POLITECNICO DI MILANO

Academic Year 2019/2020

DD

Design Document

version 1.0 – 9/12/2019

Computer Science and Engineering

Software Engineering 2

Matteo Falconi 945222 - Davide Galli 944940

Contents

1 Introduction

1.1 Purpose

1.1.1 Project Description

SafeStreets is a cross-platform service available both on app and on web.

App service is focused to develop a software-based service that allows individual basic users to report traffic violations. Those reports consist in pictures of violation, type of violation, date, time and position. When a picture is uploaded, the system runs an algorithm in order to read the license plate. Finally, all those data are stored in *SafeStreets*' databases.

The system allows also authorities registration, who can receive notifications about new violations in a certain area. When a notification occurs, an authority can reserve it taking charge of that violation.

Both basic users and authorities can access to collected data in order to analyze the streets and the relative safeness. However, a basic user can only access to anonymized data clusters, that give an idea of how many violations occur in each area; whereas authorities can also access to specific anonymized data.

Web service, instead, let *SafeStreets* to develop a functionality, in partnership with the municipalities who provide accidents data, that can cross-reference data provided by the users with the accidents one, in order to identify unsafe areas and suggest possible interventions.

1.1.2 Goals

Basic users:

[G.BU1] Basic users can report traffic violations.

[G.BU2] Basic users can view a data clustering about violations that had occurred.

Authorities:

[G.A1] Authorities should choose to receive anonymous notifications in real time about new violations.

[G.A2] Authorities should view and reserve a violation.

[G.A3] Authorities can view both data clustering and specific data about violations that had occurred.

Municipalities:

[G.M1] Municipalities can identify potential unsafe zones.

[G.M2] Municipalities can receive a safety report with suggestions to reduce accidents.

1.2 Scope

1.2.1 World

There are three main types of actors in our world: citizen, authorities and municipalities.
Citizen are interested in reporting traffic violations and receiving information about violations in certain areas, authorities and municipalities are interested in exploiting the data gathered from the citizen: the former want to get notified when new violations occur in order to generate traffic tickets, the latter want to identify unsafe zones and to receive possible solutions.
SafeStreets is the service that acts as a bridge between these actors' needs.

1.2.2 Phenomena

Phenomena that occur in the world and that are related to the system application domain are:

- Traffic violations occur in a city;
- Authority patrols an area and makes traffic tickets;
- People are interested in analyzing violation data;
- Municipality wants to reduce the number of accidents.

The system shares also some events with the world in order to communicate with it. Phenomena that occur in the world and are observed by the machine are:

- A user registers and logs in filling the various form;
- A basic user fills the violation data and sends a new report;
- An authority manages notifications, enabling or disabling them;
- An authority researches a violation among those of civilians in which he/she may be interested;
- An authority examines the details of a violation;
- An authority reserves a violation;
- App user views mined data on a map in his/her smartphone;
- Municipality analyzes unsafe zones;
- Municipality views safety report with suggestions for reducing accidents.

On the other hand, aspects generated by the machine and observed by the world are:

- The system tracks the position of users;
- The system uploads, receives and confirms data insert by users through an acknowledgement (login credentials, new violation reports, etc.);
- A connection error occurs, the system notifies the issue to users;
- The system generates notifications about new violations;
- The system loads safety reports for the municipalities, with suggestions to reduce accidents;
- The system loads and renders graphically data to the user (violations list, detail of a violation, etc.)
- The system loads into a map the mined data as highlight zone and shows them to users.

Finally, phenomena inside the machine and not visible directly from the world are:

- The system stores and reads data (users' data, report data, etc.);
- The system deletes data account;
- The system checks authority's personal ID;
- The system retrieves data from municipality database;
- The machine mines data, clustering both violation data and accidents data ;
- The machine compiles safety report;
- The algorithm analyzes photos reading the license plate of a vehicle;

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

User	Any kind of person who uses the system (basic user, authority and municipality).
App User	Any kind of person who uses the system through the App (basic user and authority).
Basic user	Citizen who can report a traffic violation and view a data clustering about violations that had occurred.
Authority	Recognized entity which can empower the law (ex. local police).

Municipality	Authority recognized by the State who holds the government in an area.
Unsafe zone	Area of the city where accidents happen frequently.
Safety report	Document with all information about the safety of a municipality. It consists in three parts: raw data, accidents' analysis and possible solution. *(RASD, section 2.2.4)*
Report / Violation report	Organized set of information collected by basic users in order to denounce a traffic violation.
Matricula / Personal ID	A code able to identify uniquely an authority, stored in state DB.
Area / zone	Region on a map of at least 5 Km^2.
Highlight zone	Area on a map where are shown the data cluster. The information shown consists in location, type of violation and occurrences.

