

AA 2019/2020

DD – design document

Version 1.0 – 9-12-2019

Computer Science and Engineering – Software Engineering 2

Amedeo Carrioli – 866256, Andrea Ceruti – 945604

Professor Elisabetta Di Nitto

POLITECNICO DI MILANO

Table Of Contents

[**1.** **Introduction** 3](#_Toc26173127)

[**1.1** **Purpose** 3](#_Toc26173128)

[**1.2** **Scope** 3](#_Toc26173129)

[**1.3** **Definitions, acronyms, Abbreviations** 4](#_Toc26173130)

[**1.4** **Reference documents** 5](#_Toc26173131)

[**1.5** **Document structure** 5](#_Toc26173138)

[**1.6** **Revision History** 6](#_Toc26173139)

[**2.** **Architectural design** 6](#_Toc26173141)

[**2.1** **Overview** 6](#_Toc26173142)

[**2.2** **High level components and their interaction** 7](#_Toc26173150)

[**2.3** **Component view** 8](#_Toc26173157)

[**2.4** **Deployment view** 13](#_Toc26173180)

[**2.5** **Runtime view** 14](#_Toc26173193)

[**2.5.1** **Login** 14](#_Toc26173194)

[**2.5.2** **Reporting making** 14](#_Toc26173195)

[**2.5.3** **Functionalities through the map** 14](#_Toc26173196)

[**2.5.4** **Data crossing** 14](#_Toc26173197)

[**2.5.5** **Reporting history** 14](#_Toc26173198)

[**2.6** **Component interfaces** 14](#_Toc26173199)

[**2.7** **Selected architectural styles and patterns** 15](#_Toc26173202)

[**2.8** **Other design decisions** 15](#_Toc26173203)

[**3.** **Algorithm design** 15](#_Toc26173204)

[**3.1** **License plate recognizing algorithm** 15](#_Toc26173205)

[**3.2** **Making suggestions algorithm** 15](#_Toc26173206)

[**4.** **User Interface design** 15](#_Toc26173207)

[**4.1** **Mock-ups** 17](#_Toc26173213)

[**4.2** **UX diagrams** 17](#_Toc26173214)

[**4.2.1** **Private mobile app** 17](#_Toc26173215)

[**4.2.2** **Authority mobile app** 17](#_Toc26173216)

[**4.3** **BCE diagrams** 17](#_Toc26173217)

[**4.3.1** **Customer mobile app** 17](#_Toc26173218)

[**5.** **Requirements traceability** 17](#_Toc26173219)

[**6.** **Implementation, integration and test plan** 17](#_Toc26173220)

[**7.** **Effort spent** 17](#_Toc26173221)

1. **Introduction**
   1. **Purpose**

The DD purpose is to give an overall description of Safestreet. In particular, we analyze the architecture, in terms of computational components and interactions among those components.

Unlike the RASD, it will have a more detailed description of the backend of the application, underlining the aspects of the application such as the high-level architecture, the runtime behavior and the algorithm design.

* 1. **Scope**

Here we give an overview of the application based on the descriptions presented on the RASD.

SafeStreets is an application designed to report street violations committed by vehicles on the road.

The users are allowed to make reportings of those violations through the application. In order to do that, a user must be signed in.

Two types of users can access the application and its functionalities: private customers and authorities. The first ones can use the basic functionalities, for example make reportings or interacting with the map on which the reportings made by other users are shown. The second ones, authorities, besides the basic functionalities can also make an information cross with the application, and the application, in specific cases, can send suggestions to authorities, in order to prevent more violations.

The violations reported by the users are stored in a map obtained from GoogleMaps API, which has areas colored with different colors (green, yellow, red) depending on the frequency of the violations in that specific area.

When an area is red for a specific type of violations, for example parkings on the sidewalk, a suggestion is made by the S2B and sent to authorities.

Users, on the map, can filter date and type of the violation (parking, traffic lights, accident and speed violations) and the interval of time they want to see the map (today, last week…).

They can also type on a specific area, the map will zoom in, showing all the reportings in that area, in the position they were uploaded with, then the users can type on a single reporting, and the information about it are shown, for example date, time, position, picture if present.

Authorities, as said earlier, can crosse information with the application, sending accident reportings and receiving from the application parking violation reportings, because are the only reliable repotrings that the application can guarantee.

