RASD

Carlo Dell'Acqua, Adriana Ferrari, Angelica Sofia Valeriani

December 12, 2019

POLITECNICO
MILANO 1863

Contents

1 Introduction **3**
- 1.1 Purpose . . . 3
  - 1.1.1 General Purpose . . . 3
  - 1.1.2 Goals . . . 3
- 1.2 Scope . . . 4
  - 1.2.1 World and Shared phenomena . . . 4
- 1.3 Definitions, acronyms and abbreviations . . . 5
  - 1.3.1 Definitions . . . 5
  - 1.3.2 Acronyms . . . 5
  - 1.3.3 Abbreviations . . . 5
- 1.4 Revision History . . . 5
- 1.5 Reference documents . . . 5
- 1.6 Document Structure . . . 6

2 Overall description **7**
- 2.1 Product Perspective . . . 7
- 2.2 Product Functions . . . 10
  - 2.2.1 Show safe and unsafe areas . . . 10
  - 2.2.2 Submit and confirm violations . . . 10
  - 2.2.3 Allow authorities to take action . . . 11
- 2.3 User Characteristics . . . 11
- 2.4 Assumptions, dependencies and constraints . . . 11
  - 2.4.1 Domain assumptions . . . 11
  - 2.4.2 Dependencies . . . 12
  - 2.4.3 Constraints . . . 12

3 Specific requirements **14**
- 3.1 External Interface Requirements . . . 14
  - 3.1.1 User Interface . . . 14
  - 3.1.2 Hardware Interfaces . . . 16
  - 3.1.3 Software Interface . . . 16
  - 3.1.4 Communication Interface . . . 16
- 3.2 Functional Requirements . . . 16
  - 3.2.1 R1: Allow any citizen of legal age, as a visitor, to register on the mobile phone app or the web app and to become a user by providing a document . . . 16
  - 3.2.2 R2: Allow a user to send pictures about street and parking violations . . . 19

3.2.3 R3: Validation of reported violations . . . 22
3.2.4 R4: Allow a user to see the areas where a violation is more likely to happen . . . 25
3.2.5 R5: Allow authorities to see data about violations and who committed them . . . 27
3.2.6 R6: Allow authorities to register with a special user profile 28
3.2.7 R7: Detection of unsafe areas and possible interventions . 31
3.2.8 R8: Traffic ticket generation from validated reports 33
3.2.9 Traceability Matrix . . . 35
3.3 Performance Requirements . . . 35
3.3.1 Performance features . . . 35
3.3.2 Evolution of the system . . . 36
3.3.3 Performance testing . . . 36
3.4 Design Constraints . . . 36
3.4.1 Standards compliance . . . 36
3.4.2 Hardware limitations . . . 36
3.4.3 Other constraints . . . 36
3.5 Software System Attributes . . . 37
3.5.1 Reliability . . . 37
3.5.2 Availability . . . 37
3.5.3 Security . . . 37
3.5.4 Maintainability . . . 38
3.5.5 Portability . . . 39

4 Formal using Alloy **40**
4.1 Signatures . . . 40
4.2 Facts . . . 41
4.3 Assertions . . . 42
4.4 Predicates . . . 43
4.5 Alloy Execution Results . . . 43

5 Effort spent **48**

References **50**

Chapter 1

Introduction

1.1 Purpose

1.1.1 General Purpose

SafeStreets is a service that aims to improve the safety of the general traffic. This is achieved by creating a community of users who are able to report any violation they see while the system manages all the aspects of data validation and statistical analysis. Different services contribute to this purpose:

- The first service offered by the front end application is the report service. Any registered user can submit violation reports and SafeStreets will validate them as described in the following sections with the help of the community.
- The second service offered by the front end application is the the Unsafe Areas Map. SafeStreets will provide statistics about areas that have a higher risk of violations based on the reports it receives and the danger of the infractions. Data can also be collected from public services if available to increase accuracy.
- The third service is the ticket generation. Traffic policemen will have access to a dedicated section of the application where SafeStreets will collect validated reports. This will enable any registered policeman to take actions against those violations.

1.1.2 Goals

The following goals describe the key points of our system:

- [G1]: the system will provide an easy interface for violation reports to all citizens of legal age
- [G2]: the system will provide aggregate anonymous data to show unsafe areas
- [G3]: the system will show violation reports to registered authorities

- [G4]: the system will integrate a validation procedure that will be followed strictly for each submitted report
- [G5]: the system will provide secure authentication methods
- [G6]: user data will only be available to the user itself, authorities and SafeStreets authorized employees
- [G7]: approved violation reports have legal validity

1.2 Scope

SafeStreets is a service that is available to any citizen having a valid ID number. The service creates a direct channel to communicate with authorities and with other users about the observed traffic violations, promoting street safety and helping others who might be in need, for example by reporting a vehicle which is blocking an access for disabled people. Registered users can submit reports that are going to be validated by other end-users, creating a network of trust. The reports they file provide information about the violation, the location and the license plate of the subject. This makes reports detailed enough for approval by public officers, who are able to review them and take proper action to discourage further infractions.

1.2.1 World and Shared phenomena

In this section a list of World and Shared phenomena are shown in order to better distinguish the separation between the system (the machine) and the world in which it operates.

World phenomena

- A violation occurred
- A User decides to report a violation
- A User wants to check where violations are more likely to occur
- An external organization wants a list of the violations occurred in an area
- An Authority acknowledges about the violations

Shared phenomena

- The user logs in the application
- A user reports the violation using the SafeStreets application
- A user request the unsafe areas map on the application
- Users verify a report by answering the poll generated by a notification of violation
- The authority is informed by the system about violations reported in his jurisdiction
- The authority generates the traffic ticket for a validated report

1.3 Definitions, acronyms and abbreviations

1.3.1 Definitions

- **Safe (or Unsafe) Area**: A geographical region, usually a set of streets, where less (or more) accidents occur than the average based on neighboring regions
- **Report**: A set of data containing all the information about a traffic violation
- **System**: The SafeStreets platform
- **Web Application**: A Rich Internet Application that enables users to access the functionalities of the system through a modern web browser without having to manually install any other software

1.3.2 Acronyms

- **RASD**: Requirements Analysis and Specification Document
- **API**: Application Programming Interface
- **GPS**: Global Positioning System

1.3.3 Abbreviations

- **G*n***: n_{th} goal
- **D*n***: n_{th} domain assumption
- **R*n***: n_{th} requirement
- **Web App**: Web Application

1.4 Revision History

Version	Major changes
1.0.0	First release
1.1.0	Update UML sequence diagrams (error corrections)
1.1.1	Centered images

1.5 Reference documents

- **Assignment:** SafeStreets Mandatory Project Assignment
- **Previous project example:**
  - **Assignment:** Mandatory Project Assignment AY 2018-2019
  - **Example document:** RASD to be analysed - A.Y. 2019-2020

1.6 Document Structure

1. **Introduction**: The first section is a general description of the system's scope, purpose, and its goals. It also includes references of the document and definitions, abbreviations and acronyms used along the paper

2. **Overall description**: The second section includes shared phenomena, requirements and domain assumptions. It also clarifies users' needs and characteristics

3. **Specific requirements**: The third section includes all the requirements of the application, both functional and non functional. Besides, it also includes scenarios and associated use cases, with sequence diagrams in order to clarify events-flow

4. **Formal analysis using Alloy**: The fourth section includes the Alloy model that describes the given system

5. **Effort spent**: The fifth section includes the detailed information about the time spent for each part of the document and how the work has been divided between group members

6. **References**: The sixth section includes all the references about papers and documents used to support this document

Chapter 2

Overall description

2.1 Product Perspective

The core of SafeStreets is the opportunity to create a community that cooperates to reduce traffic violations: for this to be possible, it is necessary to integrate third party services with our software. Our software works with the APIs provided by Google Maps to best help the users identify the location of the violation they want to report. We also work with different municipalities that help us provide better statistics by sharing with us their data about traffic violations, so our software needs to interact with their APIs and reduce the differences in communication standards as much as possible. In order to encourage citizens to become SafeStreets users, we will create partnerships with different brands. Through these partnerships we will be able to send prizes to users that contribute the most to the community.

