



POLITECNICO DI MILANO 1863

SOFTWARE ENGINEERING 2 PROJECT

Requirement Analysis and Specification Document (RASD)

---

# SafeStreets

Version 1.0

---

*Authors*

Tiberio Galbiati  
Saeid Rezaei

*Supervisor*

Dr. Matteo Rossi

**Copyright:** Copyright © 2019, Tiberio Galbiati - Saeid Rezaei – All rights reserved

**Download page:** <https://github.com/TiberioG/GalbiatiRezaei.git>

November 4, 2019

# Contents

<b>Table of Contents</b>	<b>2</b>
<b>List of Figures</b>	<b>4</b>
<b>List of Tables</b>	<b>4</b>
<b>1 Introduction</b>	<b>5</b>
1.1 Purpose	5
1.2 Scope	5
1.2.1 Description of the given problem	5
1.2.2 Goals	5
1.3 Definitions, acronyms, abbreviations	5
1.3.1 Definitions	5
1.3.2 acronyms	5
1.3.3 abbreviations	6
1.4 Revision history	6
1.5 Reference Documents	6
1.6 Document Structure	6
<b>2 Overall Description</b>	<b>7</b>
2.1 Product perspective	7
2.1.1 Class Diagram	7
2.2 Product functions	7
2.2.1 Report violation	7
2.2.2 Explore Data	7
2.2.3 Issue a ticket	8
2.2.4 Generate statistics	8
2.3 User characteristics	8
2.4 Assumptions, dependencies and constraints	8
2.4.1 Dependencies	9
<b>3 Specific Requirements</b>	<b>10</b>
3.1 External Interface Requirements	10
3.1.1 User Interfaces	10
3.1.2 Hardware Interfaces	15
3.1.3 Software Interfaces	15
3.1.4 Communication Interfaces	15
3.2 Functional Requirements	15
3.2.1 Use Cases diagrams	16
3.2.2 Use Cases Description	17
3.2.3 Requirements	23
3.3 Performance Requirements	23
3.4 Design Constraints	23
3.4.1 Standards compliance	23
3.4.2 Hardware limitations	23
3.4.3 Any other constraint	23
3.5 Software System Attributes	23
3.5.1 Simple User Interface	23
3.5.2 Reliability	23

3.5.3	Availability . . . . .	24
3.5.4	Security . . . . .	24
3.5.5	Maintainability . . . . .	24
3.5.6	Portability . . . . .	24
<b>4</b>	<b>Formal Analysis Using Alloy . . . . .</b>	<b>25</b>
<b>5</b>	<b>Effort Spent . . . . .</b>	<b>26</b>

## List of Figures

1	Login screen . . . . .	10
2	. . . . .	11
3	Login screen . . . . .	12
4	Login screen . . . . .	13
5	Login screen . . . . .	14
6	Login screen . . . . .	15

## List of Tables

# 1 Introduction

## 1.1 Purpose

## 1.2 Scope

### 1.2.1 Description of the given problem

### 1.2.2 Goals

- [G1] Allow users to notify authorities about traffic violations
- [G2] Allow users to send pictures with metadata of violations
- [G3] Allow users to mine information recorded
- [G4] Have at least two different privilege for mining data
- [G5] Generate traffic tickets
- [G6] Generate statistics about issued tickets
- [G7] Be sure every information uploaded is never altered

## 1.3 Definitions, acronyms, abbreviations

### 1.3.1 Definitions

- Heatmap : A heatmap is a graphical representation of data that uses a system of color-coding to represent different values
- Enduser : a regular citizen which will use the app
- Authority user : someone who's working for an authority like (police, municipality etc.) recognized
- Geocoding : the process of converting addresses (like a street address) into geographic coordinates (latitude and longitude)
- Reverse geocoding: the process of converting geographic coordinates into a human-readable address

### 1.3.2 acronyms

- ALPR : Automated Licence Plate Recognition
- GUI : Graphical User Interface
- GDPR : EU General Data Protection Regulation
- API : Application Programming Interface