1.3.2 Acronyms

S2B	Software to Be
IEEE	Institute of Electrical and Electronics Engineers
DB	Database
DBMS	Database Management System
API	Application Programming Interface

1.3.3 Abbreviations

…

1.4 Revision history

Date	Version	Log
9/12/2019	v. 1	First DD release

1.5 Reference Documents

...

1.6 Document Structure

This document is composed into 6 sections, organised as follow:

- *Section 1*
- *Section 2*
- *Section 3*
- *Section 4*
- *Section 5*
- *Section 6*

2 Architectural Design

2.1 Overview

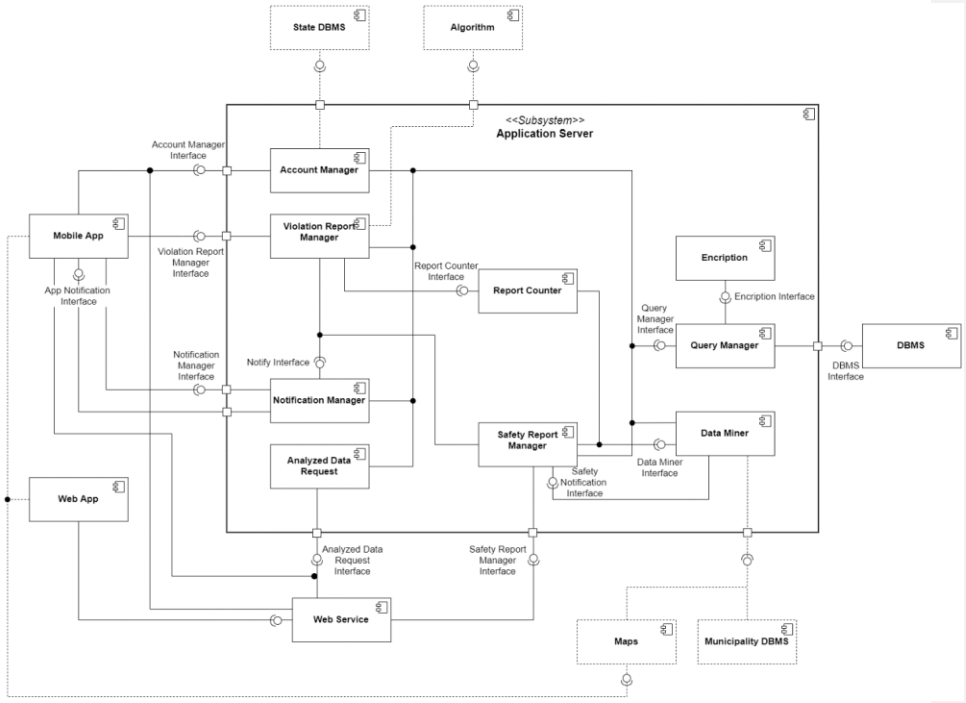
Commentato [MF1]: Faccio io

2.2 Component View

In this section we will analyze every high-level component in terms of its subcomponents and provide the main interface interaction between different components.

External services, such as *Maps* and *Algorithm*, are presented as black-box and expose only the interfaces used by our system.

For details on component interfaces see Section 2.5.



2.2.1 App

The application component is the front-end of the system, through with user interact with the system. It shall only render interfaces in order to exploit all system functionality and all computational tasks oversee the Application Server, so the front-end shall be able to communicate frequently with it. The only logic incorporated in the application component

is the checking of basilar incorrect data type, e.g. incomplete form, invalid date or email without @ character.

Map pictures are provided thanks to maps provider's API which render maps directly on the client side.

The application component consists in two subcomponents as shown in FIGUR XX:

Mobile app is provided to basic user and authorities in order to manage user's account, report a violation, view violation details, view statistic map and receive notifications.

Web app, instead, is provide to partner municipality for managing account, requesting a safety report and viewing unsafe areas.

2.2.2 Web Service

A ***web server*** is required in order to provide a web site for the municipality.