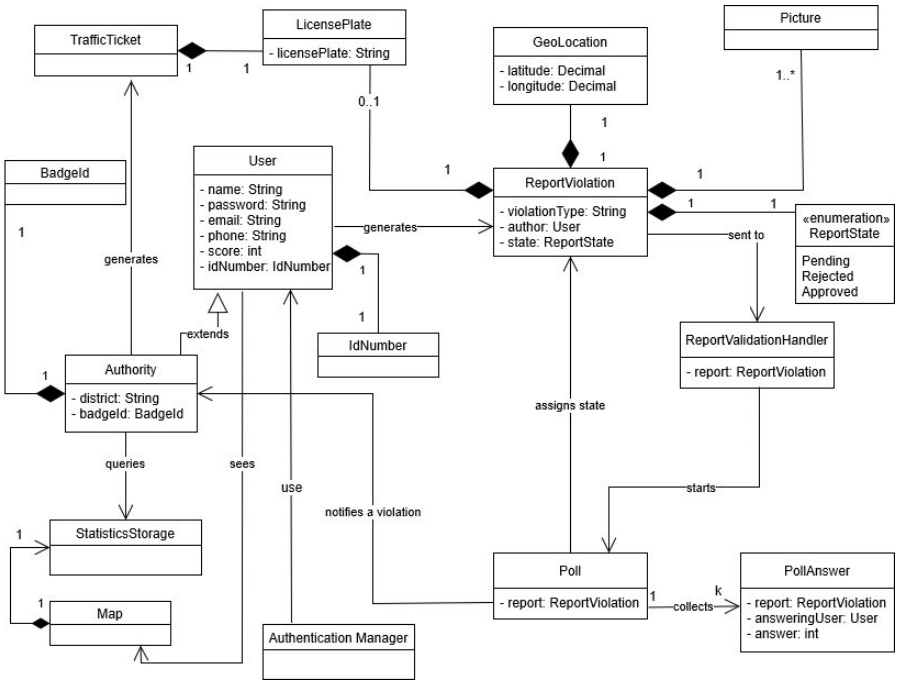

Figure 2.1: *UML class diagram*

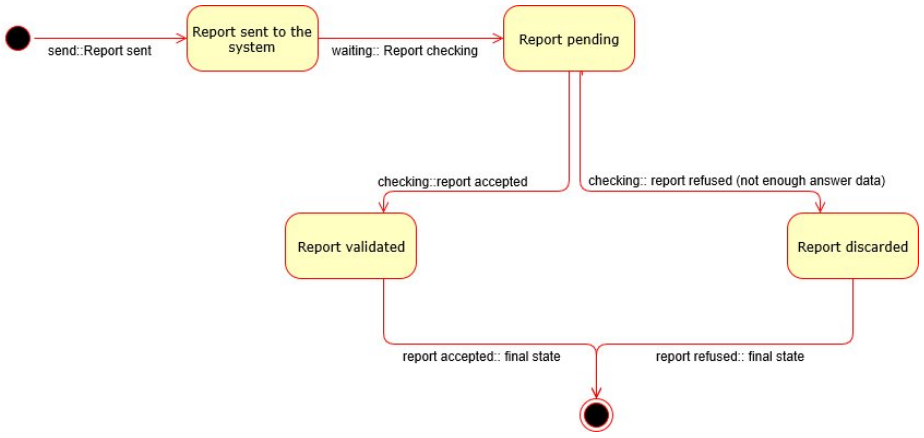

Figure 2.2: *State diagram 1 - Report*

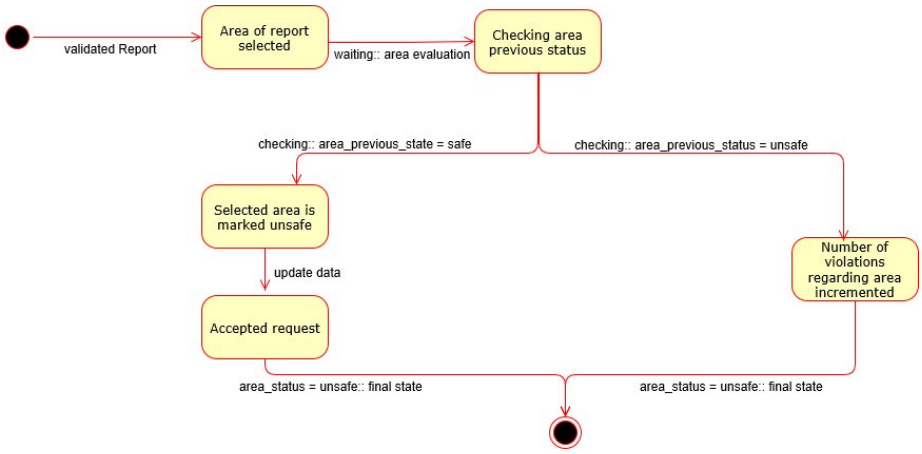

Figure 2.3: *State diagram 2 - Area*

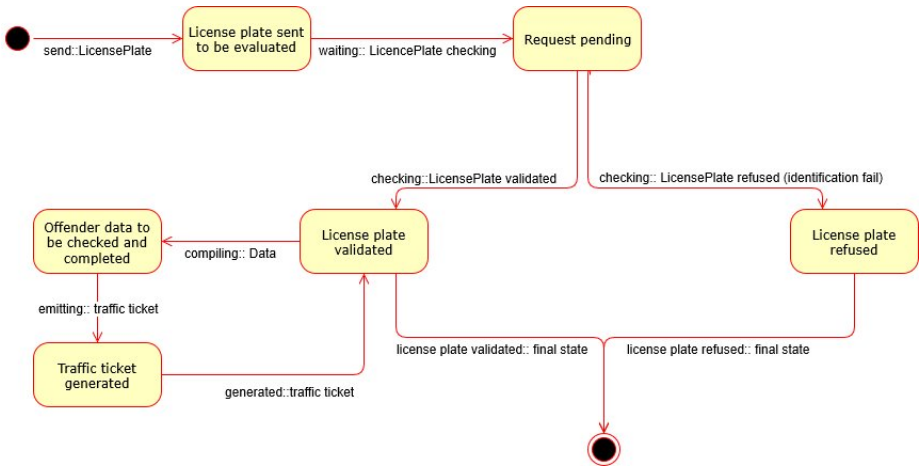

Figure 2.4: *State diagram 3 - License plate*

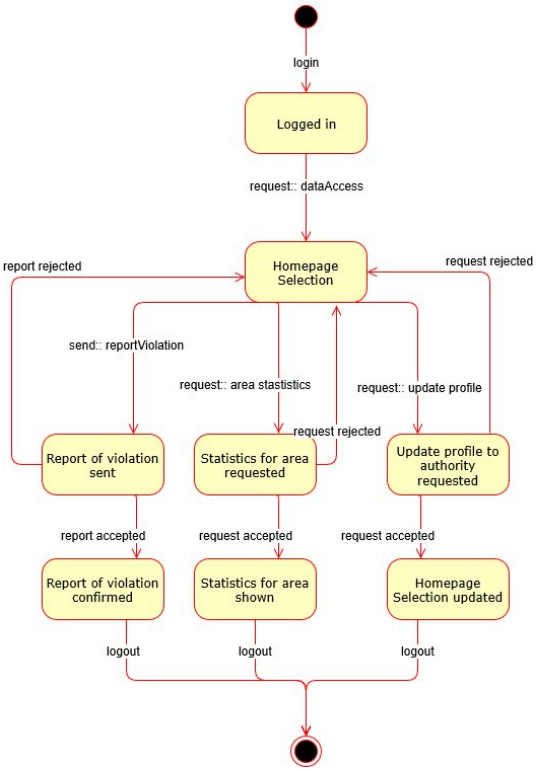

Figure 2.5: *State diagram 4 - User login*

2.2 Product Functions

The following section contains the main product functions of SafeStreets. Some of them are available to everyone, some only to registered users and others only to authorities.

2.2.1 Show safe and unsafe areas

Who has access:
Anyone

Description:
SafeStreets helps people understand which parts of their town, or any customizable region, are more dangerous because of traffic violations. A map highlights areas differently according to the frequency and severity of violations that occur. The data used to build these statistics is both entered by SafeStreets users and, whenever possible, integrated by data provided by the municipality.

2.2.2 Submit and confirm violations

Who has access:
Registered users

Description:
SafeStreets core functionality is allowing users to submit documentation about traffic violations they see. Only registered users have access to this functionality because it is necessary to be able to connect a report to the person who filed it, as intentionally filing fake reports may have legal consequences. In order to keep fake reports to a minimum, all violations need to be approved by an established amount of SafeStreets users and only then are they shown to the authorities. The ratio of approved versus non approved reports of a user, as well as the actual amount of violations they reported, is also taken into account when computing the reliability of a user.

2.2.3 Allow authorities to take action

Who has access:
Registered authorities

Description:
Authorities have access to another section of SafeStreets, which allows them to see more data about who submitted and committed violations. They can check the latest reports and either send a patrol in the area or, if the violations respects certain requirements, even directly send tickets to the culprit.

