POLITECNICO
MILANO 1863

Software Engineering 2

Requirements Analysis and Specification Document

Amorosini Francesco
Casali Alice
Fioravanti Tommaso

24 October 2019

Contents

Introduction

This document represents the Requirement Analysis and Specification Document (RASD) for SafeStreets: an application that aims to improve the safety of urban areas by giving its users the possibility to report traffic violations and accidents to authorities. The goal of this document is to supply a description of the system in terms of its functional and non functional requirements, listing all of its goals, discussing the constraints and the limits of the software, and indicating the typical use cases that will occur after the release. This document is addressed to the stakeholders, who will evaluate the correctness of the assumptions and decisions contained in it, and to the developers who will have to implement the requirements.

1.1 Purpose

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations and accidents occur. In order to report a traffic violation, users have to compile a report containing a picture of the violation, the date, time, position, and type of violation. SafeStreets stores all the information provided by users, completing it with suitable metadata (timestamps, sender's GPS if available, etc.). In particular, in case the user had not provided any information regarding the license plate of the car breaking the traffic rules, SafeStreets runs an algorithm to read it from the submitted picture. Moreover, the application allows both end-users and authorities to mine the information that has been received: this function allows users to perform useful statistics, for example to find out which area is more dangerous or which vehicles commit the most violations. If the municipality offers a service that allows users to retrieve the information about the accidents that occur in its territory, SafeStreets can cross this information with its own data to identify potentially unsafe areas, and suggest possible interventions. In addition, the municipality could offer a service that takes the information about the violations coming from SafeStreets, and generates traffic tickets from it. In this case, SafeStreets ensures that the chain of custody of information coming from users is never broken, thus information is never altered. Information about

issued tickets can be used by SafeStreets to build statistics, for example to have a feedback on the effectiveness of the application.

1.1.1 Goals

- [G.1] The System allows Users to access the functionalities of the application from different locations and devices.
- [G.2] The System allows Guests to authenticate themselves either as Authority or Citizen.
  - [G.2.1]The System offers different levels of visibility to different roles.
- [G.3] The System allows the Citizens to document traffic violations and accidents to Authorities by compiling a Report.
- [G.4] The System stores the reports provided by the Citizens.
  - [G.4.1] If the Citizen has not provided any information about the license plate, the System runs an algorithm to read it from the submitted picture.
- [G.6] The System analyzes the stored information to detect unsafe areas.
  - [G.6.1]If allowed the Municiplaity, SafeStreets can cross their information about accidents with its own in order to improve the analysis.
  -
  - [G.6.2]If the System collects multiple reports of the same violation in a given area, it can suggest possible interventions to the Authorities.
- [G.7] If allowed by the Municiplaity, SafeStreets can access their information about issued tickets to build statistics and evaluate its effectiveness.
  - [G.7.1] The System ensures that the chain of custody of the information coming from the Citizens is never broken.

1.2 Scope

According to the *World* and *Machine* paradigm, proposed by M. Jackson and P. Zave in 1995, we can distinguish the *Machine*, that is the portion of the system to be developed, from the *World*, that is the portion of the real-world affected by the machine. In this way, we can classify the phenomena in three different types: World, Machine and *Shared* phenomena, where the latter type of phenomena can be controlled by the world and observed by the machine or controlled by the machine and observed by the world.
In this context, the most relevant phenomena are organized as in the following table.

World	Shared	Machine
Traffic violation: circumstance in which someone doesn't respect the traffic rules	**Registration/Login**[1]: a Guest can sign up to the application or log in if already registered.	**DBMS query**: operation performed to retrieve/store data
Accident: event that caused damage to things or people.	**Commit of a report**[1]: action of sending a report documenting a traffic violation or an accident.	**API queries**: request for third-part services.
	Send notifications[2]: to notify the Autorithies when a report is submitted.	**Encrypt data**: to compute the hash function to ensure that reports are never altered and that user's credentials are safe.
	Build statistics[2]: computation of statistics to find unsafe areas, to see which veichles commit multiple infractions and also to monitor the trend of issued tickets.	**Role Isolation**: to grant different levels of visibility to Citizen and Authorities
	Safety suggestions[2]: to give suggestions for possible safe interventions (e.g add a barrier).	

Table 1.1: In the table above, *1* refers to shared phenomena controlled by the world and observed by the machine, whereas *2* refers to the phenomena controlled by the machine and observed by the world

.

1.3 Definitions, Acronyms and Abbreviations

1.3.1 Definitions

- *System*: the totality of the hardware/software applications that contribute to provide the service concerned.
- *Guest*: someone who has yet to sign up and who is not able to access any feature of the application.
- *User*: a registered user who has logged in.
  - *Citizen*: end-user who can send Reports of traffic violation. It has limited visibility over the stored information.
  - *Authority*: user who has access to the history of Reports and is notified whenever a new Report is recieved. It has full visibility over the data exposed by the System.
- *Municipality*: a single administrative division with its own local government. It stores sensitive information about the Authorities and their work (g.e. traffic tickets).
- *Traffic violation*: occurrence of drivers that violate laws that regulate vehicle operation on streets and highways(e.g double-way parking).
- *Report*: documentation of a traffic violation. It contains a picture of the violation, the date, time, position, and type of violation. Some fields can be omitted in case an accident is being reported.
- *Hash*: a mathematical algorithm that maps data of arbitrary size to a bit string of fixed size. It is a one-way function, that is, a function which is practically infeasible to invert.
- *Chain of custody*: chronological documentation that records the sequence of custody, control, transfer, analysis, and disposition of physical or electronic evidence.
- *Third-party services*: services used by the System in order to provide extra functionalities (e.g. image recognition).

