

Politecnico di Milano
A.A. 2019-2020
Software Engineering 2: “SafeStreets”
Requirements **A**nalysis
and
Specifications **D**ocument

Frederik Saraci, Lorenzo Amata

November 15, 2019



POLITECNICO
MILANO 1863

Contents

1	Introduction	4
1.1	Purpose	4
1.2	Actual System	4
1.3	Purpose	4
1.3.1	Goals	5
1.4	Glossary	5
1.4.1	Definitions	5
1.4.2	Acronyms	5
1.5	Abbreviations	6
1.6	Overview	6
1.7	References	6
2	Overall description	7
2.1	Product perspective	7
2.2	Product Functions	13
2.3	User characteristics	13
2.4	Domain assumption, dependencies and constraints	13
3	Specific Requirements	15
3.1	External interface requirements	15
3.1.1	User Interfaces	15
3.2	Functional requirements	15
3.2.1	Requirements	15
3.3	Non-functional requirements	16
3.3.1	Performance	16
3.3.2	Reliability	16
3.4	Security	16
4	Use Cases	17
4.1	Scenarios	17
4.1.1	Scenario 1	17
4.1.2	Scenario 2	17
4.1.3	Scenario 3	17
4.1.4	Scenario 4	17
4.1.5	Scenario 5	17
4.1.6	Scenario 6	17
4.1.7	Scenario 7	17
4.2	Use case diagram	18
4.2.1	Registration	19
4.2.2	Login	20
4.2.3	Report violation	21
4.2.4	User Statistics request	22
4.2.5	User information edit	23
4.2.6	Generate traffic ticket	24
4.2.7	View Suggestions	25
4.2.8	Authority Statistics request	26
4.2.9	Verification	27

1 Introduction

1.1 Purpose

The purpose of a Requirement Analysis and Specifications Document is the process of discovering the purpose for which a software system was intended, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. It is also concerned with the relationship of software's factors such as goals, functions and constraints to precise specifications of software behavior, and to their evolution over time and across software families.

1.2 Actual System

The system we are to develop is brand new, there is no similar previous system

1.3 Purpose

SafeStreets is a crowd-sourced application which aim at providing cooperation among citizens (end users) and authorities (municipality, local police ecc.) in order to make more livable our urban environments. Indeed, by means of SafeStreets authenticated users can notify authorities when traffic violations occur, in particular parking violations. Before notifying traffic violations the user has to fill the registration form providing all the mandatory fields (including a valid ID document) in order to be recognized by the authority. The application allows users to send pictures of the various type of violations (vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on), including their date, time, position, type of violation, notes and license plate. SafeStreets stores the information provided by users, completing it with suitable metadata, in particular when it receives a picture, it runs an external algorithm (OCR) to read the license plate, the license plate is also provided by the user during the report, especially if the environment conditions are not favorable (insufficient lighting). The information provided by users can also be used by themselves or authorities (whose type of visibility may vary depending on the type of the user who is interfacing the application) for statistic purposes, e.g. highlighting the streets (or areas) with the highest frequency of violations, or show a graph with the vehicles that commit most violations or the type of violations for a specific street. SafeStreets can even cooperate with municipality's services (if available), crossing the information about incidents in the territory of the municipality with its own stored data to identify potentially unsafe areas and suggest possible interventions in order to prevent possible future incidents or violations. SafeStreets is intended to be a means to help people becoming more civilized improving the efficiency of authorities to guarantee the respect of the traffic laws, the municipality (and in particular local police) could use the data stored in SafeStreets about traffic violations to generate traffic tickets. In this case, the authenticity of the stored data must be guaranteed, if it is not the data must be discarded. From the opposite side, SafeStreets can use the information about the generation of traffic tickets to build statistics about the effectiveness of SafeStreets initiative.