### **1.3.3 abbreviations**

## **1.4 Revision history**

## **1.5 Reference Documents**

## **References**

- [1] OpenALPR Technology Inc. , OpenALPR documentation <http://doc.openalpr.com>
- [2] Ministero delle infrastrutture e dei Trasporti, DECRETO LEGISLATIVO 30 aprile 1992, n. 285 Nuovo codice della strada, <https://www.normattiva.it/uri-res/N2Ls?urn:nir:stato:decreto.legislativo:1992-04-30;285!vig=>
- [3] GOOGLE inc, Google Maps Platform Documentation | Geocoding <https://developers.google.com/maps/documentation/geocoding/start>
- [4] GOOGLE inc, Google Maps Platform Documentation | Heatmap <https://developers.google.com/maps/documentation/javascript/heatmaplayer>

## **1.6 Document Structure**

## 2 Overall Description

### 2.1 Product perspective

#### 2.1.1 Class Diagram

The following class diagram is a high-level class diagram which should be intended as a model of the application structure. During the implementation part more classes and attributes can be created and used.

**User** The class **user** is the father of the two possible kind of user: **EndUser** and **Authority** which are needed because our application is intended to be multi-user and with at least two privileges for data that can be viewed and possible functions accessible.

**Location** Every user is in a **Location** class used to represent location as latitude and longitude coming from the SO of the smartphone.

**Reverse Geocoding** The interface **Reverse Geocoding** is used to communicate with the external service to get a readable address from the coordinates as explained in section 2.4.1.

**Violation** The class **Violation** is used to store all the data relative of the reported violation. The *kind* attribute is selected by the user in the fill form **??** from a list of possible kind of violations. The photo is here reported an attribute but can also be implemented as a separate class. It's mandatory that one picture is associated to every violation. In the **Violation** class are also stored the raw latitude and longitude in case there will be need of those data later, as an example if it's impossible get precise location using reverse geocoding.

**Automated License Plate Recognition** The interface **Automated License Plate Recognition** is needed to interact with the external ALPR service which receives a picture and returns a string containing every licence plate present in the picture. This interface is used to complete the attribute *licence plate* of every violation.

**MinedInfo** Classes **MinedStreets** and **MinedOffenders** are used to represent the data coming from the database of all violation and processed to offer different kind of visualization.  
end add here class diagram + verbal description

### 2.2 Product functions

#### 2.2.1 Report violation

#### 2.2.2 Explore Data

The app will offer the possibility to the users to visualize the data collected. Two kind of visualizations are offered:

1. Streets with the highest frequency of violations
2. Vehicles that committed the most violations

In order to get those data the system will periodically query the database of violations in order to create a table where the count of violation is stored, both for streets and vehicles. There will be a section in the app called "Explore Data" where will be able to choose which kind of data to visualize.

