

AA 2019/2020

Computer Science and Engineering

Software Engineering 2 Project

SafeStreets

**RASD**

Requirement Analysis and Specification Document

version 1.0 --- 04/11/2019

|  |  |
| --- | --- |
| Authors: | Professor: |
| Aida Gasanova | Elisabetta Di Nitto |
| Alexandre Batistella Bellas |  |
| Ekaterina Efremova |  |

**Table of Contents**

[1 Introduction 4](#_Toc23978012)

[1.1 Purpose 4](#_Toc23978013)

[1.1.1 General purpose 4](#_Toc23978014)

[1.1.2 Goals 4](#_Toc23978015)

[1.2 Scope 5](#_Toc23978016)

[1.3 Definitions, acronyms and abbreviations 6](#_Toc23978017)

[1.3.1 Definitions 6](#_Toc23978018)

[1.3.2 Acronyms 6](#_Toc23978019)

[1.3.3 Abbreviations 6](#_Toc23978020)

[1.4 Revision history 6](#_Toc23978021)

[1.5 Reference documents 6](#_Toc23978022)

[1.6 Document structure 7](#_Toc23978023)

[2 Overall Description 8](#_Toc23978024)

[2.1 Product perspective 8](#_Toc23978025)

[2.2 Product functions 11](#_Toc23978026)

[2.2.1 Violations management 11](#_Toc23978027)

[2.2.2 Interventions management 11](#_Toc23978028)

[2.2.3 Traffic tickets management 11](#_Toc23978029)

[2.3 User characteristics 12](#_Toc23978030)

[2.3.1 Actors 12](#_Toc23978031)

[2.4 Assumptions, dependencies and constraints 12](#_Toc23978032)

[3 Specific Requirements 13](#_Toc23978033)

[3.1 External interface requirements 13](#_Toc23978034)

[3.1.1 User interfaces 13](#_Toc23978035)

[3.1.2 Hardware interfaces 13](#_Toc23978036)

[3.1.3 Software interfaces 13](#_Toc23978037)

[3.1.4 Communication interfaces 13](#_Toc23978038)

[3.2 Functional requirements 13](#_Toc23978039)

[3.3 Performance requirements 13](#_Toc23978040)

[3.4 Design constraints 13](#_Toc23978041)

[3.4.1 Standards compliance 13](#_Toc23978042)

[3.4.2 Hardware limitations 13](#_Toc23978043)

[3.4.3 Any other constraint 13](#_Toc23978044)

[3.5 Software system attributes 13](#_Toc23978045)

[3.5.1 Reliability 13](#_Toc23978046)

[3.5.2 Availability 13](#_Toc23978047)

[3.5.3 Security 13](#_Toc23978048)

[3.5.4 Maintainability 13](#_Toc23978049)

[3.5.5 Portability 13](#_Toc23978050)

[3.6 Other requirements 13](#_Toc23978051)

[4 Formal analysis with Alloy modeling 14](#_Toc23978052)

[5 Appendices 15](#_Toc23978053)

[5.1 Used tools 15](#_Toc23978054)

[5.2 Hours of effort spent 15](#_Toc23978055)

1. Introduction
   1. Purpose
      1. General purpose

This document represents the Requirement Analysis and Specification Document (RASD). The main purpose of this document is to fully describe the software product in order to help developers model it.

This document describes SafeStreets application, which can help the prevention of traffic violations, and in particular parking violations, by sending the information obtained by ordinary pedestrians who are users of this application to the authorities. Both sides, users and authorities, can use this data for useful purposes, for example, see areas that have a high frequency of violations, or even the vehicles that commit the most violations. Besides this, the application can cross its own data with external data from municipally (if available) to identify potentially unsafe areas and suggest possible interventions. Lastly, the application can provide information to a municipally system that emits traffic tickets to people that committed violations, since the information came from the application is with guaranteed integrity.

* + 1. Goals
* G1: The system must allow users to report traffic violations, and in particular parking violations;
* G2: The system must identify the license plate automatically from the picture sent by a user, storing the information, beside all provided by the user, in a local database;
* G3: The application must allow users and authorities to mine the information stored in the system;
* G4: The application must cross its information with the information about the accidents that occur on the territory of the city;
* G5: The system must suggest possible interventions to potentially unsafe areas;
* G6: The system must offer to users and authorities truthful information about the registered violations;
* G7: The application must build statistics from the issued tickets;

The goals G4 and G5 can only be considered if the municipality offers a service that allows information retrieval about the accidents that occur on the territory of the municipality.

* 1. Scope

The SS service is a crowd-sourced application offered to common users and authorities that want to follow the violations occurred on the municipality territory.



