Politecnico di Milano
AA 2019/2020

POLITECNICO
MILANO 1863

RASD – Requirement Analysis and Specification Document

Version 1.1 – 17/11/19

Author: Ines Hafdaoui

Professor: Elisabetta Di Nitto

Index

1. Introduction

1.1 Purpose

1.1.1 General purpose

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations. The application allows users to send pictures of violations, including their date, time, and position, to authorities. Examples of violations are vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on.
SafeStreets stores the information provided by users, completing it with suitable metadata. In particular, when it receives a picture, it runs an algorithm to read the license plate (one can also think of mechanisms with which the user can help with the recognition), and stores the retrieved information with the violation, including also the type of the violation (input by the user) and the name of the street where the violation occurred (which can be retrieved from the geographical position of the violation). In addition, the application allows both end users and authorities to mine the information that has been received, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations. Of course, different levels of visibility could be offered to different roles.

1.1.2 Goals

- Provide users with the possibility to notify authorities when parking violations occur
- The application allows users to send pictures of violations, including their date, time, and position, to authorities
- Allow both end users and authorities to mine information and stats about parking violations

1.2 Scope

The following table analyzes the world and the shared phenomena

Phenomenon	**Shared**	**Who controls it**
User wants to notify a parking violation	N	W
User logs in the application	Y	W
System verify whether the account exists	N	M
The application shows the homepage	Y	M
The application requests the access data again showing that those previously entered do not correspond to an account	Y	M
User compiles form with the information and send a picture	Y	W
When picture is received system runs an algorithm to read the licence plate	N	M
System retrieves the geographical position of the violation using GPS	N	M
The application stores the retrieved information	N	M
System checks that the same violation has not already been reported	N	M
The application warns user that the violation has already been reported	Y	M
System updates stats	N	M
The application confirms the notification has been registered	Y	M
The application sends the notification to authorities	Y	M

1.3. Definitions, acronyms, abbreviations

1.3.1. Definitions

- Parking violation

  With parking violation is intended the act of parking a motor vehicle in a inappropriate place or in an unauthorized manner. The system considers among parking violation:

  - parking in a reserved place for example in a bus stop or in handicapped zone without an appropriate permit or in front of a garage entrance or in a parking reserved for employees of a company;
  - parking in areas that are not parking i.e. in places not used for parking for example on pedestrian crossing, on a sidewalk (unless specifically allowed by signs), parking in, too close to or within an intersection, railroad crossing or crosswalk;
  - double parking;
  - parking at a parking meter without paying, or for longer than the paid time;
  - vehicle unauthorized for example with expired or missing license plate or license plate ‘tabs’ without proper safety vehicle inspection decal;
  - parking in a wrong way for example outside marked squares or parking facing against the direction of traffic.

- End user

  End users will be used to refer to common citizens who can be registered or not, who can be interested in consulting statistics

- Authorities

  With authorities we refer to anyone representing an authority

- User

  With user we refer to citizens i.e. to every person who is not part of any authority body and who is already signed up

