Politecnico

di Milano

2019/2020

RASD – Requirement Analysis and Specification Document

Version 1.2 && 07/12/2019

Author:

938653 – Berk MORKOÇ

Professor:

Matteo Rossi

Table of Contents

1) Introduction

1.1. Purpose

The purpose of this document is to provide a detailed description of the SafeStreets system. The description will be done through a presentation of the aimed solution, showing and listing its goals. This document is addressed to the developers who have to implement the requirements and could be used as a contractual basis.

The SafeStreets system is designed as a crowd-sourced application used to report the violations (completing it with suitable meta-data) in the traffic through the pictures taken by the users, to the authorities and then provide safer traffic.

1.2. Scope

1.2.1. Description of the given problem

Every day, people especially live in big and crowded cities are able to see many violations made in traffic and this situation might be annoying for some reason such as not being able to take your car out from the parking area because of someone's illegal park in spite of you parking legally. In these cases, the sufferer should invoke authorities about the situation to impose sanctions. Doing it might be hard if you have no traffic police or any other responsible person/authority around you. Having a picture related to the violation occurred is adequate through the SafeStreets system. The application allows people to report the traffic violations by pictures taken through the users, to the authorities. Thus, it is aimed to reduce violations made in traffic. The system has also the ability to highlight the streets where the most violations. Hereby, authorities and users are going to be able to see and know where the most violations occur.

1.2.2. Goals

- ❖ [G1] Allow users to report traffic violations.
  - [G1.1] Allow users to take or upload pictures related to the violations through the system.
  - [G1.2] Allow users to fill a form that includes details belong to the type of violation.
  - [G1.3] Detect the license plate of the vehicle that violated the rule(s) and get the date and time of the violation.
  - [G1.4] Detect the place of the violation and store the information collected/detected.
- ❖ [G2] Highlight the streets that the most violation occurs and provide this information to the users as well as the authorities.
- ❖ [G3] The system must be able to notify the authorities about the violation sent by the users.

1.3. Definitions, acronyms, and abbreviations

1.3.1. Definitions

- **User:** The person who has the SafeStreets application and is able to report traffic violations.
- **Authorities:** Institution or institutions responsible for traffic safety.
- **Meta-data:** Metadata is "data that provides information about other data".
- **Violation:** It refers to any breach made in traffic.
- **Crowd-source:** It is a sourcing model in which individuals or organizations obtain goods and services, including ideas and finances, from a large, relatively open and often rapidly-evolving group of internet users; it divides work between participants to achieve a cumulative result.
- **Library:** The source codes that have been prepared and merged in order to be used for software developments.

- **Illegal:** The operations or actions which do not follow the rule/rules determined by the government.
- **Illegal Parking:** Parking that does not follow rules.

1.3.2. Acronyms

- GPS = Global Positioning System
- DBMS = Database Management System
- UI = User Interface
- RASD = Requirement Analysis and Specification Document
- DB = Database
- ISO/IEC = International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC)
- API = Application Programming Interface

1.3.3. Abbreviations

- [Gn]: n-goal
- [Dn]: n-domain assumption
- [Rn]: n-functional requirement

1.4. Revision history

- Version 1.0
  - First release.
- Version 1.1
  - The use case and sequence diagrams have been updated and database components were deleted from the diagrams.
  - Some of the figures' descriptions were positioned right behind the middle of the figure.
- Version 1.2
  - Requirements were edited.

1.5. Reference Documents

- Specification document: “Mandatory Project Assignment AY 2019---2020”
- IEEE Std 830---1998 IEEE Recommended Practice for Software Requirements Specifications
- Alloy doc: *http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html*

1.6. Document Structure

The RASD document consists of six parts (except for the appendix) as indicated below:

- **Chapter 1:** This is the first chapter of the document and it refers to the introduction. In this section, it is aimed to describe the problem as well as the purposes that the application has to reach. Also, it is given some definitions belong to the document in order to provide a better understanding.
- **Chapter 2:** This chapter hosts to the overall description of the project in which its boundaries are identified, and the actors involved in the system’s usage life cycle are described, listing all the necessary assumptions. Moreover, class and state diagrams are provided in order to better insight related to the structure of the project.
- **Chapter 3:** This chapter is the part of the document and it includes requirements for software, hardware, user interfaces. It also has a list of scenarios to show how the system acts in the real world, followed by the description of functional requirements, use case and sequence diagrams. All the requirements are given with the related domain assumptions and requirements in order to satisfy the goals. Consequently, the constraints and software system attributes such as reliability, availability, security, maintainability, and compatibility are given with the description.
- **Chapter 4:** In this chapter, the alloy model of the system is given with its results and explanations.
- **Chapter 5:** This section shows the effort spent.
- **Chapter 6:** This is the last chapter which refers to the references of the document.

2) Overall Description

2.1. Product perspective

The SafeStreet application is designed to allow people to report traffic violations. Hereby, authorities will be notified and informed about the violation through the SafeStreet application. To report traffic violations, the application has to have an ability to allow users to take a picture or choose a picture from their photo gallery. Once the violation picture has been collected by the user, the application starts running an algorithm to detect the license plate of the vehicle that violated the rule(s). Afterward, the user needs to select the type of violation on the application. In the next step, the location, date and time of the violation needs to be detected. For that reason, the application tries to access GPS embedded on the user's device in order to determine where the violation occurred. The date and time are determined in two different ways; getting the operating system's date and time if the picture is taken through the application or if it is not then time information is manually inputted by the user. Lastly, collected data/information is stored in a DBMS. Thus, the application allows both the users and authorities to see the most violations occur where and by who on the map.

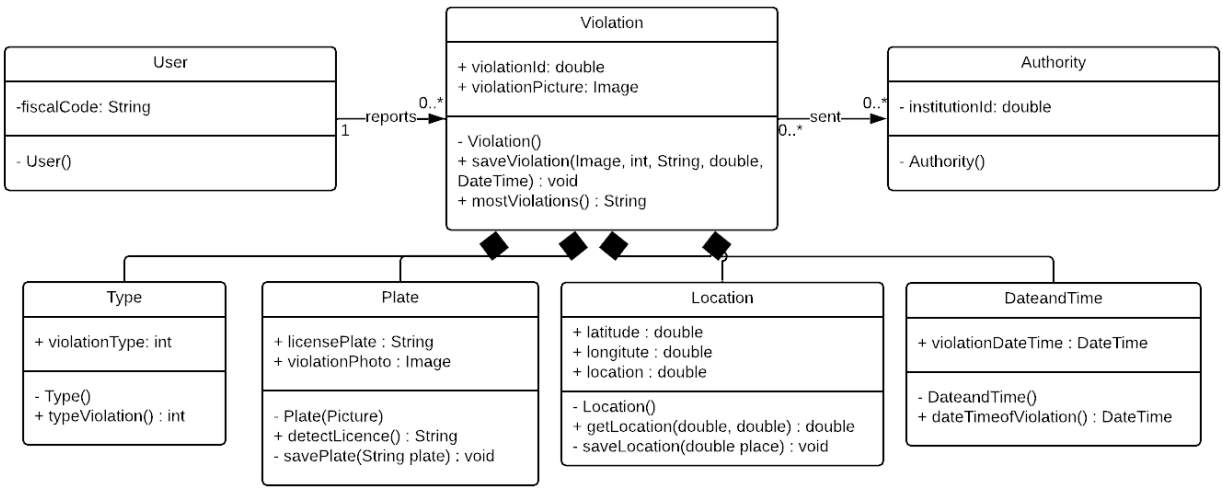

Figure 1 - Class diagram

Figure 1 shows a model of the application domain and it contains essential attributes and classes of the system. The SafeStreets application is designed to be used between the users and the authorities. The connection between these two end-users is the violations made. In other words, the users report the violations occurred and these data are directed to the authorities. When a violation occurred, that violation consists of the knowledge such as the type, location, date, time and license plate of the vehicle that violates the rule(s).

