Software engineering project

SafeStreets

RASD : Requirement Analysis and Specification Document.

Authors:
Mattia Micheloni,
Francesco Montanaro,
Amedeo Pachera.

Professor:
M.Rossi.

Version 1.0
08/11/19

Contents

1 Introduction **1**
1.1 Purpose . . . 1
1.1.1 General purpose . . . 1
1.1.2 Goals . . . 2
1.2 Scope . . . 2
1.3 Definitions, acronyms, abbreviations . . . 4
1.3.1 Definitions . . . 5
1.3.2 Acronyms . . . 5
1.3.3 Abbreviations . . . 5
1.4 Revision history . . . 5
1.5 Reference documents . . . 6
1.6 Document's overview . . . 6

2 Overall description **7**
2.1 Product prospective . . . 7
2.2 Product function . . . 8
2.2.1 Violation reporting system . . . 8
2.2.2 Authority portal . . . 9
2.2.3 Statistics . . . 9
2.3 User characteristics . . . 9
2.4 Constraints . . . 10
2.5 Assumptions and dependencies . . . 10

3 Specific Requirements **17**
3.1 External interface requirements . . . 17
3.1.1 User Interface . . . 17
3.1.2 Hardware interfaces . . . 20
3.1.3 Software interfaces . . . 20
3.2 Functional requirements . . . 20
3.2.1 User . . . 20
3.2.2 Authorities . . . 23

3.2.3 Requirements 28
3.3 Performance requirements 32
3.4 Design constraints 32
3.4.1 Standards compliance 32
3.4.2 Hardware limitations 32
3.4.3 Any other constraint 32
3.5 Software system attributes 32
3.5.1 Reliability 33
3.5.2 Availability 33
3.5.3 Security 33
3.5.4 Maintainability 33
3.5.5 Compatibility 33

4 Formal Analysis Using Alloy 34
4.1 Signatures 34
4.2 Facts 36
4.3 Assertions 38
4.4 Predicates 40

5 Effort Spent 46

Chapter 1

Introduction

This chapter explains the universe where SafeStreets application is located, all the goals, purposes and scopes of the system and gives some keys of interpretation of the document (abbreviations, definitions..).

1.1 Purpose

This Sections focuses on the description of the purpose of the system in the section 1.1.1 and lists all the goals in the section 1.1.2

1.1.1 General purpose

SafeStreets is a crowdsourced application whose purpose is to create a service to support authorities in spotting traffic violations. The aim is to improve the safety and the order of the city. It is thought to give users the chance to report when a violation occurs by sending pictures to the system, which can store information and made them available to authorities. The software's main features are to simplify the reporting process by automatically extracting metadata from the pictures (such as timestamp, geographical position and license plate of the vehicle involved); by exploiting the available data, the system can provide both the users and authorities statistics concerning areas with the highest frequency of violations or the vehicles which commit the most ones. Moreover, by crossing the information of the accidents that occur on the territory with the stored ones, the system can highlight potentially unsafe areas and suggest possible interventions. Finally, the software has to ensure the consistency and reliability of the information in order to let the authorities to generate traffic tickets.
The figure 1.1 shows the Logo of SafeStreets.

Figure 1.1: Logo SafeStreets

1.1.2 Goals

- G_1 : Users can report traffic violations to authorities by sending pictures to the system.
- G_2 : The System has to make easier the process of reporting violations by extracting metadata from the pictures.
- G_3 : Authorities can access sensitive data for specific pourposes, such as traffic tickets generation, so their consistency and reliability must be granted.
- G_4 : Both the users and authorities can access to statistics (based on accidents and violations) built by the system.
- G_5 : System can suggest intervention to authorities, based on shared data obtained by crossed information between police and SafeStreet.

1.2 Scope

SafeStreets is a software offered both to common users and authorities. Users do not need to be registered on the application to access its services. Anyone who wants to report a traffic violation can do so by uploading at most 6 pictures to the system; at least one them must contain the license plate of the vehicle(s) involved. In order to guarantee the reliability of the service the pictures can be only taken by using the application itself; in this way the reporting data could not be modified or altered. The system will simplify the reporting process by adding date, time and the geographical position of

the user's device; moreover, by launching an algorithm, it will automatically extract the license plate directly from the pictures themselves. Before sending the report to the system, users can specify the type of violation by choosing it from a predefined set. If the procedure ends without errors, the system accepts the report and stores it, waiting to be reviewed and validated by authorities, otherwise it is refused. Recognized authorities must log into the application, using their own credentials, in order to access the information concerning the reports made by the users. They can therefore proceed with the validation of the appropriate reports and eventually modify their description (if considered incorrect) or the license plate number (if wrongly extracted). The system will then delete the invalid reports. Thanks to the reliability of the data and the information about violations, authorities can even decide to generate traffic tickets. Finally, they can upload information about the accidents that occurs on the territory of the municipality. The software provides also a service to offer both the users and authorities statistics computed on the available data. Obviously, in order make the statistics consistent and reliable, the data used to compute them are the only ones which have been validated by authorities. In particular: It can highlight the streets or the areas with the highest frequency of violations, such as the vehicles that commit the most ones or the trend of the infractions that occur over time. By crossing the information of accidents on the territory with its own data, the system can identify potentially unsafe areas and suggest possible interventions in order to improve its safety. The application domain concerning the set of events recognized by the system is listed below and represented by the "world and machine phenomena" criterion (set visible in figure 1.2):

World phenomena

- W_1 : The user spots a violation.
- W_2 : The authorities generate a ticket.

Shared phenomena

- S_1 : The user takes pictures of the violation.
- S_2 : The user inputs the violation type.
- S_3 : Authorities log in to the system.
- S_4 : Authorities access to the reports.
- S_5 : Authorities edit the reports.

- S_6 : Authorities validate the reports.
- S_7 : Authorities upload accidents information into the system.
- S_8 : The system shows the statistics.
- S_9 : The system suggests possible interventions.

Machine phenomena

- M_1 : The system extracts metadata from the pictures.
- M_2 : the system accepts the reports.
- M_3 : the system refuses the reports.
- M_4 : The system stores the reports.
- M_5 : The system deletes the invalid reports.
- M_6 : The system compute statistics.

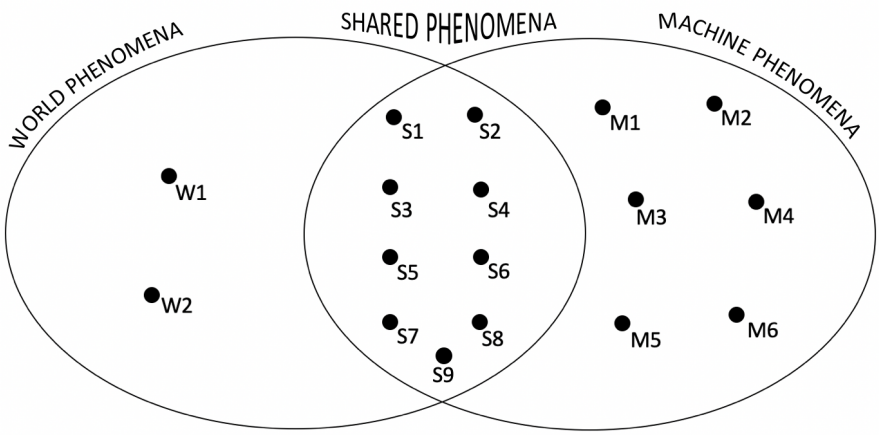


Figure 1.2: Sets of Phenomena

1.3 Definitions, acronyms, abbreviations

This section gives some definitions in 1.3.1, acronyms in 1.3.2 and abbreviation in 1.3.3 which will be use in the document, in order to explain some concept and help the general understanding.

1.3.1 Definitions

- User: The customer of the application who can upload reports concerning traffic violations and have access to the system's statistics services.
- Authority: The costumer of the application who can access the information on reports made by the users in order to validate or modify them, and eventually generate tickets. They can also upload on the system information about accidents that occur on the territory and have access to the system's statistics services.
- Traffic Violations: The set of all the infractions that occur on the territory which are recognized by the system.
- Metadata: The information extracted by the system from a report such as license plate, geographical position, data, time and traffic violation type.

1.3.2 Acronyms

- API: Application Programming Interface
- GPS: Global Positioning System
- UI: User inteface

1.3.3 Abbreviations

- Gn: nth Goal.
- R_n : nth Requirement.
- D_n : nth Domain Assumption.
- W_n : nth World Phenomena.
- S_n : nth Shared Phenomena.
- M_n : nth Machine Phenomena.

1.4 Revision history

- Version 1.0 : 08/11/2019

1.5 Reference documents

- Specification document: “Mandatory Project Assignment AY 2018/2019”.
- Lecture slides of professor M.Rossi and E.di Nitto of Politecnico di Milano.
- IEEE 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

1.6 Document's overview

The RASD document is structured by five chapters as describes below:

- Chapter 1: It represent an introduction. It is formed by a general description of the purpose of the application and a list of the goals which the system has to achieve. In addition, it presents a detailed description of the software's functionalities, specifying, in particular, the application domain through the “world and machine phenomena” criterion.
- Chapter 2: It contains an overall description of the project such as a detailed characterization of the software's main functions. It presents also the general factors that can affect the system's behavior, such as user characteristics and project constraints.
- Chapter 3: This Chapter represent the body of the document. It contains the main requirements of the software, and the mockups related to the two different interfaces to the system. It lists some scenarios to show how the system acts in real world situations. They are followed by all the requirements necessary to reach the given goals and linked with the related domain assumptions. Lastly, the non-functional requirements are defined through performance requirements, design constraints and software system attributes.
- Chapter 4: It contains the Alloy model of some important aspects, with all the related comments and documentation in order to show in a formal way how the project has been structured.
- Chapter 5: This final chapter contains the effort, in terms of hours, which each member of the group spent working on the project.

