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

Frederik Saraci, Lorenzo Amata

POLITECNICO
MILANO 1863

Contents

1 Introduction

1.1 Purpose of the document

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 constrains 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 etc.) 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 guest have 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 lines 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. As we all know, a single vehicle is able to commit multiple traffic violation at time, in order to make the system simpler: traffic violations committed by a vehicle will be individually reported using one or more pictures (if needed) for each of them and the type of the traffic violation, avoiding multiple type commit of traffic violations for a single vehicle. 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 even provided by the user during the report, especially if the environment condition 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 with 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. Logged Users will be able to see a pie chart that show the type of vehicle that commit most violations and the type of violations for a specific street, while the Logged Authority Users will see highlighted streets with highest frequency of 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 particular a single traffic ticket for each traffic violation committed by the vehi-

cle. In this case, the authenticity of the stored data must be guaranteed before issuing a traffic ticket, indeed the Local policeman is in charge to check it and eventually notify that the data have been manipulated by the user, and in this case the data must be discarded and the user is banned. After generating a traffic ticket the local policeman (Logged Authority User) notifies that the traffic ticket has been generated by using a flag. 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

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
- *Logged Authority User or Authenticated authority or Logged 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
- ID: ID document number

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

1.8 Revisions

- **V 1.0** 10th November 2019
- **V 1.1** 12th November 2019: **Chapter 1.3:** More accurate explanation of avoiding report of multiple traffic violations committed by a vehicle. More details on who check the feasibility of suggestions provided by SafeStreets More details on when a Logged User is banned More details on which kind of statistics are showed to the Logged User and Logged Authority Users More details on the fact that a single traffic violation is generated for each traffic violation (even in case of multiple report for a single vehicle) **More details on chapter 2.1:** Authority check the authenticity - correction Validation Data by authority - added State diagrams correction and details added
- **V 1.2** 30th November 2019 : removed advanced function 1, added D.A.12 and D.A.13, class diagram update, D.A.11 moved as R18.

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. 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 but it is up to the Local Police to check the authenticity of a report. The authority users is even in charge of **validating the ID document number** and hence identity of a guest user that wants to register to our S2B. 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.