The state diagram *(Figure 2*) has been prepared to provide a better understanding about what the system does and what purposes it is used. Since the made violation can correspond to more than one rule violation, the diagram (*Figure 2*) contains a loop, referring to the selection of the violation. However, *Figure 2* is deprived of some pieces of knowledge such as detection of the location, date and time in order to keep it simple and clear.

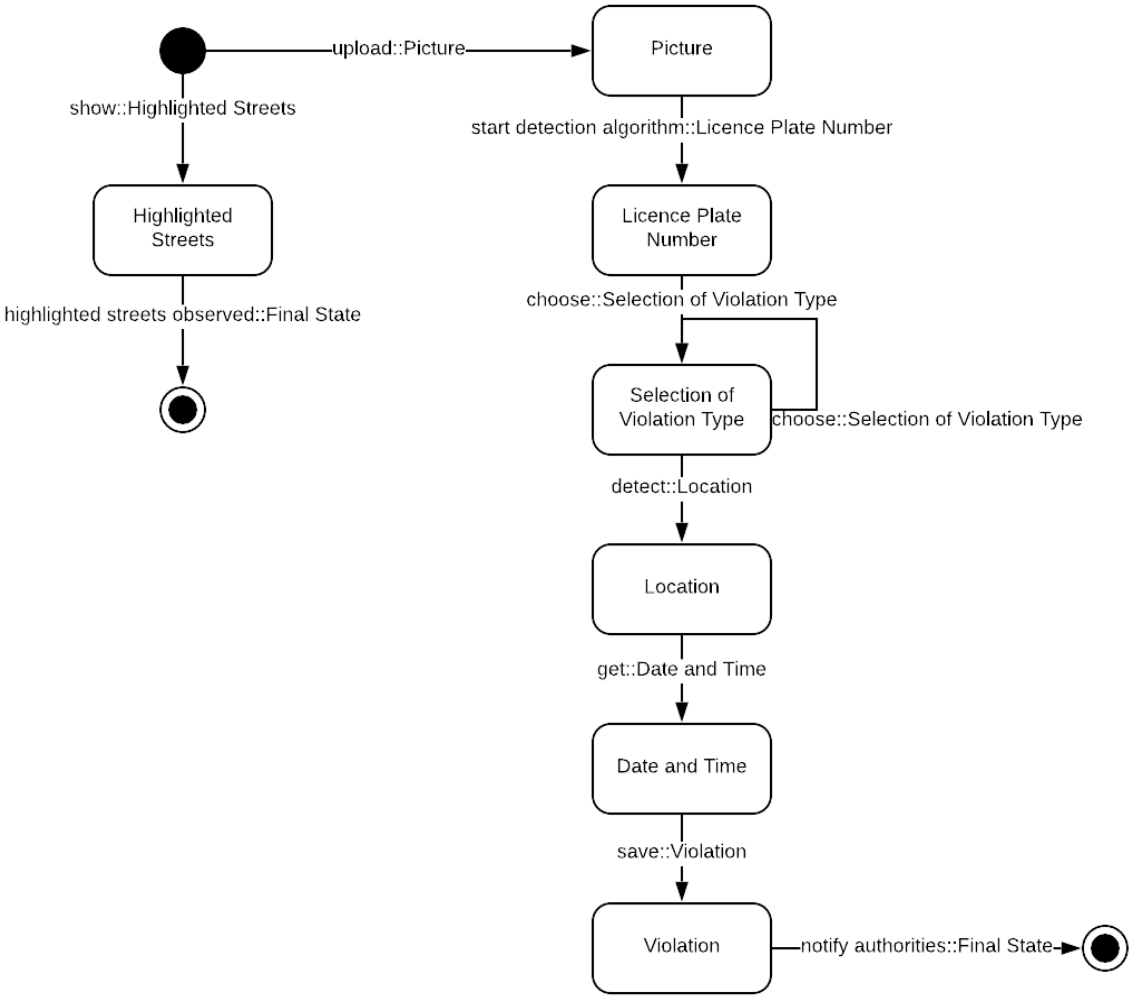

Figure 2 - State Diagram: Overview of the System

2.2. Product functions

In this section, it is aimed to offer all the goals identified by indicating the most important functions of the system.

2.2.1. Reporting service

This product function is the main function of the SafeStreets system and it is going to provide the users to report the violation occurred. This service has been explained a couple of times in the previous parts hence it is the essential part of the system. Basically, it notifies the authorities when violation information has been provided by the users. This process is respectively done as follows:

- The picture is collected.
- The license plate is detected through an algorithm.
- The type of violation is collected.
- The location, date and time are determined.
- Authorities are notified and the violation is saved.

2.2.2. Highlighting service

This product function has been built up in order to provide both the users and the authorities to see and check where the violations most occur. This service takes advantage of recorded violations. It shows the places of the violations on the map with different colors (Red refers to high frequency, blue means medium frequency and green is low-level frequency.). To be able to provide this service; having easy-to-integrate structure and a large library of APIs with extensive documentation, supplying the most commonly used features is the most appropriate mapping and location provider which is going to be integrated into our system.

2.3. User characteristics

There are basically two actors in the SafeStreets system.

1) **User:** a person who has already had the application and is ready to report the violation(s).
2) **Authorities:** Institutions or units dealing with traffic violations.

2.4. Assumptions, dependencies and constraints

- **[D1]** The picture taken or uploaded by the users is related to the violation.
- **[D2]** The map service provider is always available to present the requested service.
- **[D3]** All information provided by the user is valid and correct. In other words, the chain of information coming from the users has never been broken, and the information has never been altered.
- **[D4]** GPS provides the location at most 5-meter mistake.
- **[D5]** The fiscal code provided by the user is valid and it belongs to the user who reports the violation.
- **[D6]** The user that intends to use the application has properly working internet connection.
- **[D7]** When the user has taken a picture of the violation through the app, he/she directly reports the violation without moving or going too far.
- **[D8]** The use of GPS service is allowed by the user.

Any mobile device that can connect to the internet through the connection types such as 2G, 3G, 4G or WiFi (for example a mobile phone, a tablet, etc.) is enough for both the user and the authorities to access the application.

However, since the application allows the users to take a picture of the violation, it directly tries to access to the GPS service in order to detect the location. Therefore, the users have to have a device supports the GPS feature.

3) Specific Requirements

3.1. External interface requirements

3.1.1. User Interfaces

The following prototype designs give an approximate idea of how the application interfaces look like.

Figure 3 - Main page of the application.

