

SOFTWARE ENGINEERING II PROJECT

**SafeStreets**

***RASD*** *–* ***R****equirements* ***A****nalysis and* ***S****pecifications* ***D****ocument*

*Professor*:

Matteo G. ROSSI

*Authors*:

Matteo POZZI

Sara SACCO

Andrea VENTURA

November 10, 2019

*version 1.0*

1. Introduction

**1.1 Purpose**

SafeStreets is a mobile application that relies on the help of lawful citizens to make life in the streets less stressful and more organized. The purpose of this document is to describe in depth said application in terms of functional and nonfunctional requirements, so as to help the customer and the developer be on the same page by identifying the former’s needs, and documenting these in a way that makes analysis, communication, and implementation sustainable for both parties.

**1.2 Scope**

The given problem is to create a software system that meets the stakeholders’ needs, which translate with the intent of providing people with the ability to report and notify violations, e.g. vehicles parked in the middle of bike lanes, or in spots reserved to people with disabilities, to the designated authorities.

In particular, citizens [of age] should be able to register as users by providing meaningful credentials, so as to avoid wasteful data such as fake accounts, and a way to verify them, e.g. ID or driver license. Once successfully logged in, users should be able to send pictures as proof of vehicles parked illegally and attach additional information to provide authorities with a starting point for the reviewing process, such as the date, the time, the type of violation which is to be reported and the place in which it has occurred, which can be retrieved through the geographical position of the user itself. This means the device which the user is working with should at least be equipped with a camera and a GPS system.

SafeStreets stores the information provided by its users and employs it by running an algorithm on the picture to recognize the license plate number. Such process could be made quicker by the input of the user itself, who is given the option of inserting the license plate information as plain text while filling out their submission. If that were the case, the system should use such information as a starting point for the recognition process, though the algorithm should be run nonetheless as a way of double-checking the information. The stored data can then be elaborated by both end users and the designated authorities to highlight the zones which are found to be subject to the highest amount of violations.

Furthermore, SafeStreets wants to exploit its own data by combining it with information about accidents and analyzing it in order to identify zones or streets whose safety could be improved by making interventions, possibly suggesting viable solutions as well. This functionality is developed in collaboration with a third party, i.e. the municipality, meaning its usefulness will depend on the possibility of the municipality itself to share its data and match it with the interface SafeStreets developed for the functionality.

Lastly, SafeStreets strives to assist the local police in generating traffic tickets, and possibly build various statistics of interest. To ensure the effectiveness of this service, it is necessary that the exchange of sensible data which must occur between SafeStreets and the municipality cannot be tampered with in any way, e.g. modifying the picture of the violation at hand. To avoid this scenario, SafeStreets should only accept as reliable information pictures that have been taken within the application itself, meaning it should be equipped with an internal camera system.

In the following diagram (Figure 1), we define the boundaries of SafeStreets by identifying and distinguishing between World and Machine phenomena, with particular attention to the shared ones.



*Figure 1. World and Machine phenomena.*

*1.2.1 Goals*

[**G1**] Allow Citizens to report traffic violations;

[**G2**] Allow Citizens to view a history of their past reports;

[**G3**] Allow Authorities to access stored data about submitted reports and Citizens to view data about accepted ones with limited information;

[**G4**] Allow Authorities to send reports about accidents;

[**G5**] Allow Users to highlight areas with most violations;

[**G6**] Allow Users to highlight areas that are unsafe;

[**G7**] Allow the local Authority to generate traffic tickets;

[**G8**] Allow an Authority to link issued traffic tickets to relative Citizens reports;

[**G9**] Allow Users to visualize statistics posted by system managers;

[**G10**] Allow Users to visualize possible interventions for areas that are deemed unsafe;

**1.3 Definitions, acronyms, abbreviations**

- Definitions:

**User**: a general actor which is registered into the application; all users can consult statistics about violations and highlight unsafe areas;

**Authority**: a user which receives complaints and is able to identify actual violations among them. It has the power to punish the culprits with traffic tickets;

**Citizen**: a user which is not an authority, he can send reports about violations;

**Violation**: a violation of traffic laws, in particular parking violations;

**Accident**: a traffic event involving two or more vehicles where people got injured or caused damages to the vehicles

**Report**: a notification sent by a citizen to indicate violations, containing all the meaningful information about it;

**Traffic ticket**: a sanction which force an offender of a violation to pay an amount of money, can be generated by authorities;

**Unsafe area**: an area in which many violations and accidents have been reported;

**Statistics**: a collection of data about violations, accidents or both regarding a certain area.

- Acronyms:

**RASD**: Requirements Analysis and Specifications Document;

**GPS**: Global Positioning System;

**API**: Application Programming Interface.

- Abbreviations:

[**Gn**]: n-th goal;

[**Dn**]: n-th domain property;

[**Rn**]: n-th requirement.