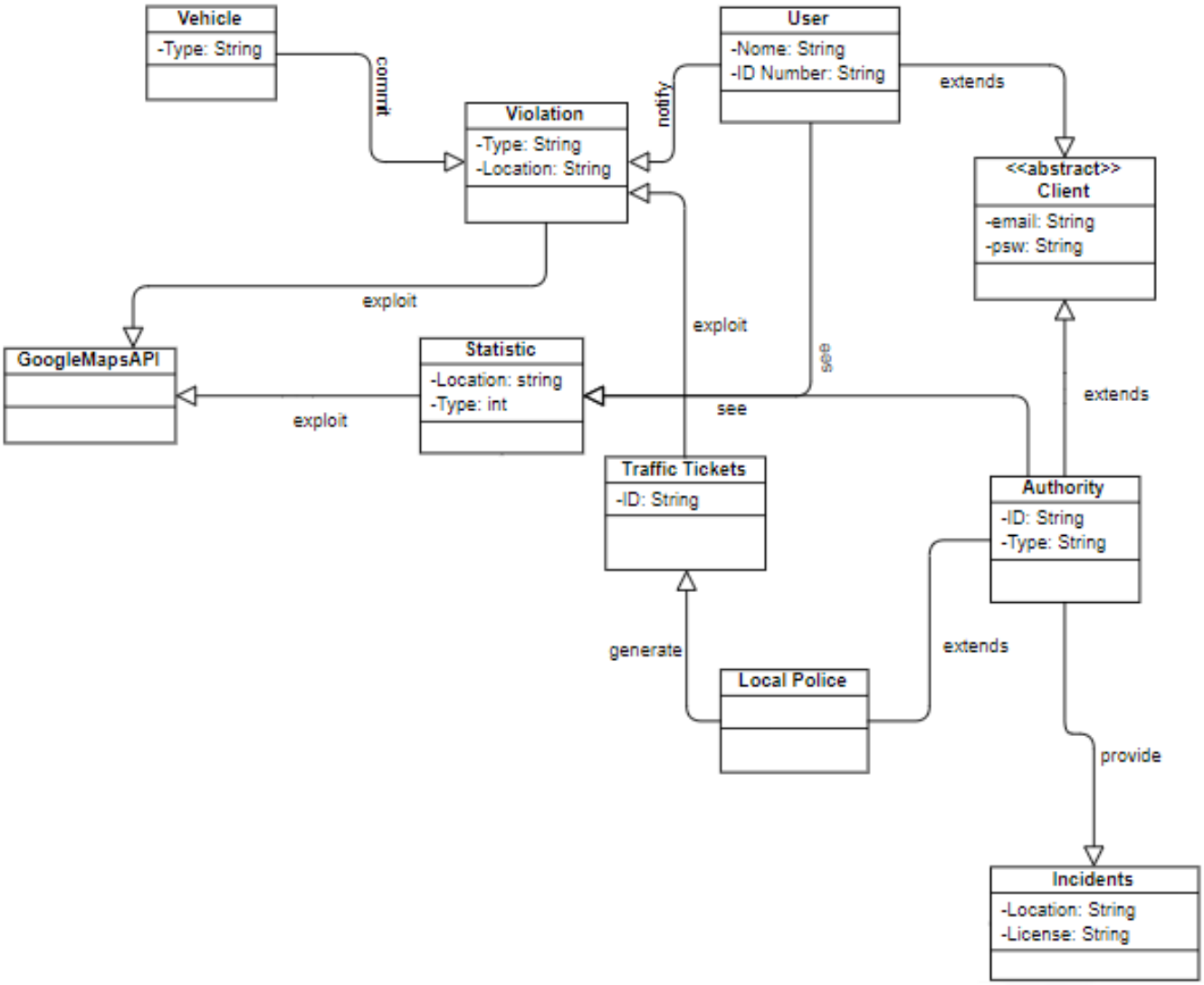


Figure 1: Class diagram

2.1.1 State diagrams

Some critical aspects are analyzed in the following state diagrams:

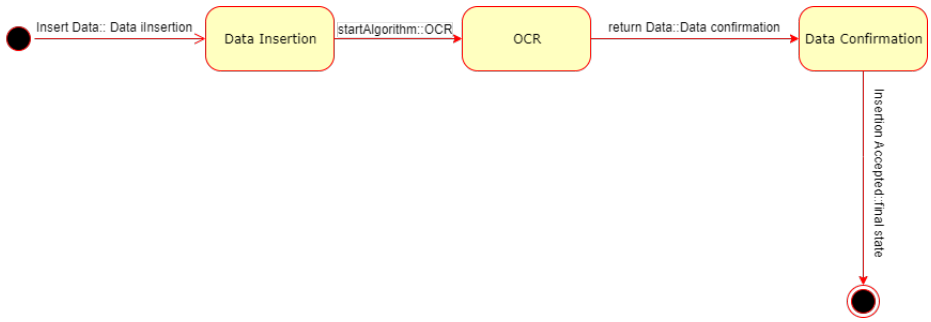


Figure 2: Data insertion

The insertion by a logged user of a traffic violation in our system. Firstly the user provide the mandatory information,then starts an external algorithm (OCR) to read the license plate from a picture, the license plate is compared with the one provided by the user during the report (data confirmation) then we can reach our final state even in case it doesn't match and the data are correctly stored. It is important to explain that the data are stored even though the license plate doesn't match because at the end of the "authenticity process" we will also have an human control given by the local policeman that generates the traffic ticket related to the traffic violation.

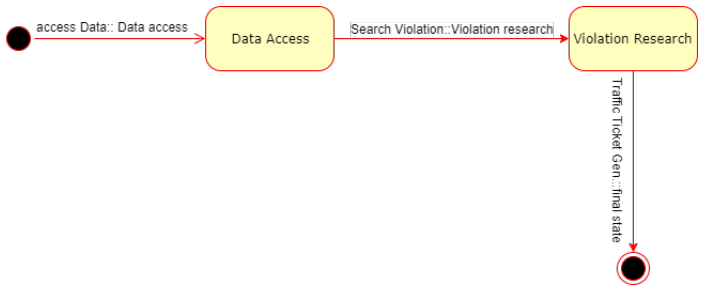


Figure 3: Traffic ticket generation

The generation of a traffic ticket by a logged authority user (type Local Police). The issuing of traffic tickets is a consequence of the basic function of our S2B (traffic violation report), in fact, the local police exploit the data stored in SafeStreets, search a violation that belongs to its authority and then issues traffic tickets.

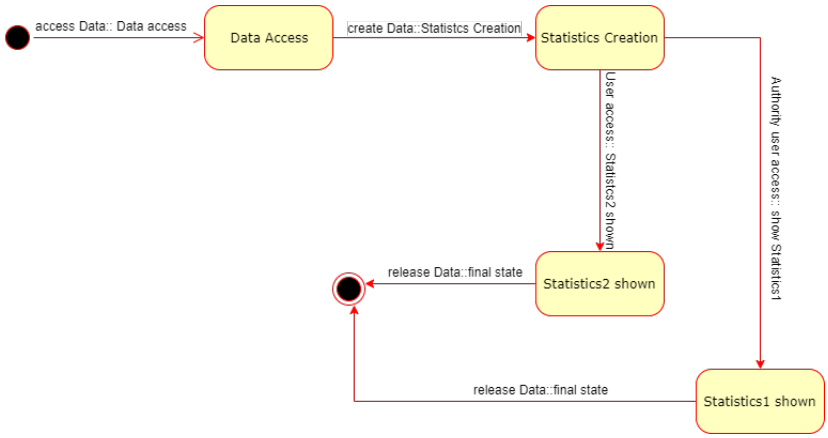


Figure 4: Statistics Generation

The view 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 in the chapter 2.1), type of the violation, date and time,license plate, note and pictures of the violation (with the license plate easily visible in of them),after that the system retrieve the data and runs an external algorithm (OCR) to read the license plate from the picture then compare it with the license plate provided by the user, even in case it doesn't match the report will be stored, the report will be discarded after the local police authenticity check by using a flag.