1.3.2 Acronyms

- **API**: *Application Program Interface*, set of routines, protocols and tools for building software applications on top of this one.
- **OCR**: *Optical Character Recognition*, software dedicated to the detection of characters contained in a document and to their transfer to digital text that can be read by a machine. In this context, OCR will be used to read license plates.

- **UML**: *Unified Modeling Language*, is a standard visual modeling language intended to be used for analysis, design, and implementation of software-based systems.
- **GPS**: *Global Positioning System*, technology widely used to get the user's position.
- **DBMS**: *Data Base Management System*, software that provides organized space memory to store information.
- **ID**: *Identification*, a unique key given to each registered User.

1.3.3 Abbreviations

- [G_i]: i-th goal.
- [R_i]: i-th requirement.
- [D_i]: i-th domain assumption.

1.4 Reference Documents

- Specification document: *SafeStreets Mandatory Project Assignment.pdf.*
- IEEE std 830-1998: *IEEE Recommended Practice for Software Requirements Specifications*.
- Alloy documentation: `http://alloy.lcs.mit.edu/alloy/documentation.html`.
- UML diagrams: `https://www.uml-diagrams.org`.

1.5 Document Structure

This document is presented as it follows:

1. **Introduction** contains a general description of the system and its goals, presenting the problem in an informal way.
2. **Overall description** gives a general description of the application, focusing on the context of the system and giving further details about shared phenomena. Furthermore, we will provide the specifications of constraints, dependencies and assumptions, that show how the System is integrated with the real world.
3. **Specific requirements** goes into details about functional and non-functional requirements and a list of all possible interctions with the System is provided using use cases and sequence diagrams.

4. **Formal analysis using Alloy** contains the Alloy model of some critical aspects of the System with all the related comments. A proof of consistency and an example of the generated world are also provided.

5. **Effort spent** shows the time spent writing this document by each member of the team.

6. **References**

Overall Description

2.1 Product Perspective

We are going to analyze all the shared phenomena that are listed in the previous section. The concepts are clarified throw class and state diagrams.

Registration/Login (world controlled, machine observed)
A Guest can sign up to the application or log in, if already registered. In the case of a new submit, the System provides a form which the Guest have to fill with his data, specifying if he/she is a Citizien or an Authority.

Commit of a report (world controlled, machine observed)
A Citizen can send a Report with a description of the violation compiling a structured field. He/she has the possibility to attach a picture. The System will check if the compiled fields are consistent and complete, then it will encrypt the Report and send it to the DBMS (Figure 2.1).

Send Notification (Machine controlled, World observed)
The System notifies the Authorities whenever a new Report is submitted by a Citizen. We assume that the notified Authorities will accordingly handle the violation reported. (Figure 2.1).

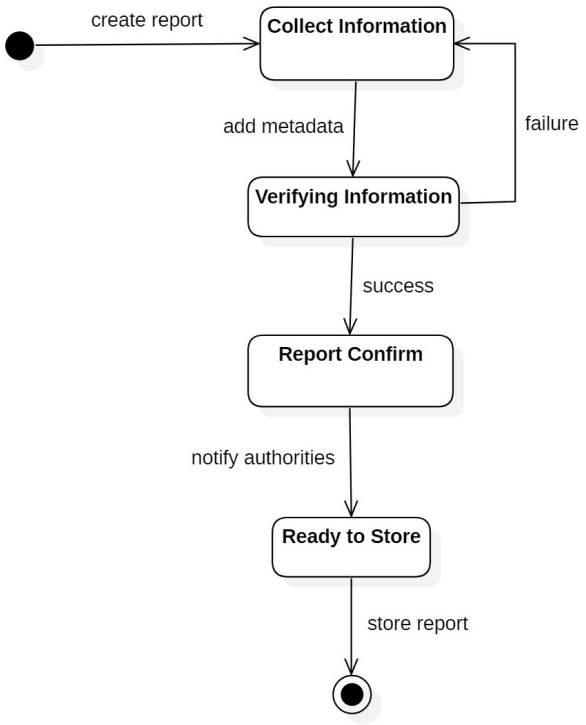


Figure 2.1: SafeStreets' statehart diagram about the collection, notification, and storage of a Report.

Build statistics (Machine controlled, World observed)
The System analyzes the stored data and computes general statistics on the violations. In this way, the System is able to find unsafe areas and which vehicles commit the most violations. All Users can access general statistics, however only Authorities can request statistics on private information such as traffic tickets. If the System is allowed to access Municipality's data, statistics can be enhanced by crossing it with SafeStreets' information (Figure 2.2).

Safety suggestions (Machine controlled, World observed)
The System gives Authorities suggestions for possible safe interventions. His knowledge is based on the statistics previously computed (Figure 2.2).