1.3.1 Goals

- Allow the users to send data about traffic violations
- Allow the authorities to access to traffic violation data
- Allow the authorities to see specific kind of statistics about traffic violations
- Allow users to see specific kind of statistics about traffic violations
- Allow the local police to use the data stored regarding the traffic violation to issue traffic tickets
- Allow to exploit the traffic ticket statistics to check its effectiveness
- Allow to cross information between the information of incidents in the area that belong to a municipality and data stored in the system to suggest possible interventions to municipalities

1.4 Glossary

1.4.1 Definitions

- *System*:: the SafeStreets software we are to develop
- *Guest or Guest User*:: the person who access the system as no logged user
- *Logged user or Authenticated user*:: authenticated person (not authority) who is interfacing with the system
- *User*: logged user or Authority user in general
- *Banned User*: A logged User or Authenticated user that cannot interface with the system anymore due to violations of terms and conditions of SafeStreets
- *Authority User or Autheticated authority*:: The authenticated public organization which use Safestreet
- *Local Police or Police*:: The department of a local police
- *Registration*:: when user provide all the necessary data for accounting him into the system
- *Violation or Traffic violation*: a violation of the traffic laws in general

1.4.2 Acronyms

- RASD: Requirements Analysis and Specification Document
- OCR: Optic character recognition
- DBMS: Data Base Management System
- DB: Database

- API: Application Programming Interface
- GPS: Global Position System
- S2B: Software to be

1.5 Abbreviations

- *i.e.*: that is
- *w.r.t.*: with respect to
- *i.d.*: id est
- *i.f.f.*: if and only if
- *e.g.*: exempli gratia
- *etc.*: et cetera

1.6 Overview

This document is structured in five parts:

- Introduction: it provides an overview of this entire document and product goals
- Overall description: it describes general factors that affect the product providing the background for system requirements
- Specific requirements: it contains all system's functional and nonfunctional requirements
- Use cases identification: it contains the usage scenarios of the system with the use case diagram, use cases descriptions and other diagrams
- Appendices: it contains the Alloy model, software and tools used, hours of work per each team member SafeStreets

1.7 References

- Specification document: "Mandatory Project Assignment AY2019-2020"
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
- Alloy doc: <http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html>

2 Overall description

2.1 Product perspective

As stated before the S2B : SafeStreets, provides different services with slight variations depending on which kind of user is interfacing with the system, while others types of services are provided only at certain kind of users. The system offers the basic service which is collecting the information about a traffic violations provided by logged users(not authorities or local police and hence Authority users) and shows statistics about them, the information inserted by logged users are the type of violations,location,date,time,etc. and for what concern the location the system is able to exploit the device's available sensors and GPS together with the **Google Maps APIs**. Google Maps has been chosen instead of other mapping and location provider since it offers a very large library of APIs with extensive documentation and so it will be easier to exploit these functionalities and also to add possible future functionalities that have to exploit these mapping services into our system. The Google Maps APIs are even used in the basic service for providing statistics about traffic violations (as mentioned before,e.g., when highlighting a street with the highest frequency of a certain type of traffic violation we do that using the Google Map APIs). Moreover,in the basic service, the S2B on receiving new data from a logged user, it runs an algorithm to read the license plate of the vehicle from the picture, even though it seems an easy job it may be very complex, therefore SafeStreets exploit an external algorithm for doing that, **OCR**. The OCR is a well known algorithm that allows to capture characters of a license plate (in this case) from a picture, to reduce the amount of error it apply several filters to discard character that don't belong to the license plate. For what concern the specific services provided by SafeStreets at the authority users (e.g. suggest interventions in unsafe areas)the system need to exploit the information about incidents in a certain territory, these **information** are provided by the **municipalities** and without them the S2B cannot provide the service. The S2B must guarantee the **authenticity** of the data stored, this situation is different than the mentioned before, in fact, for doing so we will not use external functionalities because the programming language used in the S2B already provides a mechanism for guaranteeing it. The below high-level class diagram provides a model of the application domain: it contains only few essential attributes for the various classes and does not include every class that will be necessary to define the model(useful data)of the system.