1.3.1. Acronyms

- GPS: Global Positioning System
- S2B: Software To Be

1.3.1. Abbreviations

- Stats: Statistics
- i.e.: id est, that is

2. Overall Description

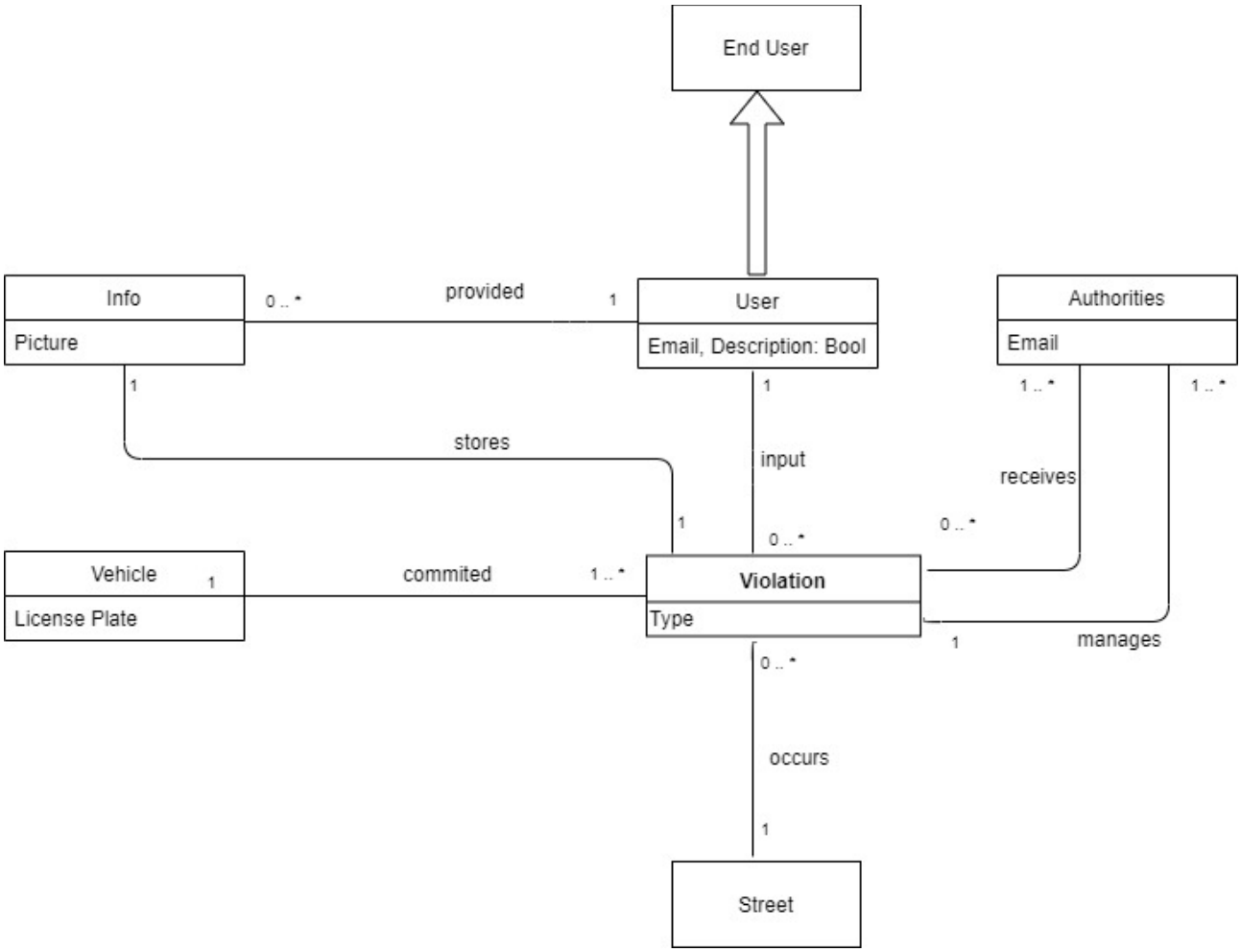
2.1. Product perspective

The application that will be realized is thought to offer the possibility for people to report parking violation to authorities who receive a notification every time a new report is registered.
Moreover SafeStreets should offer the possibility to read retrieved data and consult statistics. Obviously different functionalities are offered to different roles.

Anyone can sign up and users registered can report a parking violation.
To report a violation the user must compile a form in which he will be asked to send a picture, through which the system will acquire the license plate, insert the type of violation and give a brief description in order to help the system in recognizing the vehicle. Once completed, by submitting the form, the notification will be sent to authority.

Moreover, the application will allow end users, so common citizens and authorities, to mine the information that has been received and to see statistics that the application updates every time a user reports a violation, granting different levels of visibility to different roles.
In particular, for every violation end users can see the type of violation and the name of the street but they have not access to license plate and in general to information related to who has committed the violation.
Otherwise authorities can see every information related to the violations.

2.1.1. Class diagrams



2.1.2. State diagrams

No state diagram is represented because there is not any state changing into object.

2.2. User characteristics

2.2.1. Identification of the actors

The actors of the system are the entity who interact with the system. For this application there are three types of actors:

- user: he is signed up to the application and can logs in and report violations, he can also see stats for example about the areas where the most frequent violations occur;
- authorities: he has access to all data and stats stored in the system and is notified every time a new violation is reported;
- end users: every person that may not be signed up and that has access to the same information available to users.