- **Create User traffic violation statistics:** Using the data provided by logged users the system is able to create statistics (crossing its own data stored) that will show to the user providing different type of visibility depending on the type of the user. The S2b data are stored in a persistent memory using a DBMS, with simple query joining one or more table we can cross the information and provide statistics (e.g.join for type violation or for location)

- **Create Authority traffic violation statistics:** Similar to the previous function, the difference between the two functions is the type of statistics shown in the statistics tab in our application.

- **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. By using the flag the S2B can also check its effectiveness by looking at the amount of traffic ticket generated by means of SafeStreets.

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 OS of the user's device must be Android 6.0, it must 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 retrieved by the GPS of the user's phone is accurate and eventually showed

- **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 related to parking 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** 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.6** The authority users are provided by the developer in order to ensure the identity of the authority.
- **D.A.7** At least one of the pictures inserted by the user must contain the license plate of the vehicle
- **D.A.8** The pictures provided by a user during the notify service shows the infringement or infringements committed by the vehicle
- **D.A.9** The maps provided by the Google Maps APIs are up to date
- **D.A.10** The position of the users is always available when it interfaces with SafeStreets
- **D.A.11** The authorities verifies the new users in at most 2 days
- **D.A.12** A traffic violations is stored at most one time, even if it is reported several times
- **D.A.13** Traffic Tickets are generated only by Local Police using a notebook for traffic violations.

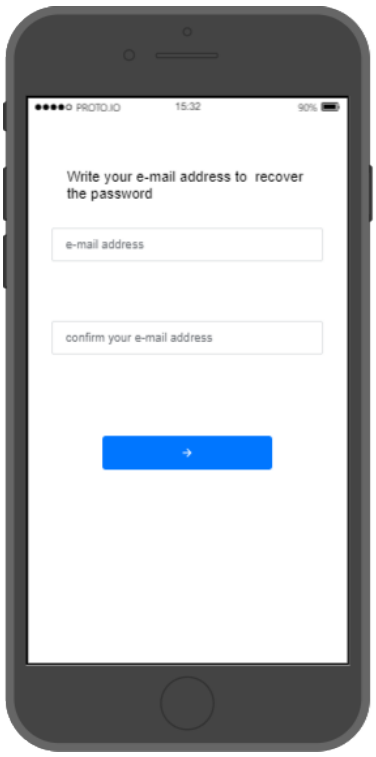
3 Specific Requirements

3.1 External interface requirements

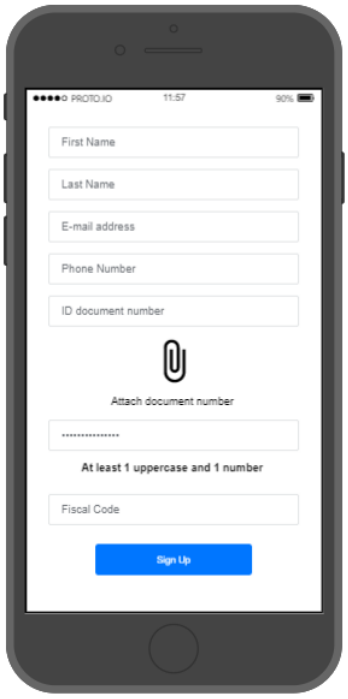
3.1.1 User Interfaces

The figures in this section are user interfaces of our S2B, in the DD chapter 3 we will also provide UX Diagrams in order to understand better the flow of events.