Chapter 2

Overall description

This chapter describe the system in an high level. The section 2.1 describe the prospective of the product and the section 2.2 describe its functions. The sections 2.3 and 2.4 lists the actors of the system and the constraints concerned all the system.

2.1 Product prospective

The SafeStreets software is thought as a system which takes as input the data sent by both the users and authorities (such as traffic violations reports or the accidents that occur on the territory) and re-elaborate them in order to output specific information (such as statistics or suggestions) with different visibility rules. Moreover, the system is built to make as simple and efficient as possible the way in which all the users interact with it. The idea is to give users and authorities different ways in which they can interface with the system. In particular users are provided with a mobile application through which they can report traffic violations and have access to the system's statistics service. They do not need to be registered on the application to benefit from its services. Authorities are provided with a website through which they can consult the reports made by the users, upload information about accidents that occur on the territory and have access to the system's statistics service.
In figure 2.1 is shown the interactions between the actors and the system. The class diagram in figure 2.2 shows with an high level of description the structure of the System.

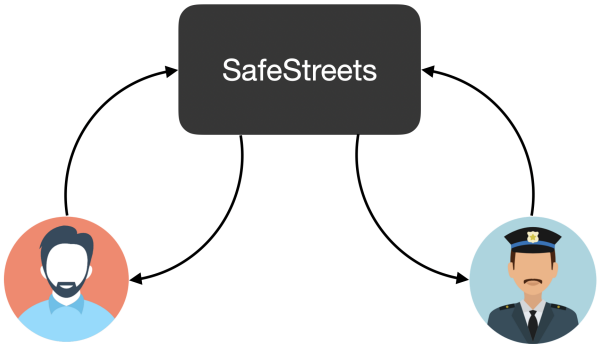


Figure 2.1: Interaction with the system

2.2 Product function

This section lists and describes the main functions of SafeStreet.
In 2.2.1 is descried the violation reporting system.
In 2.2.2 is described the portal built for authority.
In 2.2.3 is described the statistics offered by the system.

2.2.1 Violation reporting system

This part of the whole system manages the action of reporting a violation. The user uses the mobile phone interface to take pictures of a violation, with the possibility to delete some of them; the system assign to the set of picture a date and a time which are taken from the system clock in order to avoid any possible modification. The user has to take also a picture of the license plate to continue with the operation. The system derive automatically the license plate from the picture, if it's wrong the user can retake the picture, or abort the operation if the environment characteristics aren't good (bad light ..). To complete the operation the user has to choose from a predefine set the type of violation and then send it to the system. The whole pipeline guarantee the reliability and consistency of the data due to the impossibility to load a pre-modified picture, or to take picture with fake-gps or modified date and time.
In figure 2.3 is shown a state diagram representing the process of reporting a violation, from the initial to the final states.

2.2.2 Authority portal

The authorities can also access directly to stored data, with a website interface, in order to modify, validate or delete the reports made by the user; this interface allows also authorities to see all the sensible data such as license plates, in order to use it to generate traffic tickets. In this part of the system authorities can also add data about accident occurred in the city.
The Authority can also see some intervention suggestion made by the system retrieving data about accidents and violations, shown on a map and classified by category of intervention (e.g. barrier, camera ..).
In figure 2.4 is shown a state diagram representing the process of validation a violation report by authority, from the initial to the final states.
In figure 2.5 is shown a state diagram representing the process of uploading an accident report by authority, from the initial to the final states.
In figure 2.6 is shown a state diagram representing the process of suggesting an intervention by the system, from the initial to the final states.

2.2.3 Statistics

The system provides a set of statistics based on validated data, to show with two different representations data about the events occurred in the city.
In particular SafeStreets shows a map with unsafe areas (the ones with more accidents) and areas with high number of violations; in addition it provides histograms concerning the temporary trend of violations. Both representations are exploitable through different time granularity and violations or vehicle category.
In figure 2.7 is shown a state diagram representing the statistic service, from the initial to the final states.

2.3 User characteristics

The actors of the system are:

- USER: a person who uses SafeStreets to report a violation and to consult the available statistics about the safeness of the streets in the city.
- AUTHORITIES: an entity that check the violations reported by the user, validates, modifies or deletes it, acquires data to emit traffic tickets and insert data about accidents occurred in the city.

2.4 Constraints

The current constraints on the project are related to the provision of hardware resources to run some of the software's main features. In particular, to guarantee the reliability and consistency of the data the reports made by the users must contain their geographical position, so the devices unprovided of a GPS system are excluded from the violation reporting service. Most of the services offered by the system are accessible only through an internet connection, otherwise its functionalities are considerably limited. Moreover, all the devices with processors that does not support the execution of inference algorithms cannot run the license plate detection function.

2.5 Assumptions and dependencies

A number of factors that may affect the requirements specified in the document are assumed to be satisfied and listed below:

- D_1 : The mobile device has a camera.
- D_2 : The Access to the user's camera is permitted.
- D_3 : There is stable internet connection.
- D_4 : Images sent by the user are pertinent to the purpose of the service.
- D_5 : The system has an internal clock.
- D_6 : The user's device is equipped with GPS.

The system also needs some external resources to provide some of its basic services. The GPS on the mobile device returns a set of geographic coordinates, so it's necessary to use an external API to convert them in the corresponding address. Moreover, some of the statistics offered by the system, such as the highlight of possible unsafe areas or the intervention suggestions, require an external API providing a mapping service.

