1. Introduction
   1. Purpose
      1. General Purpose

This document represents the Requirement Analysis and Specification Document (RASD). In this document we will explain SafeStreets. This will be done by a detailed presentation of the proposed solution and its purpose, listing its goals, and the requirements and assumptions through which they will be achieved.

SafeStreets is a public interface aimed to public-spirit citizens who want to help keeping the streets clear. This S2B intends to provide users with the possibility to notify authorities when traffic violations occur. This materializes using a platform through which users can upload pictures of streets violations, in particular parking violations.

There could also be other types of issues that a customer can report, for example speed violations, accidents, non respected traffic lights. They’re reported in different ways.

The S2B also has a map, based on Google Maps, on which some areas are highlighted with different colors according to the number and types of violations reported (for example first the user can choose the type of violation, then the map shows different areas with different colors: a red area means that a lot of the chosen violation have occurred, yellow is medium quantity, green one means very few).

Once the violation is sent, its data are stored in SafeStreets center and analyzed by the software, in order to retrieve information to update the map.

The customers of the application are both singular users and authorities, for example the Police Department, that can find the S2B useful in order to maintain the public order.

An important point is that a user can eventually report fake violations. First, the application allows the user to report a violation even if the user’s geographical position and the violation’s positions are different (for example the user sees an illegal parked car while he is jogging but doesn’t want to stop his run to make the signalization, so he updates the violation once he gets home. This means that he could give a non accurate position of the illegal parking). Secondly, there could actually be some users that find funny reporting wrong violations, for now there is nothing we can do to fix this.

Finally, SafeStreets also wants to offer a service exploiting the information of the municipality, if it allows users to retrieve the required information. The application crosses the information given by the municipality (only accident info), which are reliable, and the ones given by the users, which are not. Then it updates the map and make suggestions regarding possible solutions to prevent violations (only in red areas).

* + 1. Goals

-[G1] The application must allow users to send reports of streets violations. In particular users can send reports even if they aren’t in the position where the infraction occurred.

-[G2] The application will have to store the information about violations and complete them with suitable metadata.

-[G3] The application must allow both end users and authorities to mine the information stored. This is done by coloring the map based on the number of violations.

-[G4] The system must update the map periodically.

-[G5] The system must be able to cross information received from municipality with its own data.

-[G6] The system must suggest to municipality possible interventions to prevent accidents.

* 1. Scope

As already mentioned, the SafeStreets system is made to provide users a service to report streets violations and a map where is possible to watch the areas in a safeness key.

The users, once they sign in with a proper username and password, are in the system and can use all the services offered by the application.

As first, the system will ask to choose either to take a look on the map or to report a street infringement. In the first case, it will be asked if the user wants to share the GPS signal with the application, so to be more precise about the areas close to him. As mentioned before, on the map are highlighted different areas with different colors, based on the frequency of violations. Then the user can interact with the map. If the user taps on a highlighted area, the number of infringements is shown.

It is also possible to select the type of violation and the interval of time (today, last week, last month) in which the user is interested in watching the map. By default, if the user doesn’t select any type of violation or interval of time, it will be shown the map of all types of violations together occurred in the last month.

Regarding to the reporting of violations, the systems allows to select the type of reporting (parking, speed, traffic light violation or accident), only in case of parking is possible to send a picture. The system will ask for a picture with visible license plate, if the doesn’t send one, he can type the license plate by himself (not mandatory).

To prevent any misuse of the reporting violation system, if the user sends a picture with visible license plate and the application analyzing pictures algorithm fulfill in recognizing the it, then that car will be shared with authorities. Otherwise, either if the user only types the license plate without sending a picture of it or the application analyzing pictures algorithm can’t recognize the plate, it is considered not reliable, so the car won’t be shared with authorities. It will only be added on the map.

For speed infraction, traffic light violation or accident reporting, is not possible to send a picture, in order to prevent the use of phones while driving. The user can only choose the type of violation.