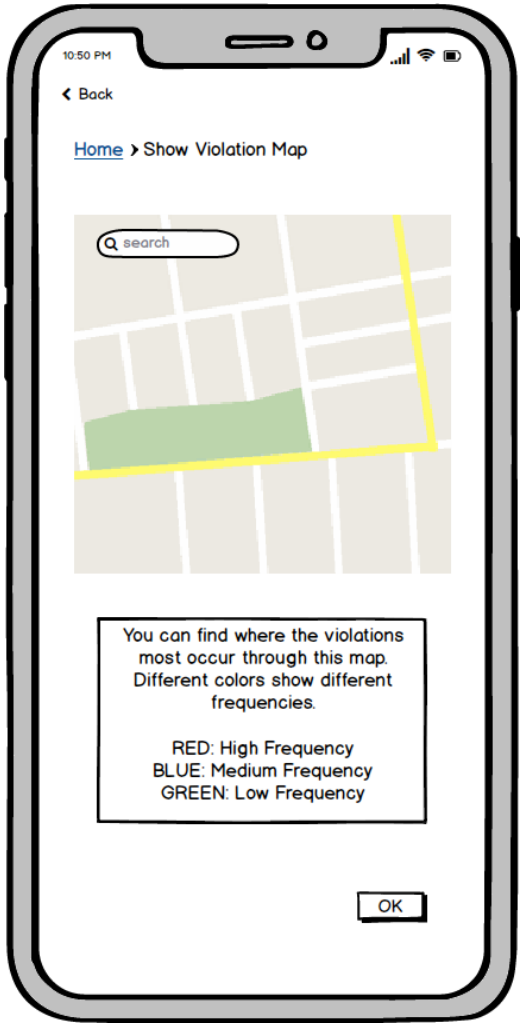

Figure 4 - Violation Map.

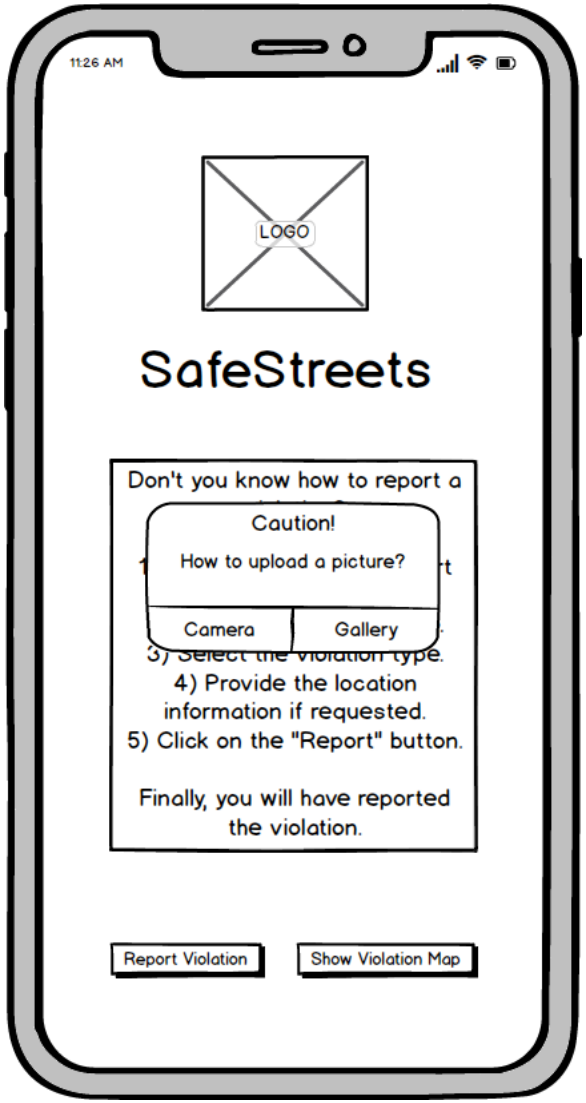

Figure 5 – When the “Report Violation” button clicked.

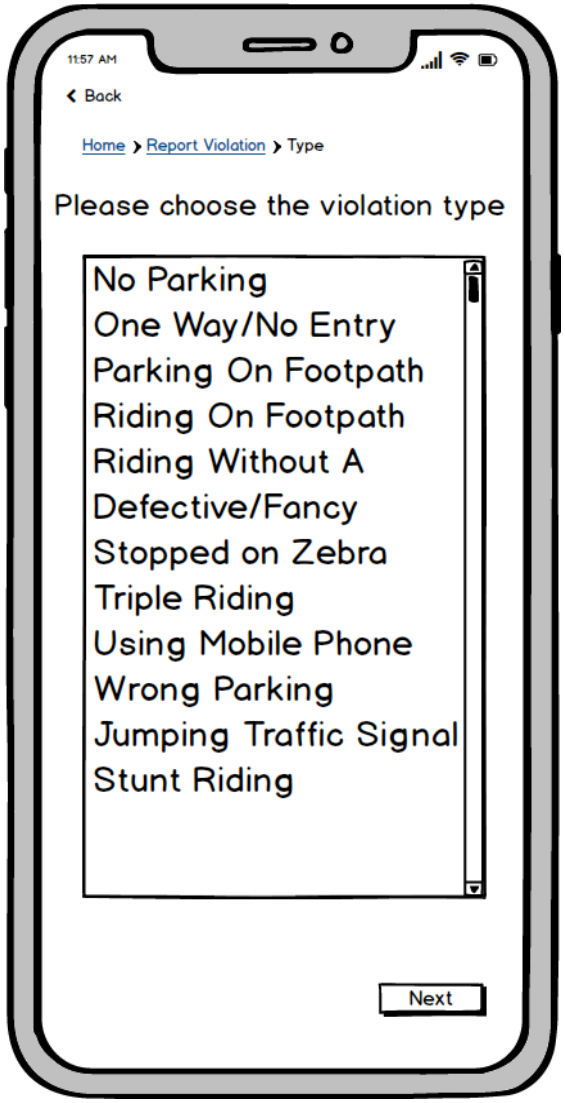

Figure 6 - Type of violation selection.

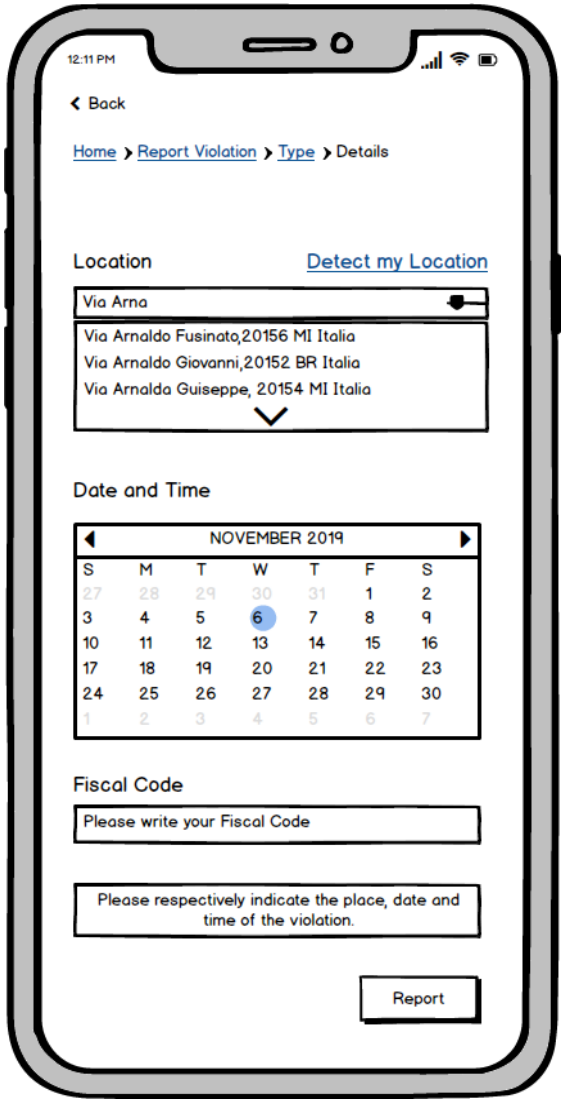

Figure 7 - Screen of details collection.