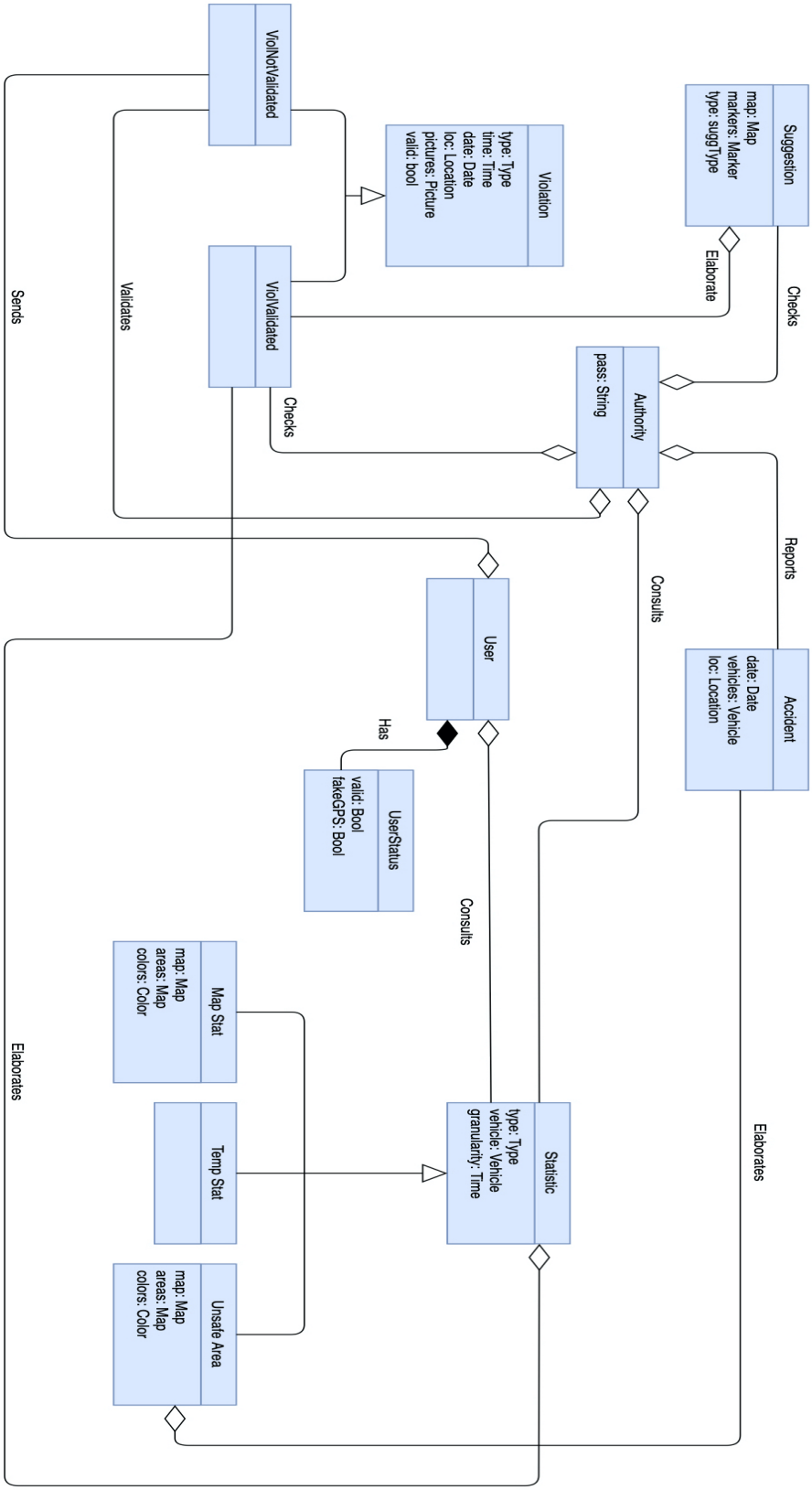


Figure 2.2: High level class diagram

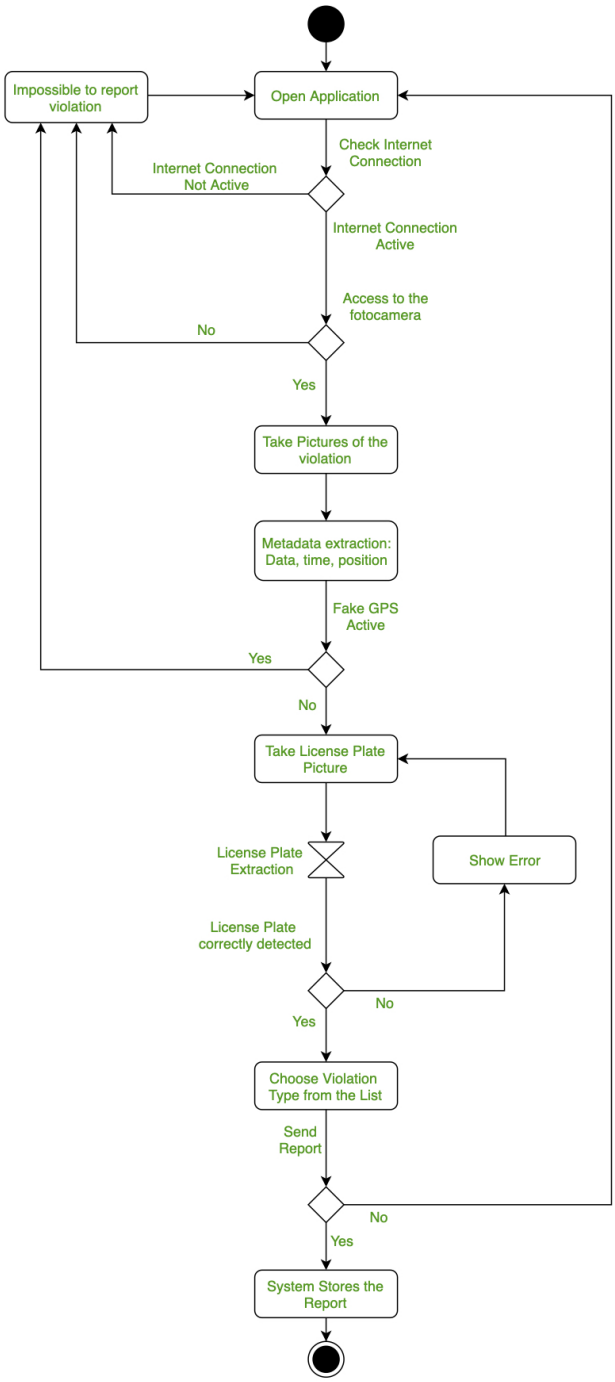


Figure 2.3: State diagram of reporting a violation

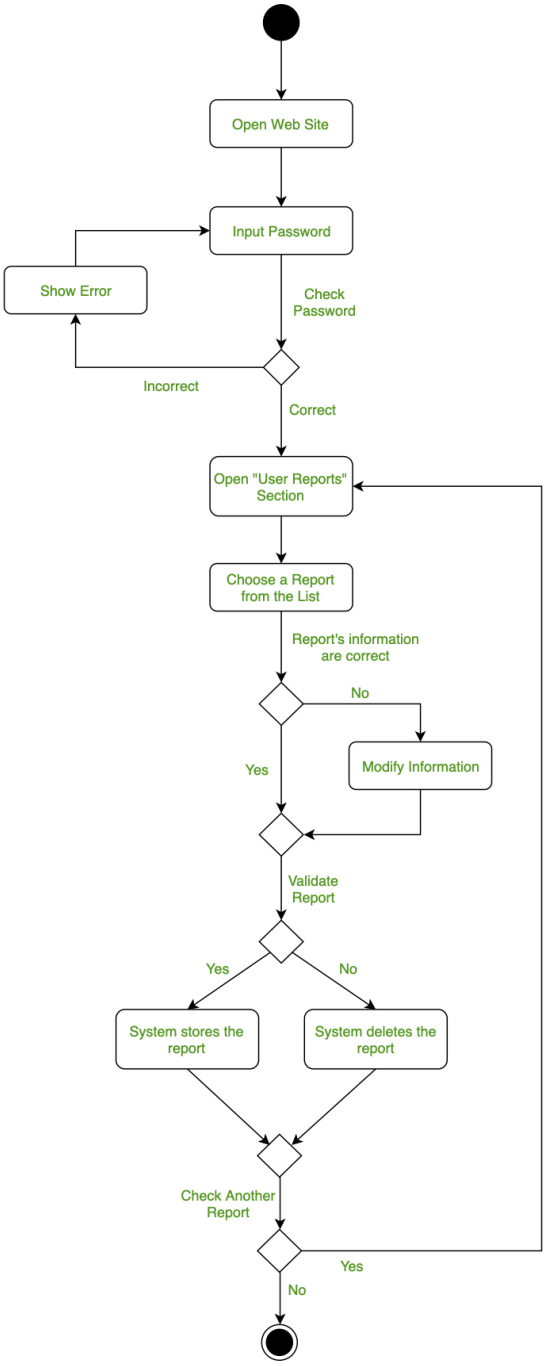


Figure 2.4: State diagram about validation of reports

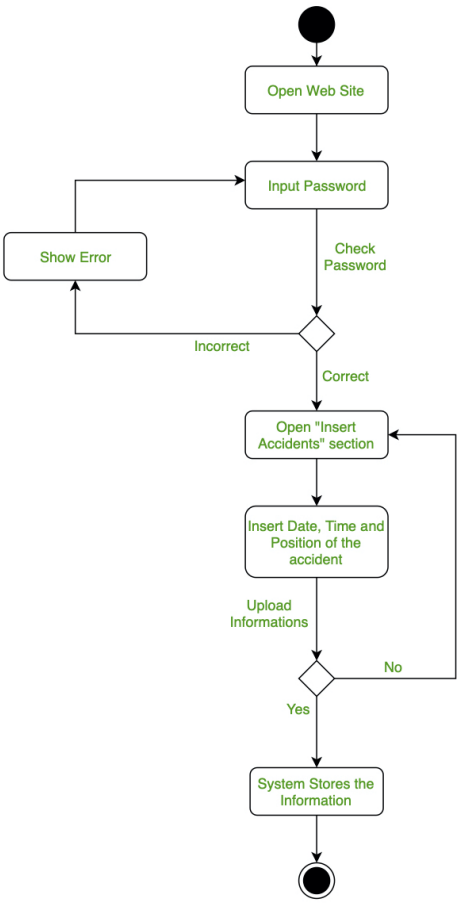


Figure 2.5: State diagram of uploading an accident

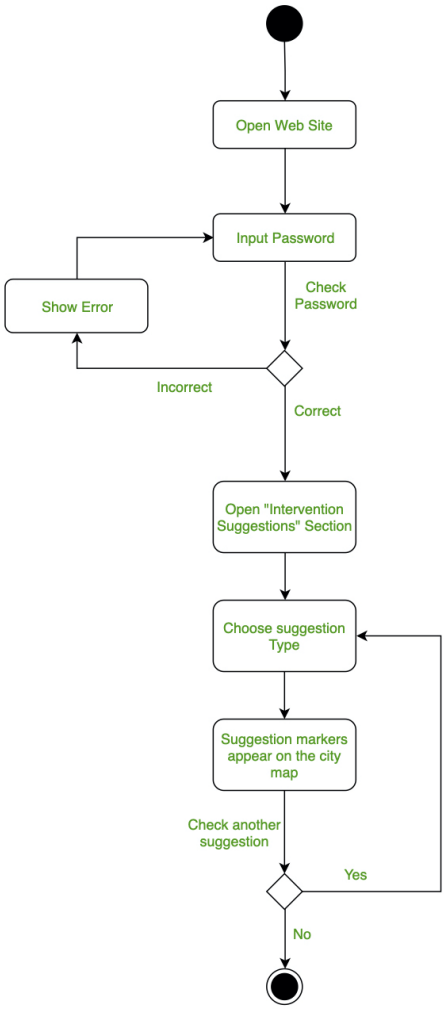


Figure 2.6: State diagram of intervention suggestion

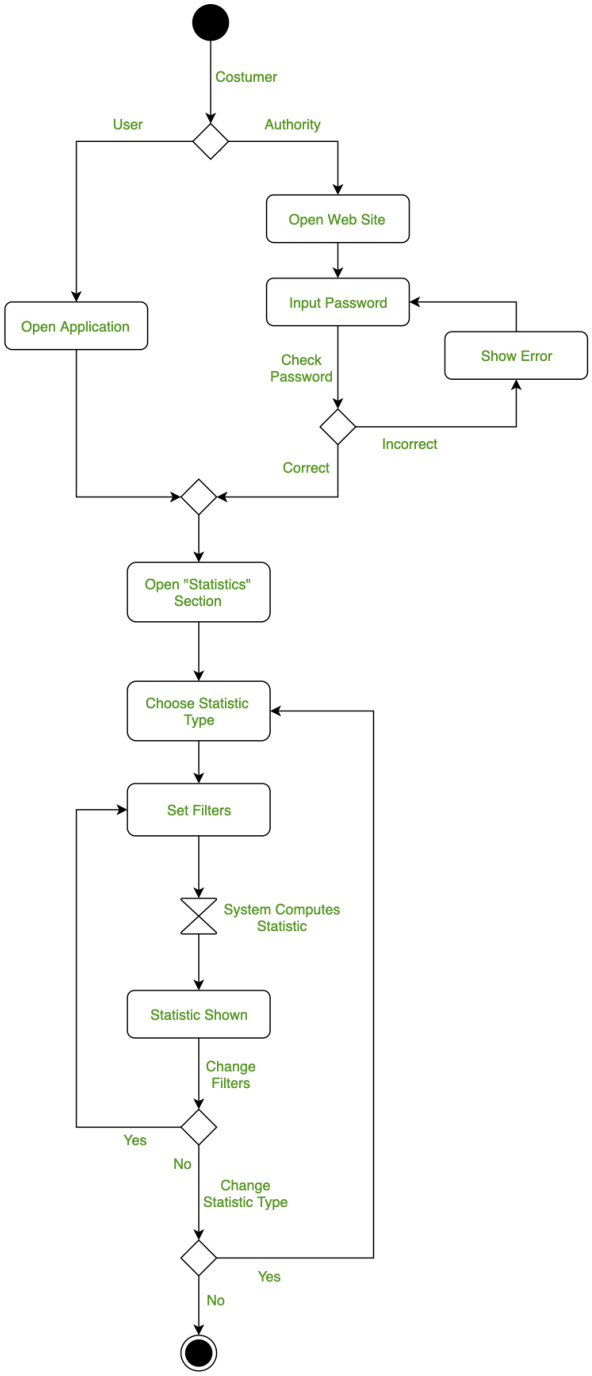


Figure 2.7: State diagram of statistic service

Chapter 3

Specific Requirements

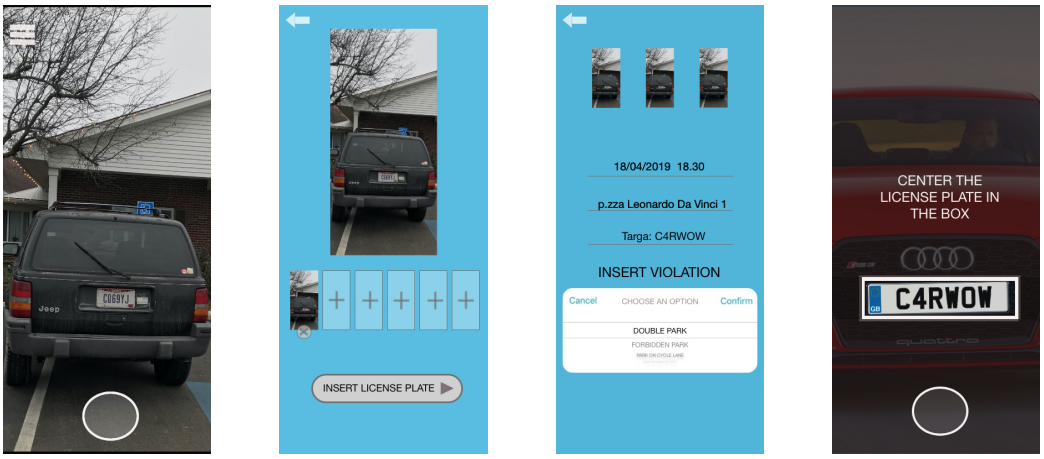
This chapter focuses on requirements of the system. The section 3.1 lists all the interfaces'requirements, the section 3.2 lists the functional requirements, the section 3.3 tells about performance requirements, the section 3.4 tells about the design constraints and the final section 3.5 lists the attributes of the system.

3.1 External interface requirements

This section focuses on interfaces'requirements. In 3.1.1 tells about the ui of the system, in 3.1.2 and 3.1.3 about hardware and software interfaces.

3.1.1 User Interface

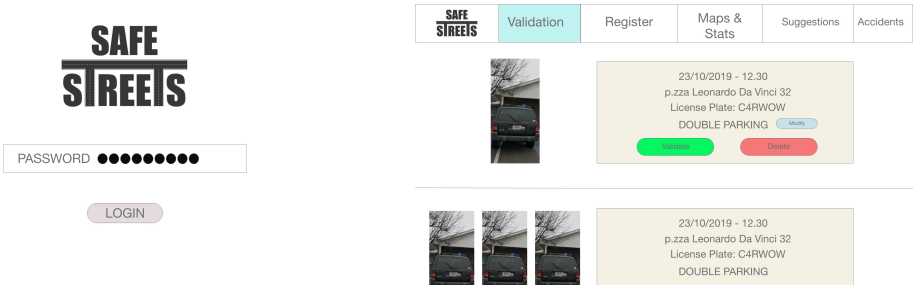
SafeStreets contains two user interfaces, one of them is the mobile phone's application, which could be used by both users and authorities to report violations and consult statistics, the other one is the website interface, used by authorities to insert data about accident and to manage data about all the reported violations.
The following mockups shows the idea of how the system should work.



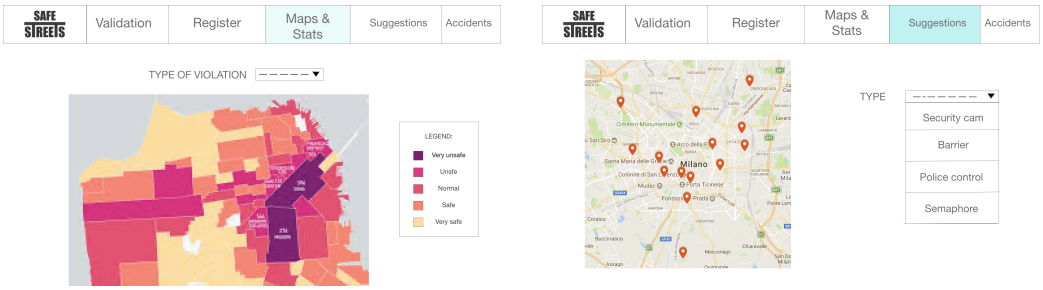
(a) First part of the action of report a violation.

(b) Second part of the action of report a violation and menu with statistics.

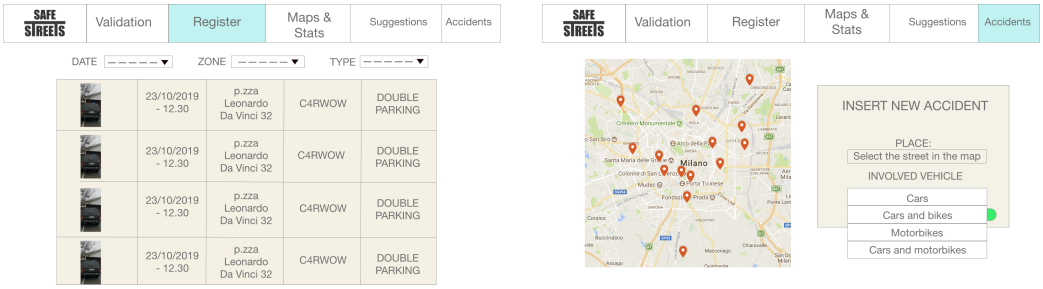
Figure 3.1: User Interface of the mobile app.



(a) Login and validate reports.



(b) Events map and suggestions



(c) Database of violations and accidents report.

Figure 3.2: User interface of authority website.

3.1.2 Hardware interfaces

The system has no hardware interface.

3.1.3 Software interfaces

The system doesn't provide any API to external applications; it only uses internal APIs as will be explained in the design document.

3.2 Functional requirements

This section lists all the system requirements of SafeStreets and their traceability matrix, in particular in the subsection 3.2.3. To explain better all the requirements the section starts with a list of **use cases and scenarios** for both user (3.2.1) and authority (3.2.2) followed by a use case diagram.

