user

Name	Log in of a user
Actor	User

Entry condition	The user opens the application.
Event flow	1. The user inserts his/her username and password in the corresponding fields. 2. The user presses the “Confirm” button. 3. The system logs the user in.
Exit condition	The user is now able to report a violation or look up for safe areas.
Exceptions	1. The username is not associated with any account: the user is notified and is asked to insert a valid email. 2. The password is incorrect: the user is notified and is asked to insert the password again.

3.2.1.2.2. Report violation

Name	Report violation
Actor	User
Entry condition	The user is logged in.
Event flow	1. The user chooses the option to report a new violation. 2. The user takes at least one photograph of the violation within the application. 3. The user selects the type of violation from a given list. 4. The user presses the “Confirm” button. 5. The system saves the report data. 6. The system retrieves and checks all the metainformation. 7. It is communicated to the user that the report has been received correctly.
Exit condition	The violation is saved in the system as “submitted” and it’s ready to be reviewed.
Exceptions	1. The system cannot retrieve the licence plate from one of the photos: the application asks the user to insert it manually. 2. Some of the metainformation of the violation report are incorrect: the user is notified and is asked to correct them. 3. something else?

3.2.1.2.3. Look up for safe areas

Name	Look up for safe areas
Actor	User

Entry condition	The user is logged in.
Event flow	1. The user chooses the option to look up for safe areas. 2. The application shows the map of the city near him using the GPS location. 3. The user inserts an address by pressing the magnifying glass icon. 4. The application shows him the requested address.
Exit condition	No exit condition.
Exceptions	1. The address is in an invalid form: the user is notified and asked to insert a valid one.

3.2.3. Municipality

3.2.3.1. Scenarios

3.2.3.1.1. Scenario 1: suggestions for the municipality

Chandler and his friends every Friday meet up in their favorite pub in town. Chandler for convenience has bought an electric motorcycle, but the parking lots dedicated for his vehicles are always occupied by some cars. While discussing about this problem with his friends one of them suggests to Chadler to download SafeStreets and report the car ifever it would happen again. Chandler does as suggested by his friend and the next week starts reporting every violation. The police officer that checks the suggestions given by SafeStreets notice the high number of reports done by Chandler and decides to do something. Only a few weeks after Chandler arrives at the pub and with great surprise he notices that no car is parked on the motorcycle parking lots thanks to some barriers. He is so happy that offers a round of beers to all of his friends.

3.2.3.1.2. Scenario 2:

As a new directive, all policemen in charge of issuing tickets in the city have to look up for the areas where in the last period there have been more reports. So before going out for his turn Bob logs in the web interface of the municipality that communicates with SafeStreets system through its API and can easily see which are the most critical areas. In this way he is able to increase the efficiency of his work.

3.2.3.1.3. Scenario 3:

Russ works at the municipality. He used to work at the issued ticket archive, he needed to sort every ticket by hand verifying that the licence plate corresponded to the correct car model using a very old interface with the local Database. Now, thanks to the service offered by SafeStreets, he does not need to sort anything by hand. Russ simply logs in the interface offered by SafeStreets and the list of all valid reposts appears on the screen; the only thing he needs to do is verify that the licence plate is valid and then press the “Convalidate” button, “reject” otherwise.

3.2.3.2. Use cases

3.2.3.2.1. Log in for municipality

Name	Log in for municipality
Actor	Municipality
Entry condition	The police officer opens web interface.
Event flow	1. The police officer inserts the username and the password in the corresponding fields. 2. The police officer presses the “Confirm” button. 3. The system logs the police officer in.
Exit condition	The police officer is now able to verify reports, retrieve suggestions and visualize the map of the city.
Exceptions	1. The username is not associated with any account: the police officer is notified and is asked to insert a valid one. 2. The password is incorrect: the police officer is notified and is asked to insert the password again.

