



POLITECNICO MILANO 1863

Software Engineering 2 Requirement Analysis and Verification Document SafeStreets

Version 1.1 - 03/12/2019

Authors:

Andrea Falanti

Andrea Huang

Professor:

Prof. Elisabetta Di Nitto

Contents

1	Introduction	4
1.1	Purpose	4
1.2	Scope	5
1.3	Definitions, Acronyms, Abbreviations	8
1.4	Revision history	9
1.5	Reference Documents	10
1.6	Document Structure	10
2	Overall Description	12
2.1	Product perspective	12
2.1.1	Introduction	12
2.1.2	Class diagram	12
2.1.3	State diagrams	14
2.2	Product functions	18
2.3	User characteristics	20
2.4	Assumptions, dependencies and constraints	20
2.4.1	Domain assumptions	20
2.4.2	Dependencies	21
2.4.3	Constraints	21
3	Specific Requirements	23
3.1	External User Requirements	23
3.1.1	User Interfaces	23
3.1.2	Hardware Interfaces	29
3.1.3	Software Interfaces	29
3.1.4	Communication Interfaces	29
3.2	Functional Requirements	29

3.2.1	Citizen	29
3.2.2	Third Party	36
3.2.3	Requirements	41
3.3	Performance Requirements	46
3.4	Design Constraints	46
3.4.1	Standards compliance	46
3.4.2	Hardware limitations	46
3.4.3	Any other constraint	47
3.5	Software System Attributes	47
3.5.1	Reliability	47
3.5.2	Availability	48
3.5.3	Security	48
3.5.4	Maintainability	48
3.5.5	Portability	48
4	Formal Analysis Using Alloy	50
5	Effort Spent	61
6	References	62

Introduction

1.1 Purpose

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations. The application allows users to send pictures of violations, including their date, time, and position, to authorities. Examples of violations are vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on.

SafeStreets stores the violation reports provided by users, that can be made from the official mobile application or from the SafeStreets website, so that users and authorities can visualize and analyze the data received by the system, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations. The data can be accessed with different levels of visibility, where most sensitive data can only be mined by authorities.

SafeStreets also provide a service aimed at helping municipalities to identify potentially unsafe areas and suggest possible interventions. To improve the accuracy of these features, a municipality can also offer a service for retrieving info about accidents in their territory, so that SafeStreets is able to cross the municipality's data with its own stored data and also analyze and suggest possible improvements to the areas with most reports.

Goals

- G1. Report certified traffic violations to authorities of city where the violations have happened.

- G2. Store data about violations and cross it with municipality's one when an API to its data is provided.
- G3. Allow municipalities to generate a list of possible reasonable interventions to resolve the violations detected in most unsafe areas.
- G4. Recognize public institutions automatically during registration to setup correctly accounts of municipalities and authorities.
- G5. Provide a synthesis of all the violations' data to users, based on their authorization level.
 - G5.1. Normal users can access only to aggregated data, without seeing any private info about a reported violation, like the license plate or the personal info of the user who reported the violation.
 - G5.2. Authenticated authorities can access to all violations' data, as aggregated data or singular reports.
- G6. Allow users to send a report of a possible traffic violation, identifying clearly the transgressor.
- G7. Allow users to check their personal info and the list of violations they reported.

1.2 Scope

The SafeStreets service is offered to common users to report traffic violations that hinder the normal flow of the traffic. It is thought to offer an aide to the public officers in detecting violations and thus provide a more regulated traffic system to the citizens. The service stands in the middle between the common citizen and the authorities, providing a real-time update of the situation on the streets.

The software will be distributed in Italy and will provide to any Italian citizen the possibility to report traffic violations by taking a photo of the transgressor's vehicle with its license plate visible and an appropriate view of the situation. The system will locate automatically the location of the violation, assuming the device of the user has the GPS service enabled. When the report is received by the system, it's stored into a database and authorities that have jurisdiction over that city can access it to

consult its details and validate or invalidate it. The user will be able to track the status of their reports from the mobile app. SafeStreets, besides receiving reports from users, will also provide the possibility to consult its stored data with different levels of authorization for privacy safety reasons. A common citizen will be able to find the most frequent violations by location or time, but no private information of the violations will be provided. Whereas, the registered authorities will be able to perform the same queries and access to all the private information of the transgressors, for example, they will be able to find the most frequent transgressor in a certain area.

To help municipalities in identifying potentially unsafe areas, SafeStreets retrieves data about traffic accidents in their territories from their specific service. It crosses this data with its own stored data and computes possible interventions to areas most affected by violations, for example suggesting to add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking.

In the following we list the different kinds of events that can occur in the possible scenarios of interaction with SafeStreets. World events are phenomena occurring in the world that are not observable and not controllable by the SafeStreets' system. Shared events are phenomena that are either controlled by the world and observed by the SafeStreets's system or vice versa. Machine events are phenomena that are observed and controlled inside the SafeStreets' system, so they are not known to the world.

World events:

- Traffic violations
- Violation detection
- Street interventions and improvements
- Authorities interventions

Shared events:

- Violation report codification
- Data visualization and analysis
- Filtered data request

- Validated violations notification to authorities
- Intervention suggestion to municipality

Machine events:

- Database queries
- Possible interventions computation
- License plate recognition
- Meta-data completion

Phenomenon	Shared	Controlled by
Traffic violations	N	W
Violation detection	N	W
Street interventions and improvements	N	W
Authorities interventions	N	W
Violation report codification	Y	W
Data visualization and analysis	Y	M
Filtered data request	Y	W
Validated violations notification to authorities	Y	M
Intervention suggestion to municipality	Y	M
Database queries	N	M
Possible interventions computation	N	M
License plate recognition	N	M
Meta-data completion	N	M

Legend:

- Y := Yes, N := No.
- W := World, M := Machine.

1.3 Definitions, Acronyms, Abbreviations

Definitions

- **Citizen or common user:** a common citizen, without any public office, who uses the application to report traffic violations;
- **Authorities:** the public officials who certify the violations reported on the application;
- **Municipality:** the public institution that provides traffic violation data to the application, and may consult and analyze intervention suggestions from it;
- **Violation:** an event that does not conform with the traffic laws and that can be reported to authorities.
- **Intervention:** a possible public intervention produced by the application aimed at solving frequent violations in some areas of the city.

Acronyms

- GPS = Global Positioning System
- S2B = Software to Be
- RASD = Requirement Analysis and Verification Document
- PEC = Posta Elettronica Certificata
- API = Application Programming Interface
- AJAX = Asynchronous JavaScript and XML
- GDPR = General Data Protection Regulation
- OCR = Optical Character Recognition

Abbreviations

- G_n = n-th goal;
- D_n = n-th domain assumption;
- R_n = n-th requirement.

1.4 Revision history

- Initial version 1.0 (10/11/2019)

- Version 1.1 (03/12/2019)

- Add subsections to "Product perspective" section to improve readability.
- Add small note about user notification in "Report validation" state diagram.
- Improve "Assumptions, dependencies and constraints" section structure and add paragraphs about "Terms and Conditions" and app permissions, improving also external API dependencies section.
- Add availability assumptions to OCR and maps service, forgotten in previous version.
- Add introduction to kinds of events, violation definition and OCR acronym.
- Delete username from citizen "registration" use case and delete definition of supervisor.
- Better link between use cases and user interface mockups, update the tabs' names in use cases.
- Add and update mockups for municipality web app (used also in design document).
- Add possibility to load one or more photos in the "notify violation" use case.
- Update D5 to conform to final modeling of the software (also reflecting some changes in "Requirements" section).
- Improve "Product functions" section and better specify features explained in other parts of the documents, so that reader is aware from the beginning of that details.
- Small changes to "Generate report" state diagram and improvements to its description.
- Small improvements to "Suggestion evaluation" state diagram.
- Improve "user interfaces" section structure to identify more easily for which type of user the mockups are related.

- Revise goal logical derivation in "Requirements" section due to new changes.
- Change deployment target geographical area from Europe to Italy, because PEC was not used at European level and we could not verify the complete functionality of the system without faulty assumptions.
- Add missing rate attribute in Intervention class of class diagram.
- Correct typos.

1.5 Reference Documents

- Specification document: "SafeStreets Mandatory Project Assignment"
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications

1.6 Document Structure

The RASD document is composed by six chapters:

Introduction: first chapter of the document, it describes informally the main goals of the system and what problems they address. It also show an analysis of the events associated to the application domain, focusing in particular to shared phenomena (events that involve both users and the machine).

Overall Description: second chapter of the document, it focuses on describing accurately all the functionalities of the application, specifying also all the actors involved in the system. At the end of this section, assumptions on the world required for the correct behaviour of the system are clearly stated.

Specific Requirements: third chapter of the document. First section describes all the interfaces required by the system, subdividing them in subgroups. Next section explicates all possible scenarios and user cases of all types of user. Other parts of the chapters specify in detail all the requirements that the system should satisfy, associating them with domain assumptions and goals to verify the completeness of the domain analysis. Last section of the chapter explains other constraints and limitation that apply to the system.

Formal Analysis Using Alloy: fourth chapter of the document. Contains Alloy model analysis, used to verify the correctness of the model described in previous section.

Effort Spent: fifth chapter of the document, shows how much effort each member of the group has spent on the various chapters of the document.

References: final chapter, contains links to material, information or documentation related to the content discussed in the document.

Overall Description

2.1 Product perspective

2.1.1 Introduction

The core concept of the application is to provide to citizens of any city a reliable and fast way to report traffic violations they see on streets, trying to improve their awareness about the topic and addressing a social problem that could potentially harm the viability of the city or cause problems to people with disability. To achieve this goal, SafeStreets behaves as an intermediary between the citizens, who have the responsibility of reporting the violations, and the authorities, who can access the report and take care of them. The system allows an asynchronous interaction between parts, avoiding time loss for both the actors.

The municipalities can also have various benefits too when registered to the system, because SafeStreets provide them a way to integrate their data about violations with the ones stored in SafeStreets' databases, calculating then the most unsafe areas and providing suggestions about possible interventions to improve them.

2.1.2 Class diagram

The following class diagrams shows the model of the application domain and its main actors and entities. Note that this class diagram is really high-level, its main purpose it's only to display the defined entities involved in the system, their principal attributes and the relations between them.

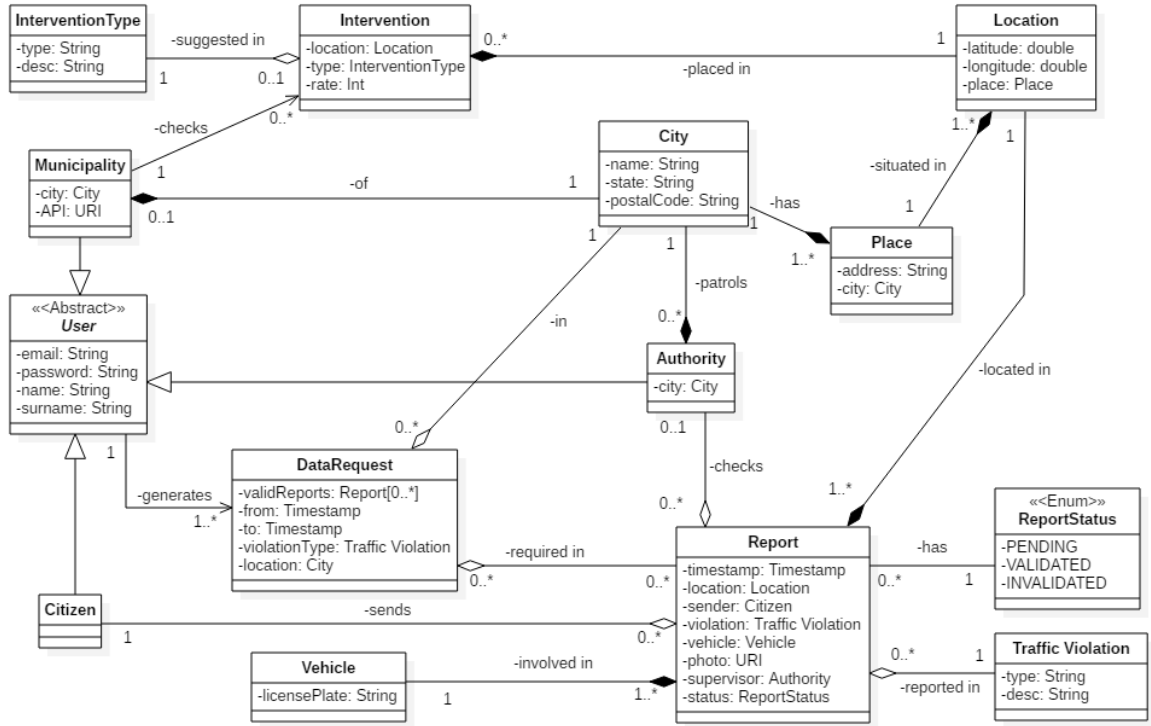


Figure 2.1: Class diagram

User abstract class represents the common structure and attributes shared between all types of accounts that reflect the role of the customers of the system. Citizens' main role is to generate reports to populate the databases of violations, while the authorities can check these reports and update their status from pending (default initial status) to validated or invalidated, based on the fact if the violation has been certified or not. Municipality accounts instead can visualize the list of interventions to improve the situation of detected unsafe areas and also rate the suggestions to give a feedback to the system. All account types can submit data requests for visualizing aggregated information relevant to them, but only authorities can visualize single reports (actually a citizen can visualize single reports too, but only the ones made by himself).