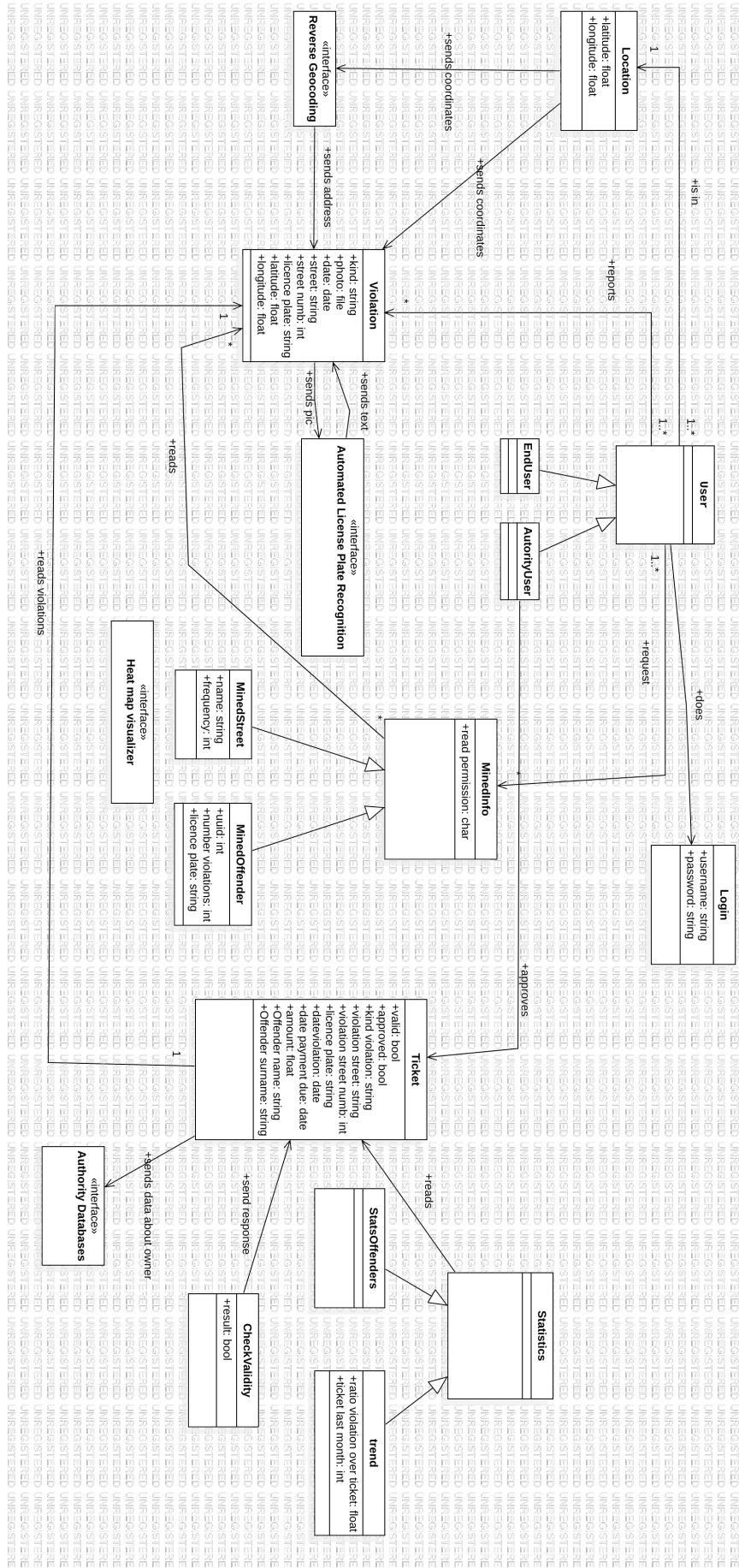


Figure 1: High-level Class Diagram



### **2.2.3 Issue a ticket**

Every time a new violation is inserted in the database, the System will use the new data available to generate a proposal of ticket, combining the data from violations with data coming from Municipality databases.

A ticket has the following structure:

1. Place where violation occurred
2. Date when violation occurred
3. Plate of vehicle
4. Article and code of violation
5. Amount to be paid
6. Date when the payment is due

The ticket will be in a pending-approval status because we want always a human check before sending it to the offender. Authority users (e.g. policemen) will check the pending-approval tickets and they can approve them or not. If ticket is not approved it will go in approval-denied status. If ticket has been approved it will have to be sent to the offender. The system will connect to the external vehicle registration database in order to retrieve the name, surname, address of the offender knowing the licence plate of his/her vehicle. Now we have all the data to print the ticket and send it via regular mail. There will be an office of police-station which will do the job.

### **2.2.4 Generate statistics**

## **2.3 User characteristics**

## **2.4 Assumptions, dependencies and constraints**

[D1] Device has internet connection

[D1] The device should acquire position with an accuracy of enough meters in order to univocally determine the road (e.g. 5 meters)

[D] We have access to an ALPR service which is able to read every licence plate in a picture and return the string

[D1] The device should take pictures with enough resolution to be able to read by the ALPR service

[D5] ALPR service has an accuracy of more than 90%

[D2] Every vehicle that can be reported should have a licence plate visible

[D3] The number and kind of violations should be finite (defined by the law)

[D4] Every authority account is verified and it's not possible to be created using the front end

[D6] We have access to the vehicle registration database where are stored licence plates, names and the addresses of the owners of every vehicle registered

[D7] We have access to a database where are stored all the codes of violations and the amount of fine for the violation

### 2.4.1 Dependencies

The app will be dependent on a third-party service to read the licence plate of the cars. (For example <http://www.openalpr.com> )

the app will be dependent on some Maps API to get the full address, knowing the coordinates of location coming from the GPS of the device.