2.3. Product functions

2.3.1. Functional requirements

In order to implement the S2B is necessary define the basic functionalities it is supposed to offer. There are different functionalities for different actors explained as follows:

- actor: user

  functionalities

  - creation and submission of a report of a violation specifying the type, attaching a picture and possibly adding a brief description of the vehicle;
  - possibility to consult data collected and statistics of interest to him;

- actor: authorities

  functionalities

  - receive a notification when a new violation is reported showing the type, the license plate, a description of a vehicle if it has been inserted by the user who has made the notification, the geographic position where the violation occurred;
  - possibility to consult all data and statistics stored in the system.

- actor: end user

  functionalities

  - possibility to consult data collected and statistics of interest to him.

2.4. Assumptions and dependencies

2.4.1. Domain assumptions

Here are presented the domain assumptions, so the functionalities we assume our software will implement, so that we don't need to explain or think of how implement them.

DA:

- the S2B stores every violation only once;
- the S2B stores the information provided by users, completing it with suitable metadata;
- it knows the geographical position of the violation by accessing to the GPS of the devise that user is using for the report;
- the type of violation is input by the user.

3. Specific Requirements

3.1. Functional requirements

3.1.1. Use case diagrams

Report Violation Event

Name	Report Violation Event
Actor	User
Entry condition	The citizen knows everything about the violation he wants to report
	1. In the homepage, the user clicks on the "Report Violation" button entering in the reporting page. 2. The user selects the type of violation, by choosing it in a list. 3. The user attach a picture by choosing between the camera icon or the image gallery icon. 4. The user can optionally insert a brief description of the vehicle for example inserting the car manufacturer, the color or the model. 5. The user clicks on the "Confirm" button.

	6. FRIEND (the S2B) checks that the violation has not been reported yet.
Exit condition	The violation is stored so FRIEND confirms the sending to the user.
Exception	FRIEND finds that the violation is already stored so it has already been reported (in stating this the application takes into account the date and time it has retrieved by the picture sent and the street name it had acquired through GPS) and warns the user that the report has already been made.
Special Requirements	The reported violation must be stored in less than 1 second.

