

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;

[**G4**] Allow Users to highlight areas with most violations;

[**G5**] Allow Users to highlight areas that are unsafe;

[**G6**] Allow the local Authority to generate traffic tickets;

[**G7**] Allow an Authority to link issued traffic tickets to relative Citizens signalizations;

[**G8**] Allow Users to visualize statistics posted by system managers;

[**G9**] 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]…[G11] |
| 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]…[G11] |
| 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. |