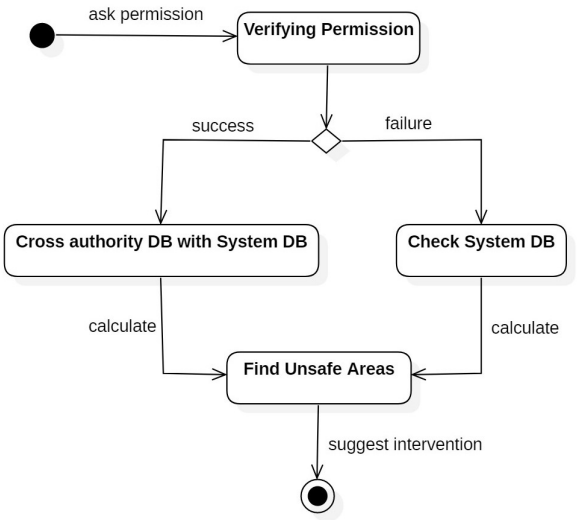


Figure 2.2: SafeStreets' statechart diagram of the creation of statistics on unsafe areas.

2.1.1 Class Diagram

The class diagram in Figure 2.3 shows a possible rapresentation of the SafeStreets' architecture. Notice that the Municipality's DBMS is only accessible by the System if SafeStreets is authorized by the AccessController. It's also important to outline that the System is not designed to provide a function to generate official traffic tickets, since only real world authorities (g.e. local police) can generate traffic tickets with legal validity.

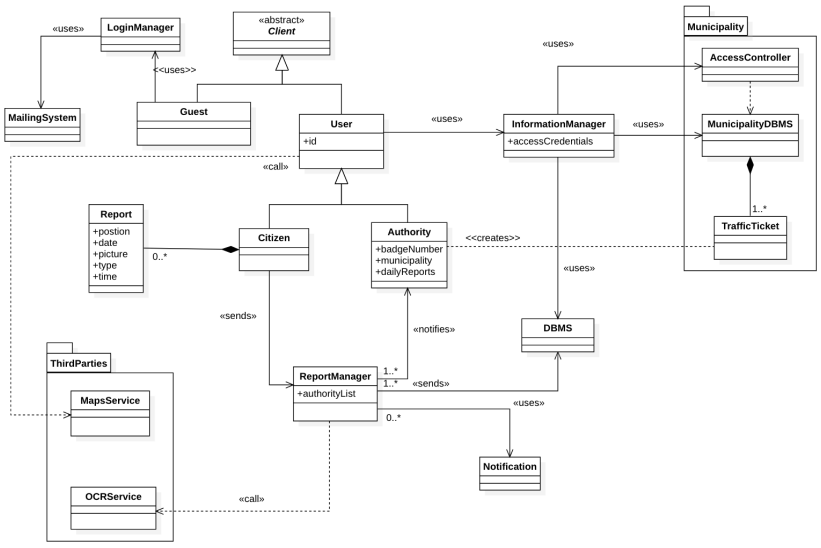


Figure 2.3: SafeStreets' class diagram.

2.2 Product Functions

In this section the most important features of the application are extensively explained:

- **Commit of a Report**: the System allows a Citizen to notify Authorities whenever a traffic violations and accidents occur. This function is achieved by compiling a Report with a picture of the violation, the date, time, position, and type of violation. Not all of these fields are always compulsory: for example, in the case the type of violation is *accident*, only date, time and position are required. Once all the information is gathered and the report is sent to SafeStreets, the Report Manager goes in the `Verify Information` status: it checks if all the obligatory fields have been compiled correctly, and then adds some useful metadata to the Report. Among the additional metadata, the systems takes note of whether or not the position has been provided by GPS, and also if the sender has added any license plate. If no license plate has been inserted, the Report Manager runs an OCR algorithm to retrive it from the picture provided.
  After all the steps above, the Report is confirmed (`Repor Confirmed` status) and a copy of it is notified to the Authorities, which we assume are going to handle it properly (g.e. dispatching a local police patrol). It is important to outline that the System is only designed to inform Authorities whenever a new report is confirmed, but by no means it provides

a function which generates official traffic tickets or dispatches patrols in case of accidents. Also, it's relevant to notice that the Report Manager contains a list of all the Authorities, so it can notify all and only those Authorities who are on duty in the involved municipality. Finally, the Report is encrypted by means of a hash function, thus no one can alter the information after its confirmation (see Section 3.5.3 for more details on security).
Once the report has been confirmed, the System enters the status `Ready to Store`: the Report is exported into the database, where it can be retrieved and dectypted only by the Information Manager.

- **Build statistics**: the System analyzes the stored data to compute general statistics and returns useful information to the Users. This function is carried out by an Information Manager, which is in charge of retrieving, decrypting and elaborating the information stored by the Report Manager. In particular, the Information Manager periodically executes the following functions:

  1. Compute a statistic which highlights the areas and vehicles with the highest frequency of violations.
  2. Identify potentially unsafe areas by analyzing Reports of accidents, and suggest possible interventions.
  3. Analyze traffic ticket trends to build statistics about the most recurrent offenders and the effectiveness of SafeStreets.

  Every User can request to see the latest result of the functions above as soon as they are available. Since both Citizen and Authorities have access to those results, it's really important to give them different levels of visibility, so that classified information doesn't end up in the hands of civilians (g.e vehicles with the highest frequency of violations and most recurrent offenders). It is also worth noting that some of the above functions are enhanced, or even enabled, only if the Municipality grants the Information Manager access to their database. In particular, the Function 2 can be enhanced by crossing Authorities' accident information with Safestreets', whereas Function 3 is entirely enabled only if the System has access to traffic tickets data.