Another important aspect to underline is that Municipalities and Authorities have also a city attribute, that identify the city on which they have jurisdiction. This allow the system to provide them respectively a list of interventions and reports that are situated in their area of interest.

Report entity could be considered central in the entire system because it contains

various references to many other entities of the model and its essential to achieve the goals of the application. Reports have attributes that specify their type, the date and hour of the violation, and their actual status. Other fields are references to other class objects, like the vehicle involved, the location of the violation, the citizen who submits the report and the authority who has supervised its certification. It's important to note that vehicle and location entities have a strict composition relation with the report entities, this means that if no more reports with that vehicle or location are stored in the system, also the vehicle and location entities are eliminated from the system.

Lastly, the location of a violation or intervention is handled by the Location-Place-City correlated entities. The separation of this entities through composition allows a higher level of granularity and maintainability. Composition is used to emphasize the fact that if no more places of a city are stored in the database, the city is also deleted from it, same thing for places if no more location coordinates of the place are needed.

2.1.3 State diagrams

Next section provides multiple state diagrams, useful to better understand the main actions that the different customers can perform with the application.

Generate report (citizens)

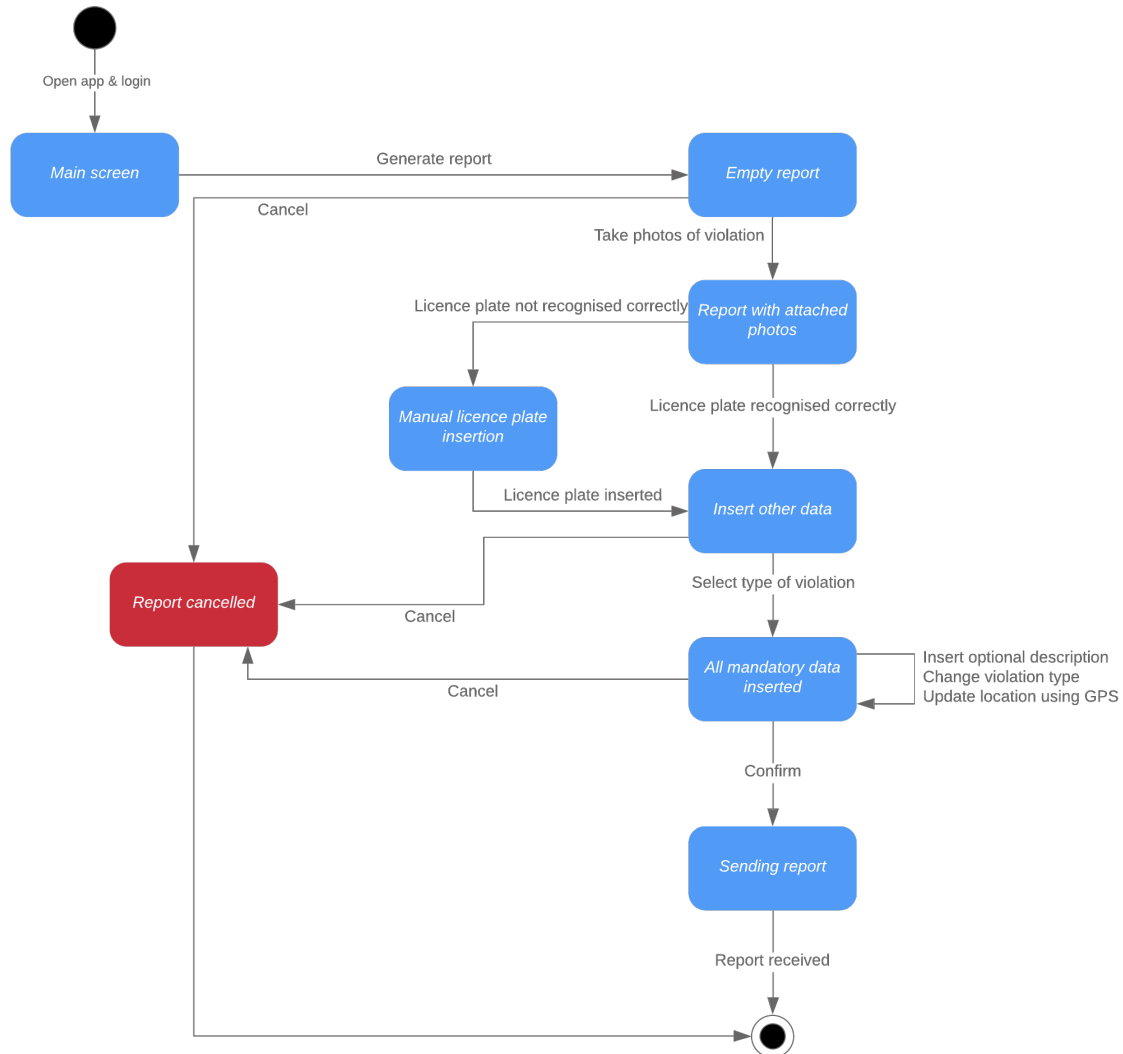


Figure 2.2: Generate report state diagram

This state diagram specifies all the operations needed to complete and submit a report. After selecting the "generate report" tab, the citizen is asked to take at least one photo of the violation, with the license plate of the transgressor's vehicle clearly visible in the first one. After confirmation, the first photo of the violation is uploaded to SafeStreets servers and used in the OCR service to recognize automatically the license plate. The result is then sent back to the user, that needs to confirm if the license plate has been recognized correctly, if not the user can insert it manually. After that, the citizen needs to select the violation type and check also that location field has been correctly populated by the GPS. If detected location is wrong, user can click a button to try

again to get the correct location from the GPS. The description field is completely optional and not required to confirm and submit the report.

Reports validation (authorities)

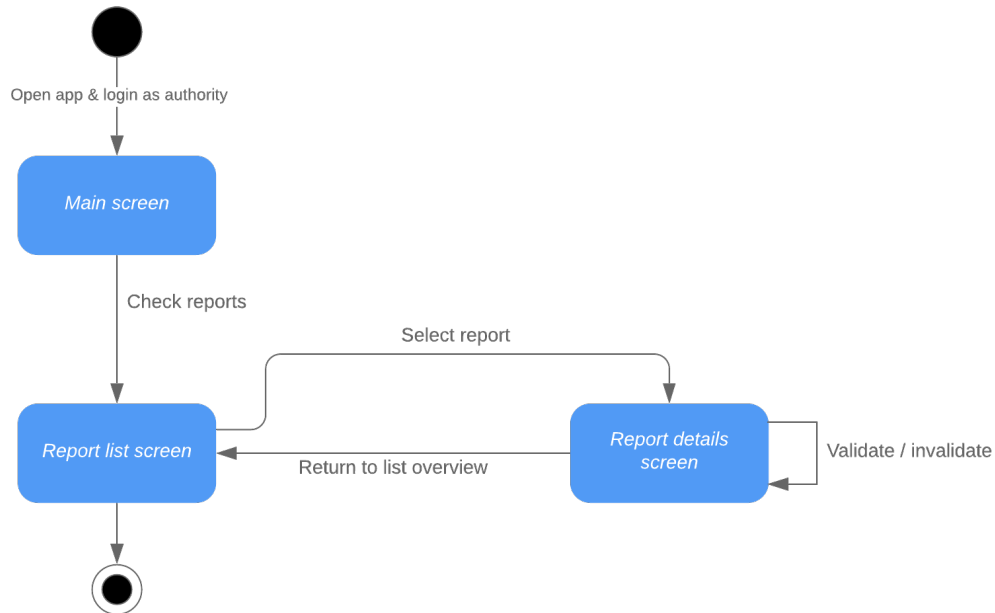


Figure 2.3: Reports validation state diagram

This diagram instead focuses on the report supervision performed by the authorities. In appropriate report section, an authority can visualize the whole list of violations occurred in their area of competence and visualize a report at a time to validate or invalidate them. Every time a report is validated or invalidated, the citizen who submitted the report is notified on SafeStreets app about the status change.

Suggestions evaluation (municipality)

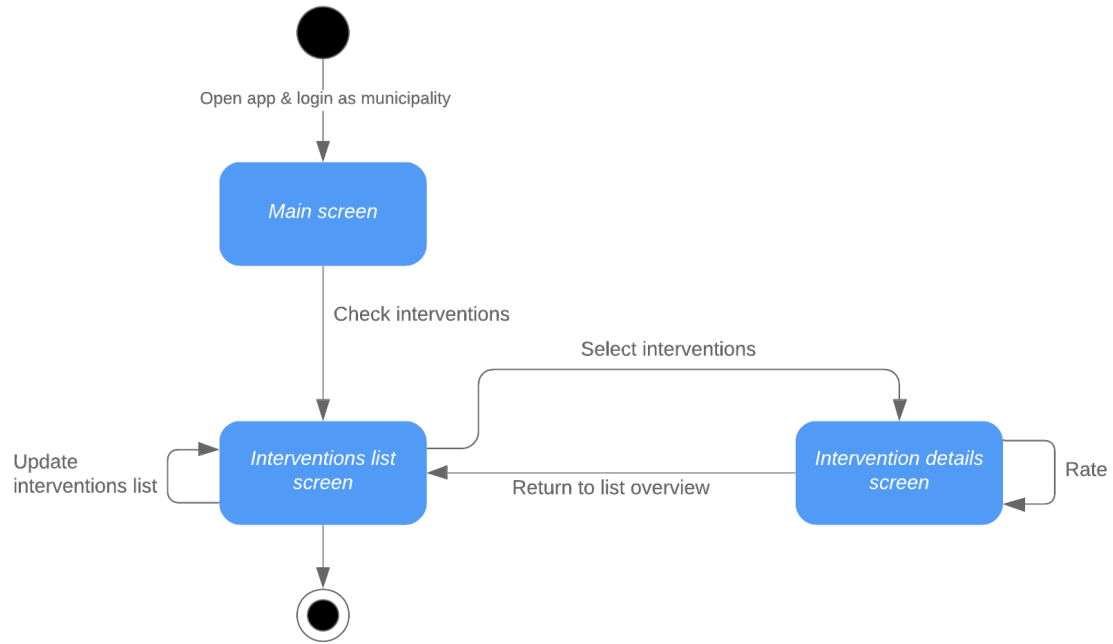


Figure 2.4: Suggestions evaluation state diagram

The flow of states that identifies the process of a municipality for checking the suggested interventions is pretty similar to the precedent. The municipalities can check, from the appropriate tab, a list of interventions relative to their city (updating it eventually) and then consult the interventions individually. They can also rate them to give a feedback to the system to improve future suggestions.

Data analysis (any user)

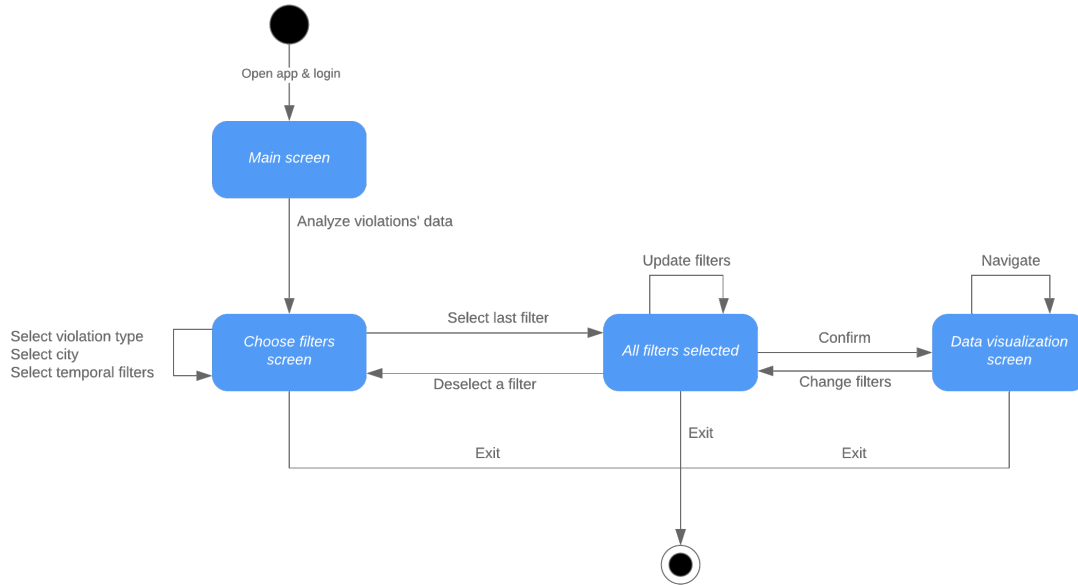


Figure 2.5: Data analysis state diagram

The final state diagram describes the process done by any type of user to analyze aggregate data of interest in a city map. In the appropriate tab, the user is asked to insert spatial and temporal filters, also specifying the type of violation (left to "all" value if not interested in a more specific research). Then the user can confirm to check a density map of the violations and a small report, containing auxiliary text data. At any time he can update the filters or end the operation.