The app will be dependent to some Maps API used to show the map and an overlay.

The app will be dependent on a smartphone, which has to provide the following features:

1. Internet connection, possibly using 2G/3G/4G in order to be available where there is no WiFi, considering the use case "on the road"
2. A camera with good resolution
3. GPS sensor

## 3 Specific Requirements

### 3.1 External Interface Requirements

This section provides a detailed description of all inputs and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

#### 3.1.1 User Interfaces

In this section we present the mockups of the GUI,



Figure 2: Login screen



Figure 3

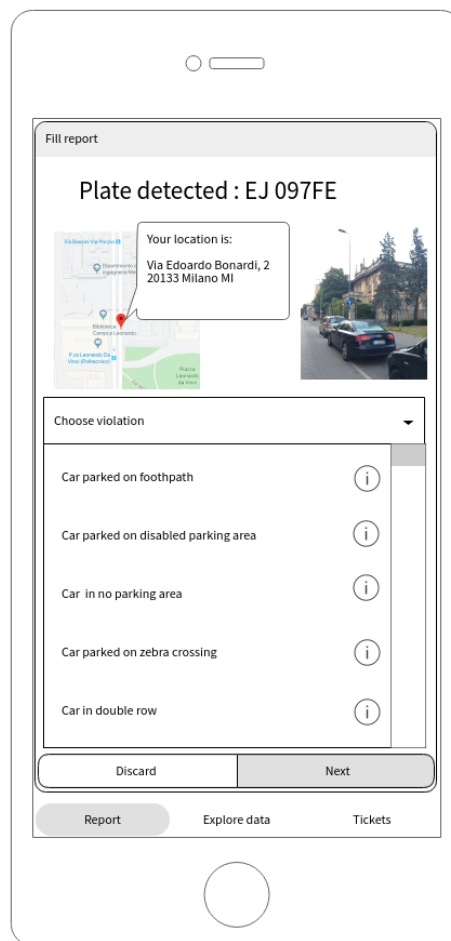


Figure 4: Login screen



Figure 5: Login screen

The image shows a mobile application interface. At the top, there is a status bar with a signal strength indicator and a battery level indicator. Below this is a header section titled "Explore data" with two tabs: "Offenders" and "Streets". The "Offenders" tab is selected. Below the tabs is a table with two columns: "ID" and "Number of violations". The table contains eight rows of data. At the bottom of the screen, there is a navigation bar with three buttons: "Report", "Explore data", and "Tickets". The "Explore data" button is highlighted.

ID	Number of violations
plate1	10
plate2	9
plate3	9
plate4	7
plate5	4
plate6	2
plate7	2
plate8	2

Figure 6: Login screen

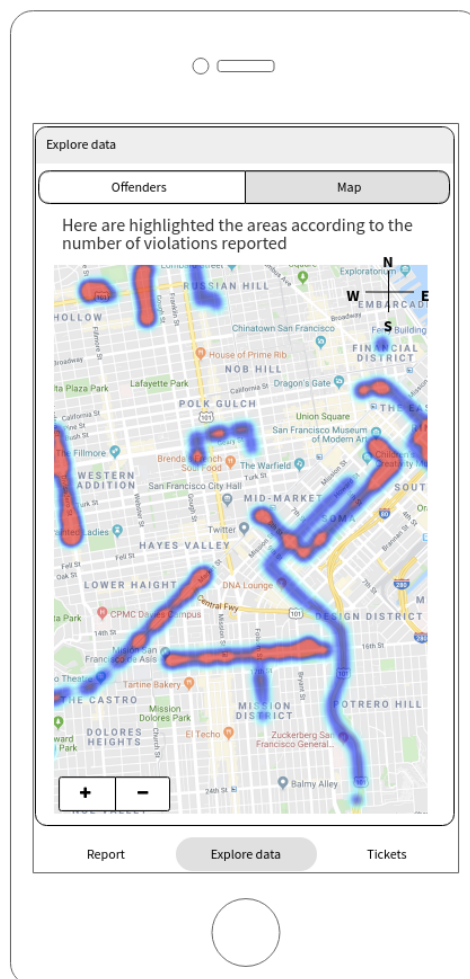


Figure 7: Login screen

### 3.1.2 Hardware Interfaces

there is no need to have hardware interfaces since we are developing a mobile application with a server side. Internet connection, GPS, and camera are all managed by the OS of the smartphone where the application will run.