2.3 User Characteristics

SafeStreets can be used from both Citizens and Authorities. It is recommended for adult users, but there are no limitation of age. We take for granted that the Users have access to Internet and are able to install and use the mobile application.
The character are:

- *Guest*: anyone who downloads and opens the app but still has to sign up or log in. He/she cannot use any of the functions provided by SafeStreets.

- *User*: a registered Guest who has to logged either as Citizen or as Authority. He/She is recognized by the System through an ID and can access any feature of the application granted for his/her role.
- *Citizen*: someone who has logged-in with his credentials, which is recognized by the System through an ID. He/she can access to the personal data menu and add new reports.
- *Authority*: someone who has logged-in with his credentials. He/she is notified every time a new Report regarding its municipality is stored by the System.

2.4 Assumptions, Dependencies and Constraints

2.4.1 Assumptions

- [D.1] The given email is assumed to be correct.
- [D.2] The sent email is assumed to be correctly received.
- [D.3] The System internal clock time used to provide notifications is correct.
- [D.4] The badge number of the Authority is assumed to be valid.

2.4.2 Dependencies

- The System needs a DBMS in order to store and retrieve Users' Reports.
- Information on issued tickets are provided by the Authorities' DBMS, which has limited access.
- The OCR tool is provided by external thrid party services.

2.4.3 Constraints

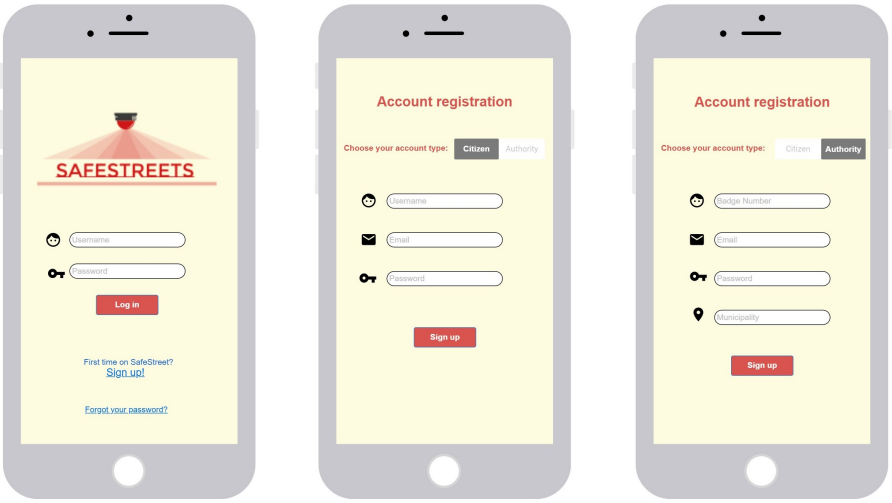
- Users must have a smartphone equipped with an OS compatible with the application.
- Users must have Internet connection.
- The System must ask Users for the permission to acquire and store personal data. Therefore, the System must offer the possibility to the Users to delete their personal account.
- The System provides statistics on issued tickets but it can not know if the tickets come from its pull of report, or if they are independent from its services.

Specific Requirements

3.1 External Interface Requirement

SafeStreets is a mobile based application. In the following sections a more detailed description of the application is given in terms of hardware, software and communication interfaces. There are also present some prototypes of User Interface through mockups.

3.1.1 User Interface



3.1.2 Hardware Interface

The application is avaiable for mobile devices that guarantee Internet access with a reasonably recent browser and have a camera with a fairly high definition.

3.1.3 Software Interface

- **Operating System**: iOS, Android.
- Web browser.
- Web Server application.
- DBMS.
- **Report System**: use third part services as OCR.
- **Mailing System**: APIs to send emails to the User.
- **Mapping System**: APIs to locate the position of the traffic violation.

3.1.4 Communication Interface

The application runs over HTTPS protocol for the communication over the Internet and with the DBMS.

3.2 Functional Requirements

In the following section are explained the functional requirements of the application.

- [R.1] A visitor must be able to register. During the registration the System will ask to provide some credentials.
- [R.2] The System must check if the Guest credentials are valid:
  - the username is not already taken by another registered User.
  - the email is in the right format.
  - the password has a minimum length.

  If credentials are correct, the System sends a confirmation email.
- [R.3] The System must store all User data such as personal information and credentials.
- [R.4] The System must allow the User to log in using his/her personal credentials.
- [R.5] The System must allow the User to change username, only if the new username is not already in use by another User, email, only if the new email is in a correct format and password, only if the new password is different from the precedent and respects the minimum length.
- [R.6] The System must send a confirmation email if username, email or password is changed. The System must replace the old credentials with the new one.

- [R.7] The System must allow the User to change password if it has been forgotten, through the personal email.
- [R.8] The User must be allowed to create reports, specifying:
  - The type of report (violation/accident).
  - The location of the violation/accident.
  - The date of the violation/accident.
  - The time of the violation/accident.
  - A picture of the violation/accident (not mandatory).
- [R.9] The System must check if the report created or edited by the User is correct.
- [R.10] The System must notify the Authorities logged if a new report submitted is located in its municipality.
- [R.11] The System must provide two different level of visibility regarding the statistics. Only the Authorities can consult statistics with private data.
- [R.12] The System must allow the User to consult the statistics.