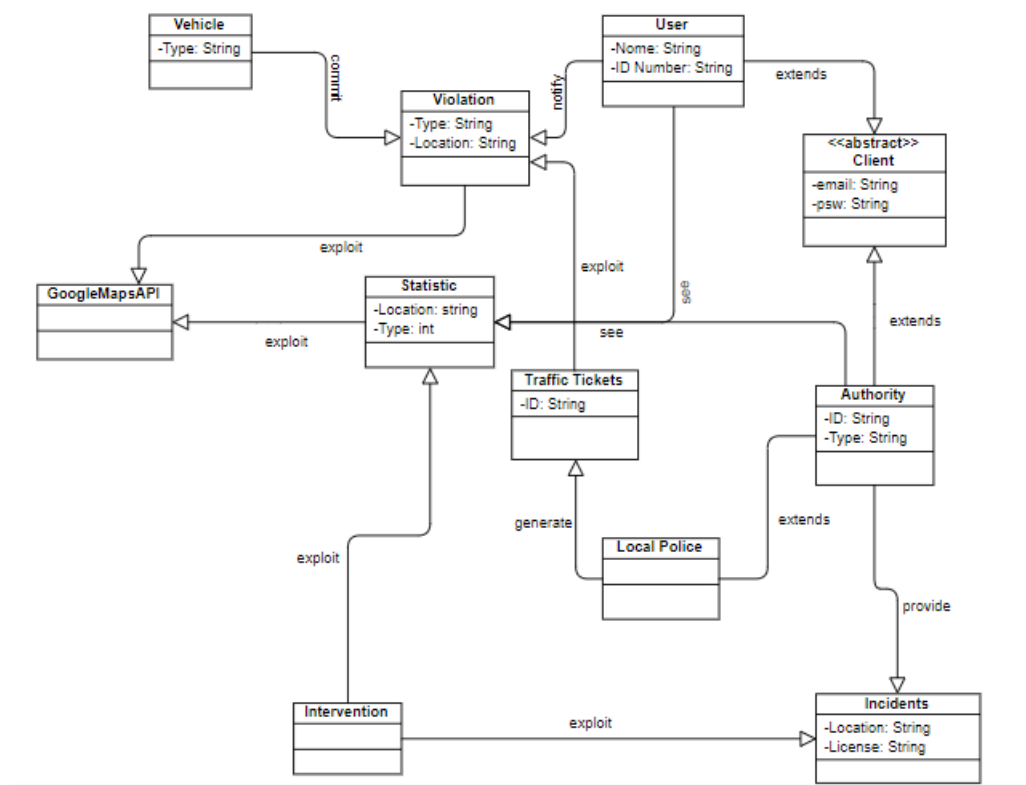


Figure 1: Class diagram

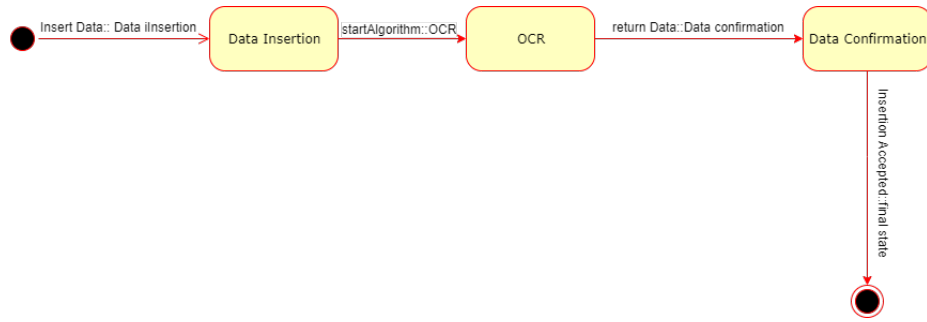


Figure 2: Data insertion

The first state diagram represents the insertion by a logged user of a traffic violation in our system. As mentioned before we need to start an external algorithm (OCR) to read the license plate from a picture, the license plate is showed to the user that have to confirm or modify it (data confirmation) then we can reach our final state and the data are correctly stored.

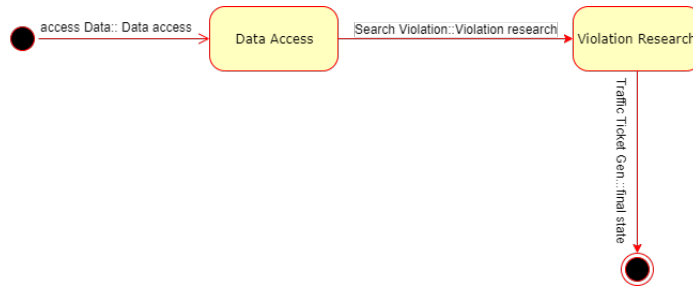


Figure 3: Traffic ticket generation

The second state diagram represents the generation of a traffic ticket by an authority user (type Local Police). This state diagram is a consequence of the previous, in fact, the local police exploit the data stored in SafeStreets to issue traffic tickets.