The main purpose of SafeStreets is to make streets safer by allowing citizens to help each other throughout reportings of streets violations. SafeStreets helps to keep the streets clear also by making suggestions to prevent more violations, these suggestions are sent to authorities, which have more power to take measures on the streets.

* 1. **Definitions, acronyms, Abbreviations**

**Definitions**:

* **Users**: a generic customer of the application.
* **Authority**: specific customers of the application. They are allowed to cross information with the application.
* **Private**: customers of the application that can’t cross information with the application.
* **Reporting**: a signalation on any street infringement made by any user, uploaded on the application.
* **Application analyzing pictures algorithm**: the algorithm through which the license plate is recognized from the picture of the vehicle.

**Acronyms**:

* **RASD**: Requirement Analysis and Specification Document
* **DD**: Design Document
* **CPU**: Central Processing Unit
* **DB**: Database
* **DBMS**: Database Management System
* **GPS**: Global Position System
* **API**: Application Programming Interface
* **TSL**: Transport Layer Security
* **SQL**: Structured Query Language
* **UI**: User interface
* **HTTP**:  HyperText Transfer Protocol

**Abbreviations**:

* [Gn]: n-th goal
* [Rn]: n-th functional requirement
  1. **Reference documents**
* Specification document: “SafeStreets Mandatory Project Assignment”.
* IEEE Std 830-‐1998 IEEE Recommended Practice for Software Requirements Specifications.
* Examples documents:

· “DD from the car sharing project”.

· “DD from Travelendar+”.

· “DD to be analysed AY 2019-20”.

* 1. **Document structure**
* Section 1: An overview of the design document, giving and describing the scope and purpose of it. Definions, acronyms and abbreviation are listed as well.
* Section 2: It gives a description of the architecture of the entire system. This is the core of the document because it contains the most relevant architecture views and decisions:
  + Overview
  + High level components and their interaction
  + Component view
  + Deployment view
  + Runtime view
  + Component interaction
  + Selected architectural style and patterns
  + Other design decision

* Section 3: It presents the two principal algorithms managed by the application. The algorithms are described with the help of a pseudo code in order to let the developers to have the highest degree of freedom.
* Section 4: Here are described in details UI (user interface) already presented in the RASD document.
* Section 5: In this section is present the requirements traceability that links the requirements wrote in the RASD to the design element of the DD.
* Section 6: Description of the implementation, testing and integration of the system.
* Section 7: Shows the effort spent by each member of the group.
  1. **Revision History**
* Version 1.0: First release

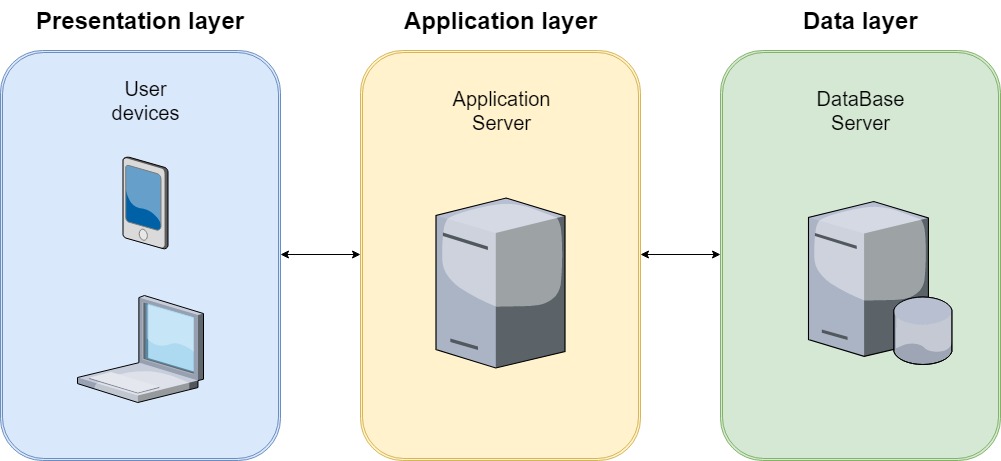
1. **Architectural design**
   1. **Overview**

This chapter is the core of this design document. Here are illustrated both the physical and logical description of the system.

This chapter is divided in sections, for each of ones we properly provide specific diagrams and descriptions of the SafeStreets system.

The image below (*diagram 1*) explains the architecture of the system in layers a tiers. This decision has been made to guarantee scalability and flexibility.