3.2.1 User

Scenario 1:
Francesco is walking down the street, while he notices a car parked in a disabled spot whitout parking pass exsposed. He thinks this kind of violation should not succed and would want to report it to the neareast authority. Unfortunately there is no one police officer to advise, but he remembers that a friend told him of SafeStreets and suddendly downloads it to report what has happened. Francesco opens SafeStreets to report the violation:

- He takes several pictures of the car to highlight the main evidences of the violation.
- He adds a picture of the license plate.
- The system runs an algorithm to compute the license plate.
- All data gathered by the system(time, date, location and license plate) are listed and Francesco can check, if they are right, and report them.

If an error occurs in the previous steps, the system throws a warning stating: “Process aborted”, therefore the report and data concerning it are discarded.
Scenario 2:
Amedeo wants to take a carefree walking around its town, for this purpose before going out he checks statistics built by SafeStreets to avoid caotic and unsafe areas.

Name	Report Violation.
Actor	User.
Entry Condition	User opens SafeStreets.
Event Flow	1.The User can take a picture of the violation clicking on the button from the homepage. 2.The User can take other pictures of the violation (they can be at most five). 3.The system forces the User to take a picture of the license plate. 4.The system checks if the User's status is valid. 5.The system retrieves time, date, location and runs an algorithm to compute the license plate from the picture. 6.The system displays all the information above. 7.The User selects the type of violation among a predefined set. 8.The User confirms and sends the report.
Exit Condition	The violation is successfully reported. The violation is aborted. If the process is aborted for any reasons the system throws a warning stating: “Process Aborted”.
Exceptions	In every steps the User can abort the reporting process and all data concerning it are descared. 2a) The User can discard a picture. 2b) If there is one picture and it is discarded, the process returns to step 1. 4a) The system is not able to retrieve a piece of the information required, so it aborts automatically the reporting process.