3.2.1 Use Cases

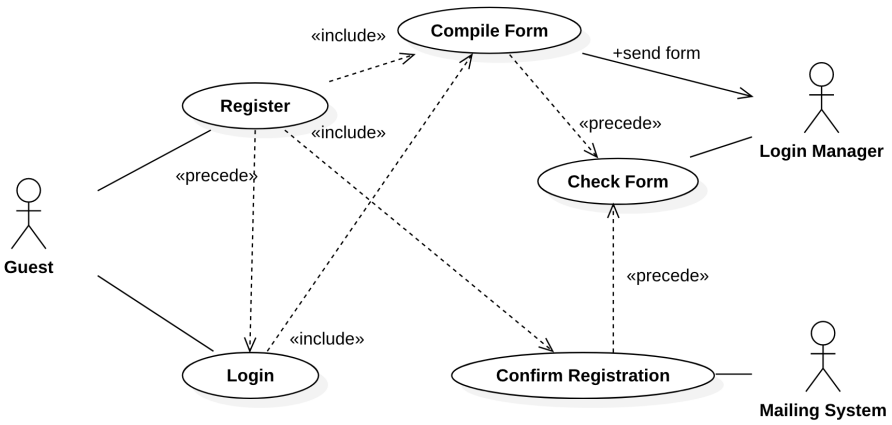


Figure 3.1: Guest use case.

Register Account

ID	UC1
Description	The ***Guest*** wants to create an account for the application.
Actors	***Guest***, ***Login Manager*** and ***Mailing System***
Preconditions	The ***Guest*** downloads and opens the application.

Flow of Events	1. The ***Guest*** taps the Sign Up button and and selects a role between Citizen or Authority. 2. Based on the the ***Guests'*** decision, the ***Login Manager*** provides the appropriate form to enter all the required data: username, email address and password. In the Authoritys' form the username corresponds to the badge number, and the municipality of belonging must be inserted as well. 3. The ***Guest*** fills the form with all the required information and then confirms the entries. The same procedure will be used for future logins. • If the ***Guest*** is an Authority, he/she has to insert his/her badge number as username and provide the municipality of belonging. 4. The ***Login Manager*** checks if all the informations are correct, in which case it generates a random activation URL and asks the ***Mailing System*** to forward the URL to the email address of the ***Guest***. 5. The ***Guest*** receives the mail and clicks on the URL. 6. The ***Login Manager*** accepts the registration and stores the data provided by the ***User*** .
Postconditions	The ***Guest*** is now able to log in.
Exceptions	1. The ***Login Manager*** realizes that invalid cerdentials have been entered (g.e. the account already exists) and shows an error message. The flow restarts from point 2.

Log in

ID	UC2
Description	The ***Guest*** wants to log in to the application.
Actors	***Guest*** and ***Login Manager***
Preconditions	A ***Guest*** wants to log in to the application and is already registered.

Flow of Events	1. The ***Guest*** opens the application and inserts his/her credentials (username and password). • If the ***Guest*** is an Authority, he/she has to insert his/her badge number as username. 2. The ***Guest*** taps the Log In button. 3. The ***Login Manager*** checks if all the credentials provided are correct.
Postconditions	The ***User*** is logged in either as Citizen or Authority.
Exceptions	1. The ***Login Manager*** recognizes invalid credentials than shows an error message. The flow restarts from point 1.

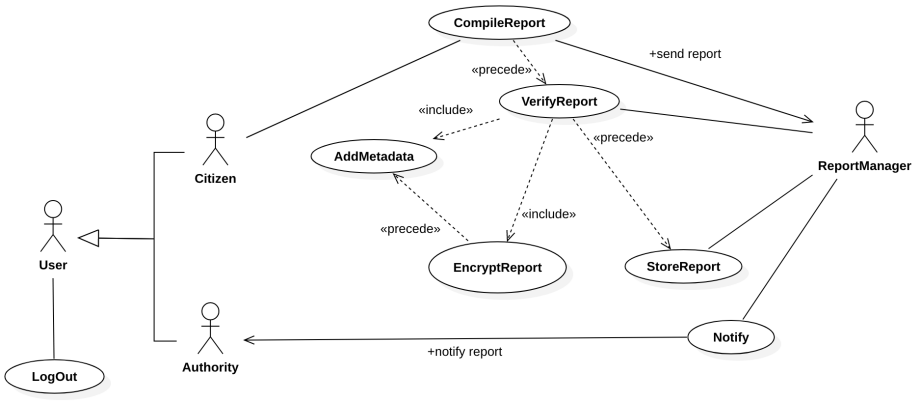


Figure 3.2: User and Report Manager use case.

Send report

ID	UC3
Description	A ***Citizen*** reports a traffic violation.
Actors	***Citizen*** and ***Report Manager***
Preconditions	The ***Citizen*** witnesses a traffic violation and wants to report it.