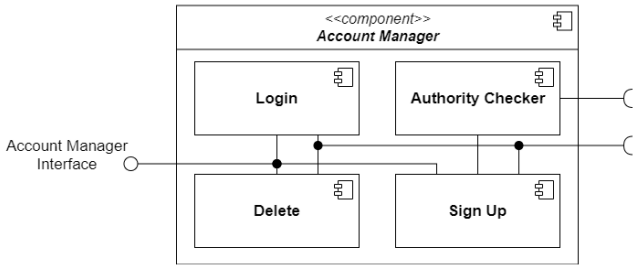
The server receives HTTPs requests from the user and forward them to the application server in order to collect all data required for rendering web pages.

2.2.3 Application Server

Application Server exploit the business logic of the S2B, its role is to compute all data needed by the system and coordinate the flow of information between application layer and data layer. Indeed, it is the only component directly connected with the DBMS and no one else can access the data, neither external system.

Components of the Application Server are:

Account Manager handles all the operations link with users' account. A connection with the DBMS is required to read, store and delate account data. In order to exploit *SafeStreets* functionality users must create an account and be authenticated by this module.



This component relies on *Sign Up* subcomponent to allow user registration, it checks if the credentials are unique or if an account already exists; through the *Authority Checker* verifies Personal ID and department of the authorities connecting to the

state DBMS, in order to let only real authorities to register. *Login* and *Delete* subcomponents handle account authentication by checking credentials and delete permanently all data of a user.

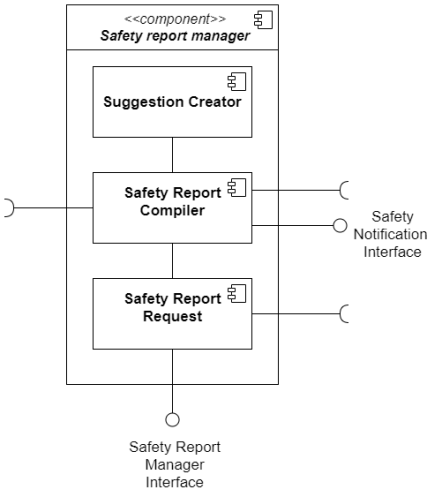
Violation Report Manager receives new report submissions by basis users and requests from authorities to load violation report data. When a new violation is reported, this component asks to the image scanning algorithm to read the license plate of the vehicle from the photos taken by the user. Lastly, it manages the reservation of a report.

Report counter keeps the account of how many reports have been recorded in order to activate the *data miner* component once XXX is reached. Every time data miner is activated, the counter is reset.

Notification Manager receive authorities' notifications preferences and stores them. It sends notifications to mobile app every time a new report is submitted in a zone interested by the authority. It also sends an email to municipalities notify them a new safety report has been generated.

Analyzed Data Request sends to mobile app and web server highlighted and unsafe areas data of the zones requested by the users, retrieving them from the Datawarehouse.

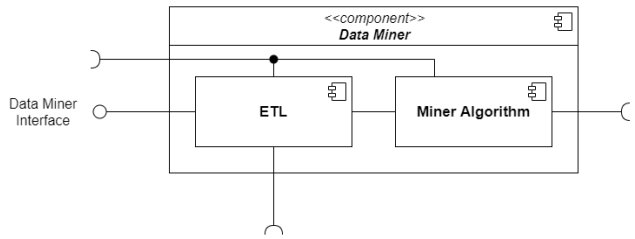
Safety Report Manager handles safety report request, creating a new one or loading the last generated.



Safety Report Request subcomponent activates the *data miner* if new report is requested, otherwise asks directly to *Safety Report Compiler* to compile the last one loading data by the database. *Safety Report Compiler* is also notified when data miner ends. *Suggestion Creator* find suggestion to reduce accidents crossing accidents

mined data with violation reports one. It works only when new report is request, otherwise suggestions are loaded from the database. For details on safety report compiling see paragraph 2.4.

Data Miner classifies both violation reports data and municipality's accidents data separately. It's activated asynchronously by *Report Counter* or *Safety Report Manager* and can notify another component when it ends.



ETL subcomponents retrieves all data needed to mine correctly violation and accidents, extraction dynamically only data not already mined. For location mining, is used information provided by *maps* external system. It also transforms data, deleting duplication (e.g. two report from two different users for the same vehicle in the same violation) and standardizing data format. Particular attention is paid to the data of the municipalities as they come from an external system and may exist inconsistencies. *Miner Algorithm* runs the mining algorithm on data provided by the *ETL*. All data are lastly store in the Datawarehouse through the DBMS. For details on mined data see paragraph 2.2.4.