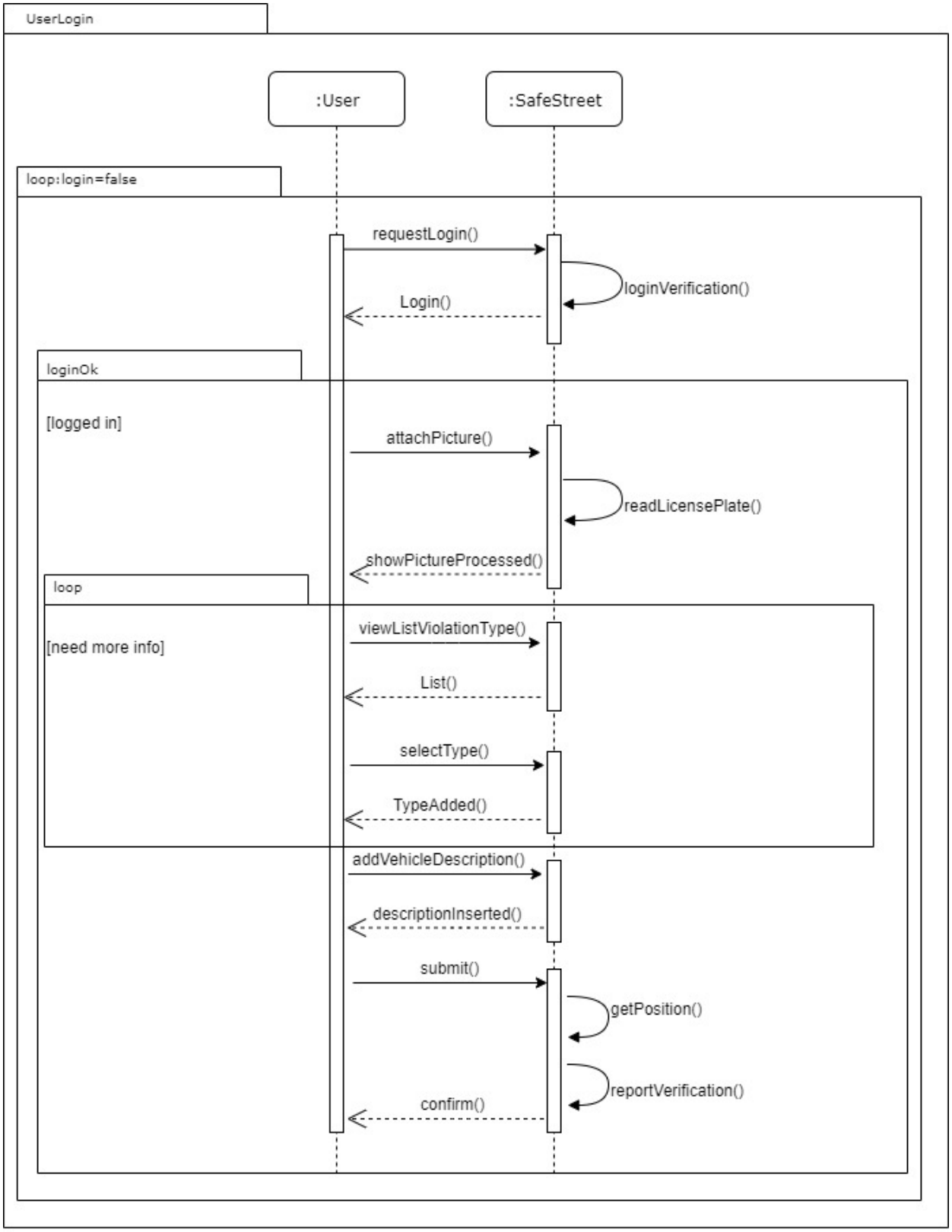
Receive Violation Notification

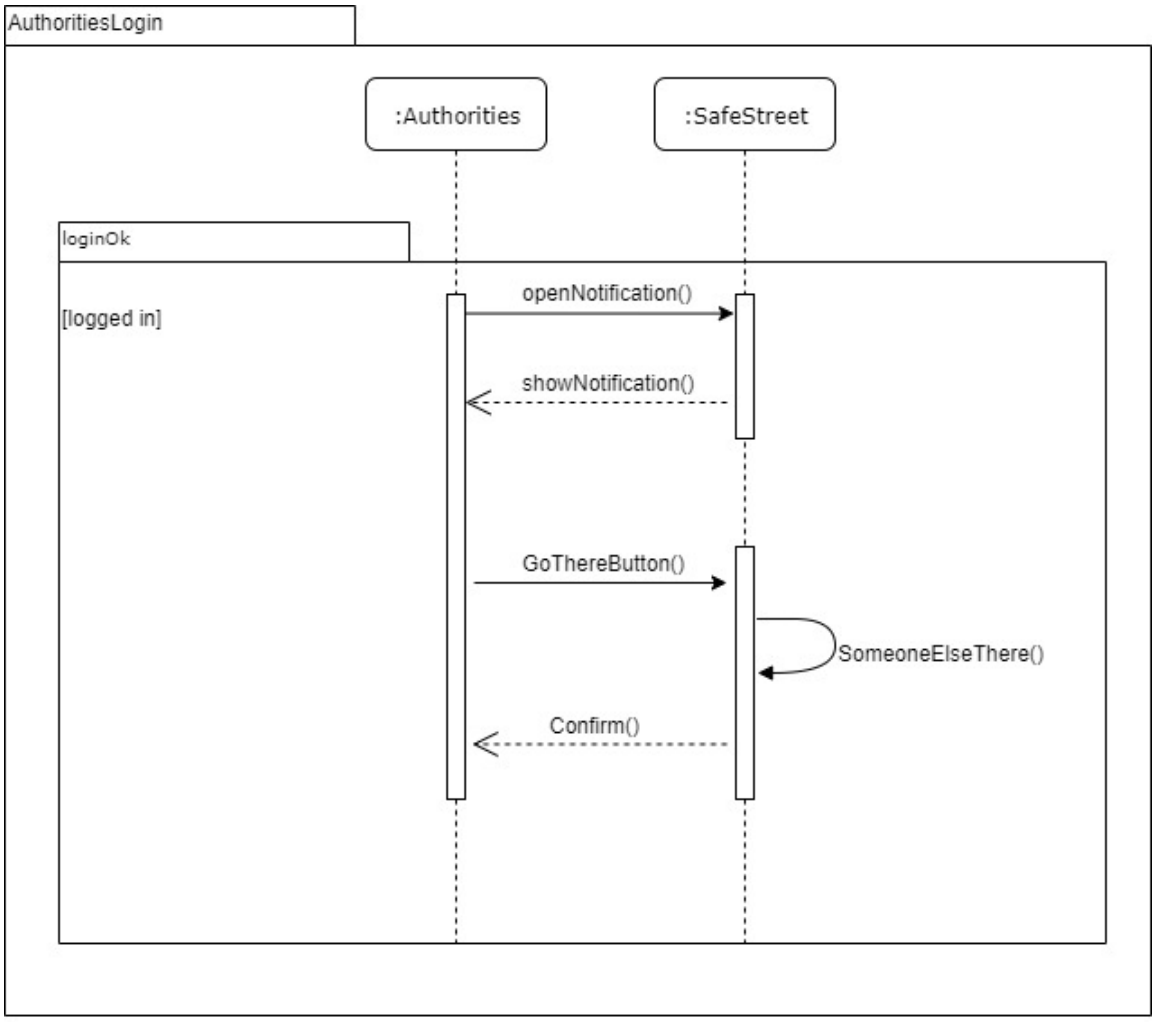
Name	Receive Violation Notification
Actor	Authorities
Entry condition	• The authorities are signed up to the application • Every violation reported can be notified to the authorities
	1. A notification appears in the screen dropdown. 2. The authorities clicks on the notification. 3. FRIEND starts and shows the form compiled by the citizen who has made the report within the information acquired by the application. 4. The authorities clicks on the “Go there” button.
Exit condition	FRIEND registers that this account has received and read the notification and that he is going in the street indicated.
Exception	Anyone else is already going there
Special Requirements	The notification must be stored as read in less than 1 second.