The user has also to choose the time and geographical position of the infraction, which is independent from the type of violation. It’s either possible to select the user’s position (using the GPS) or selecting a position on the map or typing the address.

The user can’t open the gallery on his smartphone to update a picture. This is for preventing the update of fake parking pictures that were shot in a time we can’t know. The user can open the camera through SafeStreets application and, either take the picture and send it, or take a picture and, if he doesn’t have time to send it, the S2B puts a timer of 2 hours on the photo, in which the user can re-open the application and update the picture. If the timer expires the picture gets eliminated.

If the municipality offers a service through which is possible to retrieve information about the accidents, SafeStrets analyze them and crosses them with the SafeStreets’s ones. This is done in order updated the map, which means identify unsafe areas (SafeStreets sees the municipality information as they were reliable reportings) and to give suggestions to the municipality itself to prevent more violations (for example add barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

* 1. Definitions, acronyms, abbreviations
     1. Definitions

·User: the costumer of the application that send reports. It could be a private citizen or an authority like municipality. In this case the use of the application will be different.

· application analyzing pictures algorithm: the algorithm that SafeStreets uses for recognizing the license plate of the car object of the violation.

* + 1. Acronyms

·RASD: Requirement Analysis and Specification Document

·API: Application Programming Interface

·GPS: Global Positioning System

·S2B: Software To Be

* + 1. Abbreviations

·Gn: nth goal

·Rn: nth requirement

·Dn: nth domain assumption

* 1. Revision history

·Version 1.0:

-First release.

* 1. Reference documents

- Specification document: “SafeStreets Mandatory Project Assignment”.

- IEEE Std 830-­‐1998 IEEE Recommended Practice for Software Requirements Specifications.

- Examples documents:

- RASD Sample from A.Y. 2015-2016.pdf

- RASD Sample from A.Y. 2016-2017.pdf

* 1. Document structure

The RASD is composed by 5 sections.

* Section 1:

it is the introduction of the RASD in which the problem is presented informally with natural language. It provides base information such as the product to develop and the application domain. The scope part is an analysis of the world and the shared phenomena.

* Section 2:

it presents an overall description of the project. It describes external interfaces, summary of major functions, constraints, assumption and dependencies of the S2B. Furthermore a class diagram and some state diagrams are provided to make stakeholders better understand the project, but even for giving more details on shared phenomena and the domain model.

* Section 3:

this is the body of the document. It first describes the interfaces requirements. Then it lists some scenario to show how system works in real life situations and functional requirements. Lastly we have non functional requirements such as performance requirements and design constraints. This section will be useful for the development team.

* Section 4:

here we have the Alloy formal description of the problem that includes all the relevant details.

* Section 5:

it presents the effort spent for every member of the group.

1. Overall Description
   1. Product perspective

The SafeStreets system is a completely new software application, designed from scratch. It is intended to be used as a mobile application.

The S2B receives reporting of accidents and parking, speed and traffic lights violation. It stores the information on the application data center. Then updates the map every time a reporting occurs. The map as already said is based on Google Maps’ APIs because it offers a very large library of APIs.

The application also works along with authorities. Authorities can only acquire reliable data from SafeStreets. A reporting is reliable only if it is a parking violation reporting, with a picture, and the algorithm to read the license plate fulfils. Immagine che contiene testo, mappa

Descrizione generata automaticamente

* 1. Product functions
  2. User characteristics
  3. Assumptions, dependencies and constraints

1. Specific Requirements
   1. External interface requirements
      1. User interfaces
      2. Hardware interfaces
      3. Software interfaces
   2. Functional Requirements
   3. Performance Requirements
   4. Design constraints
   5. Software system attributes
      1. Reliability
      2. Availability
      3. Security
      4. Maintainability
      5. Compatibility
2. Formal analysis using Alloy
3. Effort spent