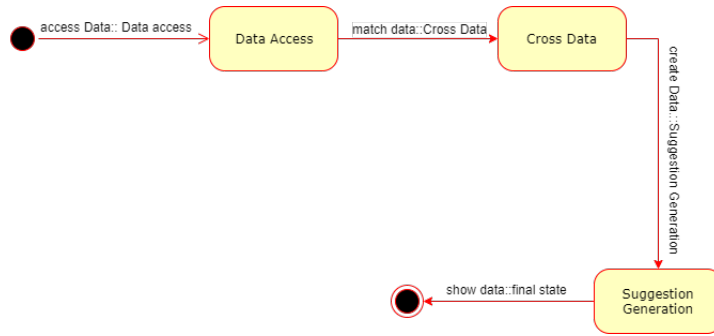


Figure 4: Suggestion Generation

After generating the traffic ticket the data is not discarded but remains stored in order to provides statistics about it. The third state diagram represents the generation of suggestions for possible interventions in unsafe areas/streets by means of SafeStreets, suggestions then showed in the Authority interface. In the cross data state, obviously, in case of unmatching data nothing is done and showed to the Users.

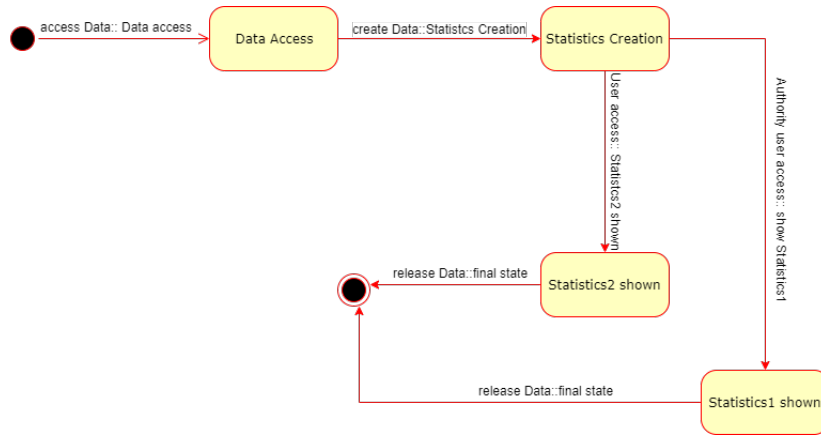


Figure 5: Statistics Generation

The last state diagram represents the show of statistics about traffic violations, is the more complex due to the different visibility that we must offer to users depending on whether they are authorities users or just users.

2.2 Product Functions

- **Traffic violation report:** this is the main function and even the basic service provided by our S2B. The system allow a user to sign up and entering his credentials, the system requires also the ID number to prevent multiple and fake sign up. A logged user after the sign in is able to notify the authorities about a traffic violation by inserting the mandatory information related to it like: location (exploit GPS and Google MAP APIs as mentioned before [2.1]), type of the violation, date and time and a picture of the violation (with the license plate easily visible in the picture), after that the system retrieve the data and runs an external algorithm (OCR) to read the license plate from the picture, then show in the user interface what was read and the user can confirm or can modify it
- **Create User traffic violation statistics:** Using the data provided by logged users the system is able to create statistics that will show to the user providing different type of visibility depending on the type of the user.
- **Create Authority traffic violation statistics:** Similar to the previous function, the difference between the two functions is the type of statistics show in the statistics tab in our application.
- **Suggest Interventions:** The system is able to cross the information about incidents in areas that belong to a municipality and traffic violations in order to provide possible interventions in unsafe areas or streets. These suggestion can be filtered using filter setted by the authority user in order to display only the suggestion about a specific municipality.
- **Traffic Ticket Generation:** The Local police, using the authenticated authority user is able to exploit the data stored regarding traffic violations in our S2B to generate traffic tickets, after generating them they will notify the system about the traffic violation to which a traffic tickets has been generated with a flag. The traffic violation will remain in our system in order to grant future statistics query.