**1.4 Revision history**

**1.5 Reference documents**

**1.6 Document structure**

The document at hand is composed of 6 chapters:

1. Introduction: it includes the goal of the project and an analysis of the world and shared phenomena;
2. Overall description: here we provide further details on the shared phenomena, as well as user characteristics and domain assumptions;
3. Specific requirements: this section provides more details on the aspects presented in Chapter 2;
4. Formal analysis using alloy: it includes a brief presentation of the main objectives of the formal modeling activity, and a description of the model itself;
5. Effort spent: it contains a quantitative description of the effort each member put into the completion the document;
6. References.
7. Overall description
   1. **Product perspective**
   2. **Product functions**
   3. **User characteristics**
   4. **Domain assumptions**
8. [D1] Each User is unique;
9. [D2] An User notification is evaluated by an Authority within a week; *(non-functional requirement)*
10. [D3] A Citizen sends a report about a violation when he notices it;
11. [D4] Information about date and time of the violation corresponds to the date and time when the report is sent;
12. [D5] Information about position of the violation is collected through GPS;
13. [D6] Picture of violations are taken at the moment and are not inserted in a second time or from already saved pictures;
14. [D7] Violations can only be reported through the mobile application; *(design choice)*
15. [D8] Each Citizen reports a certain violation once;
16. [D9] Authorities have tools for assessing if a violation included in a report is an actual violation or not;
17. [D10] Authorities generate traffic tickets only for actual violations;
18. [D11] Authorities are able to find the owner of the vehicle by the license plate, which is unique to each car vehicle;

Use Cases

|  |  |
| --- | --- |
| Name | Guest registration |
| Goals | [G1]…[G10] |
| Actors | Guest |
| Entry conditions | The guest has downloaded the application and launched it |
| Events flow | 1. The guest chooses the “sign up” option. 2. The guest selects if he wants to register either as an Authority or a Citizen. 3. The guest fills all mandatory data concerning the chosen category. 4. The guest confirms the operation by selecting the confirmation option. 5. The system saves the data. |
| Exit conditions | The Guest has become an User and can now access the application function offered to the chosen category. The system has saved the data about the User |
| Exceptions | 1. The Guest is already registered into the application. In this case the system invites him to execute the “sign in” operation. 2. One or more of the mandatory fills contain invalid input. In this case the system send a warning to the Guest and invites him to correct them. |

|  |  |
| --- | --- |
| Name | Sign in |
| Goals | [G1]…[G10] |
| Actors | User |
| Entry conditions | The User is registered to the application and is on the home page. |
| Events flow | 1. The User selects the “sign in” option. 2. The User inserts his credentials into the fields. 3. The User selects the confirmation option. 4. The system redirects the User to his personal home page. |
| Exit conditions | The system recognizes the User as registered and redirects him successfully. |
| Exceptions | 1. The User inserts invalid credentials. In this case the system warns the User and invites him to re-insert them. |

|  |  |
| --- | --- |
| Name | Check areas with most violations |
| Goals | [G5] |
| Actors | User |
| Entry conditions | The User is logged into the system and is on the home page. |
| Events flow | 1. The User selects the “view areas with most violations” option. 2. The system provides the User with a map centered on the User’s current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. Eventually, the User specifies a filter about violations to look for. 5. The User consults the areas with the most specified violations. |
| Exit conditions | The system has provided the User with all the areas within the desired range from the selected location in which the specified violations have occurred the most. |
| Exceptions | 1. The system couldn’t get the User’s actual position. In this case the system centers the map onto the last visited position and the execution gets to point 3. 2. There are no areas within the specified zone where the filtered violations (or all) occurred. In this case the system shows a message to the User and suggests him to remove some filters or to enlarge the research area. |