2.3 User Characteristics

- **Visitor**: visitors are people who visit SafeStreets without being registered. They can see statistics about safe and unsafe areas and they can also register in order to become users.
- **User**: a registered user, called simply user, is a person who registered to SafeStreets, providing their personal data. Once logged in with their credentials, a user can submit reports and vote to approve or reject reports filed by other users.
- **Authorities**: authorities are actual public officers registered to SafeStreets. They have full access to reports data and are allowed to generate tickets.
- **Municipality**: municipalities are local administrative authorities. They can provide further statistical data to help SafeStreets analysis.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

- [D1] Personal information that a visitor has to provide to become a registered user are name, surname, email, ID number and password
- [D2] Emails are unique and a user can only be associated to one email address

- [D3] Two-factors authentication can be enabled by the user by providing a phone number (SMS authentication) or by scanning a QR Code with an authenticator app (Token authentication)
- [D4] Users are enabled to submit reports only after their ID number has been verified by an API offered by public authorities
- [D5] Citizens must be at least 18 years old to register to SafeStreets
- [D6] Real world authorities can be uniquely identified through a badge number
- [D7] Municipalities expose APIs that allow SafeStreets to identify an authority by providing the badge number
- [D8] Municipalities can expose APIs that allow SafeStreets to retrieve data about other accidents that weren't notified through SafeStreets
- [D9] Once a SafeStreets report is validated, it has the same legal validity as a report filed personally by an authority, so authorities can emit tickets based on approved reports
- [D10] The safety level of an area is computed taking into account only street violations and accidents and is based on previous data
- [D11] No one can alter a report once it has been sent, it can only be validated or rejected

2.4.2 Dependencies

- SafeStreets provides suggestions for improving the conditions of the most dangerous areas, but the local authorities will be in charge of every effective measure of security taken to make streets safer
- Unless the majority of street violations is notified through SafeStreets, the software can only build statistically relevant maps and charts if the municipalities provide APIs that give access to all data about traffic tickets and accidents. This means that the safety of different areas may be based on more or less data and therefore be more or less accurate.
- SafeStreets can be used even without GPS, in that case data regarding the location needs to be manually entered by the user: this means that a device with functioning GPS will improve user experience and will also help make sure that the data submitted to the server is consistent.

2.4.3 Constraints

- The service that allows authority to remotely generate traffic tickets works at its best for violations that include a vehicle identifiable by a license plate. If the violation is about a bike or a pedestrian or the license plate for some reason isn't fully recognizable it is necessary that an officers is actually present in the reported location for any further action to be taken.

- SafeStreets can work properly and have an impact on the neighborhood only if there is a large amount of engaged users, otherwise only very few reports are submitted and validated. It's therefore necessary for the application to be also entertaining and to provide some rewards to the most dedicated users.
- While originally thought for the Italian market, SafeStreets can easily be used and integrated in other countries as long as all citizens have a unique document number and police authorities can also be uniquely identified by a badge number.

Chapter 3

Specific requirements

3.1 External Interface Requirements

3.1.1 User Interface

The following mock ups represent a basic idea of how the Mobile Application and the Web Interface are supposed to look like. Through the smart-phone application and web interface, users have access to complete SafeStreets functionalities, including the possibility of notifying violations.

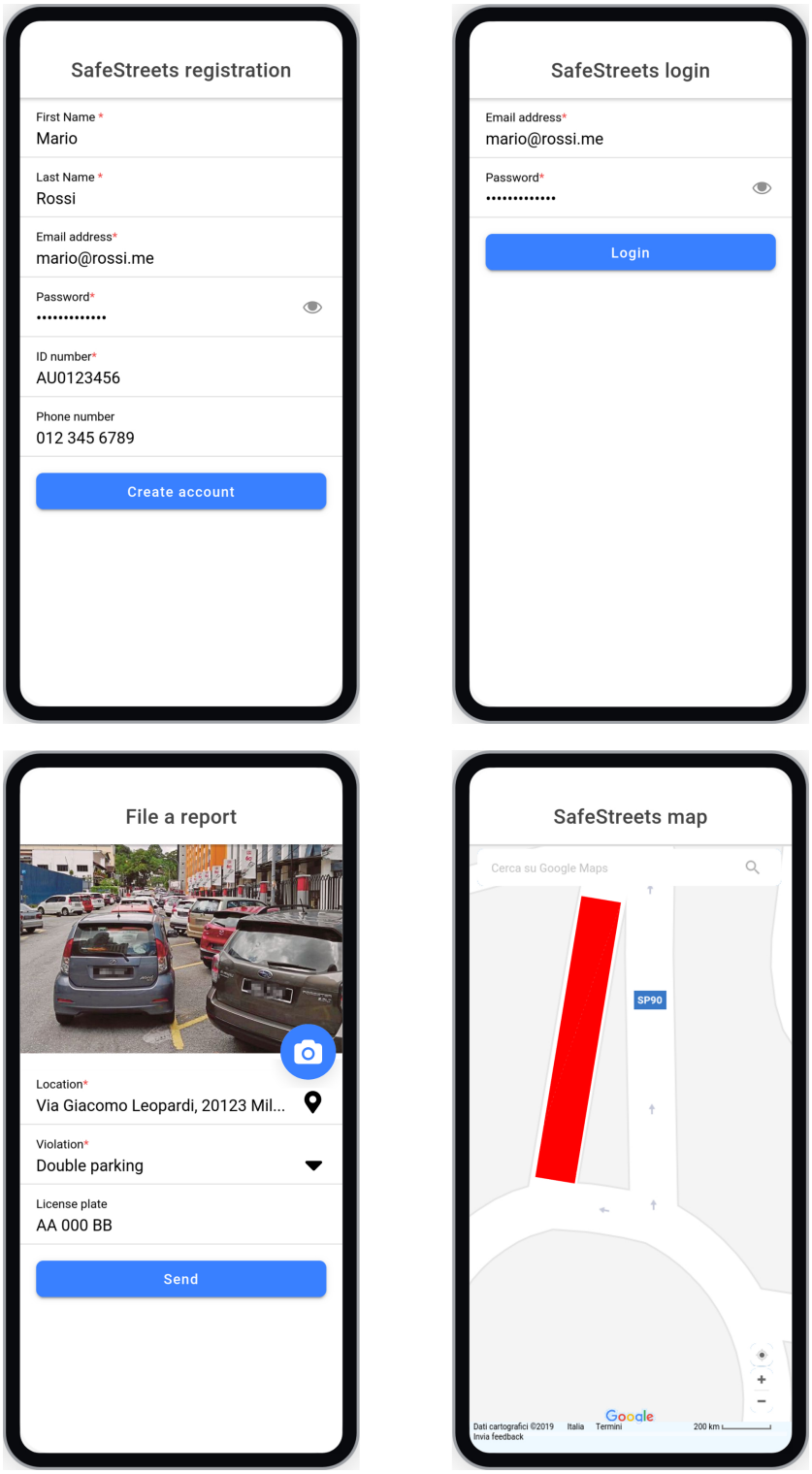

Figure 3.1: *User interface mock ups*

3.1.2 Hardware Interfaces

Since the application runs over the Internet, the hardware of both the server and the client needs to be able to connect to the Internet.

- Server-side: the hosting platform will provide all the necessary hardware interface.
- Client-side: e.g. Wi-Fi, 3G/4G.

Client-side hardware should have a camera in order to have access to all functionalities of the application. It is also recommended, although not mandatory, that client-side hardware is provided with a GPS.

3.1.3 Software Interface

An API provided by SafeStreets allows third parties to access the read-only functionalities offered by the system. This way, third party companies and applications can have access and possibly embed in their software statistical data about accidents and street safety collected by SafeStreets. On the other hand, functionalities that involve the creation of new data are meant to be used only within SafeStreets official platforms and are not exposed to third parties.

3.1.4 Communication Interface

SafeStreets will communicate with third parties and with the client application using the standard HTTPS protocol, which guarantees encryption by default.

3.2 Functional Requirements

3.2.1 R1: Allow any citizen of legal age, as a visitor, to register on the mobile phone app or the web app and to become a user by providing a document

- The system must allow the visitor to provide credentials and personal data
- The system must verify the correspondence between the ID number provided by the visitor and their personal information
- The system must allow the visitor to verify the account with an e-mail or SMS verification code
- The system must send a recap of registration to email of the visitor
- The system must verify that there are no other registered users or authorities with the same e-mail or ID number
- The user must accept users data privacy conditions to successfully register to the system
- A citizen can register to SafeStreets by providing their ID number
- The system must allow a user to register with an email and a password, that will be asked every time they log in

Scenario 1

Mario Rossi saw the advertisement of SafeStreets and decided to download the mobile application in order to be allowed to notify violations and improve safety of streets in his city. After opening the new app, he is asked to

fill a form with all his personal information, full name, email address, mobile number, ID card and phone number. To proceed with his registration, after having filled all the form, including the users data privacy conditions, Mario clicks on the "Create account" button. The system verifies the correspondence between inserted personal data and the ID card information, and if the verification is positive, Mario is asked whether he prefers to confirm his registration by phone number or by email. He chooses the alternative he prefers, then he immediately receives the verification code on the medium he chose. He can confirm his registration by clicking on the verification code. The system then sends a recap to notify the positive outcome of registration. Mario is now a Registered User of SafeStreets; he can login to notify violations on the streets and is enabled to the use of all the functions for a basic user (not an authority) that the system provides.