Name	User Checks Statistics.
Actor	User.
Entry Condition	SafeStreet opened.
Event flow	1.User pushes on menu button. 2.User pushes on “Statistics” button. 3.User can select one type of violation or all. 4.The system displays the annual statistics concerning data gathered by SafeStreets and the unsafe areas map referred to type chosen above. 5.User can select the statistics granularity (day, week, month, year). 6.The system displays statistics and unsafe areas map according to the above chosen granularity.
Exit Condition	Map and statistics are correctly displayed.
Exceptions	

Name	User Checks Unsafe Areas.
Actor	User.
Entry Condition	SafeStreet opened and “Menu” button pushed.
Event flow	1.User pushes on “Unsafe Areas”. 2.System shows graphic map with areas highlighted with different colors, according to the highest frequency of accidents.
Exit Condition	The graphic map is correctly displayed.
Exceptions	Authorities did not submit any accidents. The system will display only the map without highlighted areas and a warning stating: “Sorry, data on accidents are not available at the moment.”.

3.2.2 Authorities

Scenario 3:
Mattia, an elderly police officer, is expert of traffic violations due to many years of service. However he actually strives a lot when he is on foot patrol to emit traffic tickets, beacause he is not young anymore. Fortunately Mattia is chosen as SafeStreets police manager, in order to apply his experience on traffic violations to validate SafeStreets,Äô users reports and, moreover, to generate traffic tickets from them.
Scenario 4:
The 2019 balance of Milan is surprisingly positive, so the municipality is taking into account to partially invest the annual gain on street safety. The decision making time is not enough, why it is needed to gather a lot of data and evaluate the possible interventions, so authorities decide to rely on the SafeStreets initiative.

Name	Authority Login.
Actor	Authority.
Entry Condition	Authority opens the Login WebPage.
Event flow	1.A writes the password in the “Password” field. 2.A pushes on “Log in”.
Exit Condition	Login is performed.
Exceptions	The password is wrong. The system throws a warning stating: “Wrong password, please try again!”.

Name	Authority Check Statistics.
Actor	Authority.
Entry Condition	Log in performed.
Event flow	1.Authority pushes on “Statistics” button. 2.Authority can select one type of violation or all. 3.The system displays all the annual statistics concerning data gathered by SafeStreets and the unsafe areas map referred to type chosen above. 4.A can select the statistics granularity (day, week, month, year). 5.The system displays statistics and unsafe areas map according to the above chosen granularity.
Exit Condition	Map and statistics are correctly displayed.
Exceptions	

Name	Data Validation.
Actor	Authority.
Entry Condition	Login Performed and “Validation” field pushed.
Event flow	1.Authority chooses one type of violation. 2.The system shows all violations of the chosen type not validated yet. 3.A selects a violation. 4.The system dysplays all the data concerning the violation. 5.A pushes on “Validate” button 6.The violation is removed from the “Validation” field and added to the “Records” field.
Exit Condition	The violation is successfully validated.
Exceptions	4a) A notices some data are wrong(e.g. category of violation or license plate) and pushes on “Edit” button to fix data and after on “Confirm” when they are correctly modified. 4b) A notices some data are not pertinent (e.g. pictures of goose) and pushes on “Discard” button to delete the violation.

Name	Check Suggestion.
Actor	Authority.
Entry Condition	Login Performed and “Suggestions” field pushed.
Event flow	1.Authority can select one type of suggestion or every type together. 2.The system shows markers where suggestions are present on the map according to the chosen above.
Exit Condition	The system displays markers.
Exceptions	There are not markers for certain suggestions, why the number of violations associated to a specific suggestion is not enough for the system parameters.

Name	Submit Accident.
Actor	Authority.
Entry Condition	Login Performed and “Accidents” field pushed.
Event flow	1.Authority pushes “Add accident”. 2.Authority fills information required (date, location and vehicles involved). 3.Authority pushes on “Confirm” button.
Exit Condition	The accident is successfully reported.
Exceptions	2a). If one of the fields is not filled and the “Confirm” button is pushed, the system throws a warning stating: “Other information required”. Authority can push on “Ok” button to submit the missing information. 3a) A can push “Cancel” button to go back in the initial page.

Name	Authority Check Unsafe Areas.
Actor	Authority.
Entry Condition	Login performed.
Event flow	1.A pushes on "Unsafe Areas". 2.System shows graphic map with areas highlighted with different colors, according to the highest frequency of accidents.
Exit Condition	The graphic map is correctly displayed.
Exceptions	Authorities did not submit any accidents. The system will display only the map without highlighted areas and a warning stating: "Sorry, data on accidents are not available at the moment.".

Use case diagram

The figure 3.3 shows the use case diagram of SafeStreet. The use-case diagram above shows the main actions both Authorities and Users can perform on the system.

- Authority can browse all the violations and validate them if they are correct, according to authority standards.
- Authority can report accidents that occur on the territory and check the suggestions built by SafeStreet to make streets safer.
- Users can send violation to SafeStreet.
- Both User and Authority, finally, can check Statistics composed by unsafe areas and type and frequency of a certain violation.

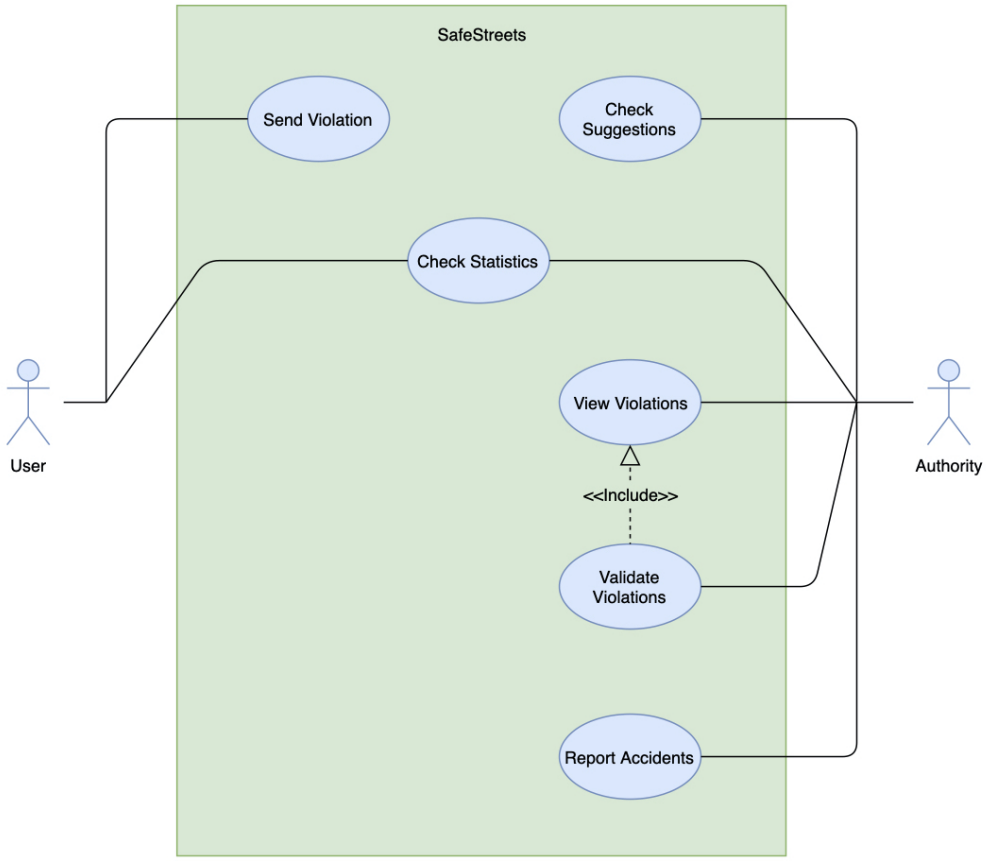


Figure 3.3: Use case diagram.

3.2.3 Requirements

G_1:Users can report traffic violations to authorities by sending pictures to the system.

- R_1: The system allows to take pictures only within the software itself.
- R_2: The report can be made only if the internet connection is available.
- R_3: The system imposes the user to include a picture of the license plate of the vehicle that committed the violation.
- D_1: The mobile device has a camera.

- D_2: The Access to the user's camera is permitted.
- D_3: There is stable internet connection.
- D_4: Images sent by the user are pertinent to the purpose of the service.

G_2 : **The System has to make easier the process of reporting violations by extracting metadata from the pictures.**

- R_4: The system gets date and time from its internal clock when the picture is taken.
- R_5: The system obtains the violation's position from the device's GPS.
- R_6: The system converts the geographic coordinates obtained in the corresponding address.
- R_3: The system imposes the user to include a picture of the license plate of the vehicle that committed the violation.
- R_7: The system automatically detects the vehicle's license plate number.
- R_{12} : The system checks that a fake GPS service is not active o the user's device.
- D_5 : The system has an internal clock.
- D_6 : The user's device is equipped with GPS.
- D_3 : There is stable internet connection.