Flow of Events	1. The ***Citizen*** logs in to the application. 2. The ***Citizen*** taps on *Report* button. 3. The ***Citizen*** compiles a Report. • He/She takes a picture of the violation, if possible. • He/She also enters the date, time, location and the type of the violation. 4. The ***Citizen*** taps the *Send* button. 5. The ***Report Manager*** checks if the report has been correctly compiled.
Postconditions	The report is sent and the ***Citizen*** recives a message that confirms the sending was successfull.
Exceptions	1. The ***Report Manager*** rejects the report and shows an error message saying what fields are wrongly compiled. The flow restarts from point 2.

Store and Notify report

ID	UC4
Description	The ***Report Manager*** recieves a report from a Citizen and notifies the Authorities about it. Then the report is stored in a database.
Actors	***Report Manager***
Preconditions	The ***Report Manager*** recives a valid report (see UC3).

Flow of Events	1. The ***Report Manager*** computes metadata (g.e. whether or not license plate has been entered, if position has been provided by GPS, etc.) and attaches them to the Report. 2. The ***Report Manager*** runs an API call to retrieve any information about the license plate by means of an external OCR. 3. The ***Report Manager*** notifis the Authorities of the municipality where the violation occurred about the report. 4. The ***Report Manager*** encrypts the report. 5. The ***Report Manager*** stores the report in the database.
Postconditions	The ***Authorities*** are notified about the report and the Report is avaiable to be taken from the database by the ***Information Manger***.
Exceptions	1. If the type of the Report recived by the ***Report Manager*** is *Accident* the report is not notify to the authority but it is only stored in the database.

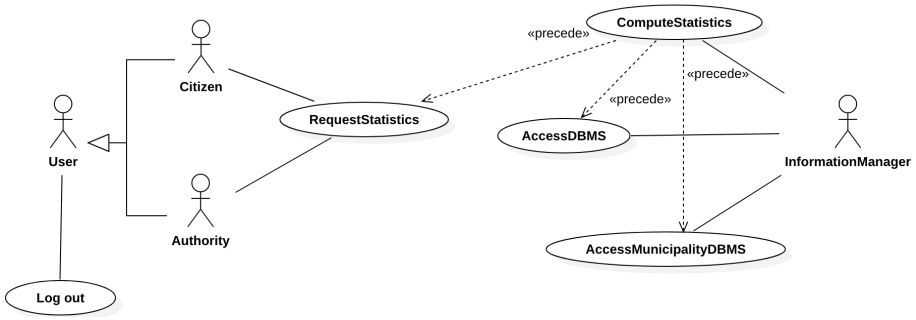


Figure 3.3: Information Manager use case.

Compute Statistics

ID	UC5
Description	A ***User*** wants to see some statistics (e.g the safety of a certain area) and ask the ***Information Manager*** about them.
Actors	***User*** and ***Information Manager***
Preconditions	The ***Information Manager*** has already computed the statistics that the user intends to consult. At set times, the ***Information Manager*** updates all the statistics due to possible new stored reports.
Flow of Events	1. The ***User*** logs in to the application and taps on the *Statistics* button. 2. The ***System*** displays a menu with all the possible statistic options to compute. 3. The ***User*** chooses the statistics in which he/she is interested in. 4. The ***Information Manager*** loads the statistics that the user selected and the application shows. • The application may use external services to show some statistics on a map.
Postconditions	The ***User*** sees the results of the statistics request.

Exceptions	1. If the ***Municipality*** doesn't allow the System access traffic ticket information, the ***Information Manager*** can't retrieve this type of data and the The ***User*** can't visualize the traffic tickets statistics. In this case, a message saying *Statistic Unavailable* is shown to the The ***User***. The flow restart from point 3.