### 3.1.3 Software Interfaces

### 3.1.4 Communication Interfaces

## 3.2 Functional Requirements

Every function should work only after successful login.

- **[G1] Allow users to notify authorities about traffic violations**

[R1] User must be able to choose the kind of violation from a list

[R2] User must be able to read detailed information about each kind of violation he can report



- **[G2] Allow users to send pictures with metadata of violations**

[R1] Application should access the camera

[R2] Date, time and position should be automatically added to the violation reported

[R3] We should require the user to send again a picture in case the plate is not visible

[R4] The user must be able to select the vehicle to report in case there are other vehicles in picture

- **[G3] Allow users to mine information recorded**

[R1] Application must be able to count occurrence of violations

[R2] Application must be able to count violation for each vehicle

[R4] Application should show the first  $n$  (input by user) vehicles with the highest number of violations

[R5] Application should visualize the areas where violation occurred

[R6] Application must use a gradient of color to show the occurrences of violation as an overlay of a interactive map

- **[G4] Have at least two different privilege for mining data**

[R1] Regular endusers can see the list of vehicles with the highest number of violations but they cannot see the licence plate, instead of that a random identifier is shown

[R2] Authority users can know the exact licence plate when mining data about offenders

- **[G5] Generate traffic tickets**

[R] Application must be able to read every violation stored and automatically generate a ticket

[R]

- **[G6] Generate statistics about issued tickets**

[R]

- **[G7] Be sure every information uploaded is never altered**

[R] The application must be able to know if a picture has been altered

### 3.2.1 Use Cases diagrams

Here are presented the use case diagrams for each main function. In the next section each use case will be verbally presented.

### 3.2.2 Use Cases Description

In the following section a description of each use case is provided. For every use case is reported: an ID defining each case, the entry conditions, the steps to accomplish the exit condition and any exception that may occur.

**ID:** [UC1]

**Name:** Sign-Up

**Actor:** Guest

**Entry conditions:**

1. A citizen who wants to use the service

**Event flow:**

1. The guest reaches the registration page containing the relative form
2. The guest fills up the form and clicks on "Sign up" to complete the process
3. The system redirects the user to his profile page and sends a confirmation email

**Exit conditions:**

- The guest has successfully registered in the system

**Exceptions:**

1. The guest left an empty field or typed something wrong an error message is displayed and the user is asked to fill the form again.

---

**ID:** [UC2]

**Name:** Login

**Actor:** User

**Entry conditions:**

1. The user has already registered

**Event flow:**

1. The user reaches the login page containing the relative form
2. The user types the username and password in the login form and click on "Login" button
3. The system redirects the user to the application homepage

**Exit conditions:**

- The user has access to the application functionalities

**Exceptions:**

1. Username and password didn't correspond or the username didn't exist, an error message is displayed and the user is asked to fill the login form again

---

**ID:** [UC3]

**Name:** Recover Password

**Actor:** User

**Entry conditions:**

1. The user has already registered

**Event flow:**

1. The user reaches the login page containing the relative form
2. The user clicks on "Password recovery" button and is redirected to the password recovery page.
3. The user inserts his email and clicks on "reset password"
4. The system sends an email to the user with a link and instruction to reset the password
5. The user chooses and types a new password and confirms
6. The application check whether the entered password is strong enough or not
7. The system redirects the user to the login page

**Exit conditions:**

- The user has changed his password

**Exceptions:**

1. The inserted email doesn't match any user in the database, it is displayed an error message and the user is asked to retype a valid email.