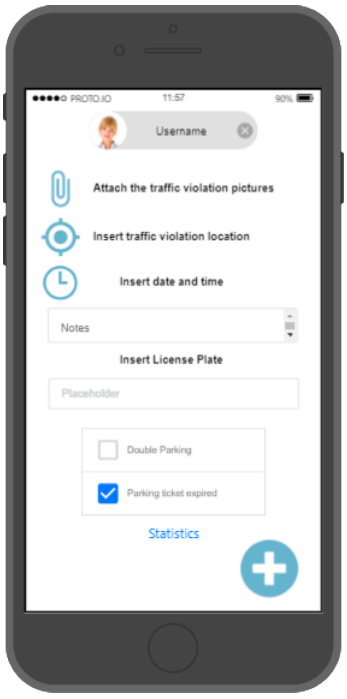
(a) Start Up Interface



(b) Forgot Password

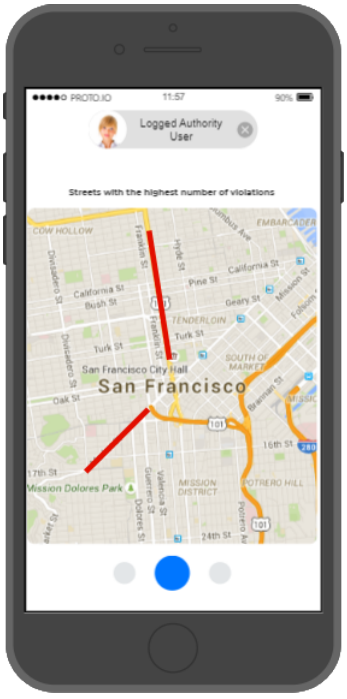


(a) Sign Up form

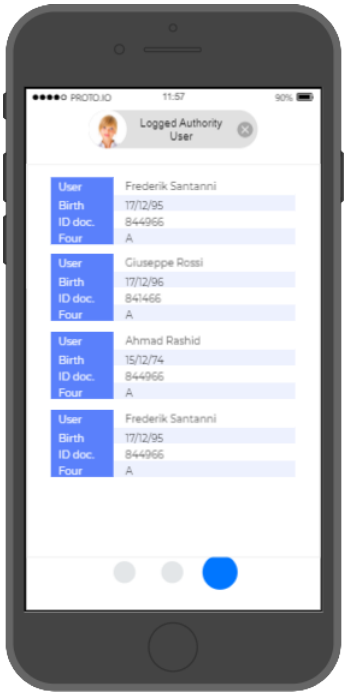


(b) Logged User interface

(a) Logged User Statistic interface

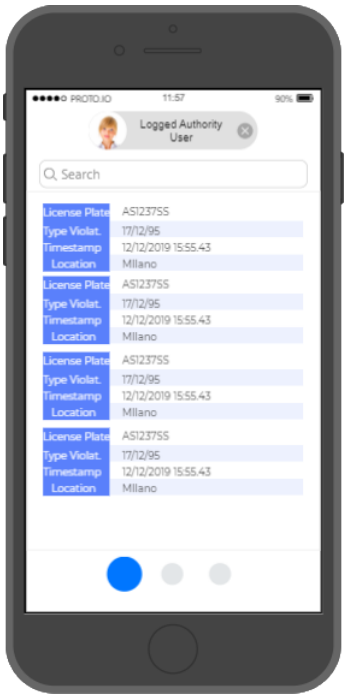


(b) Logged Authority User Statistic interface

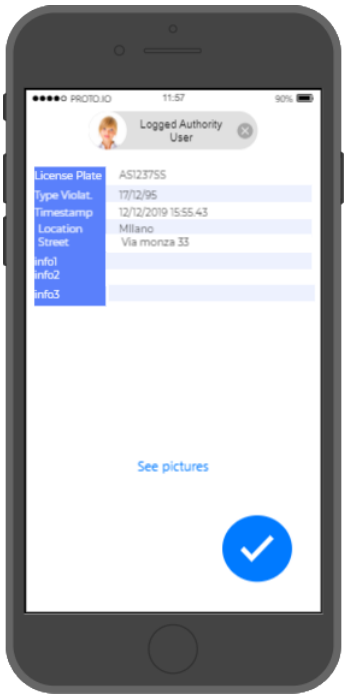


(a) Validate Users interface

(b) Validate Users interface



(a) Traffic Violations interface



(b) Generate Traffic Ticket tab

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 uniquely
- 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
- R18: After generating a traffic ticket the traffic violation is not discarded by our system in order to provide statistics about traffic violations

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
- R18: After generating a traffic ticket the traffic violation is not discarded by our system in order to provide statistics about traffic violations

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

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
- R17:Allow the system to access the traffic ticket 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 know 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.