**Figure 1.** SS as a sharing information system

The system has various directions of action. First, the largest mass of users are ordinary people. Everyone can download the application to their phone and use it to the benefit of the welfare of their city.

Every time when the citizens see the violation, they can take a photo and upload it to this application. Based on this information, a register of violations will be compiled for mapping threats on the streets of the city.

When the app receives a picture, it runs an algorithm to read the license plate, and stores the retrieved information with the violation, including also the type of the violation and the name of the street where the violation occurred. In addition, the application allows both end users and authorities to mine the information that has been received, by highlighting the streets (or the areas) with the highest frequency of violations, and the vehicles that commit the most violations. In this case there are more user's levels lake the municipality and authorities, and different levels of visibility are offered to different roles.

Another, also a very important part of users is the municipality. Its role is divided into two parts. First, the municipality can upload accident information to the application, thereby complementing existing databases. The application mixes information from users and from the municipality. As a result, we get a new map with more relevant information about the city. Secondly, the municipality can receive information about offenses from the application. This application initially checks the accuracy of the information (including date, Photoshop using, etc.). Based on this information, the municipality, in cooperation with the police, may issue fines.

Police could also offer a service that takes the information about the violations coming from SS and generates traffic tickets from it. In this case, mechanisms should be put in place to ensure that the chain of custody of the information coming from the users it never broken, and the information is never altered.

* 1. Definitions, acronyms and abbreviations
     1. Definitions
* **User:** the “normal” customer of the application that send the information about the violations to authorities or extract the information that have been received (to use it for useful purposes);
* **Authorities:** the customer of the application that receive the information about violations that have been received from “normal” customers;
* **Customer:** general *SafeStreets* customer;
* **Municipality:** a town or district that has local government;
* **Violation:** general traffic violation, and in particular parking violation;
  + 1. Acronyms
* **API:** Application Programming Interface
* **GPS:** Global Positioning System
* **UI:** User Interface
* **AI:** Artificial Intelligence
* **RASD:** Requirement Analysis and Specification Document
* **SS:** SafeStreets
  + 1. Abbreviations
* **Gn:** n-th goal
* **Dn:** n-th domain assumption
* **Rn:** n-th functional requirement
  1. Revision history
* 1.0.0 – Release version
  1. Reference documents
* Specification document: “Mandatory Project Assignment AY 2018-2019”
* IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
* UML diagrams: https://www.uml-diagrams.org/
* Alloy doc: <http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html>
  1. Document structure

The RASD document is composed by five chapters, as outlined below.

**Chapter 1** is an introduction: it describes the purpose of the system informally and also by making use of the list of goals which the application has to reach. Moreover, it defines the scope, where the aim of the project is defined in greater detail and the application domain and the most important shared phenomena are shown.

**Chapter 2** offers an overall description of the project. Here the actors involved in the application’s usage lifecycle are identified and the boundaries of the project are defined, listing all the necessary assumptions. Furthermore, a class diagram is provided, aid to better understanding the general structure of the project, with all the related entities. Then some state diagrams are listed to make 10 the evolution of the crucial objects clear. Finally, the functions offered by the system are here more clearly specified, with respect to the previously listed goals.

**Chapter 3** represents the body of the document. It contains the interface requirements, which are: user interfaces, hardware interfaces and software interfaces. It then lists some scenarios to show how the system acts in real world situations, followed by the description of the functional requirements, using use cases and sequence diagrams. All the requirements necessary in order to reach the goals are given, linked with the related domain assumptions. Lastly, the non-functional requirements are defined through performance requirements, design constraints and software system attributes.