---

**ID:** [UC4a]

**Name:** Report a violation - taking picture

**Actor:** User

**Entry conditions:**

1. User is logged in

**Event flow:**

1. User enters the section "Report a violation"
2. System opens the camera of smartphone and ask user to take a picture of the violation
3. The system reminds the user that violation and the licence plate of the vehicle which is in violation must be visible
4. The user takes the picture
5. The system shows the picture just taken
6. The system asks the user: if there are other plates visible in the picture, which are not the one of the vehicle to be reported, use the finger to delete them
7. The system enters in "brush tool mode" and the user covers the other licence plates
8. When done, user press continue button
9. The system sends the picture to the ALPR service which returns the string containing the plate decoded
10. The system shows now on the screen the "report violation form"

**Exit conditions:**

1. User must continue to next [UC4b]

**Exceptions:**

1. If no plate is found, the user has to repeat this use case, starting from taking the picture again
2. If the ALPR service returns more than one plate, the user is informed that must delete the notrequired plates and the system goes agin to the "brush tool mode"
3. If user doesn't continue to the next use case: e.g. presses exit button, or closes the app formore than 10 minutes, the picture taken is discarded

---

**ID:** [UC4b]

**Name:** Report a violation - fill the form

**Actor:** User

**Entry conditions:**

1. User has successfully completed the precedent [UC4a]
2. User is in the fill-form section of the app

**Event flow:**

1. The system sends GPS location to the external service to get the complete address of the user
2. the form is pre-filled with the address that is given by the external service
3. The user must choose from a list of violations the one referred to the picture taken which wantsto report. In the UI every row contains the name of the violation and a "info" button
4. the user can choose to send the form or exit

**Exit conditions:**

1. The violation is correctly inserted and stored

**Exceptions:**

---

**ID:** [UC4b1]

**Name:** Report a violation - fill the form - violation infopage

**Actor:** User

**Entry conditions:**

1. User is in Use case [UC4b]
2. User has pressed the "info" button of a violation from the list

**Event flow:**

1. System shows a brief decription of the selected violation

**Exit conditions:**

1. User goes back to Use case [UC4b]

**Exceptions:**

---

**ID:** [UC5a]

**Name:** Mine information - street heatmap

**Actor:** User

**Entry conditions:**

1. User is logged in

**Event flow:**

1. User enters the section "Explore data"
2. The user chooses to get the map about streets with highest frequency of violations
3. The system queries in descending order the table where for each streets is associated the count of violations
4. The app shows a map with an overlay which highlights the areas with a gradient color according to the number of violations occurred

**Exit conditions:**

User wants to go back to "Explore data" area **Exceptions:**

1. If there are no records the app will report no data available message

---

**ID:** [UC5b]

**Name:** Mine information by Authority - offenders

**Actor:** AuthorityUser

**Entry conditions:**

1. AuthorityUser is logged in

**Event flow:**

1. AuthorityUser enters the section "Explore data"
2. The system asks which kind of data the AuthorityUser wants to know
3. The AuthorityUser chooses to get the data about vehicles that committed the highest number of violations
4. The system queries the table where for each licence plate is associated the count of violations
5. The system will report in a tabular way the plate of the vehicle and the count of violations committed
6. If the AuthorityUser scrolls down, the system will offer the chance to load more rows

**Exit conditions:**

1. AuthorityUser wants to go back to "Explore data" area

**Exceptions:**

1. If there are no records the app will report no data available message

---

**ID:** [UC5c]

**Name:** Mine information by EndUser - offenders

**Actor:** EndUsers

**Entry conditions:**

1. User is logged in

**Event flow:**

1. User enters the section "Explore data"
2. The system asks which kind of data the user wants to know
3. The User chooses to get the data about vehicles that committed the highest number of violations
4. The system queries the table where for each licence plate is associated the count of violations
5. The system will report in a tabular way an anonymized identifier of the vehicle and the count of violations committed if the request comes from a Regular User
6. The system will report in a tabular way the plate of the vehicle and the count of violations committed if the request comes from an Authority User
7. If the User scrolls down the system will offer the chance to load more rows

**Exit conditions:**

1. User wants to go back to "Explore data" area

**Exceptions:**

1. If there are no records the app will report no data available message

---

Advanced function

**ID:** [UC6]

**Name:** Ticket approval

**Actor:** AuthorityUser

**Entry conditions:**

1. A new violation is inserted in database
2. AuthorityUser logged in

**Event flow:**

1. Every time a new violation is created by a EndUser the system will create automatically a ticket to be approved
2. AuthorityUser enters the section "Tickets"
3. AuthorityUser enters the section "Approve Tickets"
4. The System will show the list of tickets available for approval