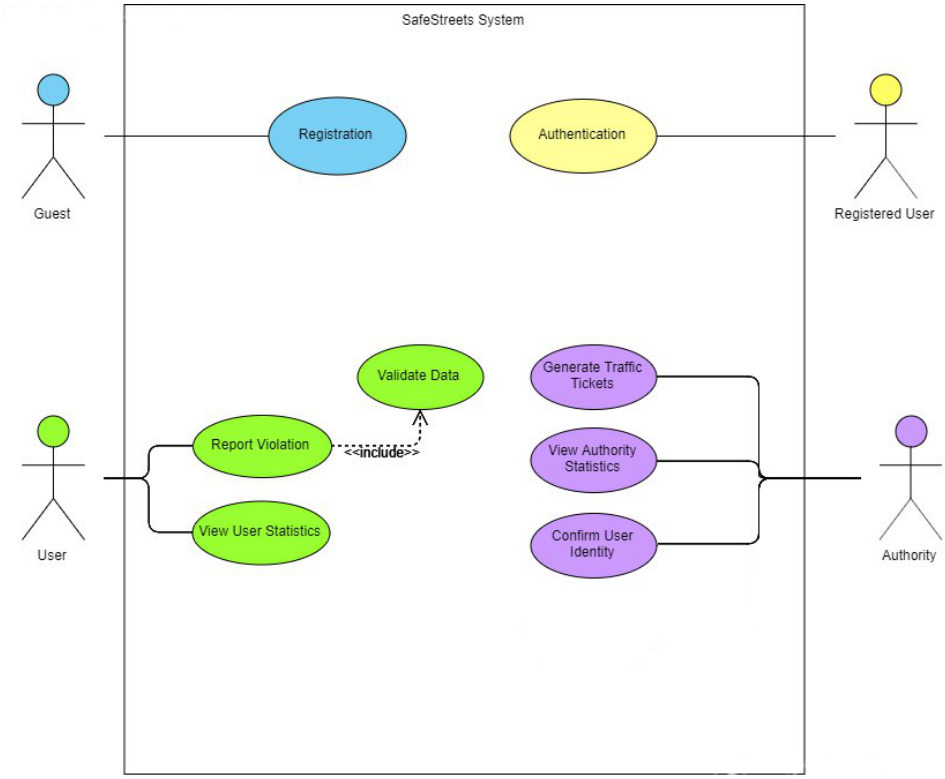
4.1.5 Scenario 5

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.

4.2 Use case diagram



4.2.1 Registration

Actors	• Guest
Entry conditions	• The user has downloaded the app
Flow of events	• 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	• The new user has successfully sent the registration form providing correct data
Exceptions	• 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	• User • Authority
Entry conditions	• The user has downloaded the app • The user has already registered to SafeStreets
Flow of events	• 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	• The user logged in successfully
Exceptions	• The credentials are wrong • The email isn't registered

4.2.3 Report violation

Actors	• Logged User
Entry conditions	• 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	• 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 • 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 • The system checks if the license plate from the OCR matches the license filled by the user
Exit conditions	• The user has successfully submitted a violation
Exceptions	• The license plate doesn't match

4.2.4 User Statistics request

Actors	• Logged User
Entry Conditions	• The user is logged in
Flow of events	• 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	• The user sees the searched statistics
Exceptions	• The service is not available at the moment of the query

4.2.5 User information edit

Actors	• Logged User
Entry Conditions	• The user is logged in
Flow of events	• The user opens the application • The user opens settings tab • The user sets the new information related to his account • The user presses 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	• The user has updated his information
Exceptions	• The system is not available at the moment

4.2.6 Generate traffic ticket

Actors	• Authority
Entry Conditions	• The authority is logged in
Flow of events	• 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	• The authority stops generating traffic ticket
Exceptions	• The service is not available at the moment of the query • There are no data related traffic violations using the filters set 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 Authority Statistics request

Actors	• Logged Authority
Entry Conditions	• The authority is logged in
Flow of events	• 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	• The authority sees the searched statistics
Exceptions	• The authority is not available at the moment of the query

4.2.8 User verification

Actors	• Authority
Entry Conditions	• The authority has downloaded the app
Flow of events	• 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	• The new user is successfully identified
Exceptions	• The authority can't verify the user