3.2.3.2.2. Verify reports

Name	Verify reports
Actor	Municipality
Entry condition	The police officer is logged in.
Event flow	1. The police officer chooses to access the confirmation section. 2. The system visualizes the information about a violation. 3. The police officer checks the correctness of the report. 4. The police officer presses the “Confirm” or “Reject” button. 5. The system updates the state of the report.
Exit condition	The report is labeled accordingly to what the police officer has chosen.
Exceptions	There are no exceptions.

3.2.3.2.3. Receive suggestions

Name	Receive suggestions

Actor	Municipality
Entry condition	The police officer is logged in.
Event flow	1. The police officer chooses to access the suggestions section. 2. The system visualizes a list of all the areas. 3. The police officer chooses an area to inspect. 4. The system visualizes in more detail the information concerning the chosen area and the possible suggestions.
Exit condition	No exit condition.
Exceptions	

3.2.3.2.4. Look up for safe areas

Name	Look up for safe areas
Actor	Municipality
Entry condition	The police officer is logged in.
Event flow	1. The police officer chooses to access the map section. 2. The system visualizes a map with all the areas controlled by the municipality. 3. The system highlights with different colours the streets where more violations and accidents occured. 4. The police officer chooses an area or a street to inspect. 5. The system visualizes in more detail the information concerning the chosen area or street.
Exit condition	No exit condition.
Exceptions	
Special requirements	

3.2.3. Requirements

The objective of this paragraph is to show the completeness of the specified requirements. To reach this objective three steps must be performed:

- Goals must adequately capture all the stakeholders' needs. This has been proved in the section "1.1.2. Goals".
- Domain assumption must be valid properties about the world. This has been proved in section "2.4.1. Domain assumptions".
- Requirements must ensure satisfaction of the goals given the context of the domain assumptions, or written in logic terms: Requirements, DomainAssumptions $\models$ Goals.

This is what is going to be proved here by showing in an informal manner which goal is satisfied by certain requirements and domain assumptions.

3.2.3.1. Satisfying goal 1

[G1] A citizen can report a violation.

- [D1] A citizen who wishes to report a violation has a mobile phone with the SafeStreets app installed.
- [R1] A citizen not yet registered must be able to sign up and become a user.
- [R2] The application must allow users to authenticate through log in.
- [R3] The application must allow users to report a violation.

3.2.3.2. Satisfying goal 2

[G2] A violation report received by the system must have enough information to be valid, i.e. has at least one picture of the violation, exactly one GPS position, exactly one timestamp, exactly one type of violation and the license plate of the vehicle.

- [R4] The application must allow reporting of violations only from devices equipped with a GPS receiver which are in the conditions to obtain a GPS fix.
- [R5] The application must allow reporting of violations only from devices equipped with a camera.
- [R6] The application must allow reporting of violations only from devices with an active internet connection.
- [R7] A user has the possibility to specify the type of the reported violation choosing from a list.
- [R8] The application creates a violation report with at least one picture, exactly one timestamp, exactly one location, exactly one type of violation and the license plate of the vehicle.

3.2.3.3. Satisfying goal 3

[G3] Users and municipality can retrieve information about violations, accidents and issued tickets in a certain area, with different levels of visibility.

- [D2] Municipality offers a service to retrieve information about accidents.
- [D3] Municipality offers a service to retrieve information about tickets.
- [R9] The system saves all the information regarding the violations reported by users.
- [R10] The system must be able to retrieve data regarding issued tickets and accidents from the municipality.
- [R11] The system must offer an interface to retrieve information about violations, accidents and issued tickets.
- [R12] The system must show to users only anonymized data.
- [R13] The system must analyze valid violations report and tag which of them are correct.

3.2.3.4. Satisfying goal 4

[G4] Municipality will be able to retrieve suggestions for possible interventions in order to increase safety.

- [D2] Municipality offers a service to retrieve information about accidents.
- [D3] Municipality offers a service to retrieve information about tickets.
- [R9] The system saves all the information regarding the violations reported by users.
- [R10] The system must be able to retrieve data regarding issued tickets and accidents from the municipality.
- [R14] The system must be able to elaborate data about violations, accidents, issued tickets and generate useful suggestions about possible interventions.
- [R15] The system must offer an interface to the municipality for retrieving useful suggestions about possible interventions.