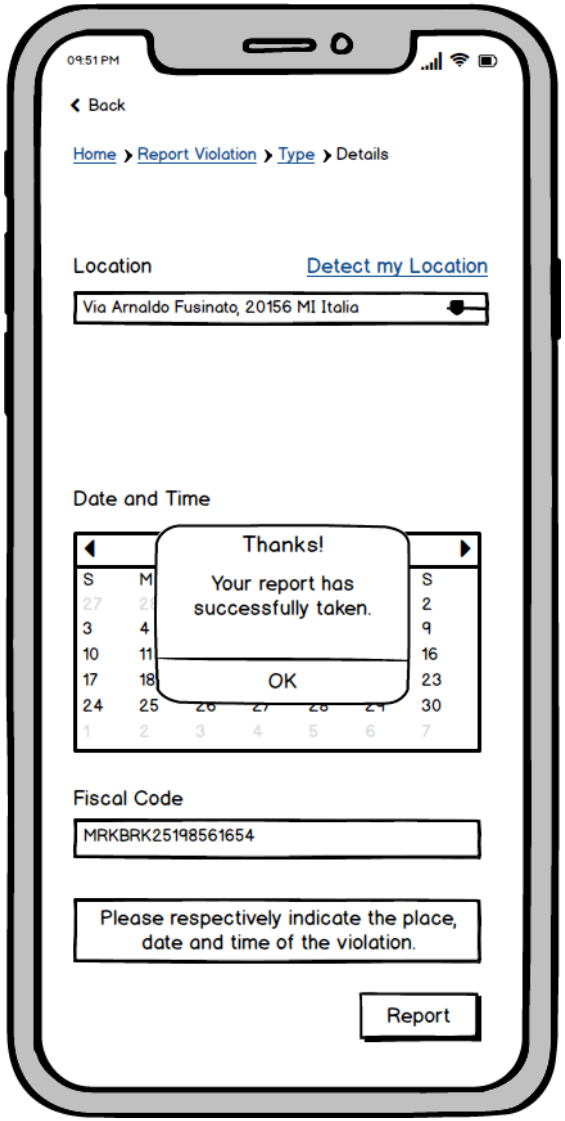

Figure 8 - Once report submitted

3.1.2. Hardware interfaces

The system has no hardware interface.

3.1.3. Software interfaces

The system has no software interface.

3.1.4. Communication interfaces

The system has an already encoded API to provide communication between the SafeStreets system and the authorities' service. This communication (API) is done by using the same DB for both side systems. *Figure 9* simply shows how communication is being revealed. It basically works as follows; the users report the violations through the app and the violations are saved to the database. Once a new data arrives at the database, it triggers the DBMS to pass the information to the authority side through the API. Thus, the violations are transmitted to the authorities.

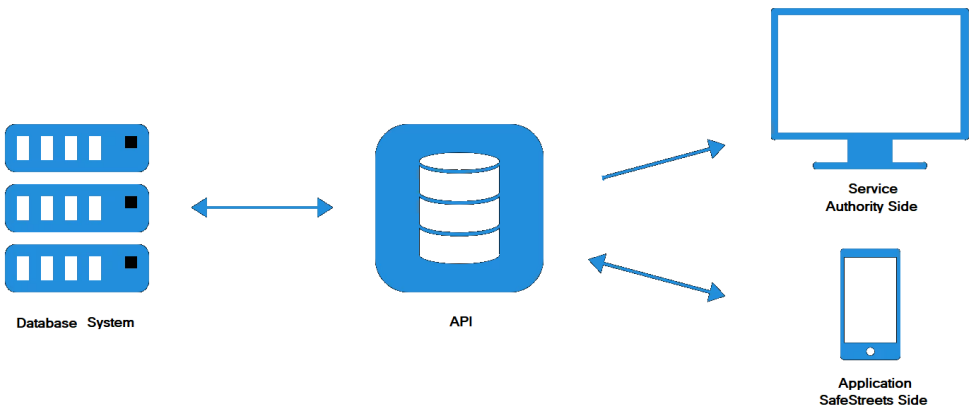

Figure 9 - Communication system

3.2. Functional requirements

3.2.1. User

Scenarios

- *Scenario 1*

  Tony is a working person and he goes to the job by his bike. A morning he is leaving his house and starting to bike to get his job. While he is on the road and biking on the cycle line, he realizes that there is a car parked at the middle of the bikeway. The only chance he has is to use the bikeway because there is no alternative cycle line that he can take. Then he stops and opens the SafeStreets application to report the situation by taking the picture of the violation. He selects the violation type as parking on the bikeway and submits the report. Thus, authorities will have been notified about the status in order to present a solution in time.

- *Scenario 2*

  Henry has just moved to Milan and does not know most of the things about the city. He is considering to rent a car in order to go to the shopping centre to buy stuffs for his new house. However, he is aware that the city is overcrowded and for that reason, he thinks that he will not find a place that he can park the car near the shopping centre. That's why he starts looking for a rental motorcycle. However, not to have a problem about the motorcycle he would like to check where he can park the motorcycle safely before he goes there. Afterward, he starts researching a source that is suitable for what he is looking for and he is ending up with finding the SafeStreets app. He accesses the application and enters the address of the shopping centre on the violation map. Then, he observes that the most common violation in the area he searches for is made to the motorcycle parking area. Regarding this observation, he finally changes his mind and decides not to rent a vehicle and he takes a taxi to carry what he has bought.

Use Cases

Name	Report a violation without having a saved picture
Actor	User
Entry Condition	The user has seen a violation.
Event Flows	1) The user opens the app. 2) The user clicks a report button to report the violation. 3) The user clicks on the “take a picture” button and uploads it. 4) The system detects the license plate of the vehicle. 5) The system gets the location, date and time information. 6) The system shows the collected pieces of information to the user. 7) The user presses the button of “Report”. 8) The system saves the violation to the database and notifies the authorities about the violation. 9) Finally, the system returns an alert to inform the user about the submission of the violation.
Exit Condition	The user clicks on the “OK” button in the alert by seen that the violation has been reported message.
Exceptions	• The user cannot find the violation type in the application. In this case, the user must choose the most related violation type(s) in the application. • The license plate of the vehicle cannot be detected by the application. In this case, the application cannot go forward and it gives an error to the user to make him take the picture again.

Name	Report a violation with a saved picture
Actor	User
Entry Condition	The user witnessed a violation, took a picture and saved it to her/his gallery.
Event Flows	1) The user opens the app. 2) The user clicks on the “Report a Violation” button. 3) The user clicks on the “Gallery” button to choose the violation picture from the gallery and uploads the picture to the application. 4) The system detects the license plate of the vehicle. 5) The system asks the user to provide the location, date and time information of the violation. 6) The user clicks on the “Report” button. 7) The system saves the violation to the database and notifies the authorities about the violation. 8) Finally, the system returns an alert to inform the user about the submission of the violation.
Exit Condition	The user clicks on the “OK” button in the alert seen that the violation has been reported message.
Exceptions	• The user cannot find the violation type in the application. In this case, the user must choose the most related violation type(s) in the application. • The license plate of the vehicle cannot be detected by the application. In this case, the application cannot go forward and it gives an error to the user to make him take the picture again.