4.2.9 Use cases sequence diagrams

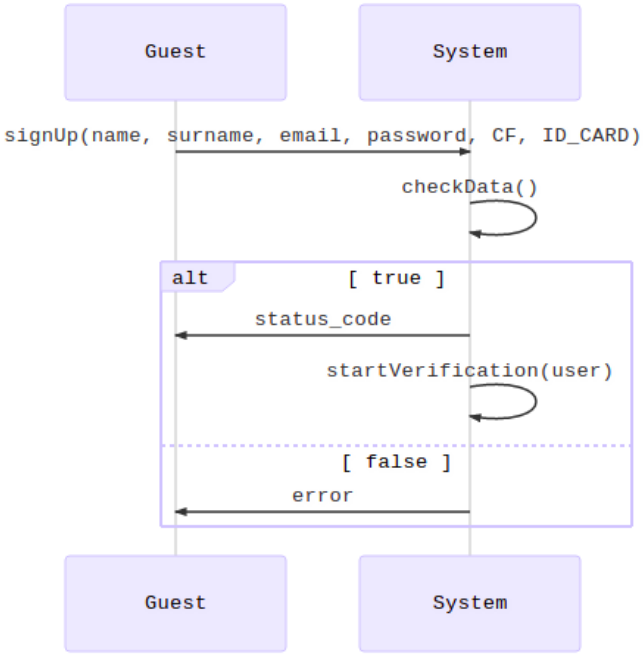


Figure 10: Registration

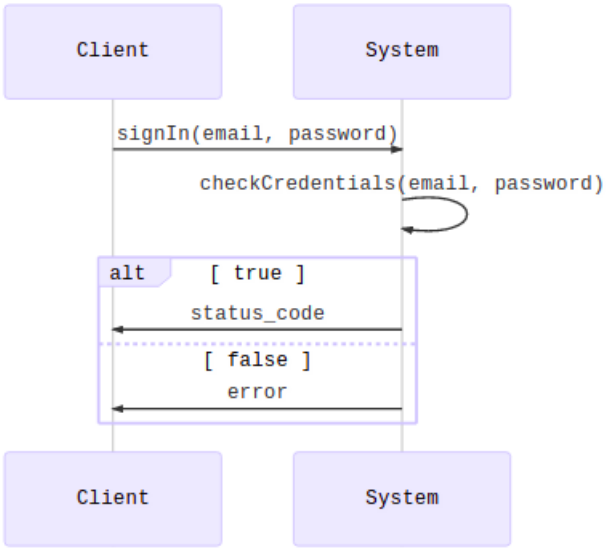


Figure 11: User login

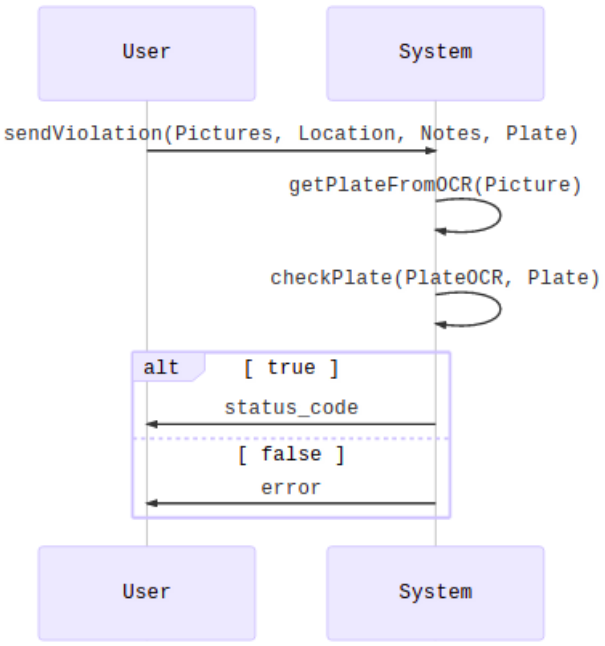


Figure 12: Report violation

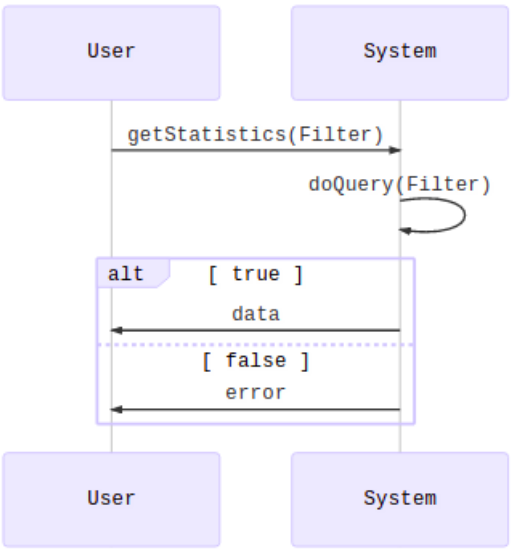


Figure 13: User statistics request

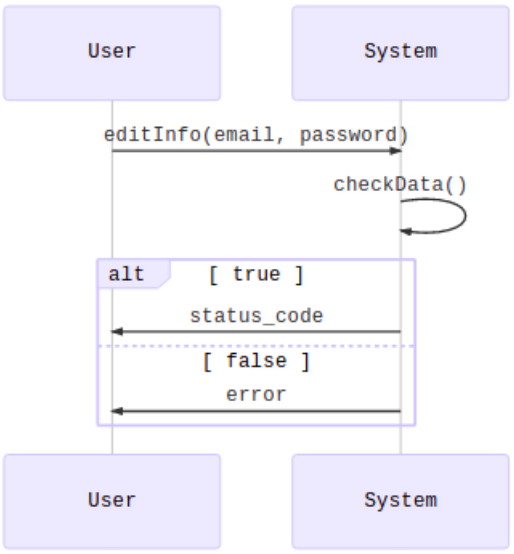


Figure 14: User information edit

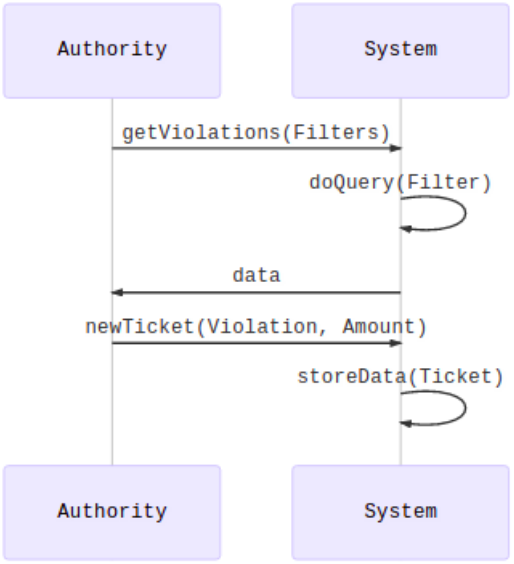


Figure 15: Generate traffic ticket

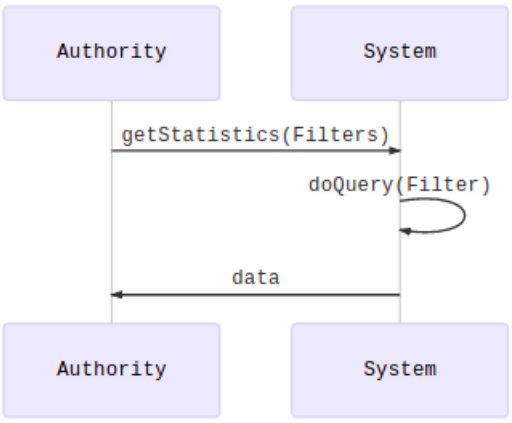


Figure 16: Authority statistics request

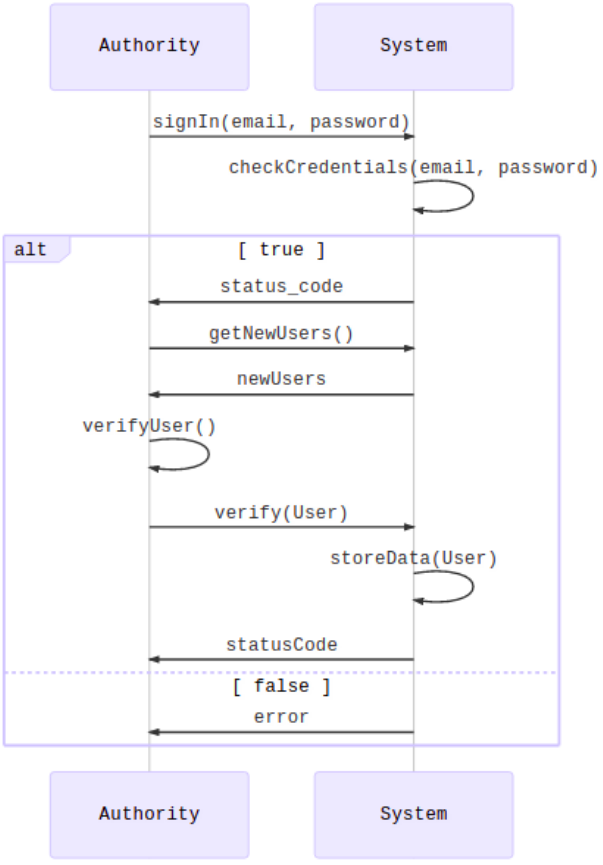


Figure 17: User verification

5 Formal Analysis Using Alloy

5.1 Introduction

In this section is provided a formal analysis of the system.
The main components of SafeStreets are modeled using Alloy and then some critical aspects are verified.
In particular we verify that users that have been banned cannot submit violations, this is critical to our model since no further checks are done by the authority in normal conditions on the status of the user.
We also check that only the authorities flagged as "Police" can generate the ticket for a given violation since authorities accounts will be given to people that don't have the right to legally issue traffic tickets.
The final thing that we check is that our system must be able to discriminate between multiple reports relating to the same physical violation to avoid multiple tickets for the same violation.
In the following pages we provide the Alloy code and the results of the analysis.