Consult Violation data and stats

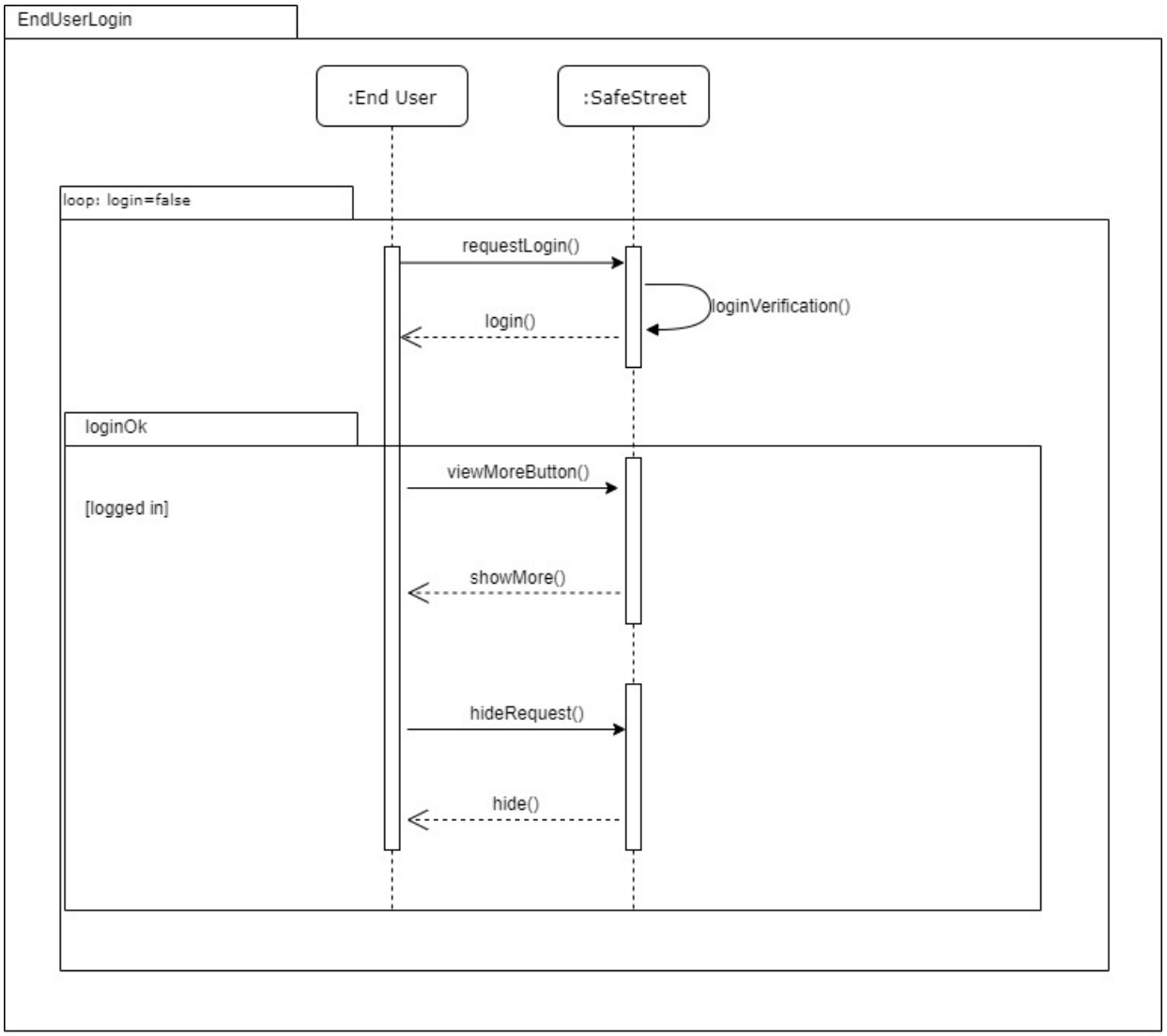
Name	Consult Violation data and stats
Actor	End users, authorities
Entry condition	
	1. In the homepage, the end user or the authorities clicks on the "View more" button entering in the data and stats page. 2. FRIEND shows stats and data about violations considering whether the request is made by authorities or anyone else. 3. The end user or authorities clicks on the "Hide" button.
Exit condition	
Exception	
Special Requirements	