2.2 Product functions

In the following sections the most important functions of the application are presented. The service of intervention suggestions to the municipalities is built on top of the storage data obtained by the reports from the citizens and optionally integrated with additional violation data provided by municipalities themselves.

Violation Reporting

This product function is the main feature of the system. After the citizen has signed up with an e-mail and a password, the user will be able to take some photos of the violation and upload them on the application, providing its violation type and optionally a description. The coordinates of the violation will be automatically determined using the GPS service of the user's device and filled in the report. After that, the user will be able to check the status of his/her report and will receive a notification when authorities validate or invalidate it. It's up to authorities to establish a proper way to validate or invalidate a report, as SafeStreets is only an intermediary between them and citizens.

Data Analysis

The system also allows the registered users to mine data from its data storage, using different levels of authorization based on account type in order to protect the privacy of the individuals and avoid a misuse of the data from the common citizens. The pool of data mined will be restricted only to report validated by authorities to provide an accurate analysis of the real data about violations, avoiding contamination from erroneous or fraudulent reports.

Intervention Suggestion generation

This product function is offered when a municipality provides its own data about traffic accidents on its territory. After the municipality has signed up and optionally provided the API to access its data, SafeStreets will cross its own data gathered from the validated reports, find the possible interventions for the most afflicted areas of the territory and notify the municipality. Then, the municipality can log in and generate a list of suggested interventions, based on reports up to that time.

Because no standards are defined about violations' data cataloging, it would be too complex and time consuming for SafeStreets to build an interface to cross data with each municipality registered to the system that use a different data structure for storing violations. Because of this consideration, for using this functionality the municipality is responsible for complying to the conventions used by SafeStreets, that will be provided and accurately explained in the website page relative to the service.

2.3 User characteristics

The actors of the system are the following:

1. **Citizen:** a common user without any public office, who registers to report violations and mine traffic violation data from the application data storage in order to identify and avoid the most unsafe areas and streets of a city.
2. **Authorities:** the public officials who will receive the reports from SafeStreets and have the possibility to validate or invalidate them. They could also mine data with a direct access to private information (license plate) of the transgressors.
3. **Municipality:** an institution that provides traffic violation data to SafeStreets and expects to get intervention suggestions in return.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

- D1. Email addresses are unique.
- D2. Users report violations when they detect them.
- D3. Municipality service about accidents is always available, when provided.
- D4. Municipality accidents' data is always accurate.
- D5. Authorities must validate or invalidate reports after they have supervised them.
- D6. The GPS sensor of a user's device has an error of at most 5m from the real position.
- D7. The internet connection is always available when the user interacts with the system.
- D8. Every city has authority institutions.
- D9. Municipality and authorities of each city have a PEC.
- D10. Italy's PEC service has an API to get the data of an institution from a PEC.

D11. License plates are unique.

D12. The first photo taken by a citizen reporting a violation must clearly show the license plate of the involved vehicle.

D13. The OCR service is always available.

D14. The maps service is always available.

We assumed D9 because an Italian law (see link) states that any municipality must have a PEC address.

2.4.2 Dependencies

The system employs external APIs to focus on its main business of providing users the possibility to report traffic violations, in particular the system will use an optical character recognition (OCR) service and a generic map API service. The OCR service will be used to recognize the license plate from the first photo of the report, while map's API will be used to allow the system to recognize places and addresses from latitude and longitude coordinates of the reports and to provide interactive maps to users in which visualize the data requested in some topic of data analysis functionality.

2.4.3 Constraints

Any user who has a device with internet connection is able to access the services offered by SafeStreets (e.g. reporting violation with images and mining data about the previously reported violations). GPS sensor is required to report a violation.

During registration, every user is asked to read the "Terms and Conditions" document, provided with an apposite link, and to accept them by ticking the checkbox, otherwise he won't be able to complete the registration and use the app.

When a citizen starts the process for reporting a violation for the first time, he will be asked to give to SafeStreets app the access to camera and localization. Camera is required to take photos of a violation, while localization is required to take the location of the violation from GPS. Both photo and location coordinates are required in a report, therefore both permissions must be accepted from the citizen to be able to generate a report.

Official apps supported mobile operating systems are Android (4.4 and above) and iOS (9.0 and above). The website can be accessed with any modern browser with AJAX support.

The additional service of suggesting possible interventions to the most unsafe areas doesn't require the municipality to provide an API to let SafeStreets access its data and augment it with its own for intervention analysis, but it's recommended to improve drastically the results given by the algorithm.

The S2B is assumed to be distributed in Italy; it needs only an email for the registration of common citizens, and a public certified email(PEC) for the authorities. Also it must be compliant with the GDPR normative for the privacy.

Specific Requirements

3.1 External User Requirements

3.1.1 User Interfaces

In this section a basic idea of how the user interface should look like when shown. It presents the basic login and signup tabs, the main tab showing the usable services, and a specific tab for each distinct service. The mockups are very similar for different kinds of users, only the services provided differ. For instance, 3.5 provide the services that only the citizens can use, whereas 3.10 shows the additional services that the authorities and municipality users can use.

Mobile app common interfaces

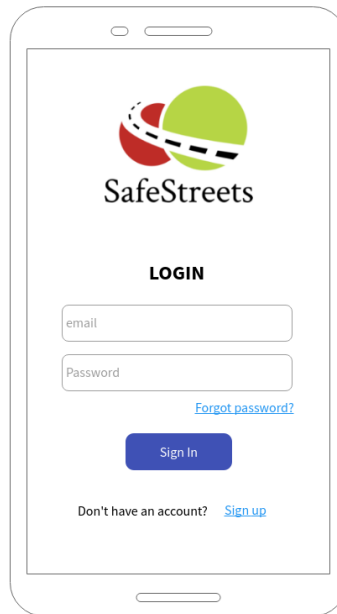


Figure 3.1: Login tab.

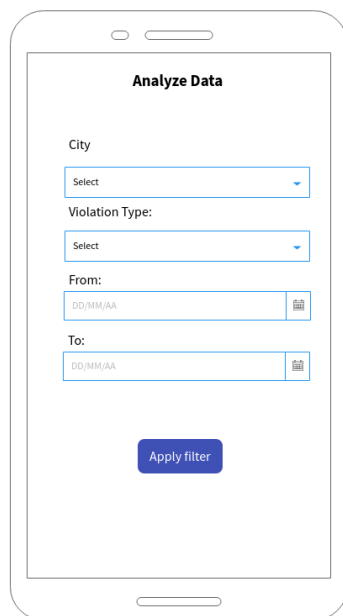


Figure 3.2: Analyze data.

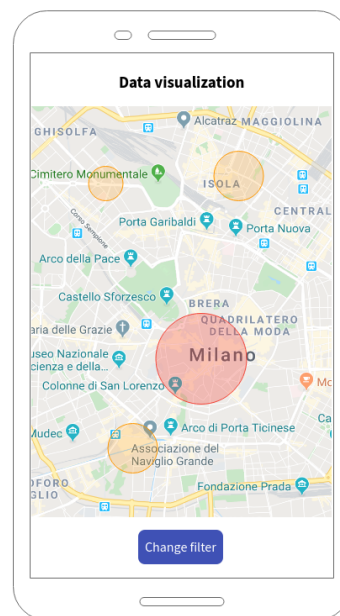
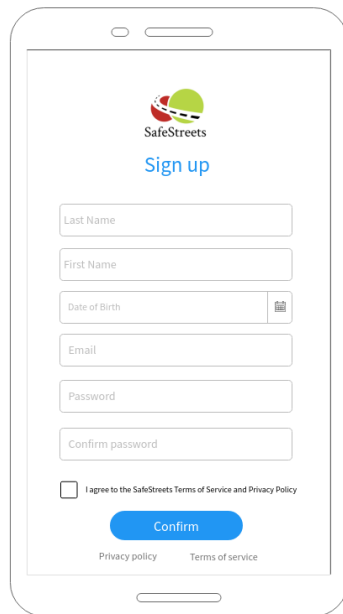
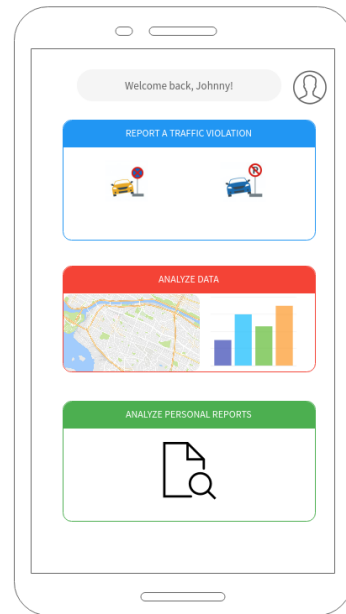


Figure 3.3: Data visualization.

Citizen interfaces



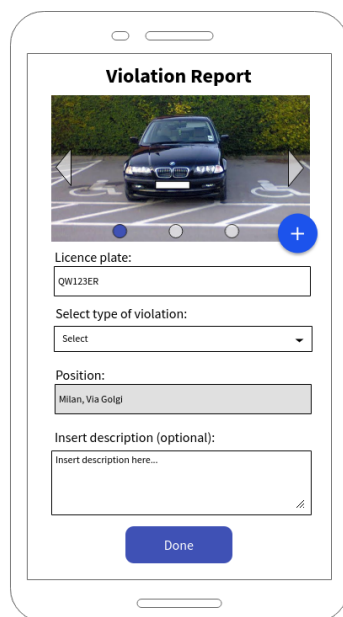
The 'Sign up' screen for SafeStreets features the app's logo at the top. Below it, there are input fields for 'Last Name', 'First Name', 'Date of Birth' (with a calendar icon), 'Email', 'Password', and 'Confirm password'. A checkbox for 'I agree to the SafeStreets Terms of Service and Privacy Policy' is located above a blue 'Confirm' button. At the bottom, there are links for 'Privacy policy' and 'Terms of service'.



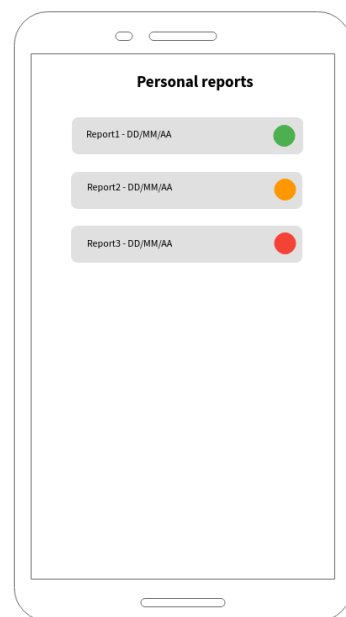
The 'Citizen main tab' screen displays a welcome message 'Welcome back, Johnny!' with a user profile icon. It features three main sections: 'REPORT A TRAFFIC VIOLATION' with icons for a car and a traffic sign, 'ANALYZE DATA' with a map and a bar chart, and 'ANALYZE PERSONAL REPORTS' with a document icon.

Figure 3.4: Citizen signup tab.

Figure 3.5: Citizen main tab.



The 'Violation Report' screen shows a photo of a car in a parking space. Below the photo is a blue '+' button. The form includes fields for 'Licence plate:' (with 'QW123ER' entered), 'Select type of violation:' (a dropdown menu), 'Position:' (with 'Milan, Via Golgi' entered), and 'Insert description (optional):' (a text area). A blue 'Done' button is at the bottom.



The 'Personal reports' screen lists three reports: 'Report1 - DD/MM/AA' with a green status indicator, 'Report2 - DD/MM/AA' with an orange status indicator, and 'Report3 - DD/MM/AA' with a red status indicator.