Use case

Visitor Registration

Name	Registration to SafeStreets
Actor	Visitor
Requirement	R1
Entry conditions	The visitor must have downloaded the mobile phone application or the visitor must have accessed the web interface
Event flow	1. The Visitor must fill all mandatory fields in the registration form 2. The Visitor must click on the "Create account" button 3. The system validate the provided data 4. The Visitor chooses how to confirm his registration (phone number/email) 5. The system sends the code on the visitor's chosen mode, mobile phone or email address 6. The Visitor clicks on the confirmation code 7. The system sends a recap of registration, as a positive outcome 8. The system saves the Visitor's data
Exit conditions	The Visitor has become a registered used
Exceptions	• The Visitor is already registered • The email is already registered • The ID card is already registered • Visitor's data don't match with ID card data • Some mandatory fields are not filled • The user is a minor

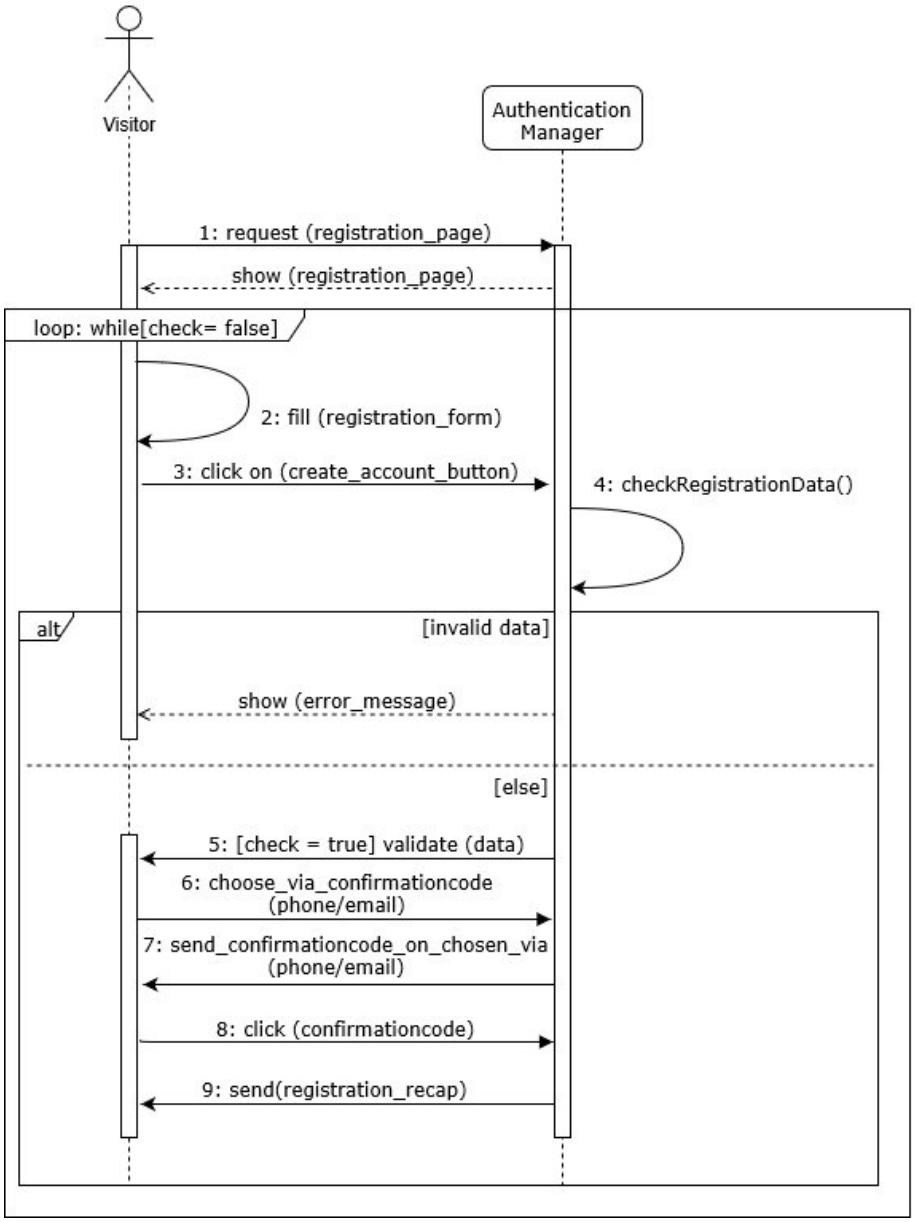

Figure 3.2: *Sequence diagram 1 - Visitor registration on mobile phone*

3.2.2 R2: Allow a user to send pictures about street and parking violations

- A user can submit a picture to the system whenever they see a traffic violation
- SafeStreets analyses the image to read the license plate of the vehicle
- The user can manually attach the license plate to the image, to help the

system with its validation

- The user must provide the type of violation, either selecting it from a list or manually typing it if it is not already in the system
- The user can help localizing the violation by inserting the name of the street where it occurred

Scenario 2

Mario Rossi is a user of SafeStreets, he has just finished working and he wants to go back home, but he is can't reach his car because of double parking. He waits for ten minutes to see if the owner of the car is coming, but then decides to notify the violation. In order to notify the violation he must have logged into the mobile application, inserting his data, so he does; then he takes two pictures representing the violation, then he clicks on the button to notify a violation, he inserts the position (Via Botticelli, Viale Romagna, 2, 20133 Milano MI), a brief description of the violation and the two pictures. He is asked to select the category of the notified violation (Double-Row Parking), or to add it if it's still not present in the system. Then, he is asked to insert the license plate of the vehicle involved in the violation in the mobile application, he does, then the app verifies that it corresponds to the license plate identified with the photograph.

Use case

User sends pictures to notify violation

Name	Notification of violation by a user
Actor	User
Requirement	R2
Entry conditions	The user must be registered to the application and he must have logged in
Event flow	1. The user takes pictures of the violation 2. The user clicks on the "Notify a violation" button 3. The user inserts data of the violation 4. The user includes the two pictures 5. The user checks whether the category of notified violation is present or not 6. The user chooses "Double-Row Parking", then submits 7. The system asks the user to insert the license plate (notified violation includes a vehicle) 8. The user inserts license plate and submits 9. The system validates the license plate
Exit conditions	The violation has been notified to the system
Exceptions	• The chosen category of the violation includes a vehicle but the license plate is not manually inserted by the user • The license plate notified by the user doesn't correspond to the license plate identified by the pictures sent • At least one of the pictures attached to the report has been taken more than 2 hours before the submission