3.2.2 Sequence Diagrams

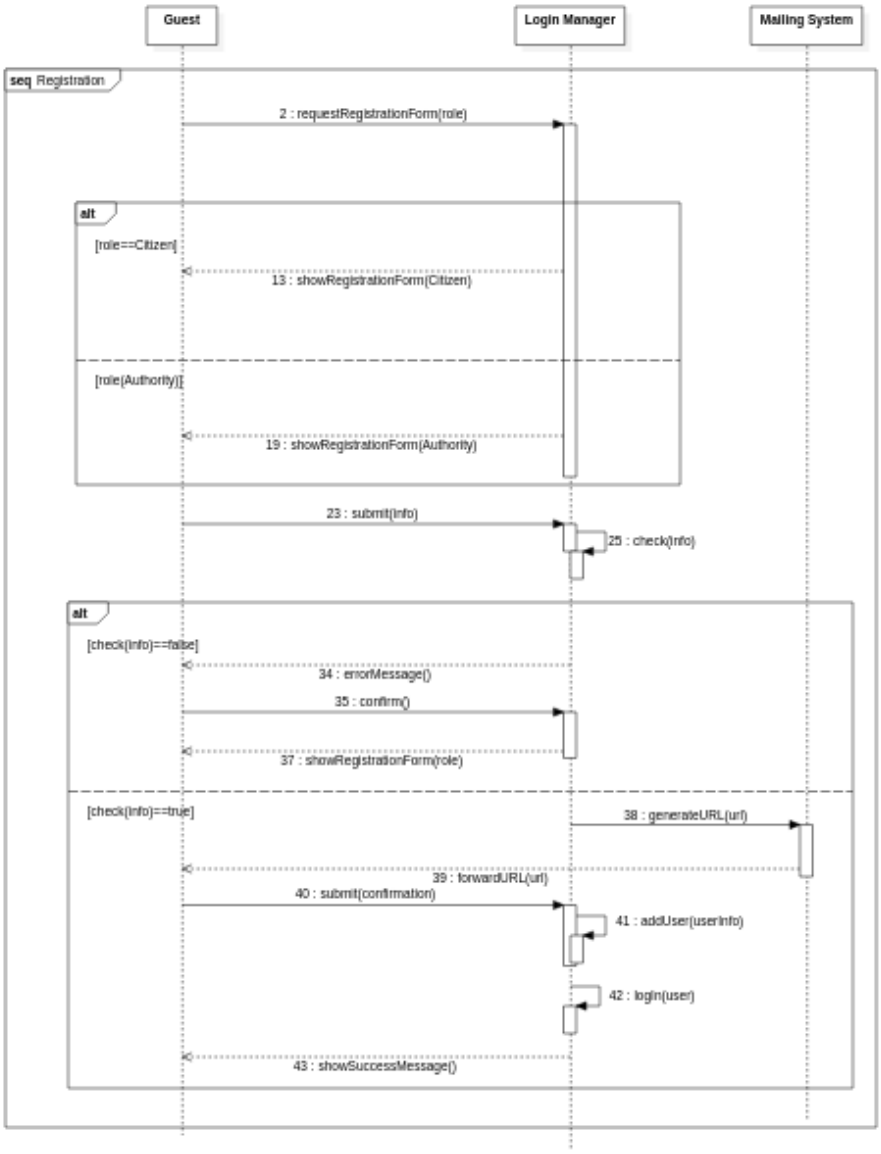


Figure 3.4: Sign up Sequence Diagram.

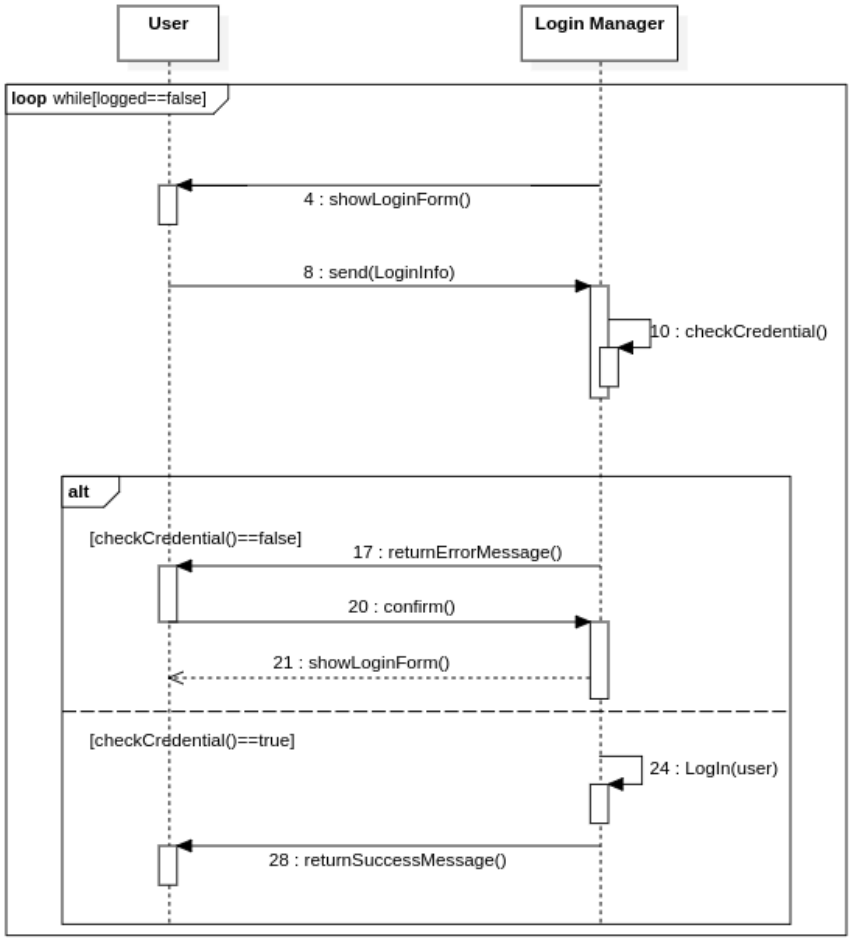


Figure 3.5: Login Sequence Diagram.