Figure 3.6: Report tab.

Figure 3.7: Personal reports.

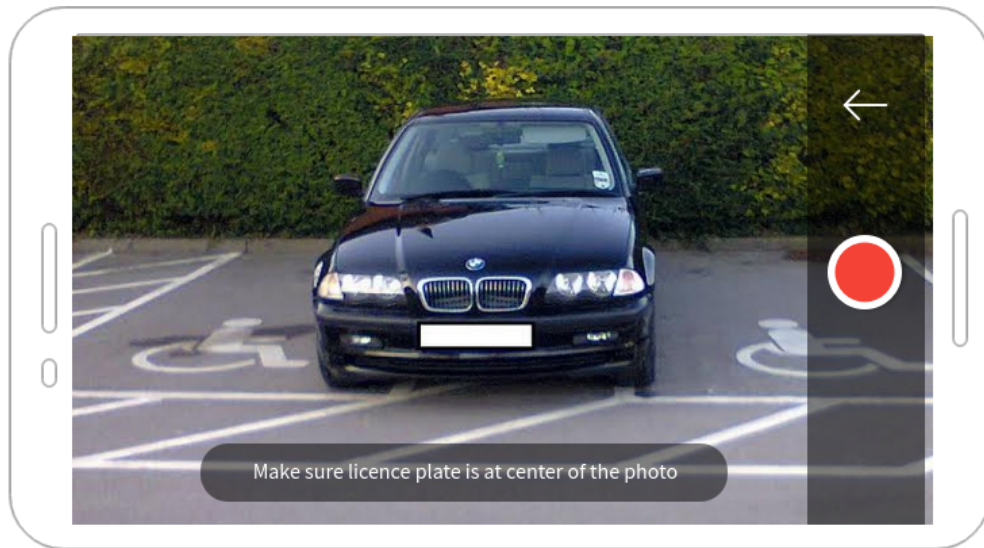


Figure 3.8: Violation photo.

Authorities interfaces

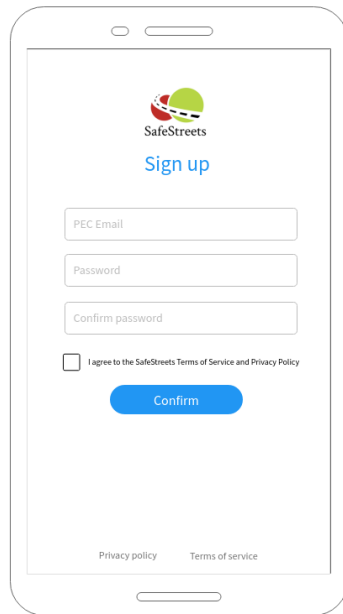


Figure 3.9: Authorities signup tab.

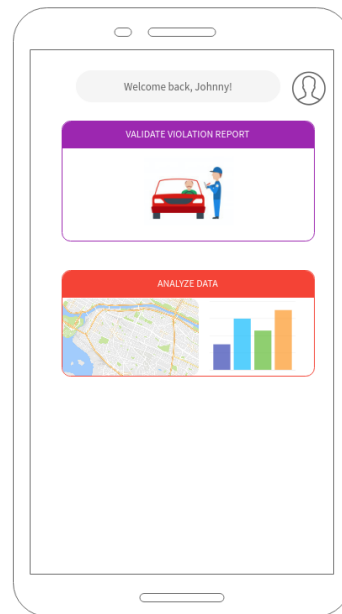


Figure 3.10: Authorities main tab.

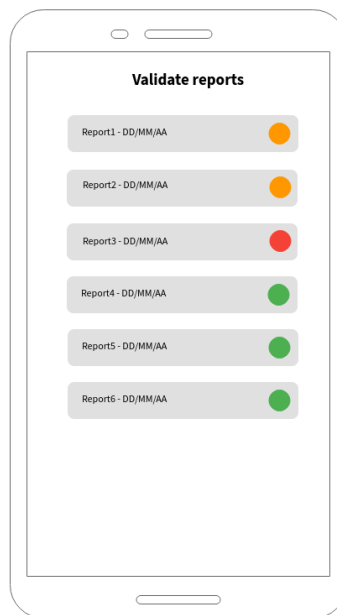


Figure 3.11: Validate reports.

Municipality interfaces (Web app)

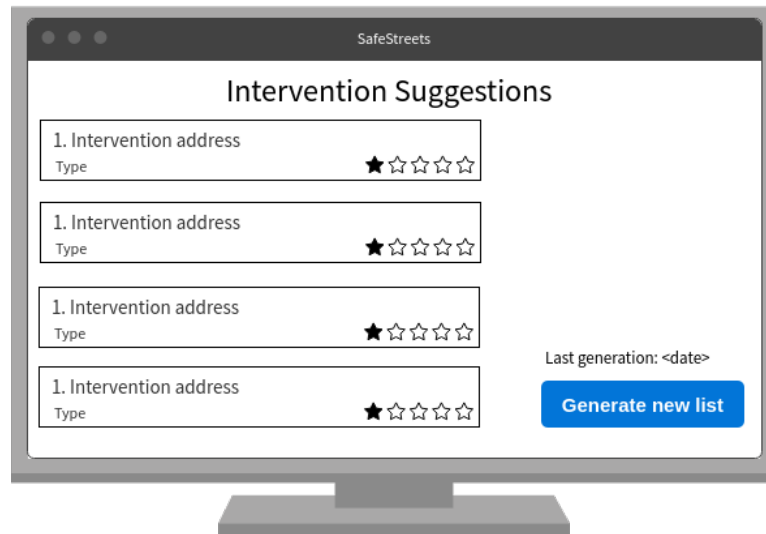


Figure 3.12: Intervention suggestions.

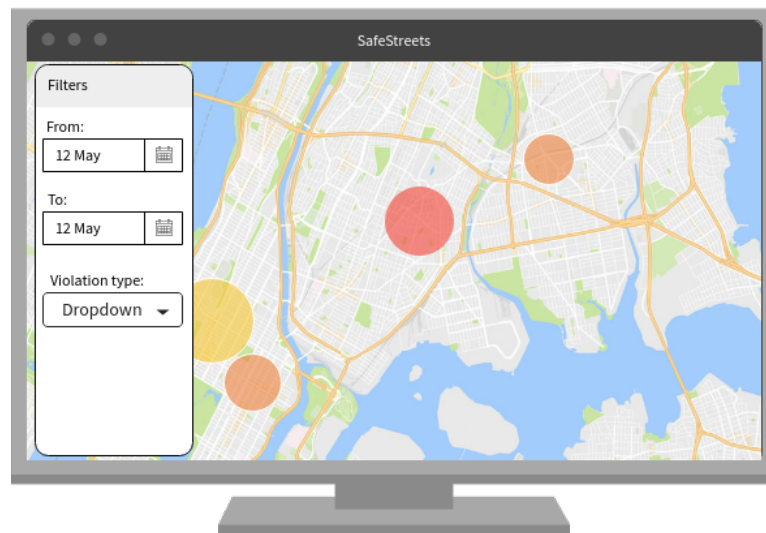


Figure 3.13: Municipality data analysis.

3.1.2 Hardware Interfaces

The system and its clients don't require any hardware interface to achieve their functionalities.

3.1.3 Software Interfaces

The system doesn't actually provide any API interface to work with other software, because all data is visible and can be analyzed within the official apps and website. API could be developed in future stages of development if partners and customers find the service valuable.

3.1.4 Communication Interfaces

No communication interfaces are needed by the system.

3.2 Functional Requirements

3.2.1 Citizen

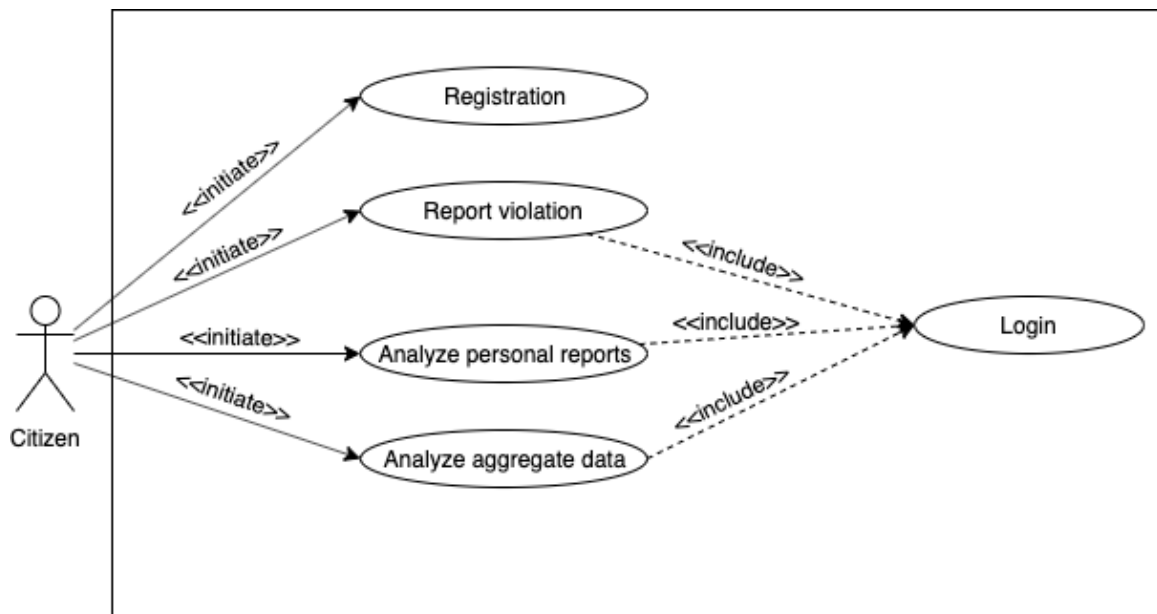
Scenarios

1. Johnny is a model citizen of Milan and he is really caring about parking violations, because they are really frequent in his neighbourhood. When he sees a violation, he opens the SafeStreets app on his smartphone and reports it with just few clicks. All he needs to do is to take a photo of the violation where the license plate is clearly visible and confirm if the system detected it correctly or not, if not he inserts the license plate manually in the field that appears after the confirmation. Position is automatically detected from the GPS because he allowed the app to access it. Finally, to send the report to SafeStreets he just needs to confirm by clicking the done button.
2. Giuseppe is a righteous citizen and recently has heard about SafeStreets but never used it. One day, he finds a double-parked car right in front of his, so he is not able to get his car out. After waiting some minutes hoping for the owner of the double-parked car to arrive without success, Giuseppe remembers about SafeStreets and decides to use it for the first time. He registers and logs

in the app, then he uploads a photo of the car of the transgressor and sends the report. While still waiting, he checks whether his report has been validated by some authorities and sees that after 10 minutes it has been validated, so he now knows that the authorities will take action about it.

3. George is a tourist who has just arrived in Rome, and plans to stay there for a week. When he goes out touring the city with the car he rented for the entire week, he uses SafeStreets to check where the most violation afflicted areas of the city are to avoid areas which can lead to a terrible experience in parking his vehicle. So he opens the SafeStreets app and enters the analyze data tab, selecting Rome as the city and a year time as time interval. George now knows where he should park his car and can have a great experience in Rome.

Use cases



Name	Registration
Actors	Citizen
Entry Condition	The citizen opens the app and has no valid account.
Event flow	<ol style="list-style-type: none"> 1. The citizen inserts his last name. 2. The citizen inserts his first name. 3. The citizen inserts his birthday. 4. The citizen inserts his email. 5. The citizen inserts the desired password. 6. The citizen confirms his password. 7. The citizen agrees to the use of his personal data and of his submitted violation reports for analysis purposes.
Exit condition	The registration is successful and the citizen is redirected to login form.
Exception	Email is already in use or confirm password field content diverge from password, in this case the citizen is alerted with a proper message on screen and asked to correct the data.

Name	Login
Actors	Citizen
Entry Condition	The citizen opens the app.
Event flow	<ol style="list-style-type: none"> 1. The citizen inserts his email. 2. The citizen inserts his password. 3. The citizen press the login button.
Exit condition	The citizen is authenticated and login is successful.
Exception	Email or password are invalid, "invalid credentials" message is displayed and the citizen needs to insert credentials again.