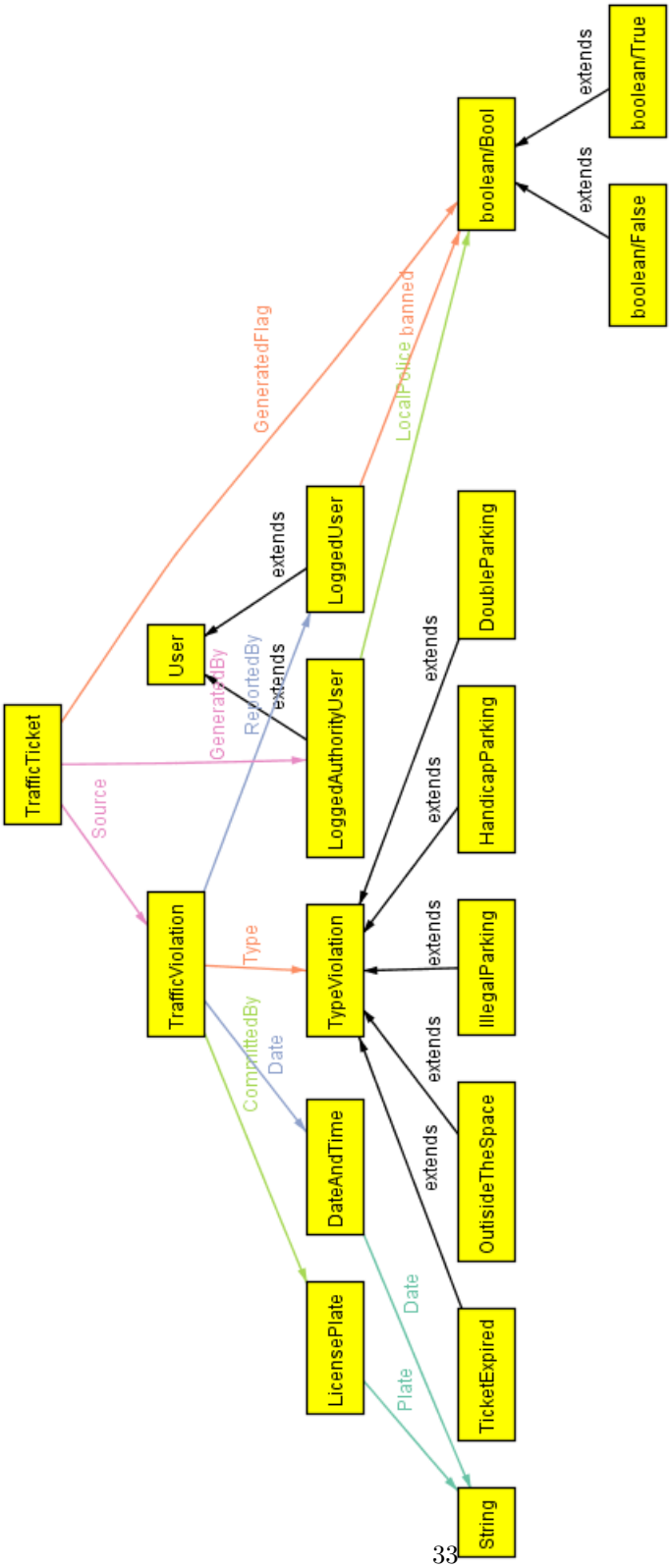
5.2 Alloy source

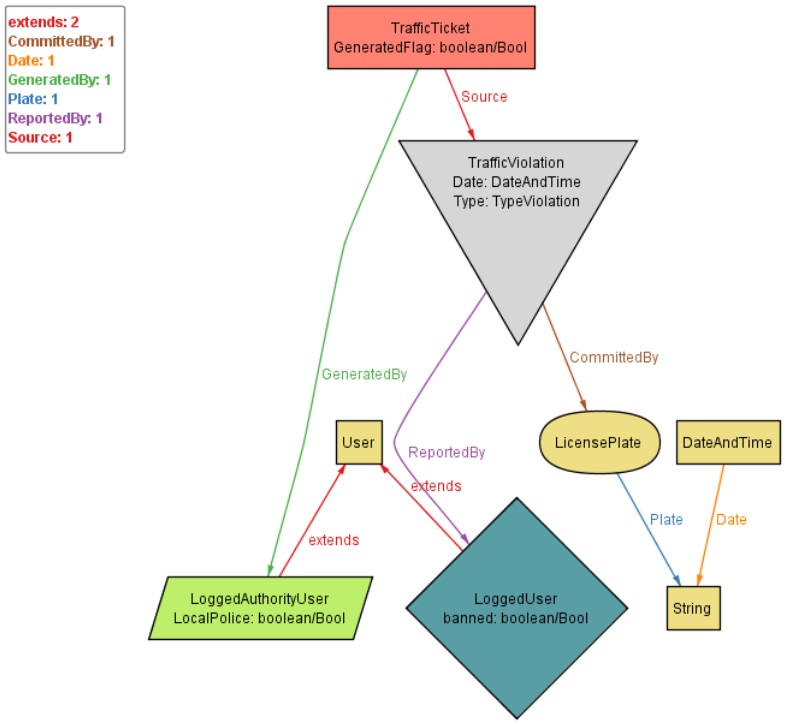
```
open util/boolean

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 OutisideTheSpace 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
```

5.3 Results


TrafficTicket
User
LoggedUser
LoggedAuthorityUser
TrafficViolation
TypeViolation
DateAndTime
LicensePlate
String
boolean/Bool
boolean/True
boolean/False
DoubleParking
HandicapParking
IllegalParking
OutisideTheSpace
TicketExpired
extends
GeneratedFlag
LocalPolice banned
GeneratedBy
ReportedBy
Source
Type
CommittedBy
Date
Plate
extends: 9
banned: 1
CommittedBy: 1
Date: 1
Date: 1
GeneratedBy: 1
GeneratedFlag: 1
LocalPolice: 1
Plate: 1
ReportedBy: 1
Source: 1
Type: 1



Executing "Check noTicketForViolationByBanned"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1668 vars. 126 primary vars. 3026 clauses. 133ms.
No counterexample found. Assertion may be valid. 12ms.

Executing "Check noMultipleTicketsForViolation"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1688 vars. 129 primary vars. 3021 clauses. 27ms.
No counterexample found. Assertion may be valid. 7ms.

Executing "Check noTicketsByNonPolice"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1656 vars. 126 primary vars. 2987 clauses. 18ms.
No counterexample found. Assertion may be valid. 5ms.

Executing "Run show for 10"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
20295 vars. 970 primary vars. 42771 clauses. 95ms.
Instance found. Predicate is consistent. 76ms.

4 commands were executed. The results are:
#1: No counterexample found. noTicketForViolationByBanned may be valid.
#2: No counterexample found. noMultipleTicketsForViolation may be valid.
#3: No counterexample found. noTicketsByNonPolice may be valid.
#4: Instance found. show is consistent.

Figure 18: Alloy analysis result

6 Hours of work

This is the amount of time spent to redact this document:

- Lorenzo Amata: $\sim$ 45 hours
- Frederik Saraci: $\sim$ 42 hours