3.1.2 Sequence Diagrams

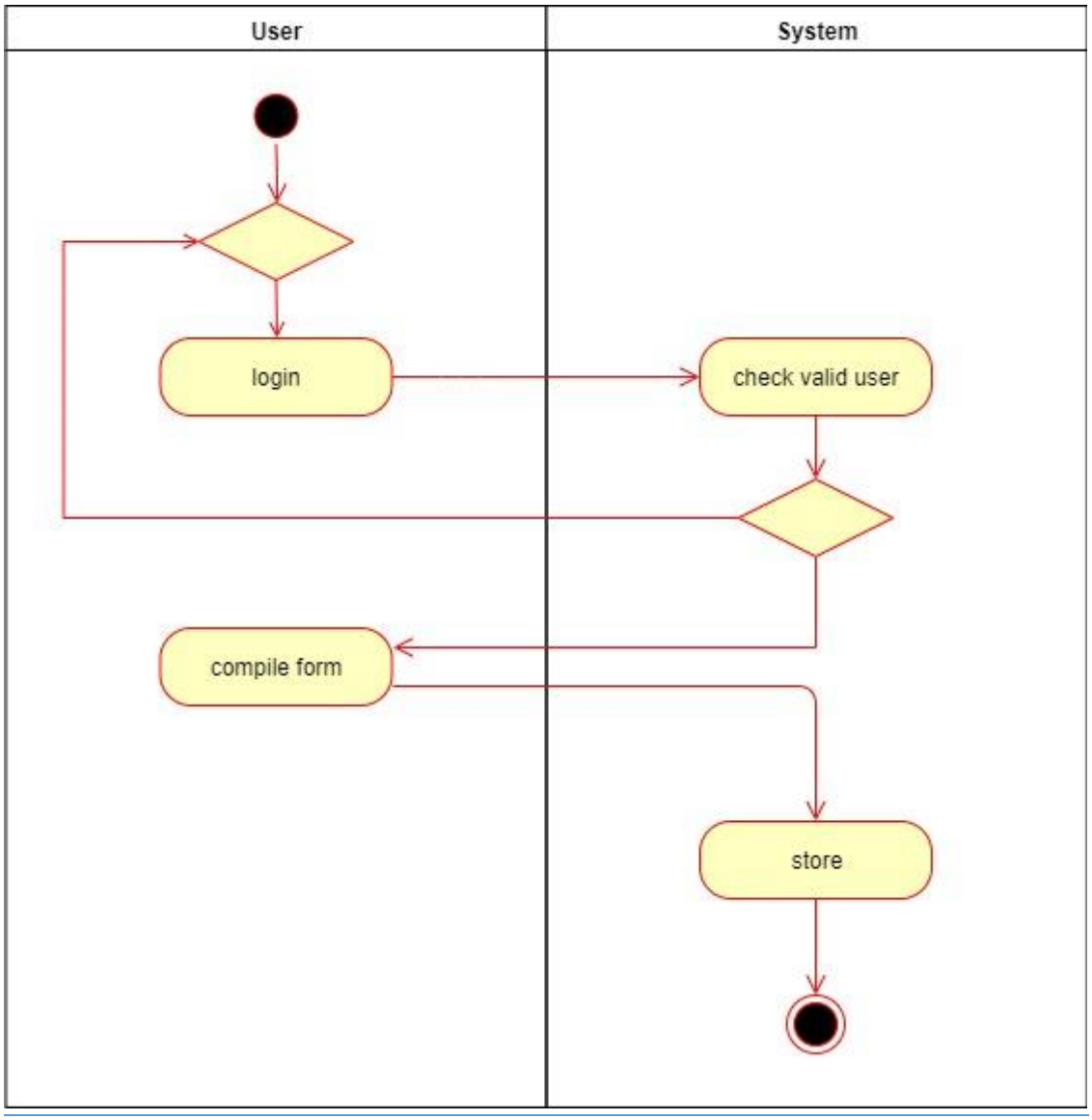



AuthoritiesLogin
:Authorities
:SafeStreet
loginOk
[logged in]
openNotification()
showNotification()
GoThereButton()
SomeoneElseThere()
Confirm()

The following one represents the scenario “Consult violation data and stats” so is valid for both end users and authorities though it is shown only for end users.



3.1.3 Activity Diagrams



The activity diagram is made only for the scenario “Report Violation Event” because it involves more activities, while it seems not necessary for the others because they involve few activities and simple processes.

4. Formal Analysis using Alloy

The Analysis using Alloy is based on Class Diagram for what concerns objects and relations modeling. Goals, requirements and domain assumptions are clearly identified by labels, so that it's easy to see that goals are granted once domain assumptions and requirements are defined. This analysis aims to ensure basics goals are reached.