Name	Report violation
Actors	Citizen
Entry Condition	Violation detection.
Event flow	<ol style="list-style-type: none"> 1. The citizen selects the "report a traffic violation" tab if not already selected. 2. The citizen takes and uploads one or more photos of the violation. 3. The citizen confirms that the license plate is correctly recognized by the app, if not he/she inserts it manually in the appropriate field. 4. The citizen selects the type of violation. 5. The citizen optionally inserts a short description of the violation in the description field. 6. The citizen confirms the violation report clicking on the "Done" button. 7. The system stores the information and completes it with suitable metadata. 8. The system sends a notification to the citizen about the success of the operation.
Exit condition	The violation report is correctly stored.
Exception	the system detects missing information and rejects the report, then asks the citizen for missing data.

Name	Analyze aggregate data
Actors	Citizen
Entry Condition	The citizen wants to know some information about the reported violations.
Event flow	<ol style="list-style-type: none"> 1. The citizen selects the "analyze data" tab. 2. The citizen selects the topic he/she is interested about. 3. The citizen may select an appropriate filter for his search. 4. The citizen selects the data of interest. 5. The system provides the data to be visualized according to the actor's authorization level. 6. The citizen visualizes the data.
Exit condition	The Actor finishes to analyze the data.

Name	Check personal reports
Actors	Citizen
Entry Condition	The citizen opens the app, desiring to check the reports he sent in the past.
Event flow	<ol style="list-style-type: none"> 1. The citizen selects the "analyze personal reports" tab. 2. The system displays a list of all reports done by the citizen. 3. The citizen clicks on a report to see its details. 4. The citizen clicks the return button, going back to the list of reports. 5. The citizen can repeat the last two steps as many times as he pleases, to check other reports.
Exit condition	The citizen closes the reports list after he doesn't need to visualize them anymore.

Sequence diagrams

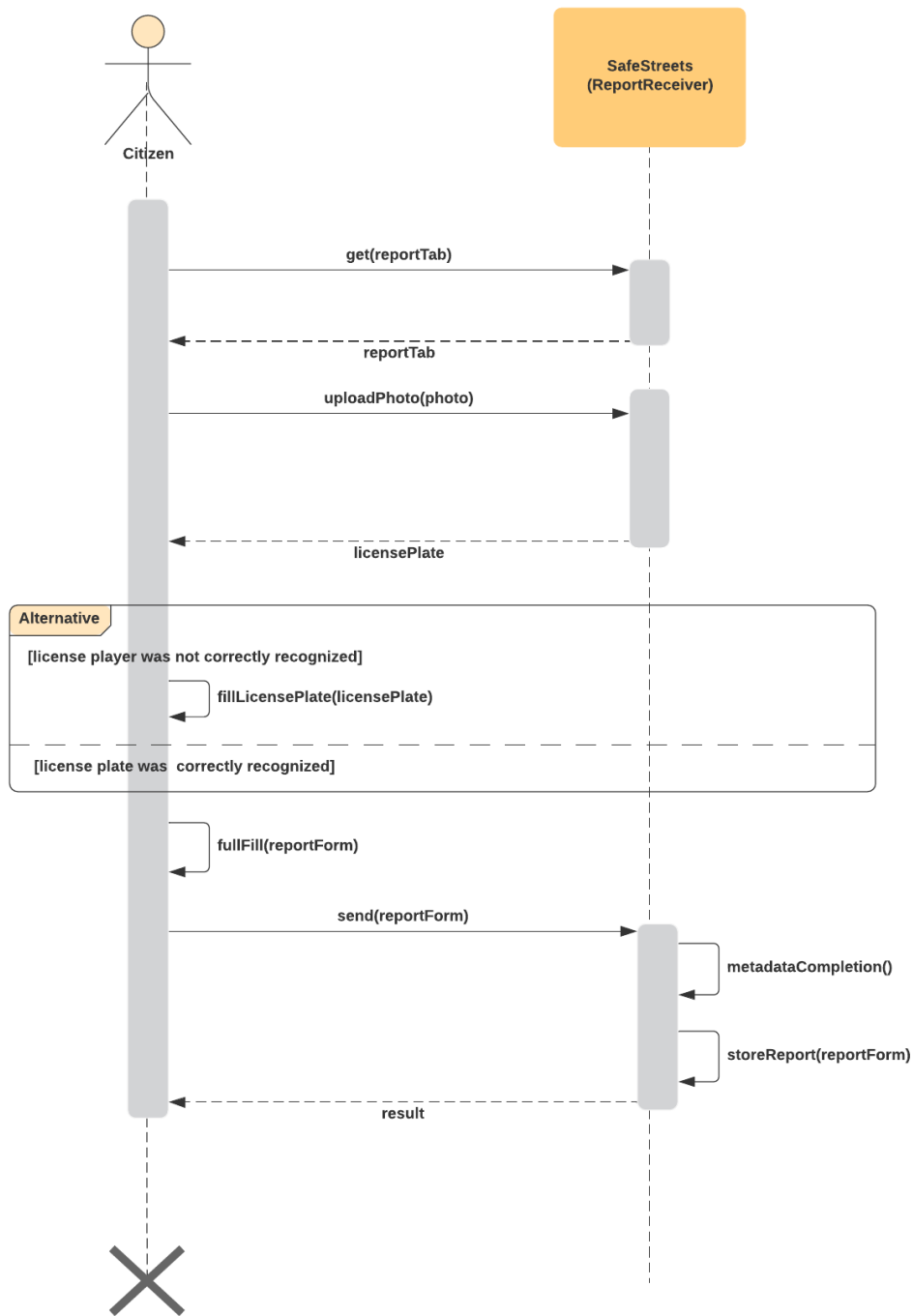


Figure 3.14: Report violation use case.

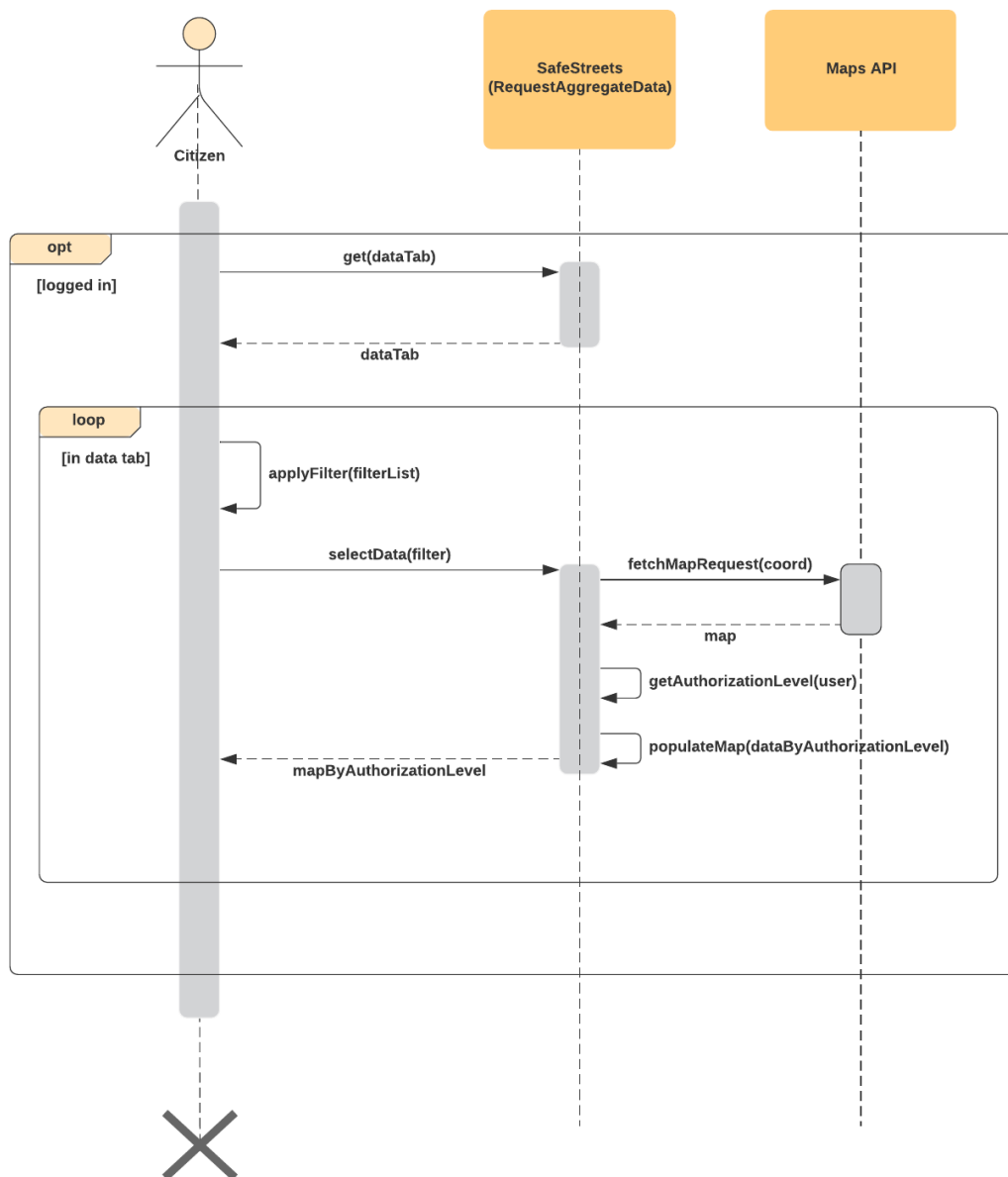


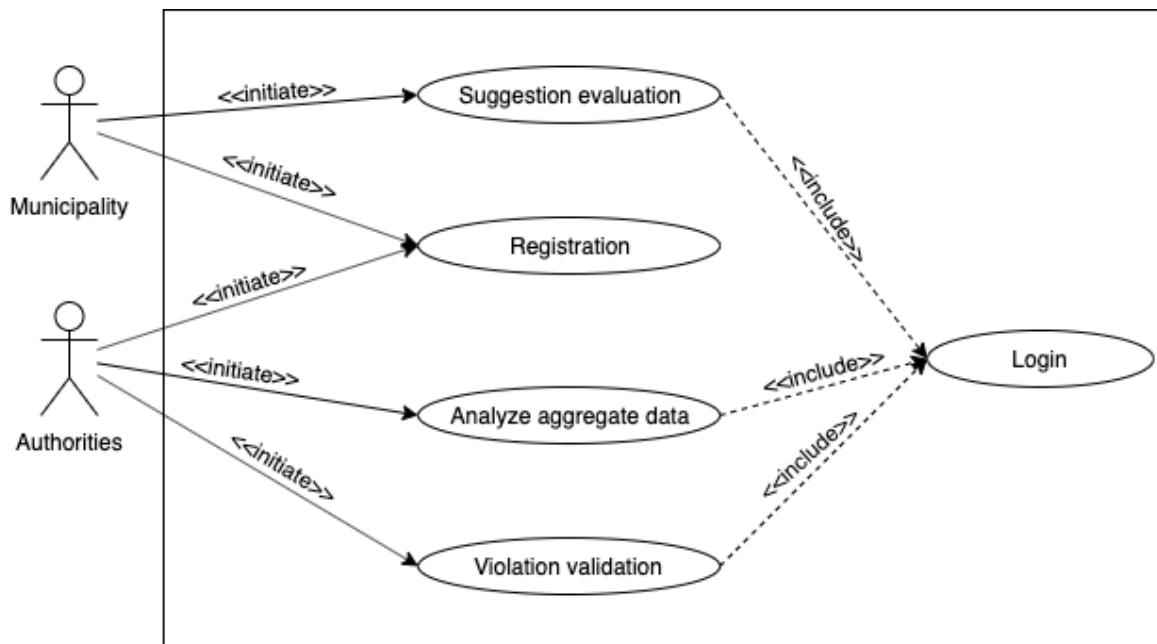
Figure 3.15: Analyze aggregate data use case.

3.2.2 Third Party

Scenarios

1. Henry is a member of the local police of the city of Casalpusterlengo, a municipality that has a partnership with SafeStreets. Every month he wants to check the most unsafe areas in the city, so that he could better patrol the city. This could be done by opening the SafeStreets app and selecting "analyze" tab, then selecting the violation density map option and using the previous month as temporal filter.
2. Lucia, a worker of Bergamo's municipality, is interested in studying possible solutions to recurrent parking violations along Via Milano. Bergamo's municipality is regularly registered to SafeStreets system, so she decides to try to check the list of interventions suggested from SafeStreets' algorithm. After she logs in the app, she just needs to check the suggestions tab to search for valid ones in that area, then she can also rate them to improve the suggestion system.

Use cases



Name	Registration
Actors	Authorities or municipalities
Entry Condition	An authority or municipality opens the app and has no valid account
Event flow	<ol style="list-style-type: none"> 1. The actor inserts his official public certified email (PEC) given for his public office. 2. The actor inserts the desired password. 3. The actor confirms his password. 4. The system checks the authenticity of the the email using its domain.
Exit condition	The email is authenticated and the actor is redirected to login form.
Exception	Email is already in use, or is not a PEC email, or confirm password field content diverge from password, in this case the actor is alerted with a proper message on screen and asked to correct the data.