G_3 : **Authorities can access sensitive data for specific pourposes, such as traffic tickets generation, so their consistency and reliability must be granted.**

- R_9: The system stores violations data.
- R_{11}: Authorities must log in to the system.
- R_{12}: The system checks that a fake GPS service is not active o the user's device.
- R_1: The system allows to take pictures only within the software itself.
- R_4: The system gets date and time from its internal clock when the picture is taken.

- R_5: The system obtains the violation's position from the device's GPS.
- D_1: The mobile device has a camera.
- D_2: The Access to the user's camera is permitted.

G_4 : **Both the users and authorities can access to statistics (based on accidents and violations) built by the system.**

- R_8: Access to the violation reports is only permitted to authorities.
- R_{13}: Authorities can modify, discard or validate the violation reports present on the system.
- R_{14}: Authorities can insert data concerning accidents that occur on the territory.
- R_{15}: The system builds statistics based on validated data.
- R_{16}: The system produces statistics concerning possible unsafe areas built on the accidents information.
- R_{17}: Authorities and users can set the temporal granularity and the category of the statistics that they want to consult.

G_5**: System can suggest intervention to authorities, based on shared data obtained by crossed information between police and SafeStreet.**

- R_{18}: The authorities can consult the interventions suggested by the system.
- R_{19}: The system has an associated suggestion for each type of violation.
- R_{20}: The system places a marker on the map where the intervention is suggested.
- R_{21}:The system allows to filter the suggestions on the map by category.

Traceability Matrix
The following table represent a mapping between each requirements and its use case, in order to underline foreach requirements what's its target.
The table shows also that some requirements are important for more than one use cases.

R1	Report Violation
R2	Report Violation
R3	Report Violation
R4	Report Violation
R5	Report Violation
R6	Report Violation
R7	Report Violation
R8	Report Violation
R9	Report Violation
R10	Authority Login
R11	Authority Login Data Validation
R12	Data Validation
R13	Data Validation Authority Check Statistics User Check Statistics
R14	Submit Accident
R15	Authority Check Unsafe Areas User Check Unsafe Areas Submit Accident
R16	Authority Check Unsafe Areas Authority Check Statistics User Check Unsafe Areas User Check Statistics
R17	Check Suggestion
R18	Check Suggestion
R19	Check Suggestion
R20	Check Suggestion

3.3 Performance requirements

The system has to keep the statistics updated daily in order to guarantee good information to all the users.
The system has to be more precise as possible in recognizing the license plates in order to guarantee the correctness of the whole pipeline in reporting a violation.

3.4 Design constraints

This section tells about the design constraints of the system, beginning with standard compliance in 3.4.1, the with hardware limitations in 3.4.2 and finally with other constraints in 3.4.3.

3.4.1 Standards compliance

SafeStreets is thought to give a service to the city of Milan, and could be extended to every other cities. In order to achieve this goal, the way in which the system locates the events and gives the statistics has to follow the standard of streets'addresses of the country (Italy in this case). If the system is extended also abroad, the algorithm that recognize the license plate has to follow the license plate's standard of the various countries.

3.4.2 Hardware limitations

The system has a direct dependence to internet connection, because all the storage system and every statistics are managed and computed by a server.
The process of reporting a violation implies the presence of a camera on the mobile phone (which is present in quiet all the devices).

3.4.3 Any other constraint

The system must respect a privacy policy while managing the license plates and its location. The system is built in a way that guarantee the visibility of this data only to authorities on their interface.

3.5 Software system attributes

This final section lists the attributes of the system:Reliability in 3.5.1, availability in 3.5.2,security in 3.5.3,maintainability in 3.5.4, compatibility in 3.5.5.

3.5.1 Reliability

The system doesn't have constraints relied to security , but it must guarantee a consistency of the information stored inside it. In order to achieve this goal, all the sensible data and information can't be modified excepted for the type of a violation. Every data is also stored permanently according to law (example in Italy every formal documents has to be keep for at least five years) in case of a request of consultation by courts.

3.5.2 Availability

The system should be "ready to use" both by user and authorities. The authorities'interface depends directly on the server, which must be available at least 90% of the time. To do this, the system uses an external service to keep its server alive (such as Google cloud platform or other services which manage any possible system's fault and problems with lots of connection).
Also the user interface should be "ready to use", but the functionality cannot be guarantee every time, because of security and consistency constraints. In order to guarantee that pictures of a violation and its metadata are consistent and not-modifiable, the user's device must have a stable internet connection in order to avoid mismatches with location and time of the violation.

3.5.3 Security

Authorities can use data from SafeStreets to generate traffic tickets, so the system must be secure. To guarantee the security of data, SafeStreets operates firstly avoiding the possibility to modify sensible data (as mentioned before), and guaranteeing the access only to authorities by a unique password which is changed periodically.
The system will also use external services to keep the whole system secure by cyber attacks.

3.5.4 Maintainability

The system must be modifiable for future implementations or upgrades, so the design of the structure will use appropriate design patterns to make any update easier and and conflict-avoidable.

3.5.5 Compatibility

The interfaces of SafeStreets are compatible with every type of user, in fact they are easy to use and the technology required is available to quite everyone.

Chapter 4

Formal Analysis Using Alloy

In this chapter is shown the use of Alloy to analyze the critical parts of the system.
In particular the most crucial points to analyze are :

- User can report a violation if hisher device settings are in line with the requisite (GPS on , no fake-gps, internet connection available, access to camera granted..).
- All the statistics built on violations must use only violations which are validated by authority.
- Only authority can access data about violations in order to accept, delete or modify them.

Any other signatures and facts are used to simplify the model in some cases or to help the building of the model in others.
In particular section 4.1 lists and explains all the signatures used , section 4.2 all the facts of the model, section 4.3 the assertions used to check the crucial points mentioned before and the last section 4.4 shows some example of worlds with Alloy's predicates.

4.1 Signatures

This section shows the piece of code in Alloy with all the signature used for this model.
The structure and the presence of the signatures are needed to build a consistent model, for example the signature **User** represent the UI (mobile application) and not the single user, but it was needed to build the model.

```
abstract sig Bool{}
one sig True extends Bool{}
one sig False extends Bool{}


abstract sig Actor{
}

/*This signature represents the User Interface.
 *the only attributes are the settings of his mobile device,
 *in order to verify if it presents the necessary requirements
 *to make a violation report.*/

sig User extends Actor{
  device: one DeviceSettings,
}
/*The Authority signature has the attributes
 *concerning the sets of the validated violation
 *reports and the accidents that have been
 *uploaded into the system.*/

one sig Authority extends Actor{
  validatedViolations: set Violation,
  insertedAccidents: set Accident
}

/*The DeviceSettings signature models
 *all the characteristics that a mobile device
 *requires to allow the User to report a violation.*/

sig DeviceSettings{
    num_cam: one Int,
    auth_camera: one Bool,
    gps_available: one Bool,
    fake_gps: one Bool,
    internet_conn: one Bool
}

abstract sig Event{}

/*To model a violation report the attributes
 *to be tracked are the ones necessary to verify
 *its consistency and admissibility: In particular
 *it presents the number of the pictures composing
 *it and a validation field.
 *The attribute author create a bind between a
 *violation and the user in order to verify some facts.*/
```

```
sig Violation extends Event{
    author: one User,
    num_pics: one Int,
    plate_pics: one Bool,
    validated: one Bool
}

sig Accident extends Event{
}

abstract sig Statistic{
}

/*sig ViolationStatistic, AccidentStatistic:
*This signatures represent the two different
*macro-categories of statistics made by the system.
*For each one there are the data sets on which they are built.
*/

sig ViolationStatistic extends Statistic{
violation: set Violation
}

sig AccidentStatistic extends Statistic{
accident: set Accident
}
```

4.2 Facts

This section shows the piece of code in Alloy that contains facts.
Some of the facts model the domain assumptions, the other are needed to build a consistent model, such as the impossibility to have a device settings with no user.

```
/*fact Camera, CameraAuthorization, GpsAvailable,
*PicsViolation, InternetAvailable:
*The following facts model the conditions under which
*a violation report can be made.
*In particular:
*A mobile device needs to have at least one camera
*and the authorization to use it must be active in order
*to make pictures (Camera, CameraAuthorization).
*The GPS of a mobile device needs to be active
*and available (GpsAvailable).
*A violation report must contain at least 2 images,
*and at least one of these must contain
*the license plate (PicsViolation).
```

```
*Finally the internet connection of the user device
*needs to be active and correctly working (InternetAvailable).*/

fact Camera{
  all u:User | u.device.num_cam>0
}

fact CameraAuthorization{
  all u:User | u.device.auth_camera=True
}

fact GpsAvailable{
  all u:User | u.device. gps_available=True
    and u.device.fake_gps=False
}

fact PicsViolation{
  all v:Violation | v.num_pics>1 and v.num_pics<7
    and v.plate_pics=True
}

fact InternetAvailable{
  all u:User | u.device.internet_conn=True
}

/*This condition imposes that all violations reports
*on which statistics are built are validated;
*otherwise their reliability would be altered.*/

fact ValidatedViolation{
  all v:Violation, s:ViolationStatistic | v in s.violation
    and v.validated=True
}


/*This conditions are necessary to ensure
*the consistency of the statistics. In particular,
*to build a statistic it is required that the data
*sets used are not empty.*/

fact UnsafeArea{
  all a: Accident | some b: AccidentStatistic |
    a in b.accident
}

/*This condition imposes that a violation report
*can be validated only by a single authority.*/
```

```
fact ViolationAndAuthority{
  all v: Violation | one a:Authority | v.validated=True <=>
     v in a.validatedViolations
}

/*This is an admissibility requirement which specifies
*that a device cannot exist without a user.*/

fact NoDeviceWithoutUser{
  all d: DeviceSettings | some u:User| u.device=d
}

/*This condition impose that all the accidents
*are bind to the Authority*/

fact AccidentAndAuthority{
  all ac: Accident |one au:Authority |ac in au.insertedAccidents
}
```