2.3 User characteristics

User can use the system to report violations of the traffic laws when these occur, the conditions that must be guaranteed from the user to use the application are:

- He must have a smartphone and be able to use it
- The smartphone muste have GPS, Internet connectivity and a camera

2.4 Domain assumption, dependencies and constraints

We assume that these assumptions hold true in the domain of our system

- **D.A.1** the positions retrived by the GPS of the user's phone is accurate (5m of tollerance)
- **D.A.2** The information about a reported traffic violation inserted by an user are correctly encoded in the system

- **D.A.3** SafeStreets contains all the possible type of a traffic violations
- **D.A.4** The algorithm used to recognize the license plate from a picture is an external library with relative APIs
- **D.A.5** The traffic violation provided by the user are real infringements of the traffic laws
- **D.A.6** A policeman search for the traffic violation that belong to his municipality before generating a traffic ticket, in fact, it will generate a traffic ticket only if the traffic violation belongs to his jurisdiction ignoring the others.
- **D.A.7** After generating a traffic ticket the traffic violation is not discarded by our system in order to provide statistics about traffic violations
- **D.A.8** The authority users are provided by the developer in order to ensure the identity of the authority.
- **D.A.9** The picture inserted by the user must contain the license plate of the vehicle
- **D.A.10** The picture provided by a user during the notify service shows the infringement or infringements committed by the vehicle
- **D.A.11** The maps provided by the Google Maps APIs are always up to date
- **D.A.12** The position of the users is always available when it interfaces with SafeStreets
- **D.A.13** The authorities verifies the new users in at most 2 days
- **D.A.14** The possible suggestions provided by our S2B must be feasible

3 Specific Requirements

3.1 External interface requirements

3.1.1 User Interfaces

3.2 Functional requirements

3.2.1 Requirements

G1: Allow the users to send data about traffic violations

- R1: The system must allow the user registration
- R2: The system must save the violations reported by the registered users
- R3: The system must be able to identify each user univocally
- R4: The system must encode correctly the traffic violation information
- R5: The system must be able to run an OCR algorithm on the images sent by the user
- R6: The system must allow an user to confirm the data read by the OCR

G2: Allow the authorities to access to violation data

- R2: The system must save the violations reported by the registered users
- R7: The system must be able to recognize the user role
- R8: The system must provide the authorities only the violations under his jurisdiction

G3: Allow the authorities to see specific kind of statistics about traffic violations

- R2: The system must save the violations reported by the registered users
- R7: The system must be able to recognize the user role
- R9: The system must be able to cross the data stored in the system in order to create statistics for the authorities
- R10: The system has to allow the authorities to retrieve statistics filtered by location
- R11: The system has to allow the authorities to retrieve statistics filtered by type

G4: Allow users to see specific kind of statistics about traffic violations

- R2: The system must save the violations reported by the registered users
- R7: The system must be able to recognize the user role
- R12: The system must be able to cross the data stored in the system in order to create statistics for the users

- R13: The system has to allow the users to retrieve statistics filtered by type of the vehicles
- R14: The system must show a graph generated with the statistic data

G5: Allow the local police to use SafeStreets system in order to issue traffic tickets

- R2: The system must save the violations reported by the registered users
- R7: The system must be able to recognize the user role
- R15: The system must ensure the integrity of the data provided by the users
- R16: The system must allow the authority to provide information about the issued ticket (multa fatta un flag)

G6: Allow to exploit the traffic ticket statistics to check the effectiveness of SafeStreets

- R16: The system must allow the authority to provide information to the system about the issued ticket (multa fatta flag)
- R17: Allow the system to access the traffic ticket data stored by the municipality

G7: Allow to cross information between the data of incidents in the area that belong to a municipality and data stored in the system to suggest possible interventions to municipalities

- R2: The system must save the violations reported by the registered users
- R18: Allow the system to access the incidents data stored by the municipality

3.3 Non-functional requirements

3.3.1 Performance

The system must be able to handle tens of thousands of registered users. The requests per minute are expected to be relatively low compared to the number of registered users but some computation intensive operations (e.g. OCR recognition) must be managed in reasonable time.

3.3.2 Reliability

The stored data are critical. The system must have an effective backup strategy that guarantees the integrity of the data. The system should be able to operate 24/7 but the uptime is not critical.

3.4 Security

Sensitive data is stored by the system. The credentials must be stored hashed and salted, the code must be checked to be immune to known vulnerabilities and the system storage must be encrypted.

4 Use Cases

4.1 Scenarios

4.1.1 Scenario 1

Elisa is walking along via Montenapoleone and sees a vehicle in double parking, she is already registered to SafeStreets, therefore decides to open the app, take a clear picture of the traffic violation and provides the mandatory data for reporting it.