Name	Login
Actors	Authorities and municipality
Entry Condition	The actor opens the app.
Event flow	<ol style="list-style-type: none"> 1. The actor inserts his email. 2. The actor inserts his password. 3. The actor press the login button.
Exit condition	The actor is authenticated and login is successful.
Exception	Email or password are invalid, "invalid credentials" message is displayed and the actor needs to insert credentials again.

Name	Analyze aggregate data
Actors	Authorities
Entry Condition	The authorities wants to know some information about the reported violations.
Event flow	<ol style="list-style-type: none"> 1. The authorities selects the "analyze data" tab. 2. The authorities selects the topic he/she is interested about. 3. The authorities may select an appropriate filter for his search. 4. The authorities selects the data of interest. 5. The system provides the data to be visualized according to the actor's authorization level. 6. The authorities visualizes the data.
Exit condition	The Actor finishes to analyze the data.

Name	Violation validation
Actors	Authorities
Entry Condition	The authorities is in the "validate violation report" tab and selects a report to be validated.
Event flow	<ol style="list-style-type: none"> 1. The authorities checks whether the reported violation is an actual traffic violation. 2. The authorities sends a result response (rejected or accepted) to SafeStreets. 3. The system updates the report status of the violation for the reporting citizen. 4. The system augments its data with the reported violation if and only if it was certified by the authorities.
Exit condition	The violation report and the system violation data are updated.

Name	Suggestion evaluation
Actors	Municipality
Entry Condition	A municipality is registered on the application and wants to evaluate possible suggested interventions.
Event flow	<ol style="list-style-type: none"> 1. The municipality selects the "evaluate intervention" tab. 2. The municipality generate or updates the list of suggestions by clicking the apposite button (optional if list was already generated in the past and the user doesn't want to update it). 3. The municipality reads and analyzes the list of suggestions. 4. The municipality can rate one or more interventions from the list.
Exit condition	The municipality closes the intervention suggestion tab.

Sequence diagrams

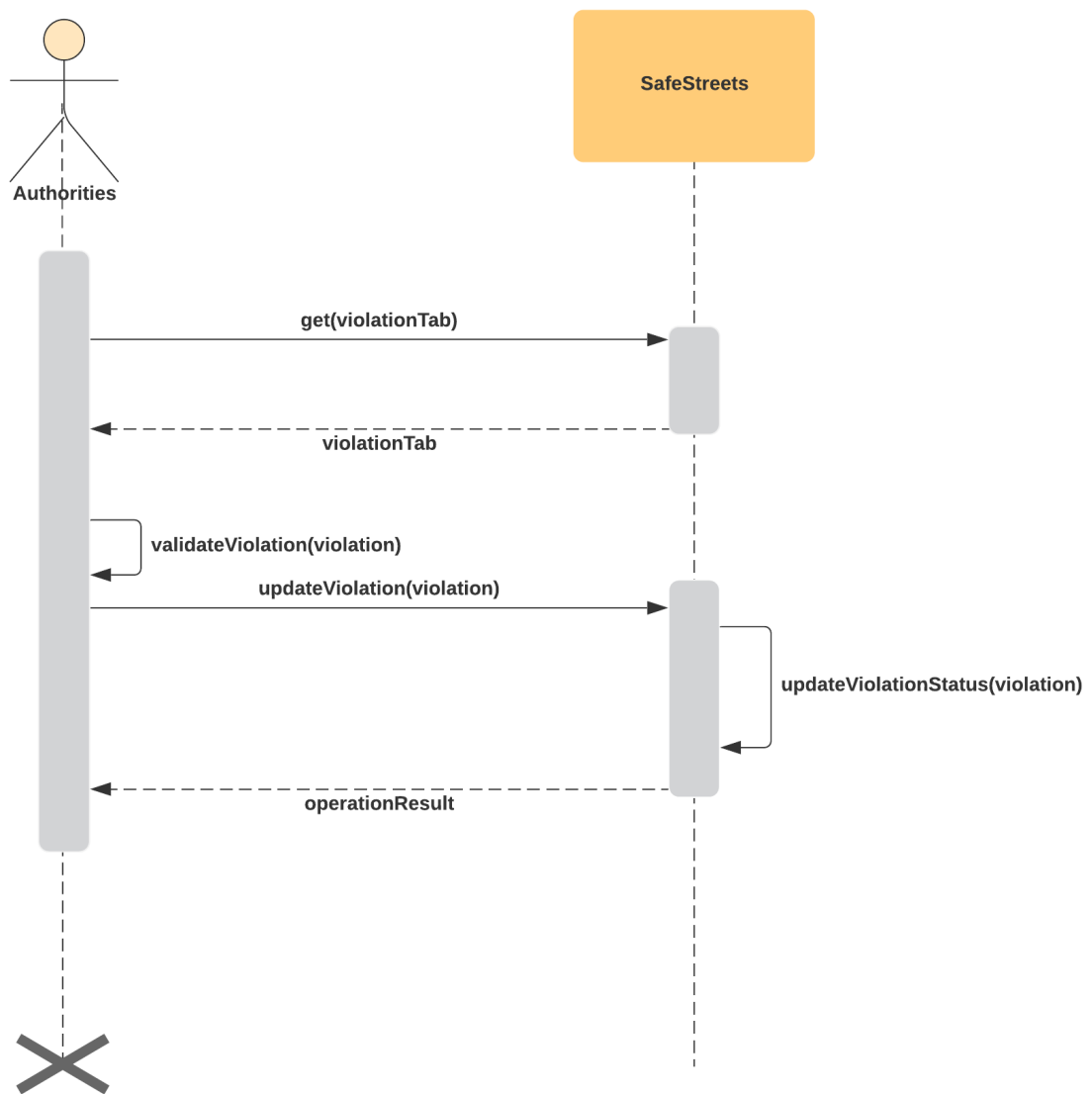


Figure 3.16: Violation validation use case.

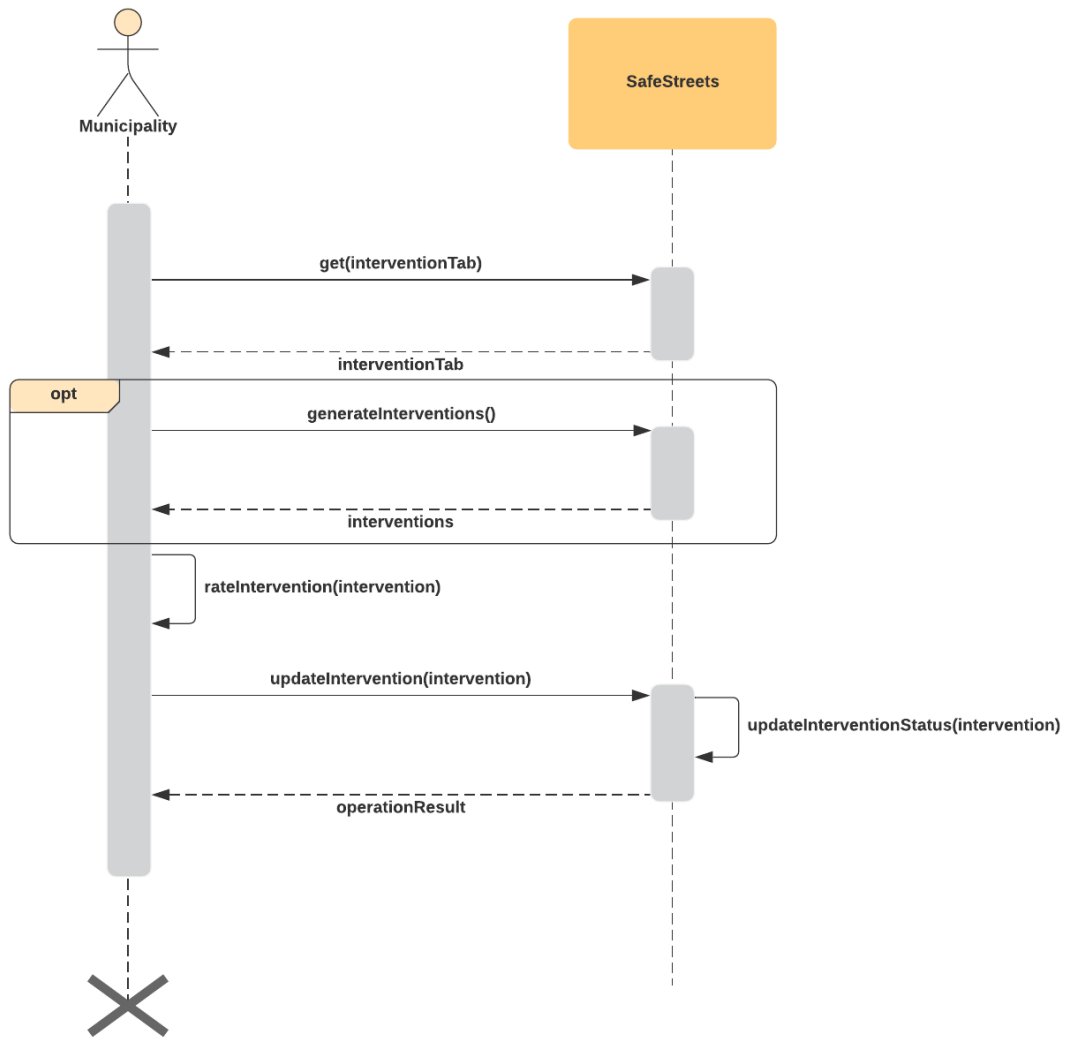


Figure 3.17: Suggestion evaluation use case.

3.2.3 Requirements

In this section project requirements are presented and relationships between goals, domain assumptions and requirements are clearly stated, to provide a logical representation of the model analysis.

Requirements:

- R1. When a traffic violation is reported, the system must allow the authorities of the city where the violation happened to verify it.

- R2. Reports about traffic violations need to be stored in a database, also associating them to the user profile who sent them.
- R3. The system needs to integrate reported traffic violations with data from the municipality when provided and analyze all the data with an algorithm to find possible interventions to address the detected main cause of the violations.
- R4. The system must be able to generate a list of possible interventions when a municipality requires it for evaluation, based on stored reports.
- R5. Traffic violations data must be available for public consultation with different levels of visibility, according to user's level of authorization.
 - R5.1. The system should allow a normal user to access only to aggregated data, without the license plate of the reported vehicle or the personal info of the user who reported the violation.
 - R5.2. The system should allow the authorities to access the data without any limitation.
- R6. The system must ascertain the identity of a public institution that want to register to the system from its PEC, completing the account with the necessary data.
- R7. The system must receive correctly the reports about the various traffic violations from the users, discarding reports with incomplete data.
- R8. The system must be able to recognize the license plate of the vehicle involved in the violation from the first photo of the report.

Goal Logical Derivation:

- G1. Report certified traffic violations to authorities of city where the violations have happened.
 - R1. When a traffic violation is reported, the system must allow the authorities of the city where the violation happened to verify it.
 - R7. The system must receive correctly the reports about the various traffic violations from the users, discarding reports with incomplete data.

- D2. Users report violations when they detect them.
- D5. Authorities must validate or invalidate reports after they have supervised them.
- D8. Every city has authority institutions.
- G2. Store data about violations and cross it with municipality's one when an API to its data is provided.
 - R2. Reports about traffic violations need to be stored in a database, also associating them to the user profile who sent them.
 - R3. The system needs to integrate reported traffic violations with data from the municipality when provided and analyze all the data with an algorithm to find possible interventions to address the detected main cause of the violations.
 - D3. Municipality service about accidents is always available, when provided.
- G3. Allow municipalities to generate a list of possible reasonable interventions to resolve the violations detected in most unsafe areas.
 - R2. Reports about traffic violations need to be stored in a database, also associating them to the user profile who sent them.
 - R4. The system must be able to generate a list of possible interventions when a municipality requires it for evaluation, based on stored reports.
 - D3. Municipality service about accidents is always available, when provided.
 - D4. Municipality accidents' data is always accurate.
- G4. Recognize public institutions automatically during registration to setup correctly accounts of municipalities and authorities.
 - R6. The system must ascertain the identity of a public institution that want to register to the system from its PEC, completing the account with the necessary data.
 - D9. Municipality and authorities of each city have a PEC.

D10. Italy's PEC service has an API to get the data of an institution from a PEC.

G5. Provide a synthesis of all the violations' data to users, based on their authorization level.

R5. Traffic violations data must be available for public consultation with different levels of visibility, according to user's level of authorization.