3.2.3.5. Satisfying goal 5

[G5] Municipality receives enough information about the violation in order to issue a ticket.

- [D4] When a device is able to obtain a GPS fix, the location provided has an accuracy of at least 20 meters.
- [D5] The municipality checks if approved violation reports can actually represent a traffic violation (for example they could check if the license plate actually corresponds to the car model in the picture).
- [R8] The application creates a violation report with at least one picture, exactly one timestamp, exactly one location, exactly one type of violation and the license plate of the vehicle.
- [R13] The system must analyze valid violations report and approve which of them may represent a correct violation.
- [R16] The system must offer an interface to the municipality for retrieving correct violations report.

3.2.3.6. Satisfying goal 6

[G6] The integrity of the violation report is guaranteed.

- [D6] All connections that use a modern encryption protocol can not be manipulated.
- [R17] The application will allow using pictures in a violation report only if the picture was taken by the application itself, preventing it to be manipulated on the device.
- [R18] All connections used by the system use modern encryption protocols.
- [R19] Data saved in the server can not be manipulated.

3.3. Performance Requirements

The system must have a scalable performance since its load is expected to undergo a lot of variations during the day. An expected real time load during peak times (6:00-10:00 and 16:00-20:00) could be approximated to < 2% of citizens living in the municipality, for every municipality. During the night the load is expected to decrease drastically.

3.4. Design Constraints

3.4.1. Standards compliance

To guarantee compatibility to other systems and to ensure that the best practices are followed the system and the organization will have to use specific standards:

- ISO 6709[1] - “Standard representation of geographic point location by coordinates”: for the format of the GPS location.
- ISO/IEC 27001[2] - “Information security management”: provides best practices for managing sensitive company information so that it remains secure.
- ISO/IEC 27701[3] - “Security techniques”: provides best practices for implementing, maintaining and continually improving a Privacy Information Management System.

The system must also be compliant to the General Data Protection Regulation (GDPR), and by complying with the ISO/IEC 27001 and ISO/IEC 27701 standards the system will meet the privacy and information security requirements of the aforementioned regulation[4].

3.4.2. Hardware limitations

Some hardware requirements need to be met in order to fully take advantage of the functionalities offered by SafeStreets:

3.4.2.1. Data mining - hardware limitations

The “Data mining” functionality needs an active internet connection with a reasonable data bandwidth.

3.4.2.2. Violation reporting - hardware limitations

There are stricter hardware constraints needed to be met in order to make a violation report. A device used to perform such functionality must have:

- An active internet connection with a reasonable data bandwidth.
- A GPS receiver.
- A camera.

These constraints are needed to create a valid violation report.

3.4.3. Any other constraint

The system will be subject to other constraints due to the presence of the communication between the system itself and the municipalities; in order to prevent an overload of the municipality's server, the system must make a reasonable amount of requests to the interfaces.
Also the SafeStreets application must follow all the requirements of the Operating Systems hosting it.

3.5. Software System Attributes

3.5.1. Reliability

To avoid faults and to reach a high reliability the S2B will be built using defensive programming techniques.
A defensive approach has been already started here, at the requirement level, by making domain assumptions that are very plausible and can actually be assumed all the time.

3.5.2. Availability

The system is not safety critical so it is possible to tolerate not so high levels of availability. A 3-nines availability (≈9 hours of downtime every year) can be enough.

3.5.3. Security

A main concern of the system is to always ensure that the information provided to the municipality, which will be used to issue tickets, can't be manipulated in any way. No one should be able to use the system to create fraudulent tickets.
Another still important concern is to always assure that the system can't be compromised in any way that could alter its behavior, for example when reporting data about violations or when suggesting possible interventions to the municipality.
Furthermore to be complied to the GDPR, user credentials and data need to be protected from external access with appropriate security measures.

3.5.4. Maintainability

The software must be written in a way that would make it easy to maintain and update, to achieve this objective components and interfaces must be specified at the most abstract level possible and appropriate design patterns must be used.