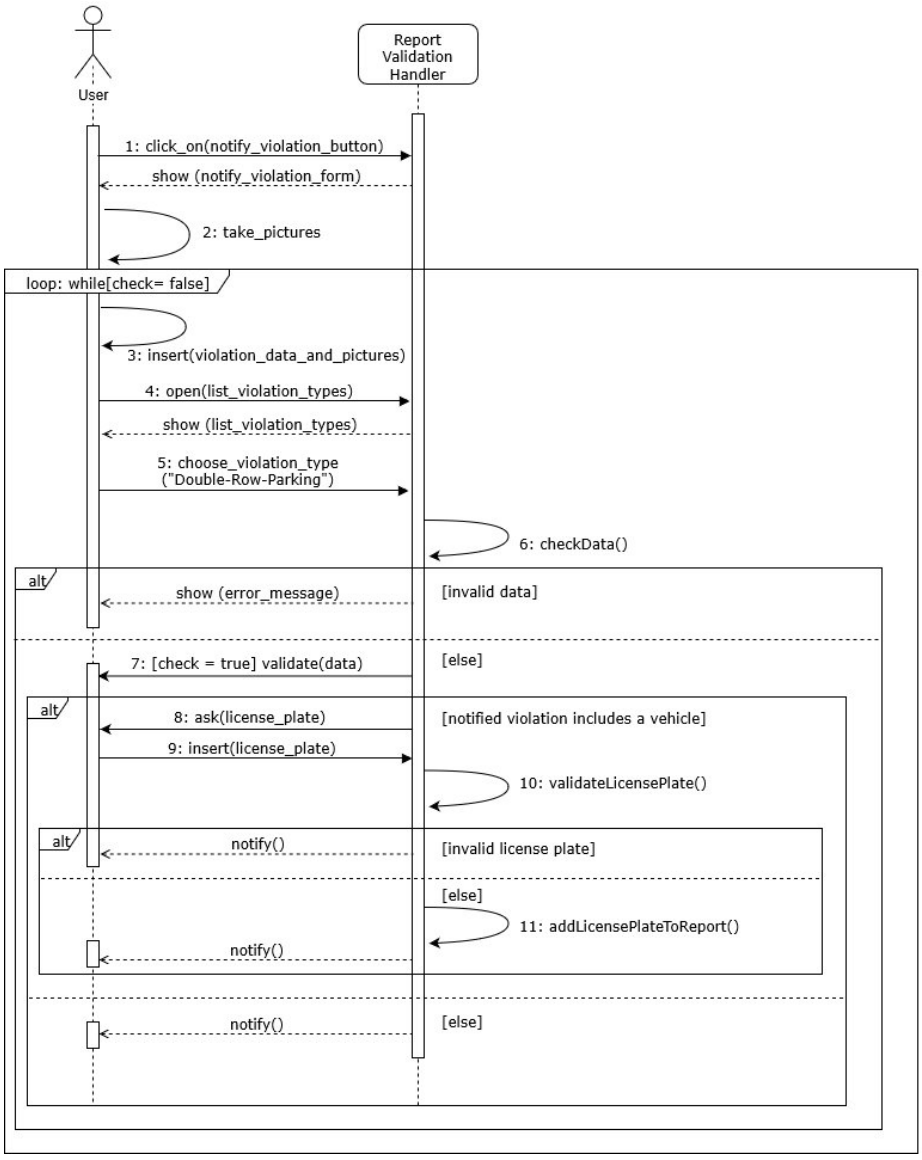

Figure 3.3: *Sequence diagram 2 - User sends pictures on mobile phone to notify a parking violation*

3.2.3 R3: Validation of reported violations

- Whenever a violation is reported the system checks whether or not the license plate field has been filled
- If the license plate is present, to prevent multiple tickets from being generated, the system verifies if the same kind of violation has already been reported in the previous h hours for the same vehicle
- If the same violation has already been reported in the previous h hours

the report is automatically discarded, otherwise it is sent to a group of k SafeStreets users for validation

- Users who receive the notification about this approval process are then asked to approve, reject or declare themselves neutral about the report
- The report is validated if and only if the algebraic sum of approvals (+1), neutralities (0) and rejections (-1) is at least v
- Every user is supposed to respond, the system expects to receive k answers
- If a user doesn't answer within a certain amount of time, their answer is considered neutral

Scenario 3

SafeStreets has received the report of Mario Rossi regarding a violation of double-row parking. To verify the violation, a notification is sent to k users. The system receives the answers that the users gave to the validation request and makes the algebraic sum of approvals and rejections. The calculated sum is $m > v$, where v is a threshold that guarantees a minimum number of approvals; as a consequence, the system validates the report and registers the violation.

Use case

The system validates a reported violation

Name	Validation of violation
Actor	k users
Requirement	R3
Entry conditions	A violation must have been notified by a user
Event flow	1. A user send a violation report to SafeStreets 2. The system sends a request to a number k of users 3. Users answer the poll 4. The system counts the answers and makes the algebraic sum m, considering that: • +1 is a confirmation of the violation • 0 is an absent response or an agnostic report of the violation • -1 is a refusal of the violation 5. The system compares the calculated sum with the memorized acceptance threshold value v (a) If $m > v$ the system validates and register the report (b) If $m \leq v$ and the sample of users was less then g the report is discarded (c) If $m \leq v$ and the sample of users was greater or equal to g the report is discarded and the user who submitted it receives a penalty
Exit conditions	The system has validated and registered the violation
Exceptions	• Another violation report has already been sent in the previous h hours with the same violation category and license plate

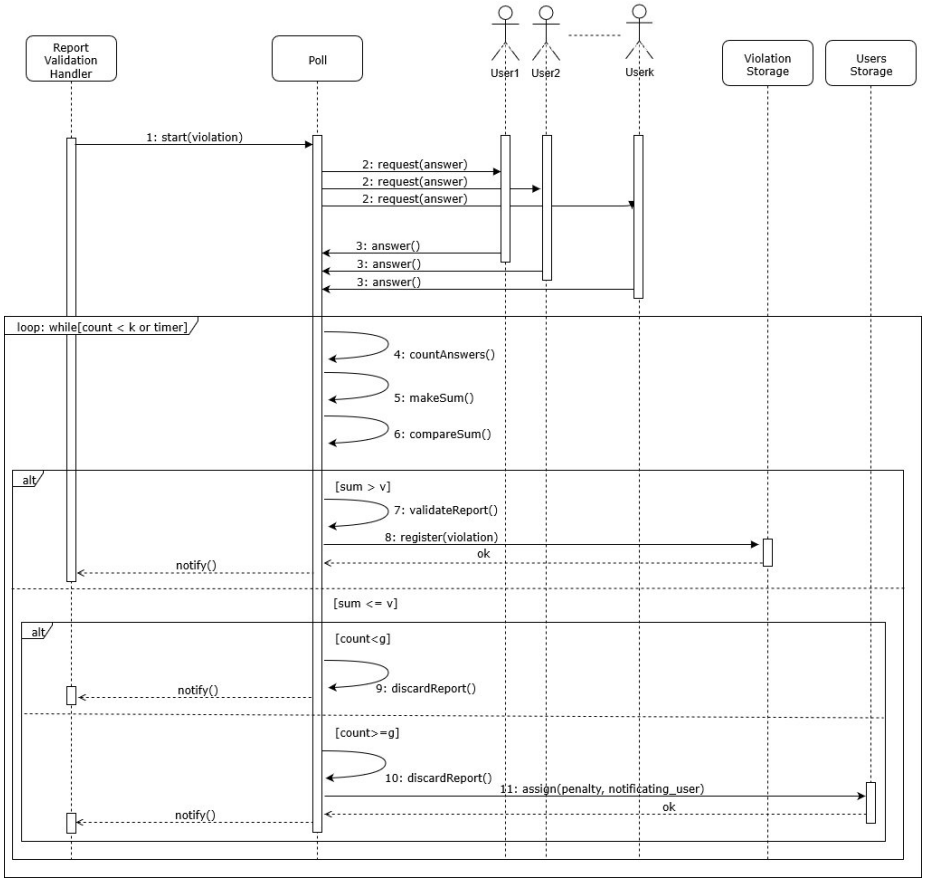

Figure 3.4: *Sequence diagram 3 - The system validates a reported violation*

3.2.4 R4: Allow a user to see the areas where a violation is more likely to happen

- SafeStreets can aggregate data inputted by users to show relevant statistics about the frequency of traffic violations
- Users can see how many violations usually happen in their neighborhood, in their current location or in an area of their choice
- A user has access to the type and amount of violations that happened
- Users can insert a specific address to check
- The system answers showing the nearest unsafe areas, if the address is located in a safe area
- The system answers showing more detailed data about frequency of car accidents and violations, if the address is in an unsafe area

Scenario 4

Mario Rossi searches for an English school for his daughter, he founds

different schools in his city on the Internet, but he wants an area not too far from his house where the little girl can securely go also on foot. He has to decide among five schools. He is a user of SafeStreets, so for every school in his list, he looks on the web interface of the application for the statistics about car accidents in all the five zones. He inserts the address of the school and the system responds with a map, showing the nearest unsafe areas if the school is in a safe area, or, if the school is in an unsafe area, showing more detailed data about frequency of car accidents and violations. This operation is repeated five times.

Use case

User looks for statistics for unsafe areas

Name	Viewing of statistics
Actor	User
Requirement	R4
Entry conditions	The user must be registered to the application and he must have logged on the web interface
Event flow	1. The user clicks on the "View statistics" button 2. The user inserts the address he looks for 3. The system shows a map with the area surrounding the inserted address
Exit conditions	The user has seen the statistics he required
Exceptions	• The area has insufficient data to provide statistically significant information

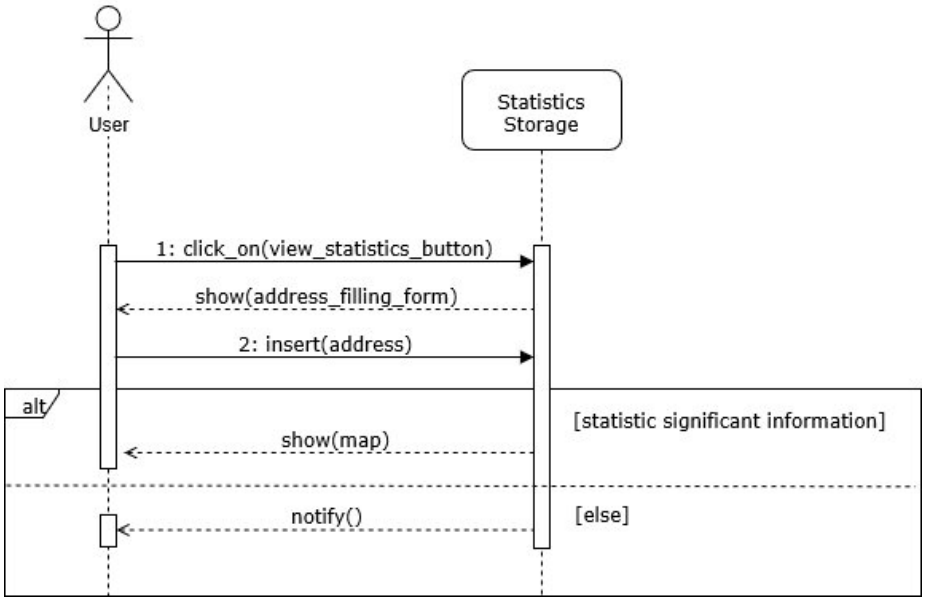

Figure 3.5: *Sequence diagram 4 - User looks for statistics for unsafe areas*

3.2.5 R5: Allow authorities to see data about violations and who committed them

- Authorities can access all data about violations that a standard user can access
- Authorities also have access to more specific data about who committed a violation, like the license plate of the car or how many infractions have been associated to a specific car
- In case the validated report includes a vehicle and the license plate is correctly identified, the authority can generate a traffic ticket

