RASD

Carlo Dell'Acqua, Adriana Ferrari, Angelica Sofia Valeriani

October 15, 2019

Contents

Chapter 1

Introduction

Chapter 2

Overall description

2.1 Assumptions, dependencies and constraints

2.1.1 Domain assumption

- Personal information that a visitor has to provide to become a registered user are name, surname, email, ID number and password
- Emails are unique and a user can only be associated to one email address
- After registration the system sends an email to the user containing a recap of the information they provided and an activation link
- Once the activation link is opened in a browser the email address of the user is verified and the user is automatically logged in
- Two-factors authentication can be enabled by the user by providing a phone number (SMS authentication) or by scanning a QR Code with an authenticator app (Token authentication)
- Once logged in a user can notify traffic violations
- Pictures with a resolution of less than 2MP are automatically rejected
- ———————————————— ALT1
- An unsafe area, represented as a circle, is described by the coordinate and the radius as the maximum acceptable distance from the center of the area
- Different unsafe areas do not overlap between them
- Every area that is not unsafe is considered safe
- Unsafe areas are determined by statistics made by the system, according to the areas where the maximum number of violations occur

- ———————————————— ALT2
- Users can view a custom map that is updated periodically and shows unsafe streets by highlighting them with a gradient of colors from yellow to red which indicates the level of danger

2.1.2 Dependencies

- The authorities will be in charge of every effective measure of security taken to make streets safer

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 license plait of the vehicle
- The user can manually attach the license 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 localizing 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 neighborhood, 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 license 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 Performance Requirements

3.2.1 Performance features

- 90% of the API calls should be completed within 4.5 seconds
- The response times during the testing phase will be measured using HP LoadRunner (or similar tools) located behind the firewall and in front of the web servers
- Real response times will be measured using the application server log files
- The system will be deployed to a machine able to serve a huge number of users in parallel

3.2.2 Evolution of the system

- At the beginning the system should be able to handle up to 10'000 users
- As the number of users will grow in unpredictable ways, at first the system will be deployed on cloud infrastructure that enables automatic up and down scaling (e.g. Azure Autoscale or AWS Elastic)
- Due to the cost of such infrastructures, as the number of users will become more stable the system will be migrated to a fixed resource machine

3.2.3 Performance testing

- Automatic monthly tests to check performance will be executed during system idle time which is determined statistically
- The performance tests should not exceed an execution time of 15 minutes

3.3 Design Constraints

3.3.1 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

3.3.2 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

3.3.3 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

3.4 Software System Attributes

3.4.1 Reliability

3.4.1.1 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 5,000 hours

3.4.1.2 Index MTTR

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

3.4.1.3 The position should be verifiable

- The system expects to receive one or two pictures
- The received pictures must be of at least 2MP
- The first picture must represent a general overview of the violation, to identify the street
- The second picture must allow the system to identify correctly the license plate, if the violation includes a vehicle

3.4.2 Availability

- The system must guarantee a 24/7 service
- The system must be available 99.9%
- 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 overload of requests
- The system can become unavailable for some seconds due to extraordinary maintenance or major updates

3.4.3 Security

3.4.3.1 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 personal data must be encrypted and safely stored
- Users' passwords must be salted and hashed before being stored on any persistent medium
- Every user can log in using their email and password, but they can optionally enable Two-Factor authentication to improve security
- Violation reports must be encrypted before being sent to the server to preserve the secrecy and integrity in the chain of custody

3.4.3.2 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 and to avoid manipulation of the data

3.4.3.3 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

3.4.4 Maintainability

3.4.4.1 Testing overview

- The system is checked in its correct functioning by software testing
- Unit tests, Integration tests and System tests are performed before and after every update

3.4.4.2 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

3.4.4.3 The code must be clean and easy to understand

- Code must follow common best practices
- It is advisable to follow the design patterns to provide a standard terminology and to make the software more adaptable to future extensions and improvements
- Complete and detailed documentation, even for low level functions, is mandatory in order to keep the maintainability on the highest level on the whole system

3.4.5 Portability

- The software shall run on Windows, macOS, Linux, Android and iOS through a modern web browser supporting at least ECMAScript 2015
- To guarantee the portability of the front end application the software shall be developed using a web framework such as React
- This approach for the front end application will enable code reuse among native mobile platforms (iOS and Android) and the Web Application

- The back end software shall be developed in a language that can be compiled to run on virtual runtime such as Java, JavaScript or C# which can target the JVM, Node.js and dotnet core respectively
- The software can be developed using the standard APIs that span different types of operating systems
- The software will then be able to be transferred with no modifications on other destination machines
- 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
Performance Requirements	1
Domain Assumptions	1

Bibliography