D5. Authorities must validate or invalidate reports after they have supervised them.

D7. The internet connection is always available when the user interacts with the system.

D14. The maps service is always available.

G5.1. Normal users can access only to aggregated data, without seeing any private info about a reported violation, like the license plate or the personal info of the user who reported the violation.

R5.1. The system should allow a normal user to access only to aggregated data, without the license plate of the reported vehicle or the personal info of the user who reported the violation.

D5. Authorities must validate or invalidate reports after they have supervised them.

D7. The internet connection is always available when the user interacts with the system.

D14. The maps service is always available.

G5.2. Authenticated authorities can access to all violations' data, as aggregated data or singular reports.

R5.2. The system should allow the authorities to access the data without any limitation.

D5. Authorities must validate or invalidate reports after they have supervised them.

- D7. The internet connection is always available when the user interacts with the system.
- D14. The maps service is always available.
- G6. Allow users to send a report of a possible traffic violation, identifying clearly the transgressor.
 - R7. The system must receive correctly the reports about the various traffic violations from the users, discarding reports with incomplete data.
 - R8. The system must be able to recognize the license plate of the vehicle involved in the violation from the first photo of the report.
- D6. The GPS sensor of a user's device has an error of at most 5m from the real position.
- D7. The internet connection is always available when the user interacts with the system.
- D11. License plates are unique.
- D12. The first photo taken by a citizen reporting a violation must clearly show the license plate of the involved vehicle.
- D13. The OCR service is always available.
- G7. Allow users to check their personal info and the list of violations they reported.
 - R2. Reports about traffic violations need to be stored in a database, also associating them to the user profile who sent them.
 - D7. The internet connection is always available when the user interacts with the system.

Traceability matrix:

Login use case is omitted for simplicity, because it would be used in almost every requirement.

Requirements	Use cases
R1	ViolationValidation
R2	NotifyViolations ViolationValidation SuggestionEvaluation
R3	SuggestionEvaluation
R4	SuggestionEvaluation
R5	AnalyzeData
R5.1	AnalyzeData-citizen
R5.2	AnalyzeData-authority
R6	Registration-authority Registration-municipality
R7	NotifyViolations
R8	NotifyViolations

3.3 Performance Requirements

The system doesn't have any critical performance requirement to fulfill its functionalities, but should be as responsible as possible to allow a pleasant experience to final users.

3.4 Design Constraints

3.4.1 Standards compliance

Because the application will be used in Italy, the system adheres to the General Data Protection Regulation (GDPR), laws concerning the privacy of the users and their data treatment (see reference documents section for more info).

3.4.2 Hardware limitations

As anticipated in "Assumptions, dependencies and constraints" section, smartphones that interact with the system's mobile apps need to have a network interface controller and an active internet connection, also GPS sensor is required for sending violation reports, to provide the exact location of the violation.

To summarize, a customer's smartphone requires:

- Internet connection (2G/3G/4G/5G/Wi-Fi)
- GPS

For accessing the SafeStreets website, both desktop and mobile devices need a modern browser with AJAX support, a network interface controller and an active internet connection.

To summarize, any device for connecting and use the website requires:

- Internet connection (2G/3G/4G/5G/Wi-Fi)
- A modern browser that supports AJAX

3.4.3 Any other constraint

Data accessibility needs to be restricted based on the level of authorization of the account in which the customer has logged in and tries to fetch the information.

Citizens' accounts have the lowest level of authorization and can visualize only aggregated data about the violation reports stored in the system. They can't by any means visualize single report for checking info about the report submitter or the vehicle involved, but can visualize violations density based on time, location and type filters.

Municipalities' accounts can perform all actions allowed to citizens, plus they can check and analyze interventions suggested by the system to resolve the detected problems that could have caused the violations in their competence area.

Authorities' accounts can perform all actions allowed to citizens too, but can also analyze individual violation reports in their competence area, accessing also sensible data like the license plate of the vehicle involved and the account who submitted the report, and have the possibility to validate or invalidate them.

3.5 Software System Attributes

3.5.1 Reliability

The system must be fault tolerant to possible server hardware failures or information

flooding, so that it could constantly provide a quality and responsive service to all customers. To reach this goal servers should be split in RACS (Reliable array of cloned services), so that when one has problems the others will take care of the requests. This structure can also be exploited to optimize the request work load among the cloned services.

3.5.2 Availability

The system is supposed to be available for 99,9% of the time, as it's important to provide the possibility to the users to report violations at any time, but it's not critical to have small downtimes in case of unexpected problems. There should be various servers scattered in countries where SafeStreets will be activated, for handling the reception of user reports without overloading the central server.

3.5.3 Security

Violation reports don't contain sensitive data, but it's really important to provide proper encryption and security measures to avoid identity leakage of the report submitter, as it could expose the users to possible repercussions. License plate info must be also omitted from aggregated report data visualizations for citizens, to avoid possible social issues. Complete access to this info must be granted only to authorities and it's really important to properly test the program to avoid information leakage from accounts with less privileges.

3.5.4 Maintainability

The system must use appropriate design architectures and patterns to fulfill its functionalities, so that in the future it will be easier to expand the application based on stakeholders' needs or adapt to possible law changes. A good architecture and structured code will also lead to easier bug fixes and consequently to faster maintenance and less downtimes, hugely helping to satisfy the quality standards granted by the application.

3.5.5 Portability

The code used in the application should be written with frameworks that supports

multiple platforms. This makes code more portable between platforms, avoiding usage of native code for each specific operative system and drastically reducing the time required to develop and maintain the application. Native code could also be used for functionalities not covered by the system or for improving critical aspects of the app, but using a common framework for developing all the code should avoid necessity of multiple teams specialized in each platform and discrepancy between the applications.

Formal Analysis Using Alloy

In this section an alloy formulation is provided to model critical aspects of the system. The analysis is mainly focused on the satisfiability of some static constraints, in particular:

- it cannot happen that two different users have the same email address.
- each city must have at most one municipality, and a municipality is associated to only one city.
- authorities have the jurisdiction of only one city, but a city can have multiple registered authorities.
- a municipality can see only intervention relative to its city.
- data requests must satisfy the filters.

```
abstract sig ReportStatus{}
one sig PENDING extends ReportStatus{}
one sig VALIDATED extends ReportStatus{}
one sig INVALIDATED extends ReportStatus{}

sig PhotoUrl{}
sig Address{}
sig State{}
sig Email{}
sig LicensePlate{}
sig Password{}
sig TrafficViolation{}
sig InterventionType{}
```

```

// Simplified version of timestamp, using only int to avoid a
    ↪ granular subdivision of time not useful for our constraints
sig Timestamp{
    timestamp: one Int
}

sig Report{
    reportTimestamp: one Timestamp,
    reportLocation: one Location,
    submitter: one Citizen,
    violationType: one TrafficViolation,
    vehicle: one Vehicle,
    photo: some PhotoUrl,
    supervisor: lone Authority,
    status: one ReportStatus,
    visualizedBy: set Authority
}{
    no supervisor iff status = PENDING
    some supervisor implies supervisor.city = reportLocation.
        ↪ place.city
}

sig Location{
    // Alloy doesn't provide a Float or Double type, so Int is
        ↪ used instead (Simplified version of GPS coordinates)
    latitude: one Int,
    longitude: one Int,
    place: one Place
}

sig Place{
    address: one Address,
    city: one City
}

sig City{
    state: one State
}

sig Vehicle{
    licensePlate: one LicensePlate
}

```

```

}

sig DataRequest{
    from: one Timestamp,
    to: one Timestamp,
    locality: one City,
    violationType: lone TrafficViolation,
    validReports: set Report
}{
    from != to and from.timestamp < to.timestamp
    // data requests must be done only on validated reports
    some validReports implies validReports.status = VALIDATED
}

sig Intervention{
    interventionLocation: one Location,
    type: one InterventionType,
    accessedBy: one Municipality
}

abstract sig User{
    email: one Email,
    password: one Password
}

sig Municipality extends User{
    city: one City
}

sig Authority extends User{
    city: one City
}

sig Citizen extends User{}

/* ----- constraints ----- */
// composition constraints
fact LocationReportInterventionCompositionConstraint {
    all l: Location | (some r: Report | l in r.reportLocation)
    ⇔ or
    (some i: Intervention | l in i.interventionLocation)
    //all l: Location, r: Report | l in r.reportLocation
}

```

```

fact TimestampReportDataRequestCompositionConstraint {
    all t: Timestamp | (some r: Report | t in r.reportTimestamp)
    ↪ or
    (some d: DataRequest | t in d.from or t in d.to)
}

fact PhotoUrlReportCompositionConstraint {
    all p: PhotoUrl | one r: Report | p in r.photo
}

fact VehicleReportCompositionConstraint {
    all v: Vehicle | some r: Report | v in r.vehicle
}

fact LicensePlateVehicleCompositionConstraint {
    all l: LicensePlate | one v: Vehicle | l in v.licensePlate
}

fact PlaceLocationCompositionConstraint {
    all p: Place | some l: Location | p in l.place
}

fact AddressPlaceCompositionConstraint {
    all a: Address | some p: Place | a in p.address
}

fact CityPlaceAuthorityMunicipalityCompositionConstraint {
    all c: City | (some p: Place | c in p.city) or
    (some m: Municipality | c in m.city) or (some a: Authority |
    ↪ c in a.city)
}

fact PasswordUserCompositionConstraint {
    all p: Password | some u: User | p in u.password
}

fact EmailUserCompositionConstraint {
    all e: Email | one u: User | e in u.email
}

fact StateCityCompositionConstraint {

```

```

    all s: State | some c: City | s in c.state
}

// other constraints
fact SameCoordinatesHaveSamePlace {
    no disj l1,l2: Location | (l1.latitude = l2.latitude and l1.
        ↪ longitude = l2.longitude) and l1.place != l2.place
}

fact SelectOnlyReportsThatSatisfyRequestFilters {
    all d: DataRequest, r: Report | r in d.validReports iff
        (r.reportTimestamp.timestamp >= d.from.timestamp and r.
            ↪ reportTimestamp.timestamp <= d.to.timestamp
        and r.reportLocation.place.city = d.locality
        and (d.violationType != none implies r.violationType = d.
            ↪ violationType))
}

fact NoMunicipalityWithSameCity {
    no disj m1,m2: Municipality | m1.city = m2.city
}

fact InterventionMunicipalityLocationConsistency {
    all i: Intervention, m: Municipality | m in i.accessedBy iff
        i.interventionLocation.place.city = m.city
}

fact ReportVisualizedByCompetentAuthorities {
    all r: Report, a: Authority | a in r.visualizedBy iff
        r.reportLocation.place.city = a.city
}

fact NoDifferentInterventionsWithSameLocation {
    no disj i1, i2: Intervention | i1.interventionLocation = i2.
        ↪ interventionLocation and i1.type = i2.type
}

fact NoPlacesWithSameAddressAndCity {
    no disj p1, p2: Place | p1.address = p2. address and p1.city
        ↪ = p2.city
}

```

```

fact NoSameReportFromSameCitizenAtTheSameTime {
    no r1, r2: Report | r1.submitter = r2.submitter and r1 != r2
}

fact NoInterventionsWithoutMunicipality {
    all i: Intervention | some m: Municipality | i.
        ↪ interventionLocation.place.city = m.city
}

/* ----- assertions ----- */
assert ReportSupervisorCanAlsoVisualize {
    all r: Report | r.supervisor in r.visualizedBy
}
check ReportSupervisorCanAlsoVisualize for 10

assert AuthorityCanAccessOnlyReportsFromHisCity {
    all r: Report | no a: Authority | a.city != r.reportLocation
        ↪ .place.city and a in r.visualizedBy
}
check AuthorityCanAccessOnlyReportsFromHisCity for 10

/* ----- worlds ----- */
// shows a world with one data request with at least one valid
    ↪ report
pred dataRequestWithValidReport {
    #(status :> VALIDATED) = 1
    #(status :> PENDING) = 1
    #validReports >= 1
}
run dataRequestWithValidReport for 3 but exactly 2 Report, exactly 1
    ↪ DataRequest, 1 City, 1 TrafficViolation, 0 Intervention, 0
    ↪ Municipality

// shows a world with one data request with zero valid reports
pred dataRequestWithNoValidReport {
    #(status :> VALIDATED) = 1
    #(status :> INVALIDATED) = 1
    #validReports = 0
}
run dataRequestWithNoValidReport for 3 but exactly 2 Report, exactly

```

```

    ↪ 1 DataRequest, 1 City, 1 TrafficViolation, 0 Intervention, 0
    ↪ Municipality

// shows a world in which interventions are accessible only from
    ↪ municipality in which they are located in
pred interventionAccessibility {
    some i1, i2: Intervention | i1.interventionLocation.place.
        ↪ city != i2.interventionLocation.place.city
}
run interventionAccessibility for 3 but exactly 2 Municipality,
    ↪ exactly 3 Intervention, exactly 2 Location, 0 Authority, 0
    ↪ Citizen, 0 Report, 0 DataRequest

// shows a world in which some reports are accessible from some
    ↪ authorities and others are not
pred reportAccessibility {
    #visualizedBy >= 1
    some r: Report | r.visualizedBy = none
}
// at least n+1 entities, where n is number of authorities, because
    ↪ at least 1 citizen is needed for reports and so n+1 emails are
    ↪ needed
run reportAccessibility for 4 but exactly 2 Authority, exactly 2
    ↪ Report, 2 Location, 0 Municipality, 0 DataRequest, 0
    ↪ Intervention

```


Executing "Check ReportSupervisorCanAlsoVisualize for 10"
 Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
 76179 vars. 3810 primary vars. 182510 clauses. 334ms.
 No counterexample found. Assertion may be valid. 175ms.