Scenario 5

A notification of a violation including a vehicle has just been made by a registered user and has been validated. The type of violation is "Stop in front of driveway" and the offender car, that has been identified with its license plate, was already present in the list SafeStreets uses to keep track of offenders. The system updates the number of infractions related to that car. An authority views the new infraction and generate a traffic ticket. The authority is also able to see the history of violations regarding the offender.

Use case

Viewing of data about violations and offenders by authorities

Name	An authority sees offenders profile and data
Actor	Authority
Requirement	R5
Entry conditions	A notification, including a vehicle, must have been made, validated and registered. The system has identified the license plate of the offender.
Event flow	1. The system updates the number of violations associated to that vehicle 2. The authority checks the history of violation related to the offender 3. The authority generates a traffic ticket
Exit conditions	The authority handled a violation report
Exceptions	• The license plate of the offender is not identified

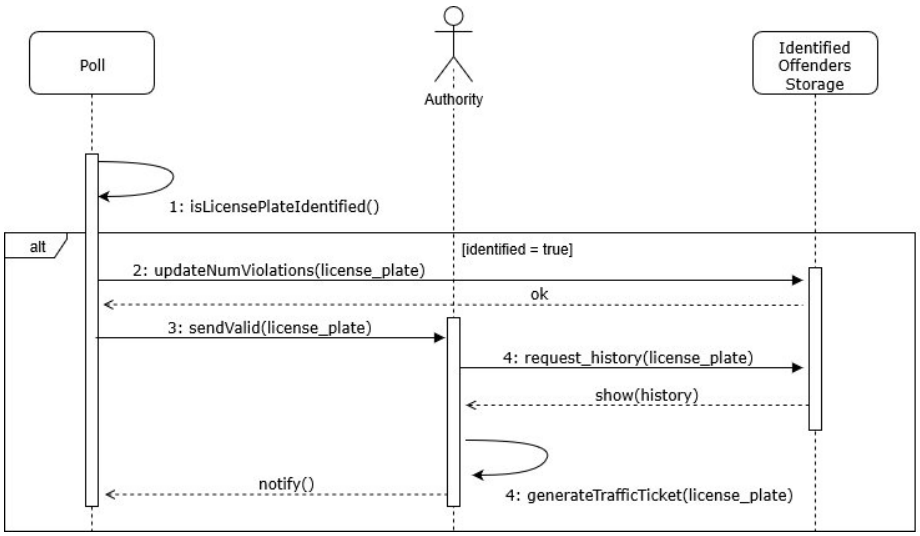

Figure 3.6: *Sequence diagram 5 - Viewing of data about violations and offenders by authorities*

3.2.6 R6: Allow authorities to register with a special user profile

- Authorities must first register as citizens
- A standard user profile can be upgraded to an authority profile if it is verified by the system

- To obtain privileged access, the user must provide a valid document that proves their authority status

Scenario 6

Paolo Brambilla is a policeman, he wants to register to SafeStreets and using his tablet he accesses the web interface, inserts his data for standard user registration (see Scenario 1) and becomes a registered user. He wants to upgrade his profile to be enabled, as a member of the local police, to access the most advanced functions of the system. He must click on the button "Upgrade profile", then he is asked to write his police ID badge number and his current district, then he clicks on the button "Upgrade to authority". The system verifies his ID badge number, sending it to the municipality that confirms or discards the identification of the policeman, checking his personal data, his area of competence and his badge. If the authority identity is validated, the profile is correctly upgraded. An confirmation email of the upgrade is sent to Paolo Brambilla's address. Now he can log to SafeStreets as an authority and has access to all the advanced functions of the system. If the authority identity is rejected, the system doesn't change anything in the user profile.

Use case

Registration of an authority

Name	Registration of an authority
Actor	User/Authority
Requirement	R6
Entry conditions	The user must be registered to the application and he must have logged in
Event flow	1. The user clicks on the "Upgrade profile" button 2. The system asks to insert data for proving their authority role 3. The user inserts the police ID badge number and their district 4. The user clicks on the "Upgrade to authority" button 5. The system sends data to municipality 6. The municipality validates data 7. The system saves the authority's data and upgrades the user profile to authority level 8. The system sends a confirmation email
Exit conditions	The user has become an authority
Exceptions	• The inserted district is not valid • The police ID badge is already registered • The police ID badge is not valid

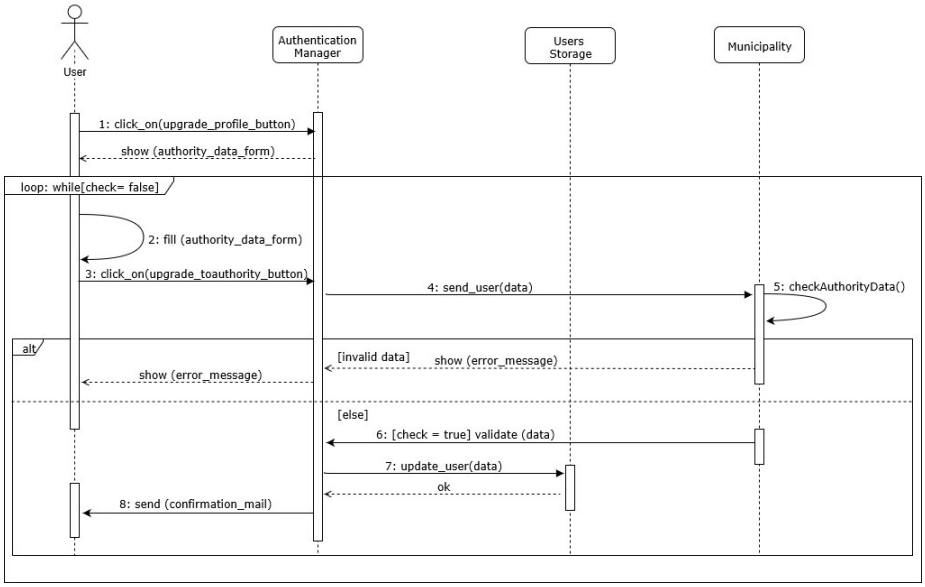

Figure 3.7: *Sequence diagram 6 - User registers as an authority*

3.2.7 R7: Detection of unsafe areas and possible interventions

- SafeStreets can cross information from different sources to identify potentially unsafe areas
- SafeStreets may be integrated with a public service, offered by the municipality, that provides such information
- Once identified, SafeStreets can suggest possible interventions to prevent accidents

Scenario 7

The system uses its own data, collected with reports of violation, and has the possibility to ask the municipality data about car accidents and violations in a specific area. This information is crossed to SafeStreets that integrates the given data with the information already saved and verified. In this way the storage keeps track of the unsafe areas, that are identified as those with violations notified by SafeStreets users, those with car accidents and damages provided by the municipality, and those areas that have both these information. On a regular basis, SafeStreets sends an email to the municipalities that can intervene on unsafe areas suggesting possible solutions identified using a special algorithm.