4.1.2 Scenario 2

Agnese is not already registered to our S2B even though she has already heard about a new application called SafeStreets that allows to report traffic violations so she downloads the app by means of her device, fill the registration form and take a picture of her ID document.

4.1.3 Scenario 3

Alessio is already registered and by opening the application he found the statistics button in the application, sets the filter and request for some specific statistic.

4.1.4 Scenario 4

The municipality of Milan hasn't reached the monthly amount of traffic tickets regarding traffic violations, therefore it is up to the policewoman Benedetta to deal with this problem. She decides to use the data stored in SafeStreets to generate the last one 100 traffic tickets needed to reach those amounts, Benedetta so sign in with the user of the Police department of Milan and starts her hard work.

4.1.5 Scenario 5

Giuseppe works for the transport ministry and he is already registered and by opening the application he found the statistics button in the application, sets the filter and request for some specific statistic.

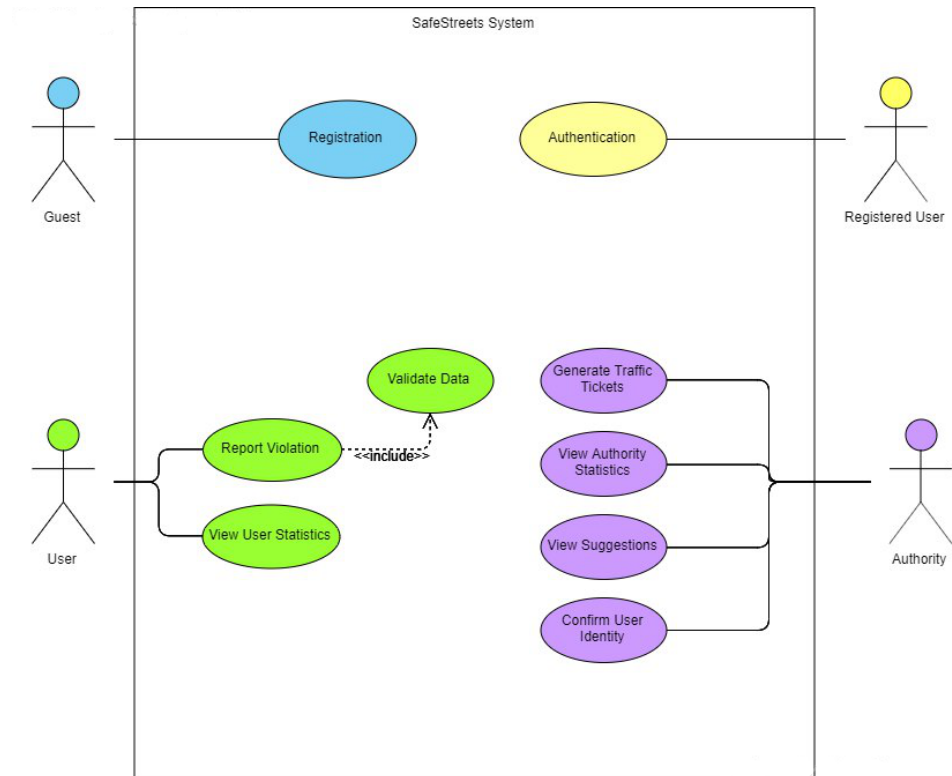
4.1.6 Scenario 6

Alessia works for the municipality of Milan, access to the application using the authenticated authority user and found 200 new users that are waiting for the validation of the ID document, so She starts her work.

4.1.7 Scenario 7

Francesco is designed to check the possible suggestions provided by SafeStreets, hence open the app, click in specific tab and start his work.

4.2 Use case diagram



4.2.1 Registration

Actors	<ul style="list-style-type: none">• Guest
Entry conditions	<ul style="list-style-type: none">• The user has downloaded the app
Flow of events	<ul style="list-style-type: none">• The guest opens the application• The guest presses the sign up button• The guest fills the first name text form• The guest fills the last name text form• The guest provide a valid email address by filling the text form• The guest provide a valid phone number by filling the text form• The guest fills the password text form• The system check if the password respect the requirements and if it is strong enough• The guest fills the Fiscal Code text form• The guest attach a photo or scan of a valid document• The guest presses the submit button• The system stores the registration data• The system starts the verification for the data provided by the user• The guest waits for the verification
Exit conditions	<ul style="list-style-type: none">• The new user has successfully sent the registration form providing correct data
Exceptions	<ul style="list-style-type: none">• The Fiscal code is already present in the system• The email already belong to a registered user• The Password doesn't respect the requirements• The Fiscal code isn't correct