4.3 Assertions

This section shows the piece of code in Alloy that contains assertions.
As was mentioned before, this assertion model the crucial points of the whole model.
In particular they verify that violations are reported if the requisites of device settings are accepted and that statistics are built only on data validated by authority.
The figure 4.1 shows the result of Alloy tool in checking assertions.

```
/*It verifies the consistency on previous written
*facts concerning the conditions for which a user
*can report a violation. In particular, through the
*search of a counterexample if verifies that there
*are no violations that do not satisfy the necessary
*requirements: fake GPS not active, internet
*connection active and the conditions on the Pictures.*/

assert UserCanReportViolation{
  no v:Violation | v.author.device.fake_gps=True
    and v.num_pics>1
    and v.num_pics<7 and v.plate_pics=True
    and v.author.device.internet_conn=False
}
```

```
/*It verifies the consistency on previous written facts
*concerning the conditions under which a statistic can be made.
*In particular there are no statistics built on not
*validated data.*/

assert BuildStatisticsViolation{
  no v:Violation, s:ViolationStatistic | v in s.violation
    and v.validated=False
}

/*This assertion checks if there are counterexamples
*concerning validated violations reports that do not
*refer to any authority.*/

assert NoViolationWithoutAuthority{
  no v: Violation | one a:Authority | v.validated=True
    and !(v in a.validatedViolations)
}

check UserCanReportViolation for 3
check BuildStatisticsViolation for 3
check NoViolationWithoutAuthority for 3
```

```
Executing "Check UserCanReportViolation for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  2480 vars. 194 primary vars. 6169 clauses. 52ms.
  No counterexample found. Assertion may be valid. 4ms.

Executing "Check BuildStatisticsViolation for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  2269 vars. 197 primary vars. 5402 clauses. 41ms.
  No counterexample found. Assertion may be valid. 4ms.

Executing "Check NoViolationWithoutAuthority for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  2237 vars. 194 primary vars. 5351 clauses. 33ms.
  No counterexample found. Assertion may be valid. 4ms.
```

Figure 4.1: Assertion checks

4.4 Predicates

This section shows the piece of code in Alloy that contains predicates.
The first world shows the model focusing on user and violation (figure 4.2)
The second (figures 4.3, 4.4) and third (figure 4.5) worlds introduce also statistics and accidents.

```
pred world1{
  #User = 3
  #Violation = 6
}

pred world2{
  #User = 2
  #Violation = 6
  #Statistic = 2
  #Accident = 3
}
```

```
pred world3{
  #User = 3
  #Event = 6
  #Statistic = 5
}

run world3 for 30 but 1 Authority
run world1 for 30 but 0 Accident, 1 Authority, 0 Statistic
run world2 for 30 but 1 Authority
```

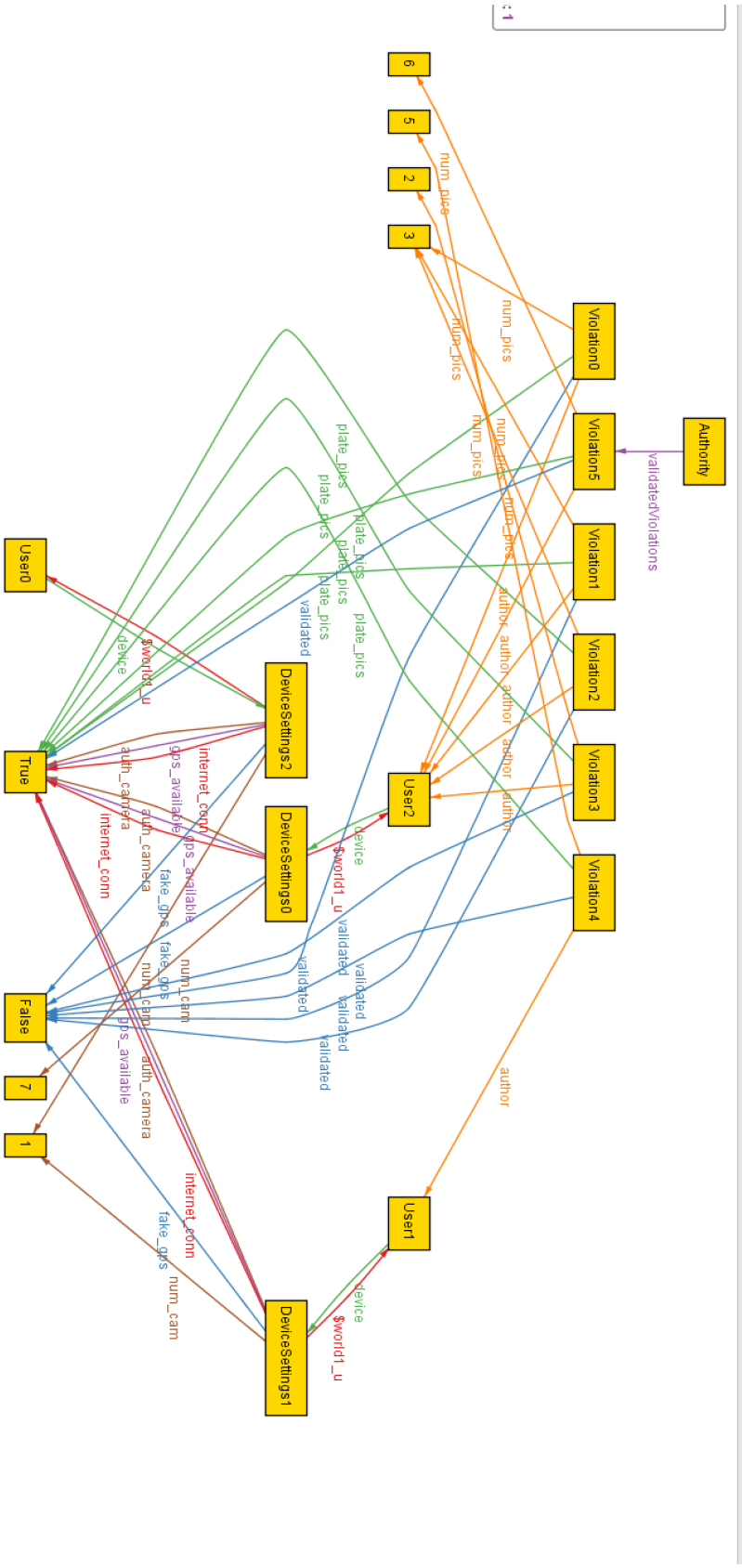


Figure 4.2: World 1 : Connections between users, device settings and violations.

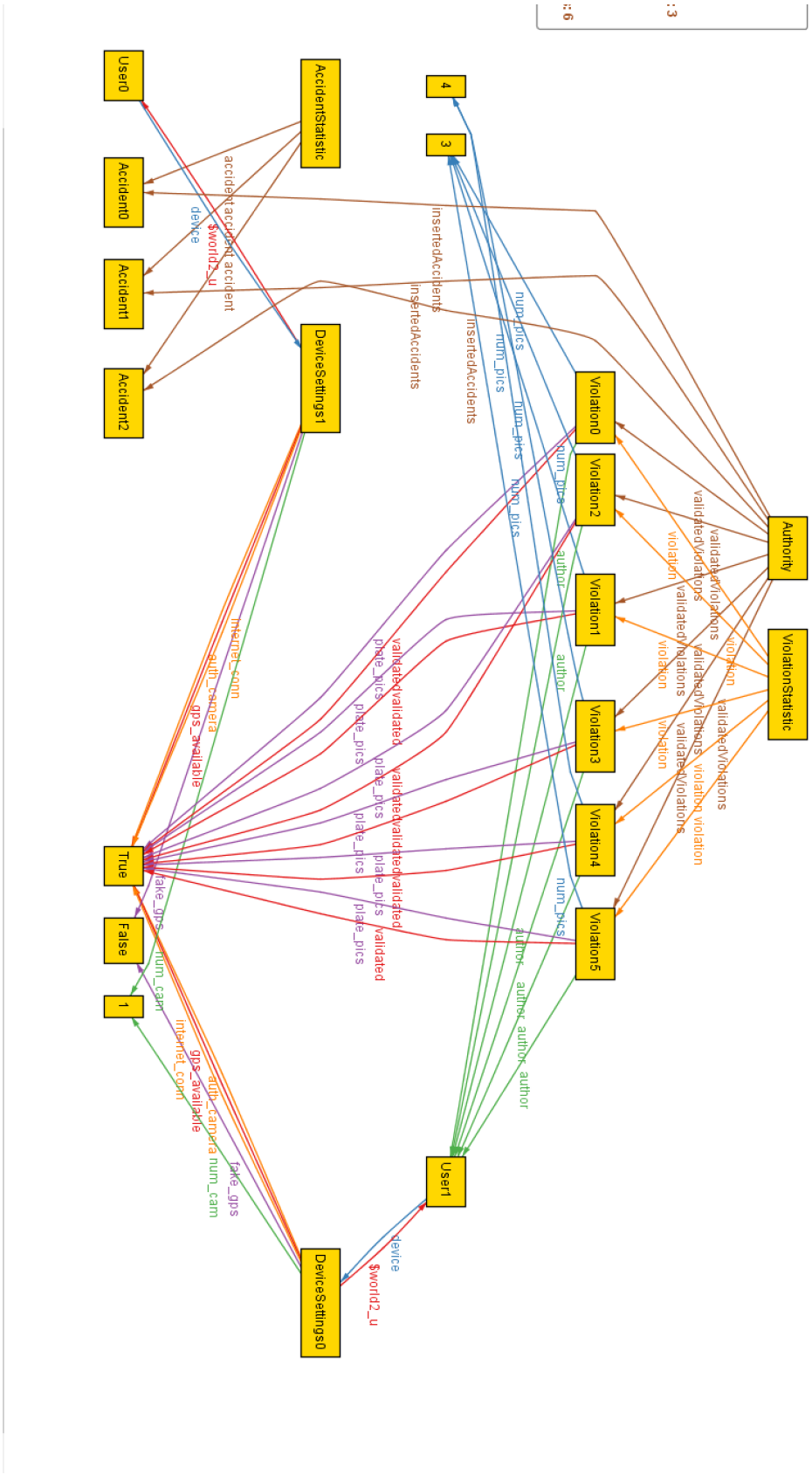


Figure 4.3: World 2 : Statistics about both violations and accidents.