Executing "Check AuthorityCanAccessOnlyReportsFromHisCity for 10"
 Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
 76611 vars. 3820 primary vars. 184125 clauses. 230ms.
 No counterexample found. Assertion may be valid. 370ms.

Executing "Run dataRequestWithValidReport for 3 but exactly 2 Report, exactly 1 DataRequest, 1 City, 1 TrafficViolation, 0 Intervention, 0 Municipality"
 Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
 3698 vars. 335 primary vars. 7966 clauses. 16ms.
 Instance found. Predicate is consistent. 29ms.

Executing "Run dataRequestWithNoValidReport for 3 but exactly 2 Report, exactly 1 DataRequest, 1 City, 1 TrafficViolation, 0 Intervention, 0 Municipality"
 Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
 3697 vars. 335 primary vars. 7966 clauses. 13ms.
 Instance found. Predicate is consistent. 29ms.

Executing "Run interventionAccessibility for 3 but exactly 2 Municipality, exactly 3 Intervention, exactly 2 Location, 0 Authority, 0 Citizen, 0 Report, 0 DataRequest"
 Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
 2411 vars. 271 primary vars. 4091 clauses. 7ms.
 Instance found. Predicate is consistent. 17ms.

Executing "Run reportAccessibility for 4 but exactly 2 Authority, exactly 2 Report, 2 Location, 0 Municipality, 0 DataRequest, 0 Intervention"
 Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
 3930 vars. 412 primary vars. 6327 clauses. 19ms.
 Instance found. Predicate is consistent. 20ms.

6 commands were executed. The results are:
 #1: No counterexample found. ReportSupervisorCanAlsoVisualize may be valid.
 #2: No counterexample found. AuthorityCanAccessOnlyReportsFromHisCity may be valid.
 #3: **Instance found.** dataRequestWithValidReport is consistent.
 #4: **Instance found.** dataRequestWithNoValidReport is consistent.
 #5: **Instance found.** interventionAccessibility is consistent.
 #6: **Instance found.** reportAccessibility is consistent.

Figure 4.1: World execution details

In the following section we present some possible models of the world, in particular the continuous lines represents the most important relations, whereas the dotted lines represents the least important ones.

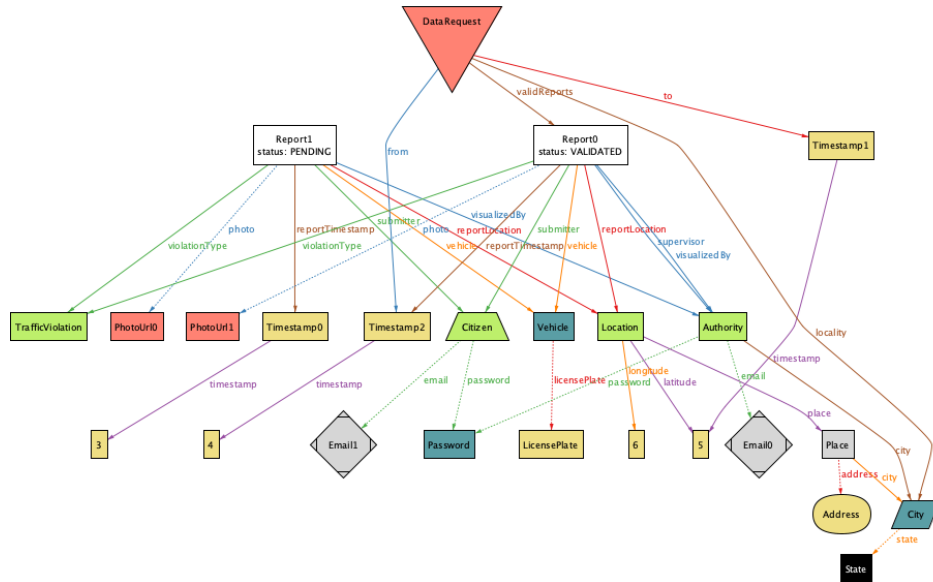


Figure 4.2: World 1

In the first world, we can see that different users have different emails (used for registration and login) and there are two reports and one data request. The first report is in a pending status even though it is visualized by an authority that has in its jurisdiction the city where the violation occurred, and it is correctly among the reports associated to the data request. The second report is in a validated status, indeed there is an authority in that city who is its supervisor; also this report satisfies the time and spatial constraints of the data request so it is in its valid reports.

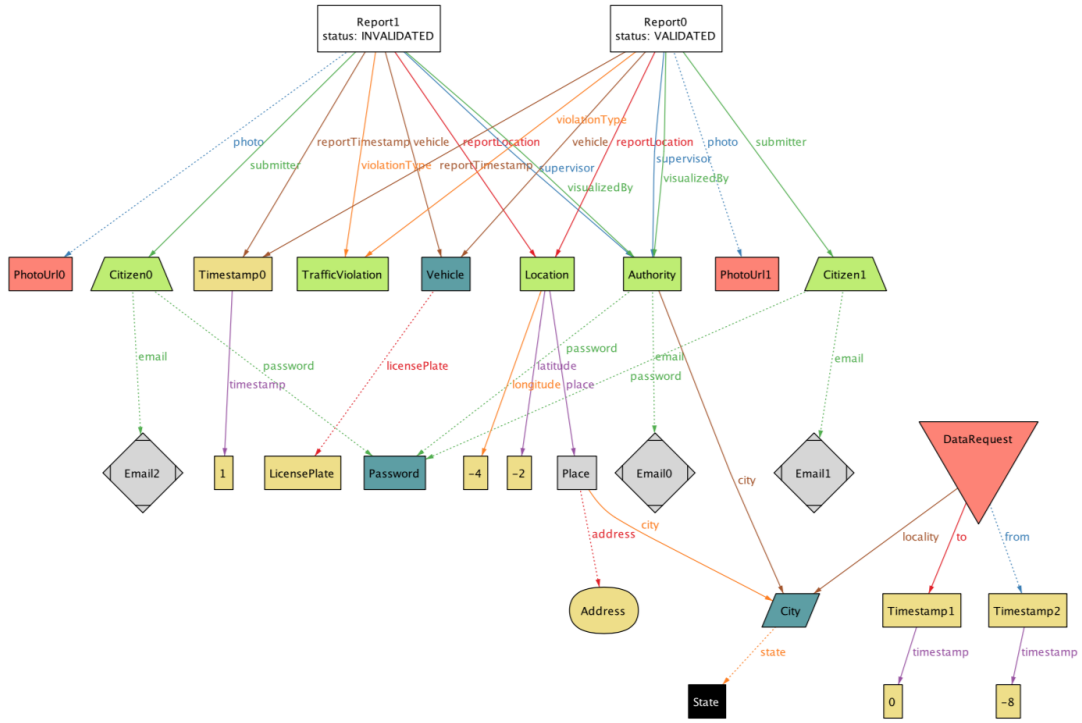


Figure 4.3: World 2

In the second world, there is one validated request, one invalidated request and one data request. The two reports are about the same traffic violation (same vehicle, same location and same timestamp), but they are reported by different citizens with two different photos. It is interesting to see that the report of a same violation can be validated in one case and not in the other, most probably when a photo describes completely the violation situation and not respectively. In this case there are no reports in the data requests, because there are no reports satisfying its filter.

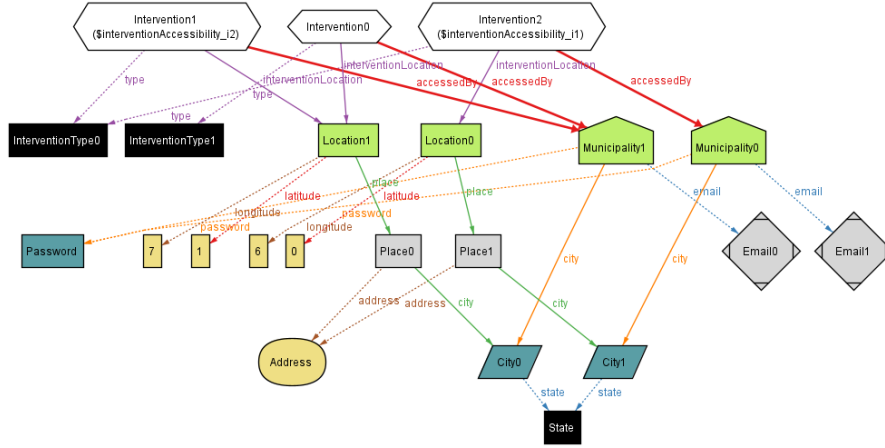


Figure 4.4: World 3

In the third world, we can check that an intervention can be seen only by municipality of the city in which the intervention is situated. As you can see, municipality1 can access intervention1 and intervention0 because their location is situated in the same city of it, same thing happens for intervention2 and municipality0. Some interventions can occur in the same location because different intervention type could be needed in that place.

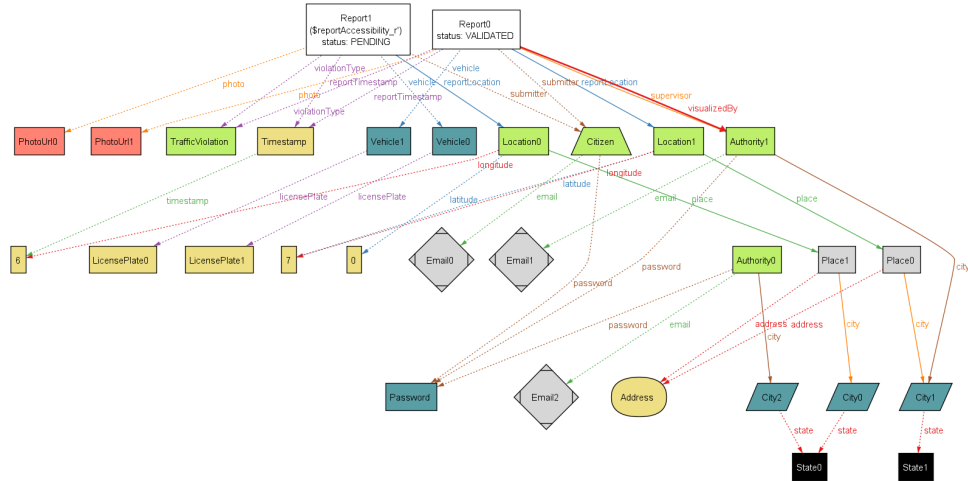


Figure 4.5: World 4

In the fourth world, we can check that a report can be visualized only by authorities who have jurisdiction on that city. Another interesting relation is that report supervisor is of course an authority that can access it, also if a supervisor is present means that report is either validated or invalidated. If a report can't be accessed, clearly its

status remain always pending because no authority can't change it.

Effort Spent

Andrea Falanti:

Document section	Hours
Introduction	7.5
Overall Description	13
Specific Requirements	15
Formal Analysis Using Alloy	10.25
Total	45.75

Andrea Huang:

Document section	Hours
Introduction	7
Overall Description	4.0
Specific Requirements	14.75
Formal Analysis Using Alloy	12.5
Total	38.25

References

- UML diagrams: <https://www.uml-diagrams.org/>.
- Alloy doc: <http://alloy.lcs.mit.edu/alloy/documentation/quickguide/seq.html>.
- Info about GDPR: https://en.wikipedia.org/wiki/General_Data_Protection_Regulation.
- List of mainstream browsers supporting AJAX: <https://jquery.com/browser-support/>.
- Circolare n. 1/2010/DDI: http://www.iuav.it/Servizi-IU/servizi-ge1/Servizi-in/PEC/Circolare-1-2010---DDI_-PEC-nelle-PA.pdf.