|  |  |
| --- | --- |
| Name | Check unsafe areas |
| Goals | [G6] |
| Actors | User |
| Entry conditions | The User is logged into the system and is on the home page. |
| Events flow | 1. The User selects the “view unsafe areas” option. 2. The system provides the User with a map centered on the User’s current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. The User consults the areas highlighted as unsafe within the specified range. |
| Exit conditions | The system has provided the User with all the areas highlighted as unsafe within the desired range from the selected location. |
| Exceptions | 1. The system couldn’t get the User’s actual position. In this case the system centers the map onto the last visited position and the execution gets to point 3. 2. There are no unsafe areas within the specified zone. In this case the system shows a message to the User and suggests him to enlarge the research area. |

|  |  |
| --- | --- |
| Name | View verified reports |
| Goals | [G3] |
| Actors | User |
| Entry conditions | The User is logged into the system and is on the home page. |
| Events flow | 1. The User selects the “view verified reports” option. 2. The system provides the User with a map centered on the User’s current position. 3. The User selects an area onto the map and, eventually, specifies a radius to explore the map in a circumference specified by it. 4. The system shows a list of all accepted reports coming from the specified area. If the User is a Citizen, the reports will be shown with limited information. 5. The User consults the list of reports. |
| Exit conditions | The system has provided the User with a list of verified reports which have been accepted by an Authority and have occurred within the specified area. |
| Exceptions | 1. The system couldn’t get the User’s actual position. In this case the system centers the map onto the last visited position and the execution gets to point 3. 2. There are no verified reports within the specified zone. In this case the system shows a message to the User and suggests him to enlarge the research area. |

|  |  |
| --- | --- |
| Name | View statistics |
| Goals | [G9] |
| Actors | User |
| Entry conditions | The User is logged into the system and is on the home page. |
| Events flow | 1. The User selects the “view statistics” option. 2. Continue later… |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Name | View possible interventions |
| Goals | [G10] |
| Actors | User |
| Entry conditions | The User has just looked for unsafe areas within a certain zone. |
| Events flow | 1. The User consults the list of all unsafe areas found. 2. The User selects one among the unsafe areas. 3. The User selects the “view suggested interventions” option. 4. The User consults all the possible interventions for the unsafe area. |
| Exit conditions | The system has provided the User with a list of all possible interventions for the selected unsafe area. |
| Exceptions | 1. There are no suggested interventions for the selected area. In this case the system shows the User a message to explain this fact. |

|  |  |
| --- | --- |
| Name | Issue report |
| Goals | [G1] |
| Actors | Citizen |
| Entry conditions | The Citizen has spotted a possible violation and wants to report it. |
| Events flow | 1. The Citizen selects the “report a violation” option. 2. The Citizen selects the type of violation from a list containing the reportable ones. 3. The Citizen selects the “take a photo” option. 4. The Citizen takes a photo of the involved vehicle making sure that the license plate is visible and readable. 5. The system analyzes the photo. 6. The system accepts the photo and takes the Citizen back to the report form. 7. The Citizen selects the “send your position” option. 8. The system collects the actual position of the Citizen. 9. The Citizen confirms the operation by selecting the relative option. 10. The system receives the data and saves them. |
| Exit conditions | The system has received the incoming data correctly and managed to store them. |
| Exceptions | 1. The system couldn’t recognize or read the license plate from the photo. In this case the system warns the Citizen about the issue and asks him to take the photo again. 2. The system couldn’t retrieve the Citizen’s current position. In this case, the system warns the Citizen about the issue and asks him to activate his GPS. 3. The system didn’t receive the data about the report. In this case the system shows the Citizen an error message and asks him to redo the whole operation. |

|  |  |
| --- | --- |
| Name | View past reports |
| Goals | [G2] |
| Actors | Citizen |
| Entry conditions | The Citizen is logged into the system and is on the home page. |
| Events flow | 1. The Citizen selects the “my profile” option. 2. The system redirects the Citizen to his personal profile. 3. The Citizen selects the “my reports” option. 4. Eventually, the Citizen enters a time filter to exclude too “old” reports. 5. The Citizen consults a list of his submitted reports in the desired time period. |
| Exit conditions | The system has shown a list of the Citizen’s past reports submitted in a time such that the filter is satisfied, including a description of the report’s status. |
| Exceptions | 1. The Citizen never submitted a report. 2. There are no reports that satisfy the inserted filter.   In both cases, the system shows a message to the Citizen describing the issue. |