Use case

Unsafe areas identification

Name	Unsafe areas identification
Actor	SafeStreets Server
Requirement	R7
Entry conditions	The system must select a specific area to identify
Event flow	1. The system queries verified and validated reports of SafeStreets about the selected area 2. v violations are returned from reports 3. The system queries municipality about car accidents in the selected area 4. a accidents are provided by the municipality 5. The system determines unsafe areas by testing the equation $v + a > 0$ on the registered areas 6. Periodically, the system sends an email containing possible solutions to the municipalities that can intervene on the unsafe areas identified
Exit conditions	New statistics have been saved by the system

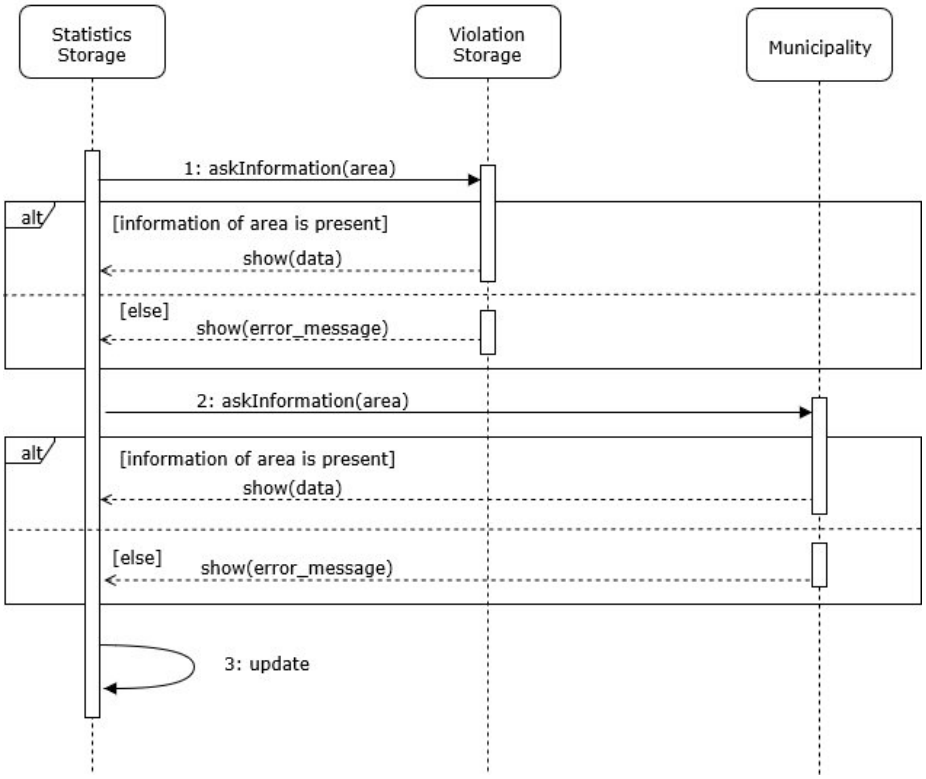

Figure 3.8: *Sequence diagram 7 - The system provides unsafe areas identification*

3.2.8 R8: Traffic ticket generation from validated reports

- SafeStreets can generate tickets for traffic violations reported from registered users
- The information is kept safe and intact through the chain of custody, from the end user to the local police officer issuing the tickets

Scenario 8

The information about the reports is stored in a database where privileges about the creation and modification of the information are distinct. Both the web interface and the mobile application can create new reports, only system administrators have full access to the database and thus it's their responsibility to guarantee that other actors don't alter the data. These conditions are necessary to allow SafeStreets tickets to have legal validity, thus enabling local authorities to automatically generate traffic ticket from the client application.

Use case

A traffic ticket is generated

Name	A traffic ticket is generated
Actor	Authority
Requirement	R8
Entry conditions	A violation has been approved
Event flow	1. The system creates new records in the database regarding the reported violation 2. Authorities can access report data in read-only mode 3. Through the web interface an authority can automatically generate a standard ticket, filling out the missing information
Exit conditions	Traffic ticket is generated
Exceptions	• A system administrator broke the chain of custody

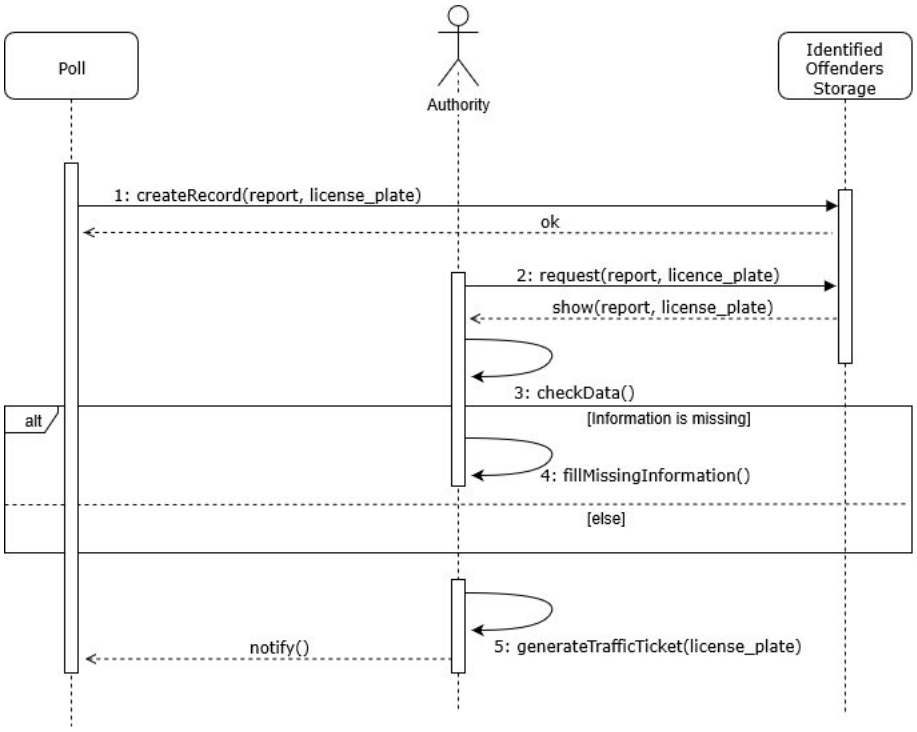

Figure 3.9: *Sequence diagram 8 - An authority provides municipality of information and traffic ticket is generated*

3.2.9 Traceability Matrix

Ref	Goals	Requirement	Domain assumptions
1	G1	R1, R2	D1, D5
2	G2	R4, R7	D8, D10
3	G3	R5, R8	D7, D9
4	G4	R3	D9, D11
5	G5	R1, R6	D2, D3, D6
6	G6	R5, R8	D1, D2
7	G7	R1, R3, R5, R6, R8	D1, D4, D5, D9, D11

To improve readability, here is a brief description of the goals and the related requirements and domain assumptions.

1. The goal is to allow citizens of legal age to report violations. This means that citizens should be allowed to register providing a document to confirm their age and identity and then SafeStreets should allow them to submit pictures along with a report.

2. The goal is to show statistics about the safety of different areas, and possibly offer some suggestions to make the worst areas safer. The statistics are going to be more reliable if the municipalities allow SafeStreets to access their data about violations, but still the safety of an area is computed only using previous data and its accuracy is strictly dependent on the amount of reports that SafeStreets has access to.

3. The goal is to show the violations to the authorities, in order to enable them to take action. This means that SafeStreets must have a way to verify that a user is actually an authority, because only authorities are allowed to generate traffic tickets.

4. The goal is to validate reports through a reliable procedure. This means that after a report is sent, it can't be modified.

5. The goal is to provide secure authentication methods, for example by allowing two-factors authentication.

6. The goal is to securely store user data and make it available only to authorized people.

7. The goal is to give legal validity to approved violation reports, so that tickets can be generated: only users with a verified profile and who are of legal age are allowed to submit reports, and once generated, a report can't be altered.

3.3 Performance Requirements

3.3.1 Performance features

- 90% of the API calls should be completed within 4.5 seconds
- The response times during the testing phase will be measured using HP LoadRunner (or similar tools) located behind the firewall and in front of the web servers

- Real response times will be measured using the application server log files
- The system will be deployed to a machine able to serve a huge number of users in parallel

3.3.2 Evolution of the system

- At the beginning the system should be able to handle up to 10'000 users
- As the number of users will grow in unpredictable ways, at first the system will be deployed on cloud infrastructure that enables automatic up and down scaling (e.g. Azure Autoscale or AWS Elastic)
- Due to the cost of such infrastructures, as the number of users will become more stable the system will be migrated to a fixed resource machine

3.3.3 Performance testing

- Automatic monthly tests to check performance will be executed during system idle time which is determined statistically
- The performance tests should not exceed an execution time of 15 minutes

3.4 Design Constraints

3.4.1 Standards compliance

- To guarantee the compatibility with the potential municipality accident information service, our backend software should communicate and be able to read data in a portable format like JSON or XML
- To transfer this data, the software should expose an HTTP REST API

3.4.2 Hardware limitations

- The frontend application should be as lightweight as possible to support the diversity of mobile device hardware
- Devices with a camera resolution of less than 2MP should be marked as incompatible with SafeStreets due to the poor quality of the pictures that are taken with them

3.4.3 Other constraints

- The language used to develop the backend application will be chosen from the ones that are most supported by the main cloud infrastructure providers, in order to have a wider choice of hosting plans

3.5 Software System Attributes

3.5.1 Reliability

3.5.1.1 Index MTBF

- Index MTBF must consider as failures those out of design conditions which place the system out of service, for example by an overload of user's requests
- MTBF is of 5,000 hours