4.2.2 Login

Actors	<ul style="list-style-type: none">• User• Authority
Entry conditions	<ul style="list-style-type: none">• The user has downloaded the app• The user has already registered to SafeStreets
Flow of events	<ul style="list-style-type: none">• The user opens the application• The user inserts the email and the password• The user presses the sign in button• The system checks if the credentials are correct• The submission is launched
Exit conditions	<ul style="list-style-type: none">• The user logged in successfully
Exceptions	<ul style="list-style-type: none">• The credentials are wrong• The email isn't registered

4.2.3 Report violation

Actors	<ul style="list-style-type: none">• Logged User
Entry conditions	<ul style="list-style-type: none">• The user is logged in• The user have found a traffic violation• The user is close to the traffic violation• The user is able to take a picture with its device
Flow of events	<ul style="list-style-type: none">• The user opens the application• The user clicks the specific button to take a picture of the traffic violation• The user attaches the picture taken• The user clicks the specific button to provide the location of the traffic violation• The user clicks the Date button and provide the actual date and time<ul style="list-style-type: none">• The user select the type(s) of the traffic violation by choosing it in a list• The user insert the license plate of the vehicle into the text form• The user clicks the report button• The system receives the violation data• The system uses the OCR service to read the license plate<ul style="list-style-type: none">• The system checks if the license plate from the OCR matches the license filled by the user
Exit conditions	<ul style="list-style-type: none">• The user has successfully submitted a violation
Exceptions	<ul style="list-style-type: none">• The license plate doesn't match

4.2.4 User Statistics request

Actors	<ul style="list-style-type: none">• Logged User
Entry Conditions	<ul style="list-style-type: none">• The user is logged in
Flow of events	<ul style="list-style-type: none">• The user opens the statistics tab by clicking the specific button• The user sets the search filters• The user clicks the submit button• The system receives the filters• The system queries the database for the data• The system replies with the matching statistic data
Exit conditions	<ul style="list-style-type: none">• The user sees the searched statistics
Exceptions	<ul style="list-style-type: none">• The service is not available at the moment of the query

4.2.5 User information edit

Actors	<ul style="list-style-type: none">• Logged User
Entry Conditions	<ul style="list-style-type: none">• The user is logged in
Flow of events	<ul style="list-style-type: none">• The user opens the application• The user opens settings tab• The user sets the new information related to his account• The user pressess the submit button• The system receives the new information• The system send a confirmation email• The user click on the link received in the email• The system update the information related to the user
Exit conditions	<ul style="list-style-type: none">• The user has updated his information
Exceptions	<ul style="list-style-type: none">• The system is not available at the moment

4.2.6 Generate traffic ticket

Actors	<ul style="list-style-type: none">• Authority
Entry Conditions	<ul style="list-style-type: none">• The authority is logged in
Flow of events	<ul style="list-style-type: none">• The authority opens the violation data tab in SafeStreets• The authority sets the search filters• The authority clicks the submit button• The system receives the filters• The system queries the db for the data requested• The system replies with the matching traffic violation data• The authority use the traffic violation data for generating traffic tickets
Exit conditions	<ul style="list-style-type: none">• The authority stops generating traffic ticket
Exceptions	<ul style="list-style-type: none">• The service is not available at the moment of the query• There are no data related traffic violations using the filters setted by the authority• There is no evidence of traffic violation in the stored picture• The data related to a traffic violation have been manipulated by the user and must be discarded

4.2.7 View Suggestions

Actors	<ul style="list-style-type: none">• Logged Authority
Entry Conditions	<ul style="list-style-type: none">• The authority is logged in
Flow of events	<ul style="list-style-type: none">• The authority opens the application• The authority clicks the specific button for suggestions• The authority sets the search filters• The system receives the filters• The system queries the db for the data• The system replies with the matching suggestions
Exit conditions	<ul style="list-style-type: none">• The authority sees the searched suggestions
Exceptions	<ul style="list-style-type: none">• The service is not available at the moment of the query