|  |  |
| --- | --- |
| Name | Evaluate submitted report |
| Goals | [G3] |
| Actors | Authority |
| Entry conditions | The Authority is logged into the system and is on the home page. |
| Events flow | 1. The Authority selects the “check submitted reports” option. 2. The system provides the Authority with a list of all not already evaluated reports from its areas of competence, with a brief description. 3. The Authority selects a report among the list. 4. The system expands the selected report including all its data. 5. The Authority checks the data of the selected report. 6. The Authority evaluates the report as accepted or rejected. 7. The system saves the new status of the report. |
| Exit conditions | The report’s new status is either accepted or rejected and the system saves the status according to the Authority’s decision. |
| Exceptions | 1. All the reports of competence of the Authority have already been evaluated. In this case the system shows the Authority a message explaining the issue. 2. The Authority doesn’t choose between accepting or rejecting the report. In this case the system warns the Authority and asks him to choose between interrupting the process or complete it. If the process is interrupted the report maintains its precedent status, else the process resumes from where it was suspended. |

|  |  |
| --- | --- |
| Name | Generate traffic ticket |
| Goals | [G7] [G8] |
| Actors | Authority |
| Entry conditions | The Authority has just evaluated a report as accepted. |
| Events flow | 1. The system asks the Authority if he wants to generate a traffic ticket from the evaluated report. 2. The Authority takes a decision on this. If it accepts, then the process continues to step 3, else it is terminated. 3. The system asks the Authority the amount of money to include in the traffic ticket. 4. The Authority inserts the amount of money. 5. The Authority confirms the operation. 6. The system emits the traffic ticket and it associates it to the license plate of the vehicle which committed the violation. 7. The system associates the emitted traffic ticket to the evaluated report. |
| Exit conditions | If the traffic ticket isn’t emitted, the report is saved just as accepted.  If the traffic ticket is emitted, it is sent to the owner of the vehicle found by license plate, the system saves the new status of the report with the issued traffic ticket. |
| Exceptions | 1. The Authority inserts an invalid amount of money (negative or too high). In this case the system warns the Authority and asks him to insert a valid value. |

|  |  |
| --- | --- |
| Name | Report accident |
| Goals | [G4] |
| Actors | Authority |
| Entry conditions | The Authority has all the information about an accident occurred in its area of competence. |
| Events flow | 1. The Authority selects the “report accident” option. 2. The Authority inserts the data about date and time when the accident occurred. 3. The Authority inserts the data about the location where the accident occurred. 4. The Authority inserts the data about the type and how many vehicles were involved in the accident. 5. The Authority inserts the data about how many people were injured in the accident. 6. The Authority inserts data about if any emergency vehicle was dispatched and if it was able to reach the place in good time. 7. The Authority inserts a brief description including any other relevant data in order to fully describe the accident. 8. The Authority confirms the operation and send the report. 9. The system receives the data about the report and saves them. |
| Exit conditions | The system has received the report and managed to store the data. |
| Exceptions | 1. The Authority inserts an invalid input in any of the fields (most likely the date). In this case the system warns the Authority and asks him to correct the input. |

|  |  |
| --- | --- |
| Name | Report unsafe area |
| Goals | [G6] missing goal? |
| Actors | System Manager |
| Entry conditions | The System Manager has noticed a dangerous area because of many accepted violations and/or accidents |
| Events flow | 1. The System Manager selects the “areas management” option. 2. The System Manager searches the area he wants to highlight. 3. The System Manager selects the area. 4. The System Manager marks the area as “unsafe”. 5. The System Manager confirms the modification. 6. The system saves the new status of the area. |
| Exit conditions | The system has successfully saved the modification. |
| Exceptions | / |

|  |  |
| --- | --- |
| Name | Suggest intervention |
| Goals | [G10] missing requirement? |
| Actors | System Manager |
| Entry conditions | The System Manager has noticed the same violation or accident occurring several times in a certain area. |
| Events flow | 1. The System Manager selects the “area management” option. 2. The System Manager searches the area he wants to revise. 3. The System Manager selects the area. 4. The System Manager selects the “suggest intervention” option. 5. The System Manager fills with a brief description the possible intervention to prevent other violations to occur. 6. The System Manager confirms the modification. 7. The system saves the suggestion description in the area. |
| Exit conditions | The system has successfully saved the modification. |
| Exceptions | / |