* + The **Presentation layer** is the logical layer that includes the UI and phisically the hardware front-end side of the system, therefore the devices of the users. For example, a smartphone for the SafeStreets mobile app and a laptop for the web. On this layer is shown the GUI (graphical user interface), differently programmed based on the device used. Also, are managed the interactions from the front-end device with the systems (when the user makes a reporting, crosses data…).
  + In the **Application layer** is the part of the system that includes the user’s messages from their device to the web server and the SafeStreets application server, through a DMZ (demilitarized zone, explained later). The corresponding layer focuses on the business logic and communication with the application server, it handles the functionalities of the application and works on data to show to the front-end side. The application server interacts with users sending notifications and questions when needed, asynchronously, and the users talk to the server providing messages as sending information of a reporting or accessing the map.
  + The **Data layer** comprehends the part of the system involving the database and the communication with it. SafeStreets also has a back-up server which only communicates and synchronizes with the database. The interaction between them stays in this layer. Also, the administration database and center are thought to be in this last tier. The application server communicates synchronously with the database to ask for information.



*Figure 1 - Layers architecture*

* 1. **High level components and their interaction**

The figure below provides a physical description of the system. The application server is put in the middle of the image to resume the idea of middle layer, in fact it provides access to the business logic layer. Connected to it are the installed firewalls to create a Demilitarized Zone (DMZ). The external networks communicate with the application server through the resources in the DMZ.

On the diagram is shown the presentation tier, which is represented by the users, that can use both the SafeStreets mobile app and the web app, exclusively for authorities. Both are connected with the web server, which constantly talks with the application server, through the DMZ.

The application server, in the middle, which communicate, on the right, with the database server. Information to the database are sent every time needed, for example when a reporting is made from a user and uploaded on the system, the database server will receive from the application server information about that reporting and will store it.

The database also talks with the backup server in order to have information cloning periodically.

Also, the administration center and database talks with the database without passing through the application server, but these components are implemented through a third-party ERP software, so, there won’t be descriptions about them in this document since they are maintained from a third party.



*Figure 2 - Physical architectute diagram*

* 1. **Component view**

This section contains diagrams showing the main components and the interfaces of the system.

The figure below (figure 3) focuses on the front-end side of the system including the services close to it, each one of them is responsible of specific purposes. For example, the User web services component provide the connections and responses from the authorities through the web application. Remember that only authorities have access to the web application in order to upload and download data in an easy way.

The employee web services component supports the interactions between the system and the SafeStreets personal for small adjustments to the web application. This part won’t be explained in detail because is not the focus of this document and doesn’t involve any other functionalities or structural decisions for the system.

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 3 - Front-end services*

In figure 4, is shown the user mobile services subsystem, which contains three components:

* Map visualizer module
* Make reportings module
* Account manager

These components provide the user mobile services the following interfaces:

* Visualize map
* Make a reporting
* Reportiongs history
* Profile manager

The components need to communicate with the GoogleMaps API and the DBMS in order to be able to give these functionalities to the system.

**Immagine che contiene mappa

Descrizione generata automaticamente**

*Figure 4 - User mobile services*

Figure 5 shows the User web services subsystem, with four components. Three are equals to figure 4, for the mobile services, but, since the web services are accessible only for authorities, a Cross data manager is needed, which provides the users to be able to download and upload data.

**Immagine che contiene mappa, testo

Descrizione generata automaticamente**

*Figure 5 - User web services*

Figure 6 is the entity relationship diagram of the system; it provides a conceptual representation of the system.

An area is on a map, is present one map for each city. The map is seen and interacts through the application with the users, identified by a userID. They can be either privates or authorities. Authorities can cross data with SafeStreets, either uploading or downloading them. All the data are saved in the SafeStreets database.

The data that authorities upload are only accidents reportings, while the downloaded ones are only parking reportings, because are the only ones that SafeStreets, through the license plate algorithm, can consider reliable.

Every reporting made is shown on the map, and it is associated with at least one vehicle (in accidents can be involved more than one).

Parkings reportings only can have a picture.

**Immagine che contiene testo

Descrizione generata automaticamente**

*Figure 6 - Entity-relationship diagram*

* 1. **Deployment view**

The image below, figure 7, shows the deployment view of the system represented with artifacts. Artifacts are considered as pieces of information produced or used by a software development process, which are deployed on nodes that can represent hardware of software environments.