3.5.5. Portability

Portability is a main concern for the system since the application must be available to different mobile operating system. Also it is wise to consider that other OSs can be introduced in the next few years so the application and the system in its entirety must be ready for the case.

4. Formal analysis using alloy

TODO

4.1. Alloy code

```
// ########## Signatures ##########
sig ViolationReport {
      pictures: set Picture,
      location: lone Location,
      timestamp: lone Timestamp,
      typeOfViolation: lone TypeOfViolation,
      licensePlate: lone LicensePlate,
      state: one ViolationReportState
}

sig Picture {}
sig Location {}
```

```
sig Timestamp {}
sig TypeOfViolation {}
sig LicensePlate {}


// ########## Enums ##########
abstract sig Bool {}
one sig TRUE extends Bool {}
one sig FALSE extends Bool {}

abstract sig ViolationReportState {
	canBeAltered: one Bool
}
one sig AT_DEVICE extends ViolationReportState {}
one sig AT_NETWORK extends ViolationReportState {
	encryptedConnection: one Bool
}
one sig AT_SERVER extends ViolationReportState {}


// ########## Goal 2 ##########

/* Represent the act of sending a violation report through the network.
* @param vDevice the representation of the violation report on the
device.
* @param vNetwork the representation of the violation report in the
network.
*/
pred sendReportToNetwork [vDevice : ViolationReport, vNetwork :
ViolationReport] {
	vDevice.state = AT_DEVICE
	vNetwork.state = AT_NETWORK
	vDevice.pictures = vNetwork.pictures
	vDevice.location = vNetwork.location
	vDevice.timestamp = vNetwork.timestamp
	vDevice.typeOfViolation = vNetwork.typeOfViolation
	vDevice.licensePlate = vNetwork.licensePlate
}

/* Represent the act of receiving a violation report from the network.
* @param vNetwork the representation of the violation report in the
network.
* @param vServer the representation of the violation report on the
server.
*/
```

```
pred receiveReportFromNetwork [vNetwork : ViolationReport, vServer :
ViolationReport] {
	vNetwork.state = AT_NETWORK
	vServer.state = AT_SERVER
	vNetwork.pictures = vServer.pictures
	vNetwork.location = vServer.location
	vNetwork.timestamp = vServer.timestamp
	vNetwork.typeOfViolation = vServer.typeOfViolation
	vNetwork.licensePlate = vServer.licensePlate
}

/* Represent the act of sending a violation report to the server from
the device.
* This basically says that there exist a violation report sent in the
network such that:
* it has the same content of the violation report sent by the device,
* and the same content of the violation report received by the server.
* @param vDevice the representation of the violation report on the
device.
* @param vServer the representation of the violation report on the
server.
*/
pred sendReportToServerFromDevice [vDevice: ViolationReport, vServer :
ViolationReport] {
	vDevice.state = AT_DEVICE
	vServer.state = AT_SERVER
	some vNetwork : ViolationReport | vNetwork.state = AT_NETWORK and
		sendReportToNetwork[vDevice, vNetwork] and
		receiveReportFromNetwork[vNetwork, vServer]
}

// [R4] The application must allow reporting of violations only from
devices equipped with a GPS receiver which are in the conditions to
obtain a GPS fix.
// [R5] The application must allow reporting of violations only from
devices equipped with a camera.
// [R7] A user has the possibility to specify the type of the reported
violation choosing from a list.
// [R8] The application creates a violation report with at least one
picture, exactly one timestamp, exactly one location, exactly one type
of violation and the license plate of the vehicle.
fact requirement4_5_7_8 {
	all v : ViolationReport | v.state = AT_DEVICE implies (#v.pictures
		>= 1 and #v.location = 1 and #v.timestamp = 1 and
		#v.typeOfViolation = 1 and #v.licensePlate = 1)
```

```
}


// [R6] The application must allow reporting of violations only from
devices with an active internet connection.
// This requirement basically says that the violation report created by
the device will be correclty received by the server.
fact requirement6 {
	all vS : ViolationReport | vS.state = AT_SERVER implies
		(some vD : ViolationReport | vD.state = AT_DEVICE and
			sendReportToServerFromDevice[vD, vS])
}

// [G2] A violation report received by the system must have enough
information to be valid, i.e. has at least one picture of the violation,
exactly one GPS position, exactly one timestamp, exactly one type of
violation and the license plate of the vehicle.
assert goal2 {
	all v : ViolationReport | v.state = AT_SERVER implies
		(#v.pictures >= 1 and #v.location = 1 and #v.timestamp = 1
		and #v.typeOfViolation = 1 and #v.licensePlate = 1)
}
check goal2 for 5


// ########## Goal 6 ##########

// [D6] All connections that use a modern encryption protocol can not be
manipulated.
fact domainAssumption6 {
	all n : AT_NETWORK | n.encryptedConnection = FALSE <=>
		n.canBeAltered = TRUE
	all n : AT_NETWORK | n.encryptedConnection = TRUE <=>
		n.canBeAltered = FALSE
}

// [R17] The application will allow using pictures in a violation report
only if the picture was taken by the application itself, preventing it
to be manipulated on the device.
fact requirement17 {
	all d : AT_DEVICE | d.canBeAltered = FALSE
}

// [R18] All connections used by the system use modern encryption
protocols.
```

```
fact requirement18 {
	all n : AT_NETWORK | n.encryptedConnection = TRUE
}

// [R19] Data saved in the server can not be manipulated.
fact requirement19 {
	all s : AT_SERVER | s.canBeAltered = FALSE
}

// [G6] The integrity of the violation report is guaranteed.
assert goal6 {
	all v : ViolationReport | v.state.canBeAltered = FALSE
}
check goal6 for 5
```