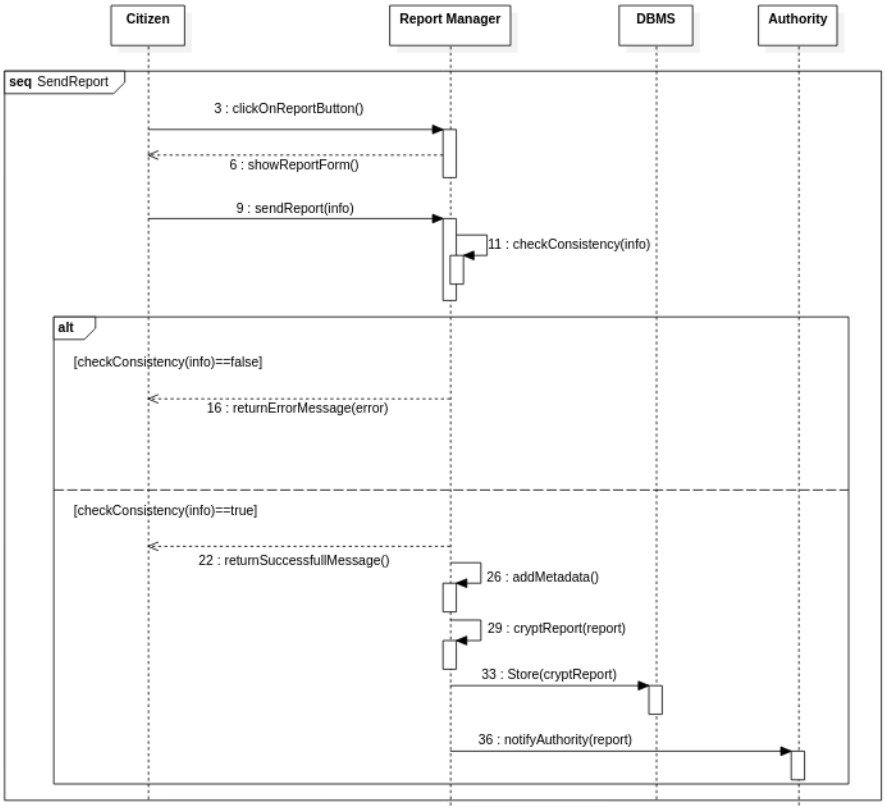


Figure 3.6: Send Report, notify and Store report Sequence Diagram.

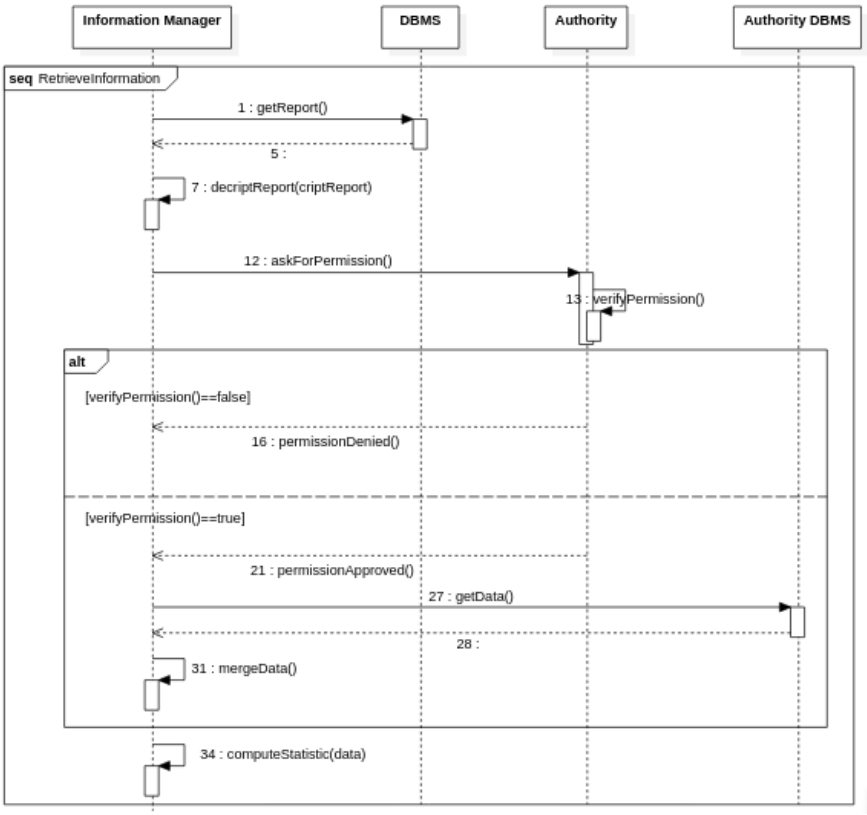


Figure 3.7: Retrieve Information Sequence Diagram.

3.3 Performance Requirements

3.4 Design Constraint

3.4.1 Standards Compliance

3.4.2 Hardware limitations

The application should be able to run, at least, under the following condi- tions:

- 3G connections at 2Mb/s
- 50 MB of space
- 2 GB of RAM

3.5 Software System Attribute

3.5.1 Reliability

The System (and all its components) should guarantees a 99% of reliability.

3.5.2 Availability

The System must guarantee an high availability, indeed it has to offer a continuous service 24 hours every day.

3.5.3 Security

User credentials and data will be stored in a DBMS that should guarantee an high level of security. To reach this goal the passwords stored in the DBMS are salted and hashed, in this way a sequence of generic characters of any length is concatenated to the password, hashed it all and stored in the DBMS. Hash is used also to stored the report so that the informations cannot be altered by anyone. So as soon as a report arrives to the Report Manager, the hash of the report is computed, encrypted to form a digital signature and then it is stored. In this way we create a snapshot of the report ensuring the identity, authenticity and non-tampered state of the informations. As mentioned before, the System uses HTTP over SSL protocol (HTTPS) to comunicate with all the services in order to guarantee privacy and protection of the exchanged informations.
If someone break into the OCR service, the System is not responsible in case of damage, due to the fact that its functionalities are provided by third-part services.

3.5.4 Maintainability

3.5.5 Portability