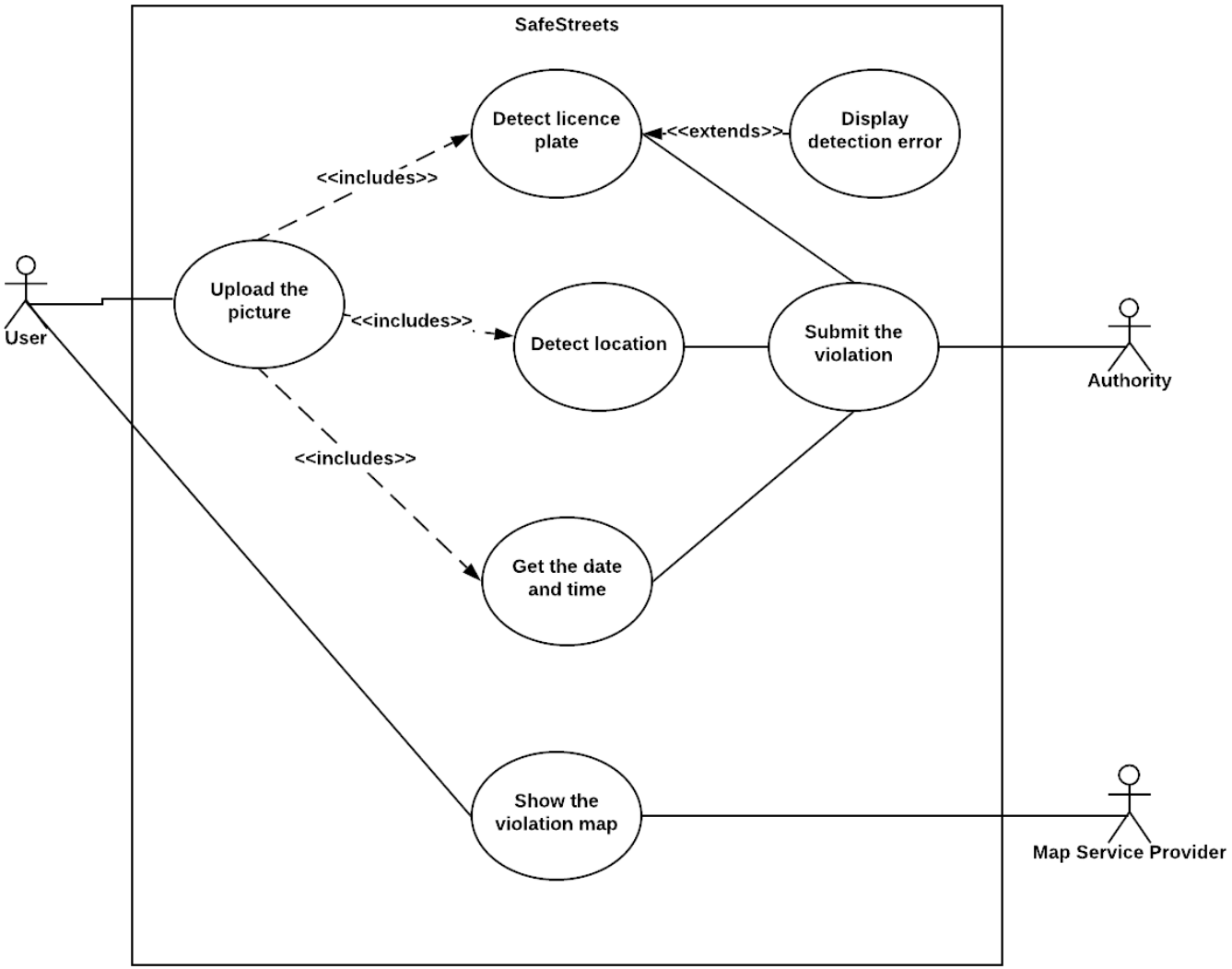

Figure 10 - Use case diagram

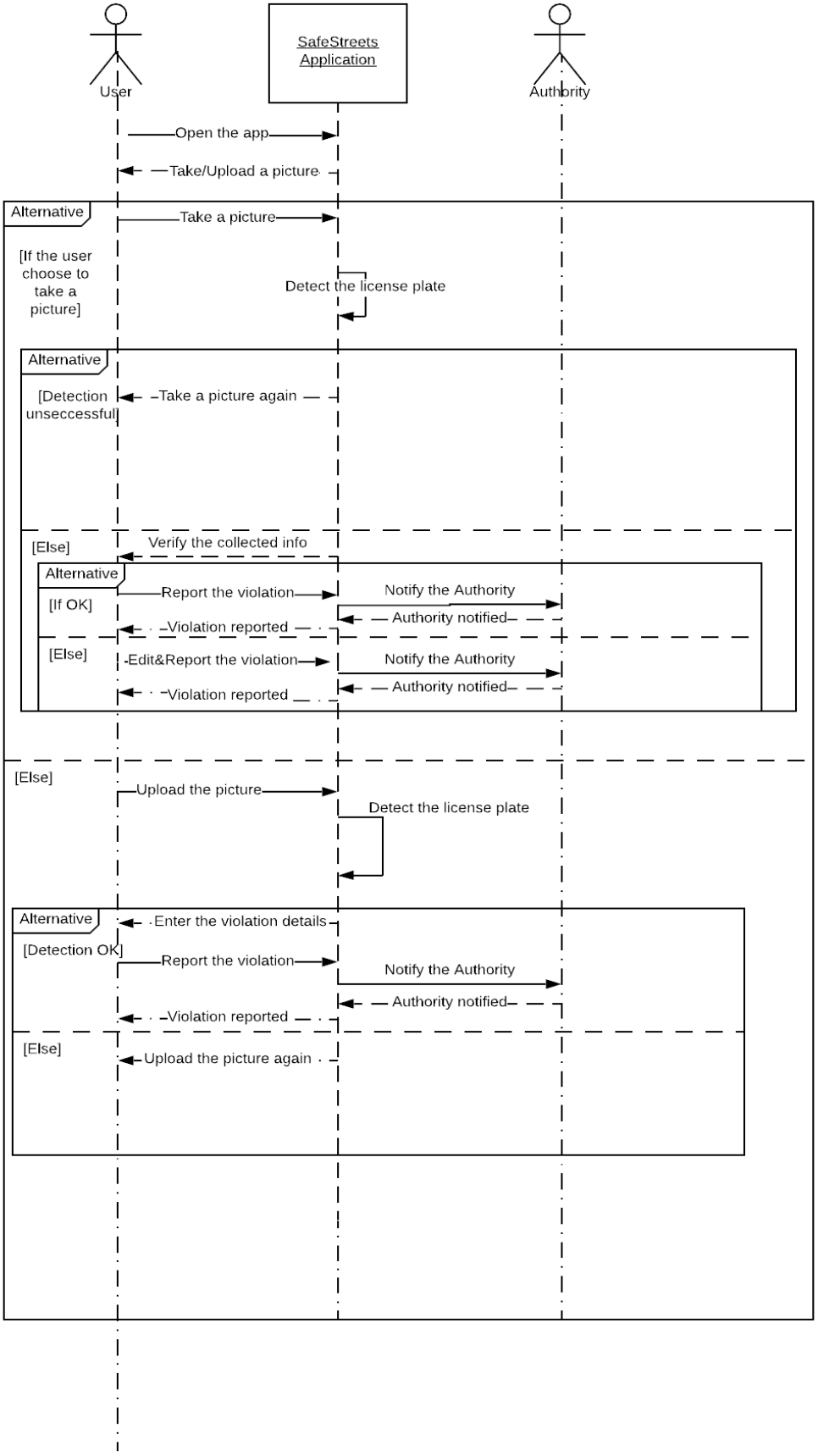

Figure 11 - Sequence diagram of "report violation" (User)