Query Manager is the only component links to the DBMS, it receives request from other components about what read, store or delate form the database and compute the query for the DBMS.

Encrypter runs encryption/decryption algorithm on some data, such as account password, by request of *Query Component*.

2.2.4 Data layer

2.3 Deployment View

Commentato [MF2]: Faccio io

2.4 Runtime View

2.5 Component Interfaces

2.6 Selected architectural style and patterns

Mvc, observer – observsable

3 User Interface Design

In the RASD document we have shown a series of mock-ups that show the application screens. Here we will extend the UI by providing the navigation flow between the screens.

The graphs should be read as follow: nodes for screens, referenced by figure number in the RASD document; arcs for buttons and square brackets for conditional branches.

3.1 Flow Graph in mobile App

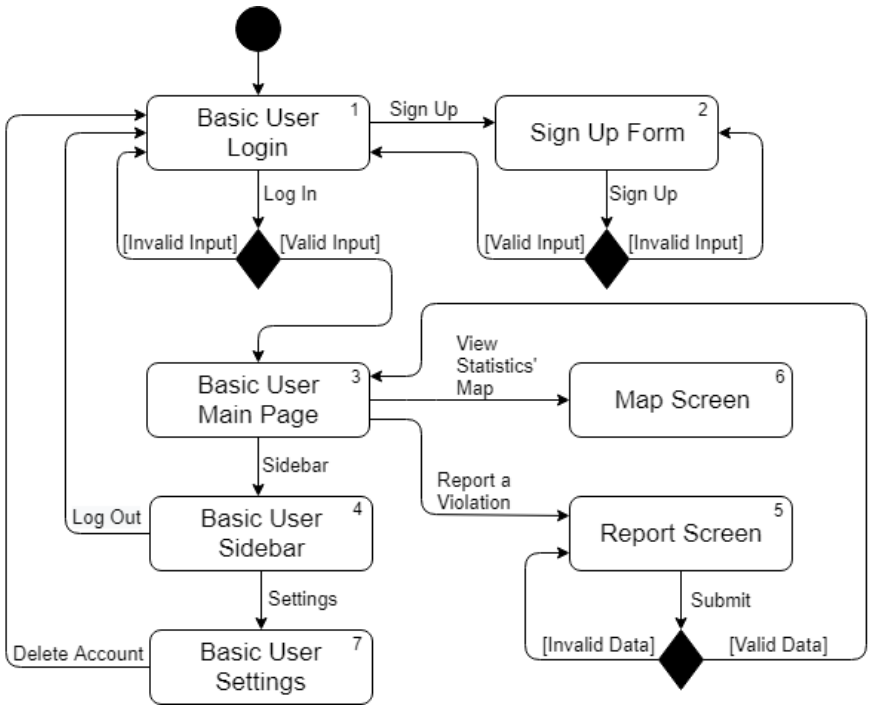


Figure 1: *Basic User's flow of screens*

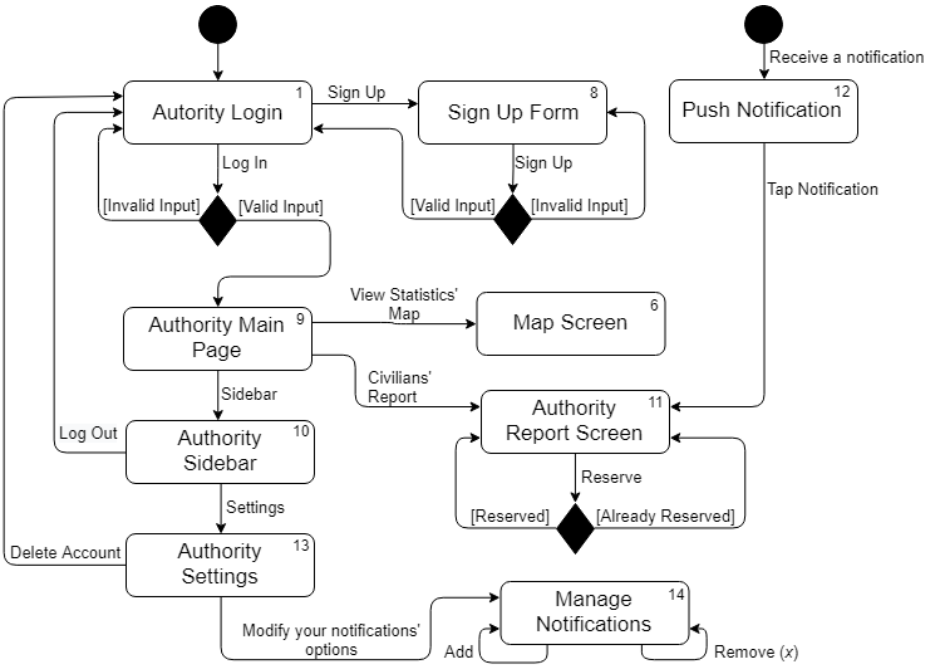


Figure 2: *Authority's flow of screens*

Although there is only one mobile application, authorities and users have two partly different flow charts; as clearly shown on *Figure 1-2.* All mobile users must log in in order to exploit the functionalities, todo

3.2 Flow Graph in Website

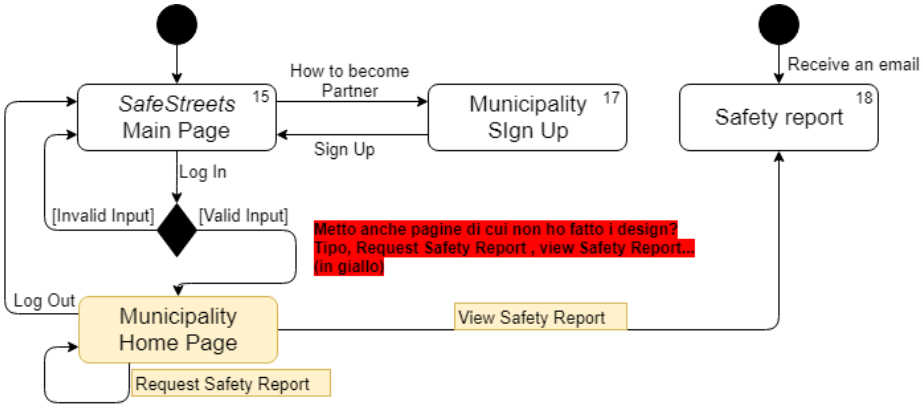


Figure 3: *Municipality's flow of screens*

4 Requirements Traceability

In the following table the requirements described in the RASD document (reported along with a short description) are matched with the components described in the Section 2.2.

Requirement	Description	Components
R.1	*User registration*	
R.2	*App users' distinguishability between basic users and authorities*	
R.3	*Authority validation upon registration*	
R.4	*Account uniqueness*	
