Front Page Test

AmorosiniCasaliFioravanti

October 2019

Contents

Introduction

This document represents the Requirement Analysis and Specification Document (RASD) for SafeStreets: an application that aims to improve the safety of urban areas by giving its users the possibility to report parking violations and accidents to authorities. The goal of this document is to supply a description of the system in terms of its functional and non functional requirements, listing all of its goals, discussing the constraints and the limits of the software, and indicating the typical use cases that will occur after the release. This document is addressed to the stakeholders, who will evaluate the correctness of the assumptions and decisions contained in it, and to the developers who will have to implement the requirements.

1.1 Purpose

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations and accidents occur. In order to report a traffic violation, users have to compile a report containing a picture of the violation, the date, time, position, and type of violation. SafeStreets stores all the information provided by users, completing it with suitable metadata (timestamps, sender's GPS if available, etc.). In particular, in case the user had not provided any information regarding the license plate of the car breaking the traffic rules, SafeStreets runs an algorithm to read it from the submitted picture. Moreover, the application allows both end-users and authorities to mine the information that has been received: this function allows users to perform useful statistics, for example to find out which area is more dangerous or which vehicles commit the most violations. If the municipality offers a service that allows users to retrieve the information about the accidents that occur in its territory, SafeStreets can cross this information with its own data to identify potentially unsafe areas, and suggest possible interventions. In addition, the municipality could offer a service that takes the information about the violations coming from SafeStreets, and generates traffic tickets from it. In this case, SafeStreets ensures that the chain of custody of information coming from users is never broken, thus information is never altered. Information about

issued tickets can be used by SafeStreets to build statistics, for example to have a feedback on the effectiveness of the application.

1.1.1 Goals

- [G.1] The System allows the Users to access the functionalities of the application from different locations and devices.
- [G.2] The System allows Guests to authenticate themselves either as Authority or Citizen.
  - [G.2.1]The System offers different levels of visibility to different roles.
- [G.3] The System allows the Citizens to document traffic violations and accidents to Authorities by compiling a Report.
- [G.4] The System stores the reports provided by the Citizens.
  - [G.4.1] If the Citizen has not provided any information about the license plate, the System runs an algorithm to read it from the submitted picture.
- [G.6] The System analyzes the stored information to detect unsafe areas.
  - [G.6.1]If the System collects multiple reports for the same problem in a given area, it can suggest possible interventions to the Authorities.
  - [G.6.2]If allowed by Authorities, SafeStreets can cross their information about accidents with its own in order to improve the analysis.
- [G.7] If allowed by Authorities, SafeStreets can access their information about issued tickets to build statistics and evaluate its effectiveness.
  - [G.7.1] The System ensures that the chain of custody of the information coming from the Citizens is never broken.

1.2 Scope

According to the *World* and *Machine* paradigm, proposed by M. Jackson and P. Zave in 1995, we can distinguish the *Machine*, that is the portion of the system to be developed, from the *World*, that is the portion of the real-world affected by the machine. In this way, we can classify the phenomena in three different types: World, Machine and *Shared* phenomena, where the latter type of phenomena can be controlled by the world and observed by the machine or controlled by the machine and observed by the world.
In this context, the most relevant phenomena are organized as in the following table.

World	Shared	Machine
Traffic violation: circumstance in which someone doesn't respect the traffic rules	**Registration/Login**[1]: a Guest can sign up to the application or log in if already registered.	**DBMS query**: operation performed to retrieve/store data
Accident:event that caused damage to things or people.	**Commit of a report**[1]: action of sending a report documenting a traffic violation or an accident.	**API queries**: request for third-part services.
	Build statistics[2]: computation of statistics to find unsafe areas, to see which veichles commit multiple infractions and also to monitor the trend of issued tickets.	**use of Hash**: compute the hash function to ensure that information is never altered.
	Safety suggestions[2]: to give suggestions for possible safe interventions (e.g add a barrier).	**Credential validation**: checking that Citizens can not access as Authorities
	Send notification[2]: to notify the Autorithies when a report is submitted.	