4.2.8 Authority Statistics request

Actors	<ul style="list-style-type: none">• Logged Authority
Entry Conditions	<ul style="list-style-type: none">• The authority is logged in
Flow of events	<ul style="list-style-type: none">• The authority opens the statistics tab by clicking the specific button• The authority sets the search filters• The authority clicks the submit button• The system receives the filters• The system queries the database for the data• The system replies with the matching statistic data
Exit conditions	<ul style="list-style-type: none">• The authority sees the searched statistics
Exceptions	<ul style="list-style-type: none">• The authority is not available at the moment of the query

4.2.9 Verification

Actors	<ul style="list-style-type: none">• Authority
Entry Conditions	<ul style="list-style-type: none">• The authority has downloaded the app
Flow of events	<ul style="list-style-type: none">• The authority opens the application• The authority presses the sign in button• The authority presses the submit button• The authority opens the registration data• The authority checks the new user data• The authority confirms the registration
Exit conditions	<ul style="list-style-type: none">• The new user is successfully identified
Exceptions	<ul style="list-style-type: none">• The authority can't verify the user

5 Formal Analysis Using Alloy

```
open util/boolean

abstract sig User{}

sig LoggedUser extends User{
    //personal information
    //other parameters
    banned: one Bool
}

sig LoggedAuthorityUser extends User{
    LocalPolice: one Bool
}

sig TrafficViolation{
    //A traffic violation can be reported by one user
    ReportedBy: one LoggedUser,
    //The violation must contain only one license plate
    CommittedBy: one LicensePlate,
    //Type of the violation, can be only one for each report
    Type: one TypeViolation,
    Date: one DateAndTime
}

abstract sig DateAndTime {
    Date: one String
}

sig LicensePlate{
    Plate: one String}

//Various violation types, currently not used
abstract sig TypeViolation{}
one sig DoubleParking extends TypeViolation {}
one sig TicketExpired extends TypeViolation {}
one sig HandicapParking extends TypeViolation {}
one sig OutsideTheSpace extends TypeViolation {}
one sig IllegalParking extends TypeViolation {}

//Traffic Ticket issued from a Traffic Violation
sig TrafficTicket{
    Source: one TrafficViolation,
    GeneratedBy: one LoggedAuthorityUser, //The logged authority user
    //must be the Police
    GeneratedFlag: one Bool
}

// If multiple violations must not have the same attributes, multiple users
// cannot report the same violation
fact DifferentLoggedUserDontReportTheSameTrafficViolation{
    all disjoint t1,t2: TrafficViolation | (t1.ReportedBy ≠ t2.ReportedBy
    )
    implies (t1.CommittedBy ≠ t2.CommittedBy or t1.Date ≠ t2.Date or t1.
    Type ≠ t2.Type)
}

// All the traffic violations reported by a banned user must be removed
fact BannedUserCannotDealWithTrafficViolation{
    no u: LoggedUser |
        no t: TrafficViolation |
            u.banned = True and t.ReportedBy = u
}

// The user must report multiple times if the vehicle is breaking more than
// one traffic law
fact MultipleTrafficViolationAreReportedJustOnce{
```

```

        no u:LicensePlate |
            all disjoint t1,t2:TrafficViolation |
                (t1.CommittedBy = u and t2.CommittedBy = u)
        implies (t1.Type ≠ t2.Type or t1.Date ≠ t2.Date)
    }

    // Authorities that aren't part of the Local Police can't generate tickets
    fact OnlyLocalPoliceCanGenerateTrafficTicket{
        no a: LoggedAuthorityUser |
            some t: TrafficTicket |
                a.LocalPolice = False and t.GeneratedBy = a
    }

    // Multiple tickets cannot be issued for the same Violation
    fact AtMostOneTrafficTicketForTrafficViolation{
        no disjoint t1, t2 : TrafficTicket |
            t1.Source = t2.Source and (t1.Source ≠ none or t2.Source ≠
            ↪ none)
    }

    assert noTicketsByNonPolice {
        no tt : TrafficTicket |
            tt.GeneratedBy.LocalPolice = False
    }
    check noTicketsByNonPolice

    assert noMultipleTicketsForViolation {
        all disjoint s1,s2:TrafficTicket |
            s1.Source & s2.Source = none
    }
    check noMultipleTicketsForViolation

    assert noTicketForViolationByBanned{
        no tt : TrafficTicket |
            tt.Source.ReportedBy.banned = True
    }
    check noTicketForViolationByBanned

    pred show{}

    run show for 10

```