RASD

Carlo Dell'Acqua, Adriana Ferrari, Angelica Sofia Valeriani

October 13, 2019

Contents

Chapter 1

Introduction

Chapter 2

Overall description

Chapter 3

Specific requirements

3.1 Functional Requirements

3.1.1 Allow a user to send pictures about street and parking violations

- A user can submit a picture to the system whenever they see a street violation
- SafeStreets analyses the image to read the licence plait of the vehicle
- The user can manually attach the licence plait to the image, to help the system with its validation
- The user must provide the type of violation, either selecting it from a list or manually typing it if it is not already in the system
- The user can help localising the violation by inserting the name of the street where it occurred

3.1.2 Validation of reported violations

- Whenever a violation is reported by a user, it is sent to a random group of k SafeStreets users
- Users who receive the notification about this approval process are then asked to approve or reject the report
- The report is validated if and only if the algebraic sum of approvals (+1) and rejection (-1) is at least v

3.1.3 Allow a user to see the areas where a violation is more likely to happen

- SafeStreets can aggregate data inputted by users to show relevant statistics about the frequency of street violations
- Users can see how many violations usually happen in their neighbourhood, in their current location or in an area of their choice
- A user has access to the type and amount of violations that happened

3.1.4 Allow authorities to see data about violations and who committed them

- Authorities can access all data about violations that a standard user can access
- Authorities also have access to more specific data about who committed a violation, like the licence plait of the car or how many infractions have been associated to a specific car

3.1.5 Allow any citizen to become a user by providing a document

- A citizen can register to SafeStreets by providing their ID number
- The system must allow a user to register with an email and a password, that will be asked every time they log in

3.1.6 Allow authorities to register with a special user profile

- Authorities must first register as citizens
- A standard user profile can be upgraded to authority profile if it is verified by the system
- To obtain privileged access, the user must provide a valid document that proves their authority status

3.1.7 Unsafe Areas Identification

- SafeStreets can cross information from different sources to identify potentially unsafe areas
- SafeStreets may be integrated with a public service, offered by the municipality, that provides such information
- Once identified, SafeStreets can suggest possible interventions to prevent accidents

3.1.8 Traffic Ticket Generation

- SafeStreets can generate tickets for traffic violations reported from registered users
- The information is kept safe and intact through the chain of custody, from the end user to the local police officer issuing the tickets

3.2 Design Constraints

3.2.1 Standards compliance

- To guarantee the compatibility with the potential municipality accident information service, our backend software should communicate and be able to read data in a portable format like JSON or XML
- To transfer this data, the software should expose an HTTP REST API

3.2.2 Hardware limitations

- The frontend application should be as lightweight as possible to support the diversity of mobile device hardware
- Devices with a camera resolution of less than 2Mp should be marked as incompatible with SafeStreets due to the poor quality of the pictures that are taken with them

3.2.3 Other constraints

- The language used to develop the backend application will be chosen from the ones that are most supported by the main cloud infrastructure providers, in order to have a wider choice of hosting plans

3.3 Software System Attributes

3.3.1 Reliability

3.3.1.1 Index MTBF

- Index MTBF must consider as failures those out of design conditions which place the system out of service, for example by an overload of users requests
- MTBF is of 600'000 hours

3.3.1.2 Index MTTR

- Index MTTR must consider time of testing and eventual solution of bugs
- MTTR must be of two hours

3.3.1.3 The system must verify position in an accurate way

- The system expects to receive two pictures
- The received pictures must be of at least 2 Mp
- The first picture must represent a general overview of the violation, to identify street and area
- The second picture must allow the system to identify correctly the license plate, if the violation includes a car
- The second picture is simply an additive detail, if the violation doesn't include a car
- In the second picture, if the violation includes a car, the license plate must occupy at least a quarter of the total picture
- The system must verify the position provided by the user that sent a picture notifying a violation by comparison to the data stored and the GPS service
- The identification of coordinates must not be less precise than 3 metres in case of parking violation

3.3.1.4 Availability

- The system must guarantee a 24/7 service
- The system must be available 99.99%
- The system must be available when performing the standard routine tests for maintenance
- The system can bear slowdowns of the service in case of extraordinary maintenance due to overloading of requests

3.3.2 Security

3.3.2.1 Secrecy of data received and of users information

- The system should never allow both non-registered users and registered users to see the identity of the person that notifies the violation
- User credentials, ID and password, have to be encrypted and safely stored
- Every user has to register with username and password, but the system provides for each user a multi-factor authentication

3.3.2.2 Integrity of data

- The information sent by users (picture, date, time and position of the violation) has to be encrypted in order to keep the integrity of message, and to avoid manipulation of data

3.3.2.3 Measures of security according to danger level

- Public statistics to identify the most dangerous areas are provided for SafeStreets
- SafeStreets makes a ranking of dangerousness (minimum, medium, high)
- The authorities can decide to improve security by highlighting streets in areas at minimum risk
- The authorities can decide to improve security by adding cameras in the areas at medium risk
- The authorities can decide to improve security by both adding cameras and doubling patrol shifts in the areas at large risk

3.3.3 Maintainability

3.3.3.1 Testing overview

- The system is checked in its correct functioning by software testing
- Unit tests are performed every month in standard conditions
- Integration tests are performed every two weeks in standard conditions
- System tests are performed every week in standard conditions, in order to guarantee performances and by regression to solve eventual bugs due to software development

3.3.3.2 The system is controlled and monitored

- The system is equipped with condition-monitoring algorithms
- The condition-monitoring algorithms identify the functions that cause alarm
- A unit test is made as soon as a function is identified as potentially at risk
- An integration test is performed after the unit test of the potentially at risk function
- A system test is performed after the integration and unit test of the potentially at risk function

3.3.3.3 The code must be clean and easy to understand

- Code must have a clear structure
- It is advisable to following the design patterns to provide a standard terminology and to make the software more adaptable to future extensions and improvements
- Complete and detailed documentation from a unit to the superior levels is mandatory in order to keep the maintainability on the highest level on the whole system

3.3.4 Portability

- The software shall run on Windows, IOS, and Linux
- The software must be developed in Java
- The software can be developed using the standard APIs that span different types of operating systems
- The software will be transferred with no modifications to support the environment on another destination machine
- Binary translation must be supported by the system as adaptability is significant and practical for a program in binary (executable) form
- The architecture must be flexible

Chapter 4

Formal using Alloy

Chapter 5

Effort spent

Carlo Dell'Acqua

Task	Time spent (hours)
Project setup	2
Functional requirements	0.5
Design constraints	0.5

Adriana Ferrari

Task	Time spent (hours)
Project setup	2
Functional requirements	1
Design constraints	0.5

Angelica Sofia Valeriani

Task	Time spent (hours)
Software System Attributes	2

Bibliography