Table 1.1: In the table above, *1* refers to shared phenomena controlled by the world and observed by the machine, whereas *2* refers to the phenomena controlled by the machine and observed by the world

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- *System*: the totality of the hardware/software applications that contribute to provide the service concerned.

- *Guest*: someone who has yet to sign up and who is not able to access any feature of the application.
- *User*: a registered user who has logged in.
  - *Citizen*: end-user who can send Reports of traffic violation. It has limited visibility over the stored information.
  - *Authority*: user who has access to the history of Reports and is notified whenever a new Report is recieved. It has full visibility over the data exposed by the System.
- *Report*: documentation of a traffic violation. It contains a picture of the violation, the date, time, position, and type of violation. Some fields can be omitted in case an accident is being reported.
- *Hash*: a mathematical algorithm that maps data of arbitrary size to a bit string of fixed size. It is a one-way function, that is, a function which is practically infeasible to invert.
- *Chain of custody*: chronological documentation that records the sequence of custody, control, transfer, analysis, and disposition of physical or electronic evidence.
- *Third-party services*: services used by the System in order to provide extra functionalities (e.g. image recognition).

1.3.2 Acronyms

- **API**: *Application Program Interface*, set of routines, protocols and tools for building software applications on top of this one.
- **OCR**: *Optical Character Recognition*, software dedicated to the detection of characters contained in a document and to their transfer to digital text that can be read by a machine. In this context, OCR will be used to read license plates.
- **GPS**: *Global Positioning System*, system widely used to get the user's position.
- **DBMS**: *Data Base Management System*, system which provides organized space memory to store information.

1.3.3 Abbreviations

- [G_i]: i-th goal.
- [R_i]: i-th functional requirement.
- [D_i]: i-th domain assumption.

1.3.4 Reference Documents

1.3.5 Document Structure

The rest of the document is organized as follows:

- **Overall description**: this section gives a general description of the application, focusing on the context of the system and giving further details about shared phenomena. Furthermore, we will provides the specifications of constraints, dependencies and assumptions, that show how the System is integrated in the real world.
- **Specific requirements** this section goes into the details about functional and non-functional requirements and a list of all possible interctions with the System is provided using use cases and sequence diagrams.

Overall Description

2.1 Product Perspective

We are going to analyze all the shared phenomena that are listed in the previous section. The concepts are clarified throw class and state diagrams.

Registration/Login (world controlled, machine observed)
A User can sign up to the application or log in, if is already registered. In the case of a new submit, the System provides a form which the User have to fill with his data, specifying if he/she is a Citizien or an Authority. Then the data are collected in the associated DBMS.

Commit of a report (world controlled, machine observed)
A User can send a report with a description of the violation compiling a structured field. He/she has the possibility to attach a picture. The System will check if the informations inserted in the fields are consistent and complete, then will send them to the DBMS.

Build statistics (machine controlled, world observed)
The System computes general statistics on the violations, and also finds unsafe areas. These general information can be seen by the Citizens. Other specific statistics are accessible only by Authorities an example is the vehicles which commits multiple infractions, and also to see how the trends of issued tickets change.

Safety suggestions (machine controlled, world observed)
The System gives suggestions for possible safe interventions. His knowledge is based on data statistics.

Send Notification (machine controlled, world observed)
The System notifies a logged Authority when a report is submitted by an User. Then the Authority takes the right legal measure based on the information that he/she gets.

2.1.1 Class Diagram

https://github.com/FrancescoAmorosini/AmorosiniCasaliFioravan

Figure 2.1: SafeStreets' class diagram.

Specific Requirements

3.1 Functional Requirements

- [G.4] The Report Manager handles the submissions:
  - [G.4.1] Before accepting any Report, the Report Manager ensures that it contains a picture of the violation, the date, time, position, and type of violation.
  - [G.4.2] The Report Manager notifies all logged in Authorities whenever a new Report is accepted.
  - [G.4.3] The Report Manager must ensure that no more than one Authority has taken in charge the same Report.
  - [G.4.4] The Report Manager notifies Citizens whenever one of their Reports has been taken in charge by an Authority.
- [G.9] The System can accedes to the stored traffic tickets with a permission from the local municipality.

their