R.5	*Users login*	
R.6	*Users can exploit functionalities only after the log in*	
R.7	*Account deletion*	
R.8	*Basic user violation report's upload*	
R.9	*Data is saved internally*	
R.10	*Data can't be erased*	
R.11	*Violation reports are anonymous for Authorities*	
R.12	*It is allowed to authorities to receive notifications regarding new violations*	
R.13	*The system shall generate notifications to authorities when a new report is registered*	
R.14	*Authority access to report anonymously*	
R.15	*Basic user does not access to single report*	
R.16	*Authority reservation of violations*	
R.17	*New violation classification*	
R.18	*Render violations graphically*	
R.19	*List of all violations for authorities*	
R.20	*Management of connection errors*	
R.M1	*Municipality's sign up*	

Commentato [MF3]: Ho segnato in giallo quelle che sono poco convinto o che non sapevo proprio come scriverle. Ho cercato di tenere un livello *minimalist*, in quato già spiegate nel dettaglio nel RASD.

Commentato [MF4]: ADD NON FUNCTIONAL REQ.

R.M2	*Availability of accidents data from municipalities*	
R.M3	*Data integration between SafeStreets and Municipality's DB*	
R.M4	*Data mining for generate unsafe zones*	
R.M5	*???*	
R.M6	*???*	
R.M7	*Safety Report generation*	
R.M8	*Municipality's access to its own Safety Reports*	

5 Implementation, integration and test plan

//todo

Date	Falconi	Galli	Theme
20/11	2.5	2.5	Section 1 Overview Diagram Component Diagram
21/11		2	ER DFM
23/11	3,5		Flow graphs
24/11	3		Section 3, Section 4
26/11		3	Component Diagram
TOTAL	9	7.5	

Appunti

Verifica valida camp (data ect.) lato client

Patterns:
adapter per potersi interfacciarsi alle diverse API di state DBMS and municipality DBMS
Observable per le notifiche alle autorità: lista di tutte le autorità divise per zona o una roba simile