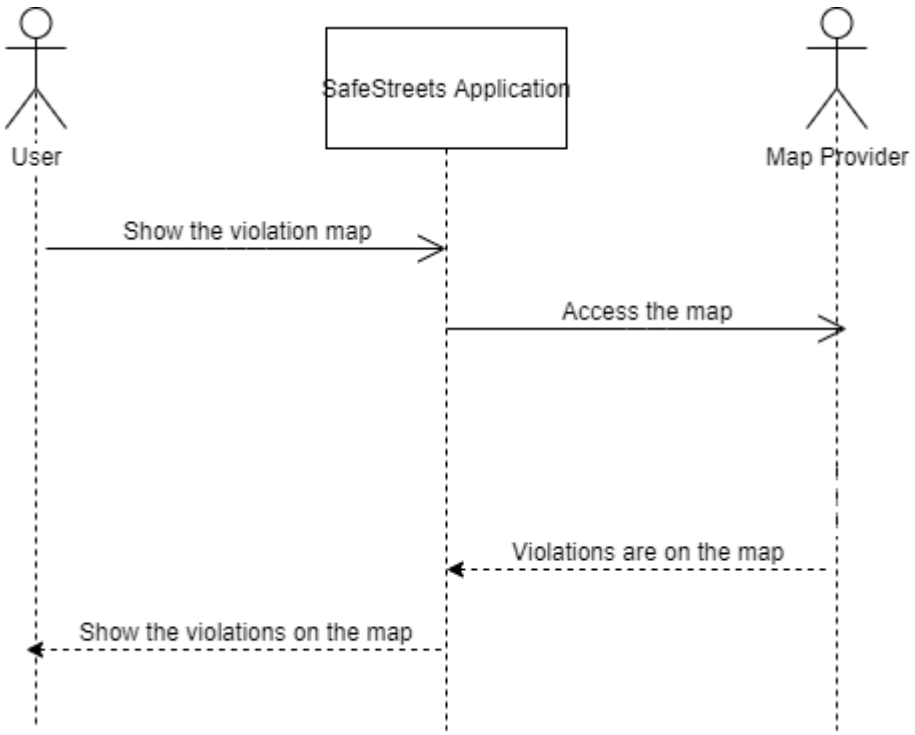

Figure 12 - Sequence diagram of "violations map" (User)

3.2.2. Authorities

Scenarios

- *Scenario 1*

  Hilal is a person who works for an authority that is responsible for traffic rules. Her boss wants her to make research related to the violation frequencies in the city. The institution that Hilal works for has the SafeStreets application system as well as the API service. Thus, she opens the application, accesses the violation map and checks where and what level of the frequency the most violation occurs in the city. Thanks to the SafeStreets system, she manages to prepare the documentation in order to present it to her boss.

Use Cases

Name	Show the violations and violation frequencies on the map.
Actor	User, Authority
Entry Condition	Both the user and authority have the application.
Event Flows	1) The user/authority opens the application. 2) The user/authority clicks on the button of "Show violation map". 3) The system sends the request to the database and gets the recorded violations to the requestor. 4) The system shows the violations.
Exit Condition	The user/authority observes the violations on the map and goes back to the main page or closes the application.
Exceptions	• The address entered is not valid. In this case, the application warns the user/authority with a corresponding error. • The map provider is in the status of the busy. In this case, the application gives the corresponding error to the user/authority.

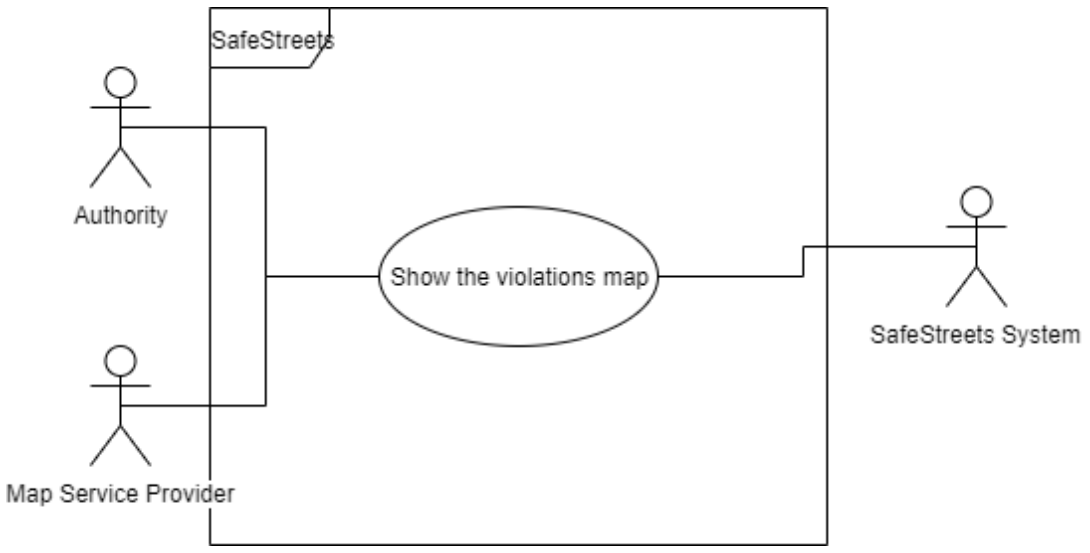

Figure 13 - Use case diagram (Authority)