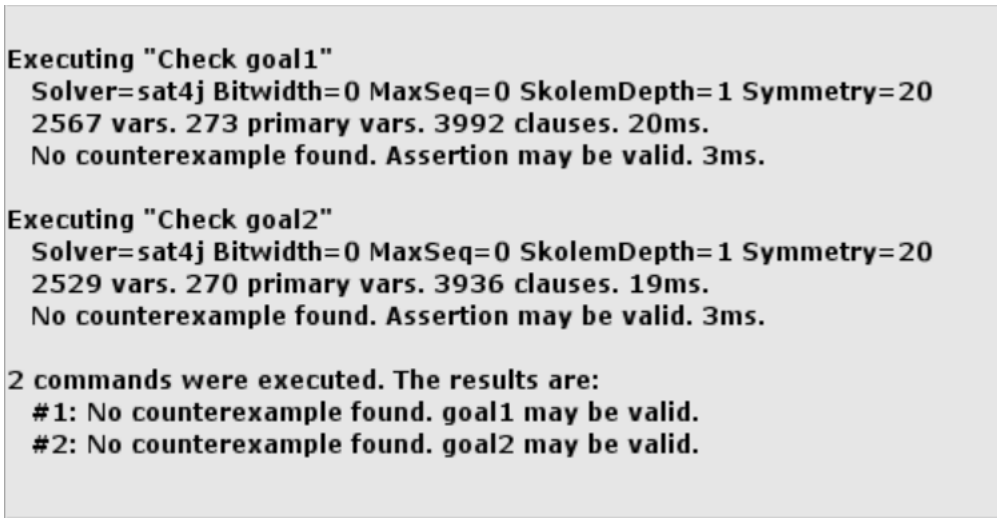
It also considers domain assumptions not explicitly mentioned before but that are implicitly required in the model.

```
open util/boolean

abstract sig ViolationType { }
one sig DoubleParking extends Violation { }
one sig OverReservedPlace extends Violation { }
one sig InAWrongWay extends Violation { }
one sig UnauthorizedVehicles extends Violation { }
one sig WithoutPaying extends Violation { }
one sig InAreasNotParking extends Violation { }

sig EndUser{ }

sig Email{ }

sig Street{ }

sig LicensePlate{ }

sig Picture{ }

sig Violation{
type: one ViolationType,
reported: one User,
street: one Street,
vehicle: one Vehicle,
managed: Bool
}{managed=False}

sig User extends EndUser{
email: Email,
description: Bool,
report: some Violation,
GPSPosition: Street
}
```

```
sig Authorities{
email: Email,
receives: some Violation,
manages: one Violation
}

sig Vehicle{
licensePlate: LicensePlate
}

sig Info{
providedBy: one User,
stores: one Violation,
picture: some Picture
}

fact NoDuplicateUsers{
no disj u1,u2:User | u1.email=u2.email
}

fact NoDuplicateAuthorities{
no disj a1,a2:Authorities | a1.email=a2.email
}

fact NoDuplicateVehicles{
no disj v1,v2: Vehicle | v1.licensePlate=v2.licensePlate
}

fact NoDuplicateInfo{
no disj i1,i2:Info | i1.picture=i2.picture}

fact ViolationInReported{
all v:Violation | lone u: User | v.reported=u and v in u.report
}
```

```
fact EveryViolationReportedBySomeone{
all v:Violation | all u:User| v!=none implies v.reported in u
}

fact domainAssumption1{
    all v:Violation | no disj u1, u2:User | v.reported = u1 and v.reported = u2
}

fact domainAssumption2{
    all i:Info | i.providedBy!=none implies i.stores!=none
}

fact domainAssumption3{
    all v:Violation | v.street=v.reported.GPSPosition
}

fact domainAssumption4{
    all v:Violation | v.type!=none implies v.reported!=none
}

fact violationManaged{
    all v:Violation| some a:Authorities| v in a.manages <=> v.managed=True
}

fact authoritiesManages{
    all v:Violation| one a1: Authorities| all a: Authorities| a.email!=a.email and v not in a.manages implies v in a1.manages
}

fact requirement{
    all i:Info | i.picture!=none implies i.stores.vehicle.licensePlate!=none
}

fact authoritiesReceived{
    all v:Violation| all a: Authorities| v.managed=False implies v in a.receives
```

```
fact domainAssumption3{
    all v:Violation | v.street=v.reported.GPSPosition
}

fact domainAssumption4{
    all v:Violation | v.type!=none implies v.reported!=none
}

fact violationManaged{
    all v:Violation| some a:Authorities| v in a.manages <=> v.managed=True
}

fact authoritiesManages{
    all v:Violation| one a1: Authorities| all a: Authorities| a.email!=a.email and v not in a.manages implies v in a1.manages
}

fact requirement{
    all i:Info | i.picture!=none implies i.stores.vehicle.licensePlate!=none
}

fact authoritiesReceived{
    all v:Violation| all a: Authorities| v.managed=False implies v in a.receives
}

assert ViolationNotified{
    all v:Violation| all a: Authorities| v in a.receives
}
check goal1

assert ViolationManaged{
    all v:Violation | v.managed=True
}
check goal2
```



As it's clear from the picture, the solver does not find any counterexample so the model is basically correct from a logic point of view.

5. References

- https://docs.alloy.co/
- https://about.draw.io/tag/user-documentation/
- *ISO/IEC/IEEE 29148*