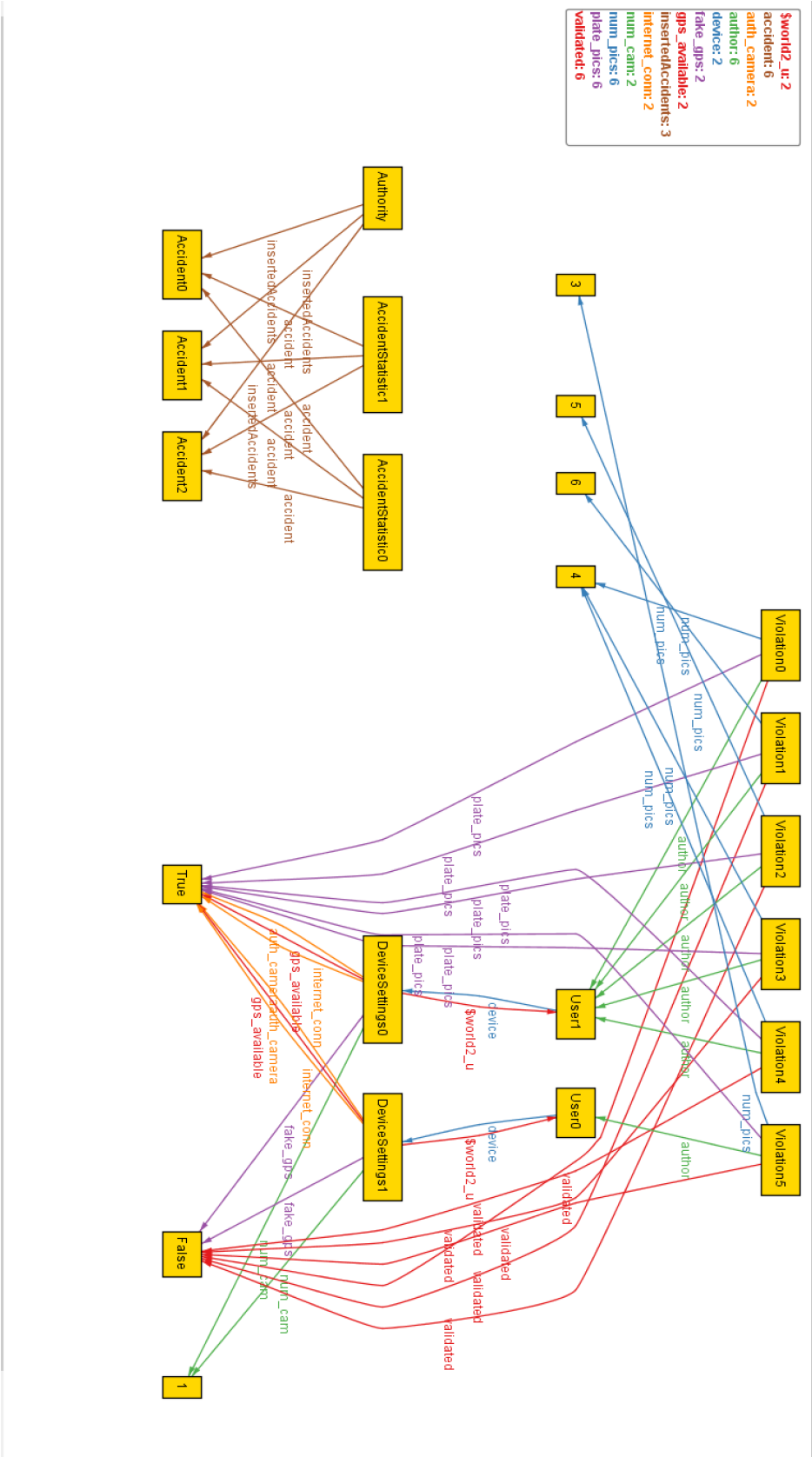


Figure 4.4: World 2 : No statistics about violations and violations not validated.

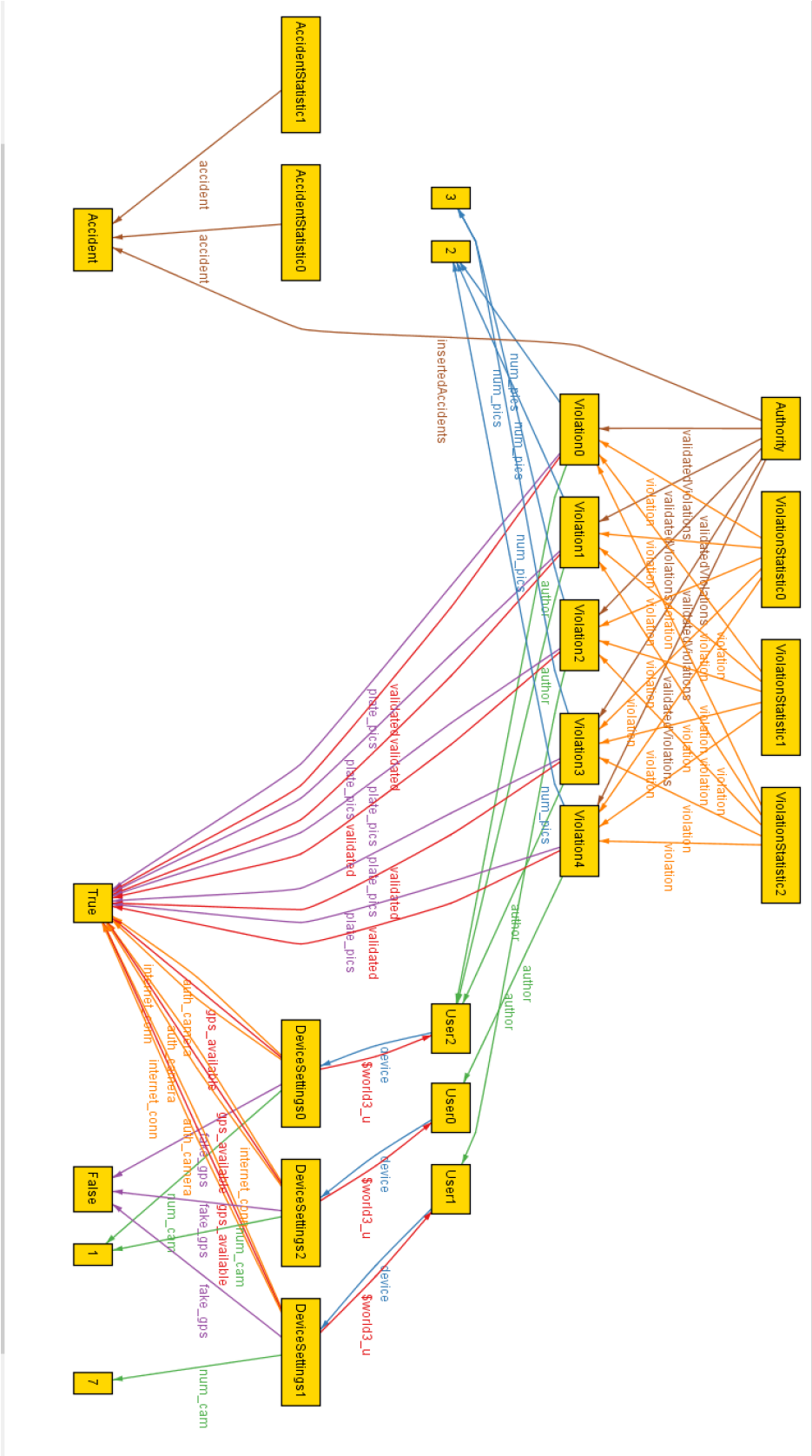


Figure 4.5: World 3 : Complete model with all signatures.

Chapter 5

Effort Spent

Mattia Micheloni

TASK	HOURS
Ch.1	1
Ch.2	4
Ch.3	18
Ch.4	12

Francesco Montanaro

TASK	HOURS
Ch.1	4.5
Ch.2	4
Ch.3	13
Ch.4	13

Amedeo Pachera

TASK	HOURS
Ch.1	2
Ch.2	3
Ch.3	13
Ch.4	14