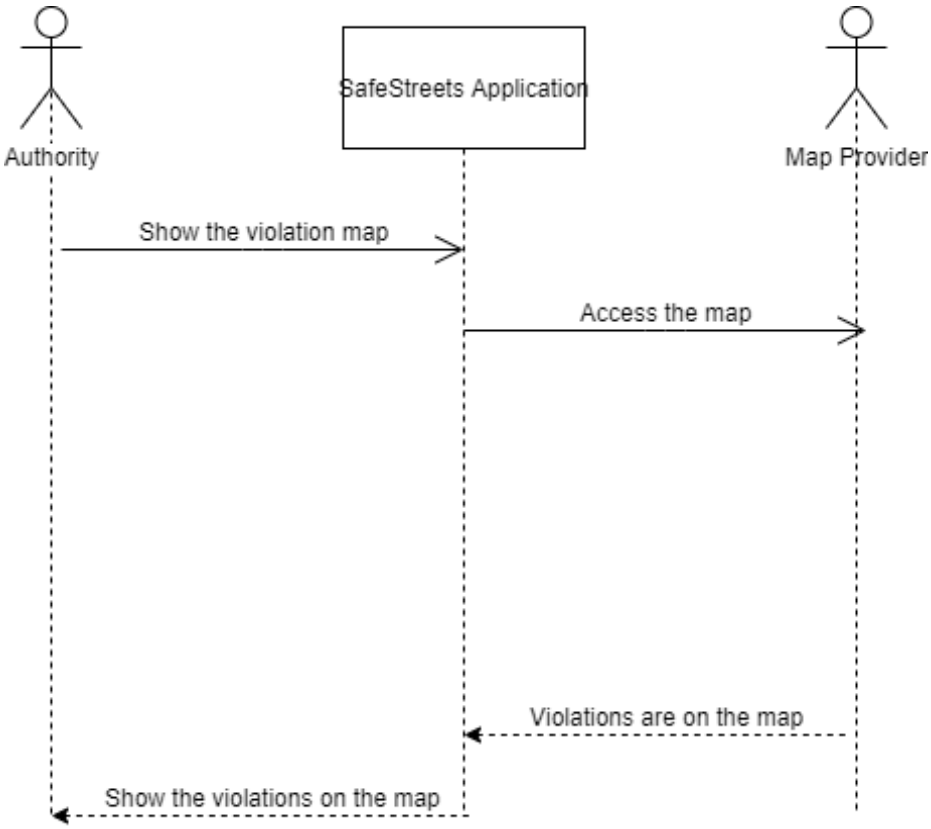

Figure 14 - Sequence diagram of "violation map" (Authority)

3.2.3. Requirements

The requirements have to satisfy the goals with the given domain assumptions. Therefore, associated requirements and domain assumptions will be given under the associated goals.

[G1] Allow users to report traffic violations.

- **[R1]** The system must save violations in real-time.
- **[R2]** The system must be able to process the reporting chain all the time.
- **[R3]** The system must have a DB that can sufficiently correspond to the needs to store the information provided by the users.
- **[R4]** The system must be able to prevent the submission of the same violation more than one time.

- **[R5]** The system must take the user back to the first step of the violation submission to make him/her provide a new picture if the license plate of the vehicle cannot be detected by the application.
- **[D1]** The picture taken or uploaded by the users is related to the violation.
- **[D3]** All information provided by the user is valid and correct. In other words, the chain of information coming from the users has never been broken, and the information has never been altered.
- **[D4]** GPS provides the location at most 5-meter mistake.
- **[D5]** The fiscal code provided by the user is valid and it belongs to the user who reports the violation.
- **[D6]** The user that intends to use the application has properly working internet connection.
- **[D7]** When the user has taken a picture of the violation through the app, he/she directly reports the violation without moving or going too far.
- **[D8]** The use of GPS service is allowed by the user.

[G2] Highlight the streets that the most violation occurs and provide this information to the users as well as the authorities.

- **[R6]** The system must be able to show the violations in real-time. So, if a new violation is saved to the database, then it must be shown on the violation map real-timely.
- **[R3]** The system must have a DB that can sufficiently correspond to the needs to store the information provided by the users.
- **[D3]** All information provided by the user is valid and correct. In other words, the chain of information coming from the users has never been broken, and the information has never been altered.
- **[D6]** The user that intends to use the application has properly working internet connection.
- **[D2]** The map service provider is always available to present the requested service.

[G3] The system must be able to notify the authorities about the violation sent by the users.

- **[R7]** The system must have an API service to provide and notify the authorities about the violations.
- **[R4]** The system must be able to prevent the submission of the same violation more than one time.
- **[D6]** The user (here the user is the authority) that intends to use the application has properly working internet connection.

3.3. Performance requirements

The system has to be able to serve authorities and a great number of users simultaneously. It has to guarantee quick, reactive and correct responses. It is important that the violation map must present a fluent service, especially while showing the whole city map with the violations.

3.4. Design constraints

3.4.1. Standards compliance

The system shows and accepts the provided date and time information in terms of *dd-mmm-yyyy hh:mm.* Additionally, It uses the 24 hours timing system. The system is improved by taking into consideration of security techniques determined in ISO/IEC Standard 15408.

3.4.2. Hardware limitations

The hardware limitations of the system can be identified as follows:

- Connection to the Internet (2G/3G/4G or WiFi).
- GPS sensor.

3.4.3. Any other constraints

The system can only provide its services to cities that have an agreement with the authorities. For that reason, the scope of the system is going to be expanded upon the coming requests by dealing with the new authorities.

3.5. Software system attributes

3.5.1. Reliability

The system has to have the ability to transfer user-supplied data to authorities without colliding with other violations. Therefore, the system must be running properly (without any bugs or delay) on the devices.

3.5.2. Availability

The availability of the system depends on the services provided by the SafeStreet application. For that reason, it is better to analyze the service's availability in parts. It is thought that the violations most likely occur during the part of the day when the people are awake. In addition, since the authorities cannot serve 7/24, it is determined that the reporting service provided by the SafeStreets application is going to be in service in the fragment of 70% during the day. However, the violation map service will be available in the fragment of %99.999 except the updates or improvements planned.

3.5.3. Security

The information and violations provided by the users are encrypted in the database system. As it has been mentioned in the section of standards compliance, ISO/IEC Standard 15408 is used to secure the system. Thus, the system is protected from any external or internal hacking attack.

3.5.4. Maintainability

The application must be designed with a structure that can be easily integrated as expected for the mapping service. Hereby, as a result of easy-integration will ease the improvements or updates planned in the future. It will be the consequence of low cost, as well.

3.5.5. Compatibility

The application offers multiple services and can be used by a variety of people. It is particularly designed for this purpose, not having complicated mechanisms. From a different viewpoint, complications bring high-level hardware requirements. In order to have a great number of user groups, the system is deprived of complications.

4) Formal Analysis Using Alloy

The written codes to model the system through Alloy is given below.

```
sig User{
report : some Violation
}

sig Authority{
gets : some Violation //??
}

sig Violation{
location : one Location,
picture : one Picture,
Date_Time : one ViolationTime
}

sig Location{
latitude:one Int,
longitude:one Int
}

sig Picture{
}

sig Date{
}

sig Time{
}

sig ViolationTime{
date : one Date,
time : one Time
}

fact "no shared photos"{
   all p: Picture |
      one v: Violation |
         p in v.picture
}
```

```
fact "no violations reported by the different users"{
   all v: Violation |
      one u: User |
         v in u.report
}

fact "every violationtime date is unique"{
   all d: Date|
      one vt: ViolationTime |
         d in vt.date
}

fact "every violationtime time is unique"{
   all t: Time|
      one vt: ViolationTime |
         t in vt.time
}

fact "no location associated to violation"{
   Violation.location = Location
}

fact "no violationTime associated to violation"{
   Violation.Date_Time = ViolationTime
}

fact "no same location values for different locations"{
   no disj l1,l2: Location |
      some l1.latitude & l2.latitude
//here we have chosen to latitude value in order to prevent the same location values
//for the different locations.
}

fact "no same location and same violationtime for different violations"{
   no disj v1,v2: Violation |
      some v1.location & v2.location and some v1.Date_Time & v2.Date_Time
//here we assume that no different violations can happen in the same location at the
//same time
}

fact "every violationtime time is unique"{
   all t: Time|
      one vt: ViolationTime |
         t in vt.time
}

fact "every reported violation is notified Authority"{
   all v: Violation |
      one a: Authority |
         v in a.gets
}
pred show{}
run show
run {} for 5 but 1 Authority
run {} for 5  but exactly 3 User
```