4.2. Alloy analyzer results

The results provided by the analyzer are reported here:

```
Executing "Check goal2 for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  3403 vars. 183 primary vars. 7070 clauses. 22ms.
  No counterexample found. Assertion may be valid. 38ms.

Executing "Check goal6 for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  3156 vars. 183 primary vars. 6230 clauses. 14ms.
  No counterexample found. Assertion may be valid. 1ms.

2 commands were executed. The results are:
  #1: No counterexample found. goal2 may be valid.
  #2: No counterexample found. goal6 may be valid.
```

5. Effort spent

Matteo Marchisciana
7 hr (World and machine phenomena, goals, domain assumptions and requirements)
5hr 15 min (Overall description)
30/10 20 min (Overall description)
31/10 30 min (Overall description)
02/11 1 hr (Overall description & Alloy)
03/11 1 hr 30 min (Overall description and review)

Andrea Marcer
7hr 5 min (world and machine, goals, requirements and domain)
1hr (Scenarios)

45min (research for use cases)
1hr 45min (use cases)
1hr 15min (use cases)
2hr 15min (Use cases)
2hr 45min(Overall review)

Dennis Motta
23/10 - 4hr 0 min - world and machine phenomena, goals, domain assumptions and requirements
26/10 - 2 hr 45 min - goals, domain assumptions and requirements
27/10 - 2 hr 30 min - performance requirements, design constraints and Software System Attributes
28/10 - 1 hr 0 min - Software System Attributes
29/10 - 2 hr 30 min - Goals, domain assumptions and requirements
31/10 - 0 hr 15 min - Overall description
02/11 - 2 hr 30 min - Alloy, Software System Attributes, design constraints, dependencies, External Interface Requirements
03/11 - 3 hr 0 min - Alloy, overall description

6. References

1. “ISO 6709”. https://www.iso.org/standard/39242.html
2. “ISO/IEC 27001”. https://www.iso.org/standard/54534.html
3. “ISO/IEC 27701”. https://www.iso.org/standard/71670.html
4. “ISO 27001 and the GDPR”. https://www.itgovernance.co.uk/gdpr-and-iso-27001
5. “a GPS fix has an accuracy of ≈ 3 meters while driving in an open area”. This is an actual measurement performed by the *GPS Status & Toolbox* app: https://play.google.com/store/apps/details?id=com.eclipsim.gpsstatus2