5. AuthorityUser selects one ticket and system will show the related details
6. System will ask the AuthorityUser if he wants to approve or not the ticket

**Exit conditions:**

1. User wants to go back to "Ticket" area
2. AuthorityUser approves the ticket
3. AuthorityUser doesn't approve the ticket

**Exceptions:**

1. If there are no tickets pending, the app will report no data available message

---

**ID:** [UC5]

**Name:** Ticket statistics selection

**Actor:** AuthorityUser

**Entry conditions:**

1. AuthorityUser logged in

**Event flow:**

1. AuthorityUser enters the section "ticket statistics"
- 2.

**Exit conditions:**

**Exceptions:**

---

**ID:** [UC5]

**Name:** Statistics - offenders

**Actor:** AuthorityUser

**Entry conditions:**

1. AuthorityUser logged in

**Event flow:**

1. AuthorityUser enters the section "ticket statistics"
- 2.

**Exit conditions:**

**Exceptions:**

---

**ID:** [UC5]

**Name:** Statistics - trends

**Actor:** AuthorityUser

**Entry conditions:**

1. AuthorityUser logged in

**Event flow:**

1. AuthorityUser enters the section "ticket statistics"
- 2.

**Exit conditions:**

**Exceptions:**

---

### **3.2.3 Requirements**

Requirements in order to satisfy the goals

- 1 test

## **3.3 Performance Requirements**

## **3.4 Design Constraints**

### **3.4.1 Standards compliance**

The app should be available for the two main operating systems of smartphones: Android OS and Apple iOS.

The traffic violations which can be reported should be compliant to the local traffic code where the app will be used.

For an use in Italy the app should be compliant to the "Codice della Strada", in particular parking violations are reported in Art. 157.

### **3.4.2 Hardware limitations**

The app will have a server side and a client side (smartphone). On server side limitations can be the size of available storage and the bandwidth. On smartphone side we have the network connectivity (3G/4G connection) and GPS limitations in some areas.

### **3.4.3 Any other constraint**

Application should be compliant to European GDPR and don't track users.

## **3.5 Software System Attributes**

### **3.5.1 Simple User Interface**

The user interface has to be as simple and intuitive as possible, the application should allow an average user to set up an account and start using the application understanding its functionality in no more than a dozen minutes. In addition there should be a complete tutorial to make it easy using the application.

### **3.5.2 Reliability**

The application provides a reliable service in which individual users can easily log in and report the violations in the most optimal way. Furthermore it Warranties that the chain of custody of the information coming from the users is never broken, and the information is never altered. This would provide a secure and reliable system. In addition, if the license plate is not readable from the picture the application should warn the user to send another photo.



### **3.5.3 Availability**

The application must offer the maximum availability, granting its service every day at any time (24/7). The lack of service must be minimal. Reporting violation and taking the information about the violation coming from SafeStreets must be active every day at any time. The lack of service is acceptable only if it is due to maintenance. In this case, users must receive a warning 48 hours before.

### **3.5.4 Security**

The application need to be safe and it does not have particular security concerns except the ones related to unauthorized login. The login of Users and especially of authorities must be very safe to avoid reporting. Moreover, the means of communication must be encrypted to save the confidentiality of information sent to SafeStreets.

### **3.5.5 Maintainability**

The application will be maintained and designed in such a way it makes it easier to maintain and it should be understandable for both the users and the authorities. Furthermore, the system will put effort in keeping the live data services (such as highlighting the streets with the highest frequency of violations or the vehicles that commit the most violation) always online.

### **3.5.6 Portability**

Portability of user data from a device to another is possible by entering personal login data. Also the application will be able to run for devices with different operating systems. Trackme wants to focus on the both Android iOS market and Apple iOS, because Android is the largest OS in the world and it is expected that the market share of Apple iOS will increase in the coming years.

## **4 Formal Analysis Using Alloy**

## **5 Effort Spent**