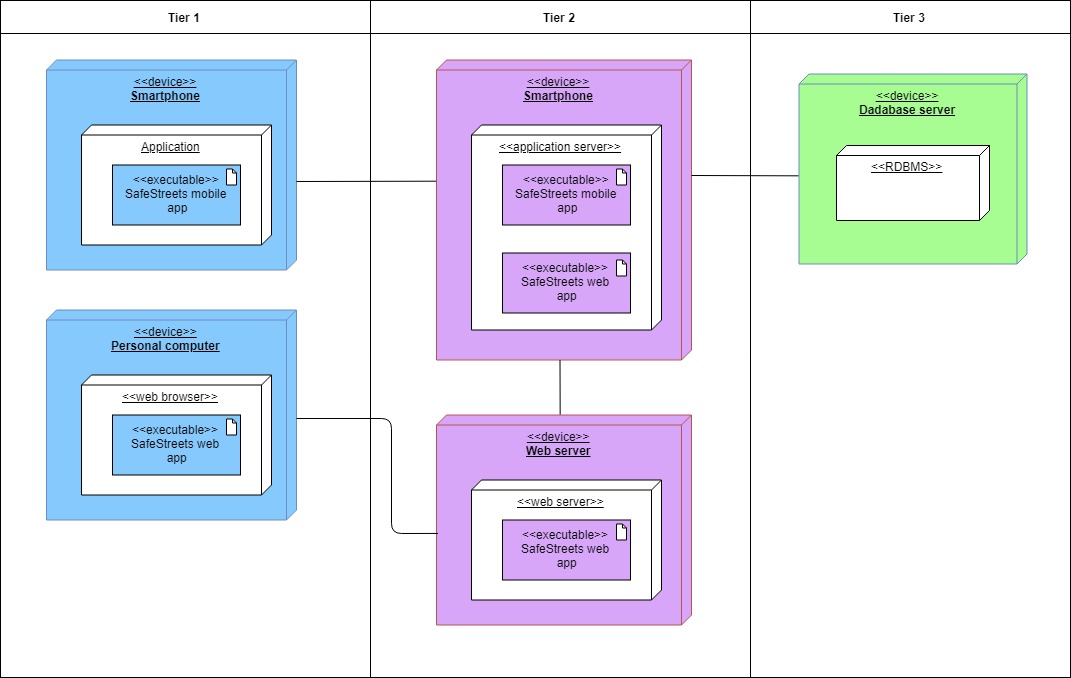
In this diagram are not shown minor components of the system as well as firewalls, due for a willing to represent only the cores of the application.

As said earlier, it is a three tiers architecture:

* In **tier 1** are put the smartphone and personal computer as hardware interfaces, because it represents the presentation logic layer, on which the users interact with the system through the SafeStreets mobile or web app. This tier relates to the application and web servers, the users ask information to them in case they want, for example, upload a reporting or watch a specific reporting on the map.

The mobile app must be compatible with Android and iOS, to make SafeStreets available for almost everyone who has a smartphone. The web app must be compatible with the most common browsers:

* + Google Chrome
  + Firefox
  + Internet Explorer
  + Safari
* **Tier 2** is related with the business logic. It comprehends the application and web servers and it communicate constantly with the previous tier in order to give services to the users. Every time a user asks for information, the request is sent to the application server, which, based on the type of request, either responds to the user or forwards the request to the database server. The principal application’s algorithms are in the application server.
* In **tier 3** the SafeStreets database server is present. It works with a RDBMS (relational DBMS) in order to make the system faster and easier to query. This type of DBMS organizes data into one or more tables or “relations” of columns and rows with a unique key identifying each row. Most of the programming within a RDBMS is accomplished using stored procedures (SPs). These are used to greatly reduce the amount of information transferred within and outside of a system. For increased security, the system design may grant access to only the stores procedures and not directly to the tables.



*Figure 7 - Deployment diagram*

* 1. **Runtime view**

In this section will be shown sequence diagrams of the main functionalities of the system.

It is a high-level description of what really happens, due for simplicity and clearness. For example, the name of the functions in these diagrams might change their name during the development process.

However, the purpose of these functions is and will remain exactly what is show in the following diagrams.

* + 1. **Login**

This first sequence diagram, figure 8, shows the Login action that every user must make at least one time.