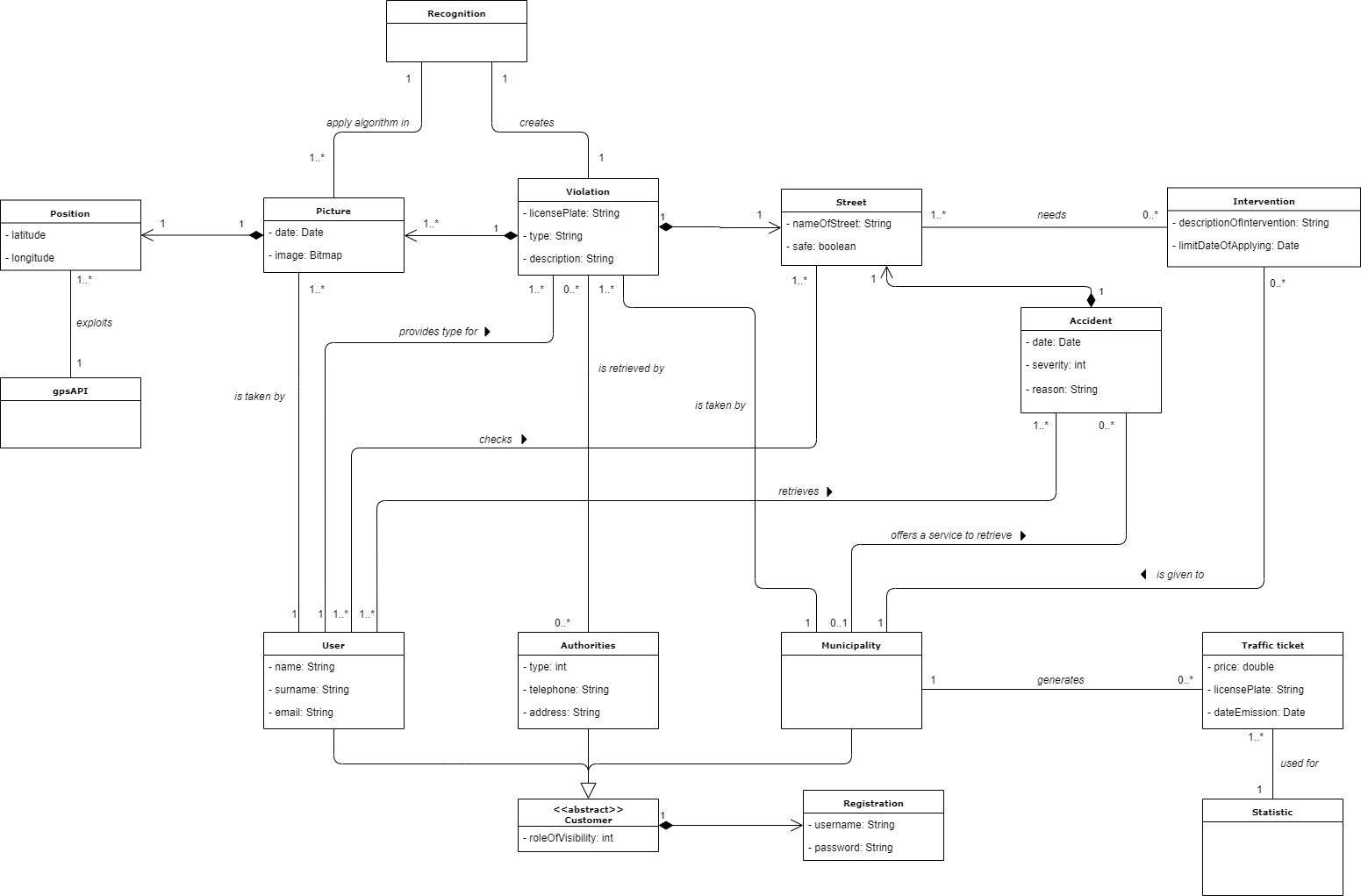
**Chapter 4** contains the Alloy model of some critical aspects with all the related comments and documentation in order to show how the project has been modeled and represented through the language.

**Chapter 5** contains the list of the tools used and shows the effort which each member of the group spent working on the project.

1. Overall Description
   1. Product perspective

The system will be developed from scratch, and it’ll make interface with the service provided by the municipality when dealing about the accidents occurred on the territory of the own municipality and when providing the violations to it as well. Besides this, the application will need to use the device’s GPS service to register the location where the pictures were taken, and also the API from an online service for showing a map inside the application, specifically for the function of highlighting the streets with higher numbers of violation.

Below there is the class diagram of the application, with a high-level visualization of the most important classes that will be implemented by the developers. In the diagram is clear the relations between each actor of the system, for example, every user can provide pictures of a seen violation, upload in the system and make it available for the authorities to check, pointing the relation between users and authorities.



**Figure 2**. Class diagram of the software project

Now we are going to model the dynamic behavior of the system to represent the condition of the system or part of the system at finite instances of time. To describe the states of different objects in its life cycle we will use statechart diagrams. States can be identified as the condition of objects when a particular event occurs. These diagrams reported below.

Изображение выглядит как устройство, счетчик

Автоматически созданное описание

**Figure 3**. State diagram 1 – Receiving the information from user.

The figure 1 represents receiving an information about a violation from user to *SafeStreets*. On the event of an information being received, we transit from our initial state to *Unprocessed information* state. The information can be rejected or accepted depending on ensuring that the chain of custody of the information coming from the users is never broken, and the information is never altered. Also, we need to be sure that the information coming from users is relevant. If the order is rejected, we transit to the *Rejected Information* state. If the information is accepted and it’s completed, we transit to the *Saved information* state. If the information is not completed, we need to finalize it with corresponding metadata. For example, when we receive the picture, it launches the license plate reading algorithm. When we have all the information about the violation, including the type of the violation and the name of the street name where the violation occurred, *SafeStreets* saves the data. After the information is completed, we transit to the final state.