Results

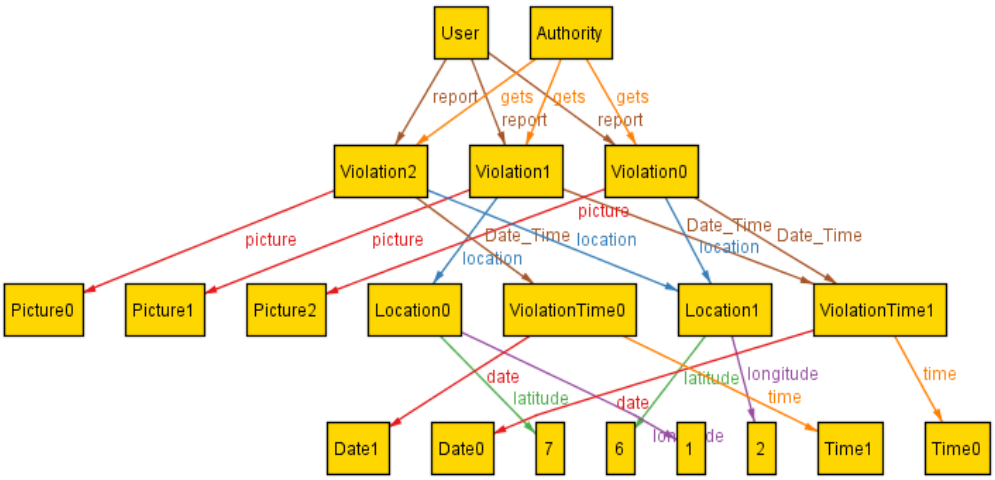

Figure 15 - Simplified model result

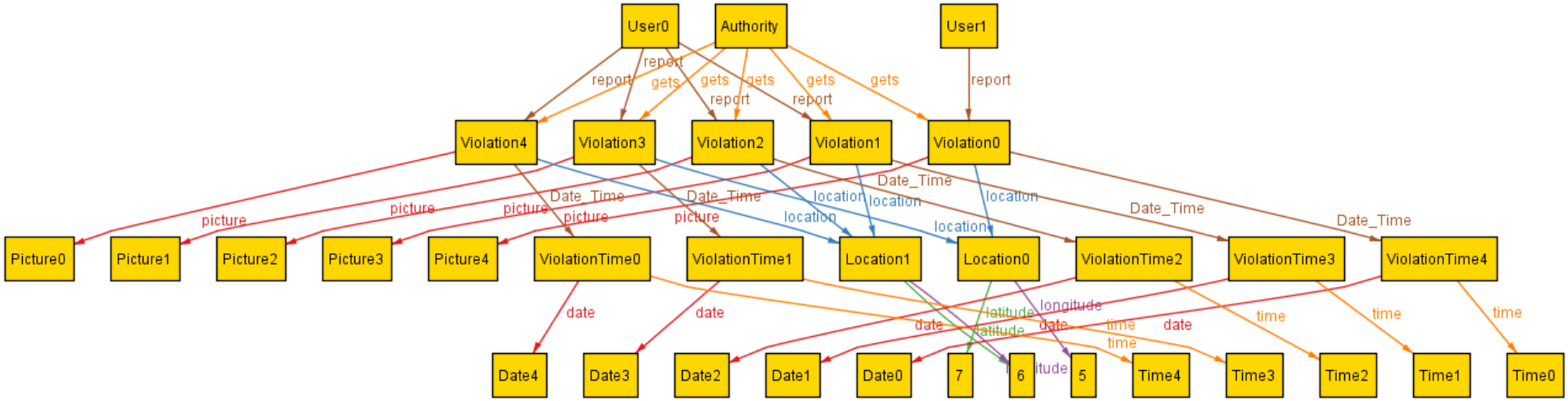

Figure 16 - Model Result

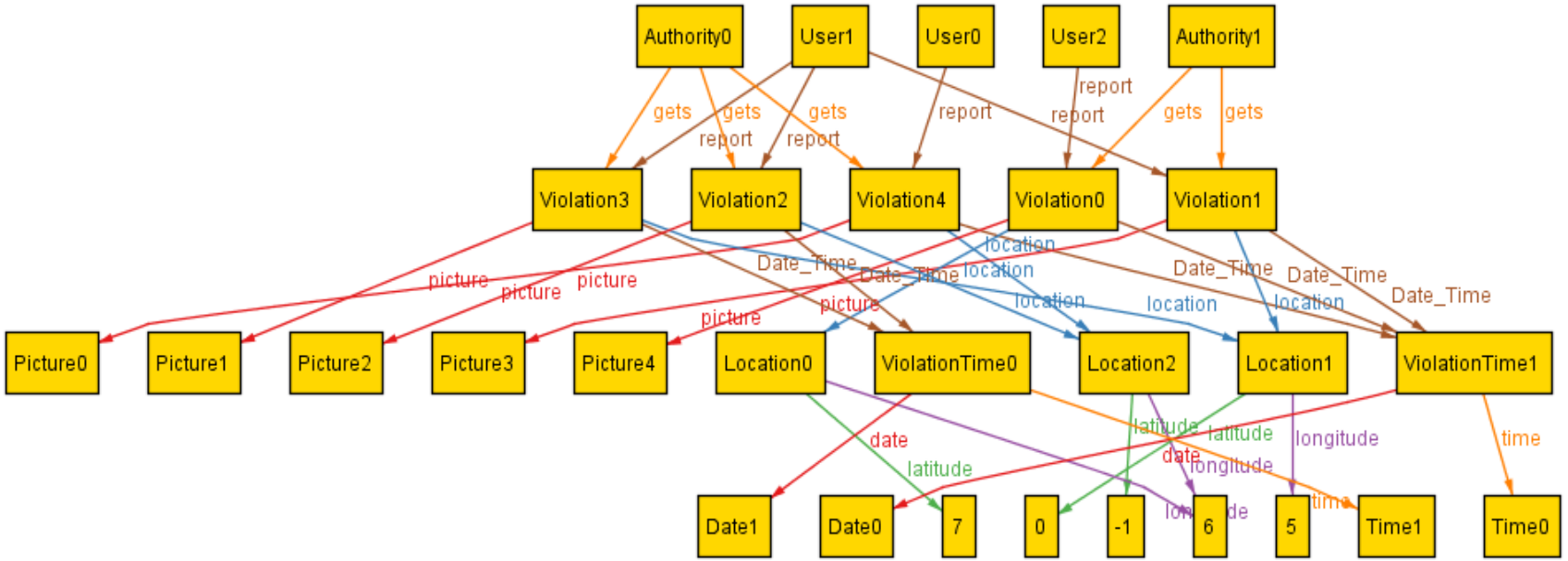

Figure 17 - Model result (Multiple Authorities)

Figure 17 has two different authorities as it is seen. The purpose of showing this if the system has agreed with more than one authority in the city, the model would be exactly as it is seen in *Figure 17*.

5) Effort Spent

Chapters of the Document	Hour(s) Spent
Introduction	3
Overall Description	4
Specific Requirements	8
Formal Analysis Using Alloy	6
TOTAL HOURS SPENT	**21**

6) References

1) https://tallyfy.com/uml-diagram/
2) https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/
3) https://esb-dev.github.io/mat/alloy-cheatsheet.pdf