Once the login page is shown on the user’s device, the user inserts email and password and sends it to the system only by clicking on the button “Login”.

Now, the system starts to send these information up to the DBMS in order to check if those credentials are allowed to enter the system. Then, if the answer from the DBMS is negative, the system blocks the login of the user, otherwise the user logs in in the system and he’s ready to use all the functionalities offered by the application.

The request is forwarded to the DBMS because the system stores the majority of information in the database, for example the credentials of the users.

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 8 - Login*

* + 1. **Reporting making**

The sequence diagram below represents the core of the application: reporting making.

The diagram shows how a parking violation is reported, but, as already said, there are also speed and traffic light violations and accidents that can be reported through the application. The processes are similar to the one described below, but with some differences. For example, when a user reports an accident, can insert the number of vehicles involved but can’t send a picture of it. The same thing happens for speed and traffic light violations. This is due to prevent the use of the phone while driving.

This process includes, of course, the insertion of information such as the license plate or the geographical position from the user. As soon as a user chooses to make a reporting, the GoogleMaps API is requested, this is because the user often selects the position of the street violation choosing directly from the map.

The information about the violation are asked in two different moments (see mock-ups in RASD). Firstly, is asked position, time and type of violation, depending from these, the application will ask different information to the user. For example, as below, when the user selects “parking violation” then the system will ask for a picture, a license plate and the specific type of parking violation (zebra crossing, bus stop, etc).

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 9 - Make a reporting*

* + 1. **Functionalities through the map**

In the following diagrams are presented the functionalities offered by the system when a user opens the application and looks and interacts with the map. We remember that, by default, when a user opens the application, it is shown the map of all the types of infractions occurred in the last month, grouped by areas of different colors based on the frequency of violations.

Again, the GoogleMaps API is required. In this specific example is shown the user selection of specific interval of time and type of violations that he wants to watch the map with. Then a specific colored area is selected, we remember that when this is done, the map zooms in and all the reporting of that area are shown. Now the user can select one of those reportings, and the system shows all the information about it, for example the time and picture, if present.

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 10 - Map functionalities*

* + 1. **Data crossing**

Figure 11, the following diagram, shows a specific functionality reserved for authorities: crossing information. In this example the authority uses the SafeStreets web app and both the actions of download and upload data are described.

When the authority wants to send to SafeStreets its information, which, as said previously, are only accidents violations, needs to upload a file .sfst, the format given and made specifically by SafeStreets for data crossing with authorities, in order to send and store data in a easy way for both sides. When the file is uploaded, it is analyzed by an algorithm which puts the reportings on the map.

When authorities download a file from SafeStreets, they download it in the same format .sfst, and the file will have both reliable parking violations and suggestions, if present.

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 11 - Data crossing*

* + 1. **Reporting history**

This diagram shows how a user accesses his personal reportings that he has made. The reportings are stored in the DBMS so the request is forward to it.

**Immagine che contiene screenshot

Descrizione generata automaticamente**

*Figure 12 - Reporting history*

* 1. **Component interfaces**

**Immagine che contiene screenshot

Descrizione generata automaticamente**

**Immagine che contiene screenshot

Descrizione generata automaticamente**

* 1. **Selected architectural styles and patterns**
  2. **Other design decisions**

1. **Algorithm design**
   1. **License plate recognizing algorithm**
   2. **Making suggestions algorithm**
2. **User Interface design**

The application UI mockups were presented in the RASD document, section 3.1.1 User Interfaces.

Below are presented two diagrams related to both privates and authorities front end application, in particular they expose the flow of the main menu that the user can navigate from their devices. The application windows are represented as colored rectangles half blue and half white, while the available actions in the windows and the credential recovery as a simple rectangles. The action to return to the previous menu is omitted for clarity.

Finally, the small green rectangle (success) represents the access to the application servers and the recognition of the user's activity ,the red one (logout) represents the logout from the application.

Immagine che contiene mappa

Descrizione generata automaticamente

Immagine che contiene screenshot

Descrizione generata automaticamente

* 1. **Mock-ups**
  2. **UX diagrams**
     1. **Private mobile app**
     2. **Authority mobile app**
  3. **BCE diagrams**
     1. **Customer mobile app**

1. **Requirements traceability**
2. **Implementation, integration and test plan**
3. **Effort spent**