Изображение выглядит как объект, красный

Автоматически созданное описание

**Figure 4.** State diagram 2 – Mining the information from SafeStreets.

In this state diagram (figure 4) before providing the information to customers, first we need to check the level of visibility. For example, both and authority and users can see highlighting the streets with the highest frequency of violations, but only authority can see vehicles that commit the most violations.

Изображение выглядит как объект

Автоматически созданное описание

**Figure 5.** State diagram 3 – Using an information from accidents service in SafeStreets.

The figure 5 shows that SS can cross the information about accidents provided by service of municipality with its own data (if the information is relevant) and suggest possible interventions (e.g., add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

Изображение выглядит как небо

Автоматически созданное описание

**Figure 6.** State diagram 4 – Using an information from SafeStreets in traffic tickets service.

On the state diagram 4 we can see that another service, provided by municipality that can use the information about violations from SS to generate traffic tickets and then provide the tickets to SS, so that SS can build some statistics.

* 1. Product functions

For this section, it’s defined all the most important product functions for the development of the project. Since that the application depends on external services, as the one provided by the municipality (if provided), it’s needed to define different functions for each service provided by each user of the system.

* + 1. Violations management

As the main function of the system (basic service), all the users of the application will be allowed to do a registration putting some of their own personal data, and choosing an available role (citizen or authority), all of this information being mandatory to insert. In case of being authority, a more complex verification will be needed, as a photo of the work license and a specific interview. Otherwise, only a photo of the personal document will be needed for verification. For municipality, a professional contact will be needed for providing the offered services from the application via own API, and no registration inside the app is need.

After registration, the users with the role of citizens will be able to upload to the application pictures taken from their mobile devices of violations seen in the street. These pictures, translated as instances of violations inside the app through the recognition algorithm, can be retrieved as a map highlighting the streets with the highest frequency of violations (for both citizen and authority) and a ranking of vehicles with most violations (only for authorities). Besides this, all the violations can be retrieved only by the municipality for applying traffic tickets on the violators.

* + 1. Interventions management

This function will only be performed by the system if there is a service from the municipality offering information about the accidents in the territory of itself. In case of this statement is accomplished, all the information about the accidents is retrieved by SS via municipality’s API, and with this, the data stored in the system (about violations) will be crossed with the new data (about accidents). If the cross can be performed, an analysis is applied and a possible intervention is created for the cross, being sent to the municipality as a suggestion.

* + 1. Traffic tickets management

Lastly, the municipality can have the benefit from the system service about the violations occurred in the territory of itself, retrieving it by an API offered by the system. Thus, it can generate traffic tickets for the offenders reported by the users through the application. From the violations management, the information provided by the users will be always verified and processed to be legit for the municipality apply all the punishments. After issuing the tickets, this information will be used by SS to build statistics about the effectiveness of the SS initiative.

* 1. User characteristics
     1. Actors
* **User:** a person who uses the application not only for viewing but also can control the situation in more detail. He gives photos of violations and upload them to applications for further verification.
* **Authorities:** this organization can track road information as a regular user. It also has access to view user information.
* **Municipality:** this is an organization that has the right to view relevant information. It can also download accident information for further processing by the application. Police could also offer a service that takes the information about the violations coming from SS and generates traffic tickets from it.
  1. Assumptions, dependencies and constraints

D1:

The user does not have access to all the information stored in the application.

1. Specific Requirements
   1. External interface requirements
      1. User interfaces
      2. Hardware interfaces
      3. Software interfaces
      4. Communication interfaces
   2. Functional requirements
   3. Performance requirements
   4. Design constraints
      1. Standards compliance
      2. Hardware limitations
      3. Any other constraint
   5. Software system attributes
      1. Reliability
      2. Availability
      3. Security
      4. Maintainability
      5. Portability
   6. Other requirements
2. Formal analysis with Alloy modeling
3. Appendices
   1. Used tools

The tools used for the development of this document were those ones listed below.

* Microsoft Office Word 2016
* draw.io
* GitHub
* Alloy Analyser 4.2
  1. Hours of effort spent

The hours spent by the group are listed below, differentiating for each participant.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task** | **Hours spent** | | |
| **Aida Gasanova** | **Alexandre**  **Batistella Bellas** | **Ekaterine Efremova** |
| Introduction | 4 | 4 | 1.5 |
| Product perspective | 5 | 4 | 0 |
| Product functions | 0 | 3 | 0.5 |
| Domain assumptions |  |  |  |
| External interface requirements |  |  |  |
| Functional requirements |  |  |  |
| Non-functional requirements |  |  |  |
| Formal analysis using Alloy |  |  |  |