3.5.1.2 Index MTTR

- Index MTTR must consider time of testing and solution of bugs
- MTTR must be less than two hours

3.5.1.3 The position should be verifiable

- The system expects to receive one or two pictures
- The received pictures must be of at least 2MP
- The first picture must represent a general overview of the violation, to identify the street
- The second picture must allow the system to identify correctly the license plate, if the violation includes a vehicle

3.5.2 Availability

- The system must be available 99.9%
- The system must be available when performing the standard routine tests for maintenance
- The system can bear slowdowns of the service in case of extraordinary overload of requests
- The system can become unavailable for some seconds due to extraordinary maintenance or major updates

3.5.3 Security

3.5.3.1 Secrecy of data received and of users' information

- The system should never allow both non-registered users and registered users to see the identity of the person that notifies the violation
- User personal data must be encrypted and safely stored
- Users' passwords must be salted and hashed before being stored on any persistent medium

- Every user can log in using their email and password, but they can optionally enable Two-Factor authentication to improve security
- Violation reports must be encrypted before being sent to the server to preserve the secrecy and integrity in the chain of custody

3.5.3.2 Integrity of data

- The information sent by users (picture, date, time and position of the violation) has to be encrypted in order to keep the integrity and to avoid manipulation of the data

3.5.3.3 Measures of security according to danger level

- Public statistics to identify the most dangerous areas are provided for SafeStreets
- SafeStreets makes a ranking of dangerousness (minimum, medium, high)
- The authorities can decide to improve security by highlighting streets in areas at minimum risk
- The authorities can decide to improve security by adding cameras in the areas at medium risk
- The authorities can decide to improve security by both adding cameras and doubling patrol shifts in the areas at large risk

3.5.4 Maintainability

3.5.4.1 Testing overview

- The system is checked in its correct functioning by software testing
- Unit tests, Integration tests and System tests are performed before and after every update

3.5.4.2 The system is controlled and monitored

- The system is equipped with condition-monitoring algorithms
- The condition-monitoring algorithms identify the functions that cause alarm
- A unit test is made as soon as a function is identified as potentially at risk
- An integration test is performed after the unit test of the potentially at risk function
- A system test is performed after the integration and unit test of the potentially-at-risk function

3.5.4.3 The code must be clean and easy to understand

- Code must follow common best practices
- It is advisable to follow the design patterns to provide a standard terminology and to make the software more adaptable to future extensions and improvements
- Complete and detailed documentation, even for low level functions, is mandatory in order to keep the maintainability on the highest level on the whole system

3.5.5 Portability

- The software shall run on Windows, macOS, Linux, Android and iOS through a modern web browser supporting at least ECMAScript 2015
- To guarantee the portability of the front end application the software shall be developed using a web framework such as React
- This approach for the front end application will enable code reuse among native mobile platforms (iOS and Android) and the Web Application
- The back end software shall be developed in a language that can be compiled to run on virtual runtime such as Java, JavaScript or C# which can target the JVM, Node.js and dotnet core respectively
- The software can be developed using the standard APIs that span different types of operating systems
- The software will then be able to be transferred with no modifications on other destination machines
- The architecture must be flexible

Chapter 4

Formal using Alloy

4.1 Signatures

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

sig ViolationCategory {}

sig LicensePlate {}

sig Location {
  latitude: one Int,
  longitude: one Int
}

sig Picture {}

sig Email {}

sig IdNumber {}

sig PhoneNumber {}

sig CharSeq {}

sig Municipality {
  name: one CharSeq,
  province: one CharSeq
}

sig User {
  name: one CharSeq,
  surname: one CharSeq,
  email: one Email,
  phoneNumber: lone PhoneNumber,
  password: one CharSeq,
  idNumber: one IdNumber
}

sig Authority extends User {
  badgeNumber: one CharSeq,
  district: one Municipality
}

sig Report {
  pictures: set Picture,
  location: one Location,
  timestamp: one Int,
  author: one User,
  licensePlate: lone LicensePlate,
```

```
    violationCategory: one ViolationCategory,
    isValid: lone Boolean
  }{
    #pictures > 0
  }

  sig PollAnswer {
    value: one Int,
    author: one User
  }{
    -1 ≤ value ∧ value ≤ 1
  }

  sig Poll {
    report: one Report,
    pollAnswers: set PollAnswer,
    isCompleted: one Boolean,
    score: one Int
  }

  sig Ticket {
    authority: one Authority,
    report: one Report,
    fine: one Int
  }

  sig Area {
    points: set Location,
    score: one Int,
    reports: set Report
  }{
    #points > 0
    #reports > 0
  }
```

4.2 Facts

```
  open signatures

  fact pollStatus {
    all p: Poll |
      p.isCompleted = True ⟺ #p.pollAnswers ≥ 5
  }

  fact uniqueIdNumber {
      no disj u1, u2: User |
          u1.idNumber = u2.idNumber
  }

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

  fact uniqueBadgeNumber {
      no disj a1, a2: Authority |
          a1.idNumber = a2.idNumber
  }

  fact uniquePoll {
      no disj p1, p2: Poll |
          p1.report = p2.report
  }

  fact onePollPerReport {
      all r: Report | one p: Poll |
          (p.report = r)
  }

  fact pollScore {
      all p: Poll | p.score = (sum a: p.pollAnswers | a.value)
  }
```

```
36
37  fact validatedReport {
38      all r: Report | one p: Poll |
39          (p.report = r
40          ∧ (
41              (
42                  p.isCompleted = False
43                  ∧ r.isValid = none
44              )
45              ∨
46              (
47                  p.isCompleted = True
48                  ∧ (r.isValid = True ⟺ p.score > 2)
49                  ∧ (r.isValid = False ⟺ p.score ≤ 2)
50              )
51          ))
52  }
53
54  fact authorDoesNotVoteTheirReport {
55      all p: Poll | no a: PollAnswer |
56          a in p.pollAnswers
57          ∧ p.report.author = a.author
58  }
59
60  fact noDoubleVoting {
61      all p: Poll | no disj a1, a2: PollAnswer |
62          a1 in p.pollAnswers
63          ∧ a2 in p.pollAnswers
64          ∧ a1.author = a2.author
65  }
66
67  fact noMultipleReports {
68      all r1, r2: Report |
69          (r1.violationCategory = r2.violationCategory
70          ∧ (r1.timestamp > r2.timestamp)
71          ∧ (r1.timestamp - r2.timestamp) < 4)
72          ⟹ r1.licensePlate ≠ r2.licensePlate
73  }
74
75  fact noMultipleTickets {
76      no disj t1, t2: Ticket |
77          t1.report = t2.report
78  }
79
80  fact pollAnswerAssociation {
81      all a: PollAnswer | one p: Poll |
82          a in p.pollAnswers
83  }
84
85  fact ticketsForValidReportOnly {
86      all t: Ticket | one r: Report |
87          t.report = r ∧ r.isValid = True
88  }
89
90  fact areaReports {
91      all a: Area | all r: a.reports |
92          r.location in a.points
93  }
94
95  fact allReportsHaveAnArea {
96      all r: Report | some a: Area |
97          r in a.reports
98  }
```

4.3 Assertions

```
1  open facts
2  open signatures
3
4  assert noDoubleTicketing {
5      no disj t1, t2: Ticket |
6          t1.report.violationCategory = t2.report.violationCategory
```

```
        ∧ t1.report.timestamp > t2.report.timestamp
        ∧ t1.report.timestamp - t2.report.timestamp < 4
        ∧ t1.report.licensePlate = t2.report.licensePlate
}

check noDoubleTicketing for 10
```

4.4 Predicates

```
open facts
open signatures

pred show {}

pred someValidatedReport {
    some p: Poll | one r: Report |
        p.report = r
        ∧ r.isValid = True
        ∧ p.isCompleted = True
}

pred someTickets {
    some t: Ticket | one r: Report |
        t.report = r
        ∧ r.isValid = True
}

pred someAreas {
    #Area = 1
}

run show for 2 but
    exactly 1 Report,
    exactly 2 User,
    exactly 2 IdNumber,
    exactly 1 Authority,
    exactly 1 LicensePlate,
    exactly 2 PhoneNumber,
    exactly 2 Email,
    exactly 1 Municipality
run someValidatedReport for 1 but
    exactly 1 Report,
    exactly 6 User,
    exactly 0 Authority,
    exactly 6 Email,
    exactly 6 IdNumber,
    exactly 5 PollAnswer,
    exactly 1 Poll
run someTickets for 1 but
    exactly 1 Report,
    exactly 6 User,
    exactly 1 Authority,
    exactly 6 Email,
    exactly 6 IdNumber,
    exactly 5 PollAnswer,
    exactly 1 Poll
```

4.5 Alloy Execution Results

The results of the execution of the Alloy Analyzer are reported in the following pages

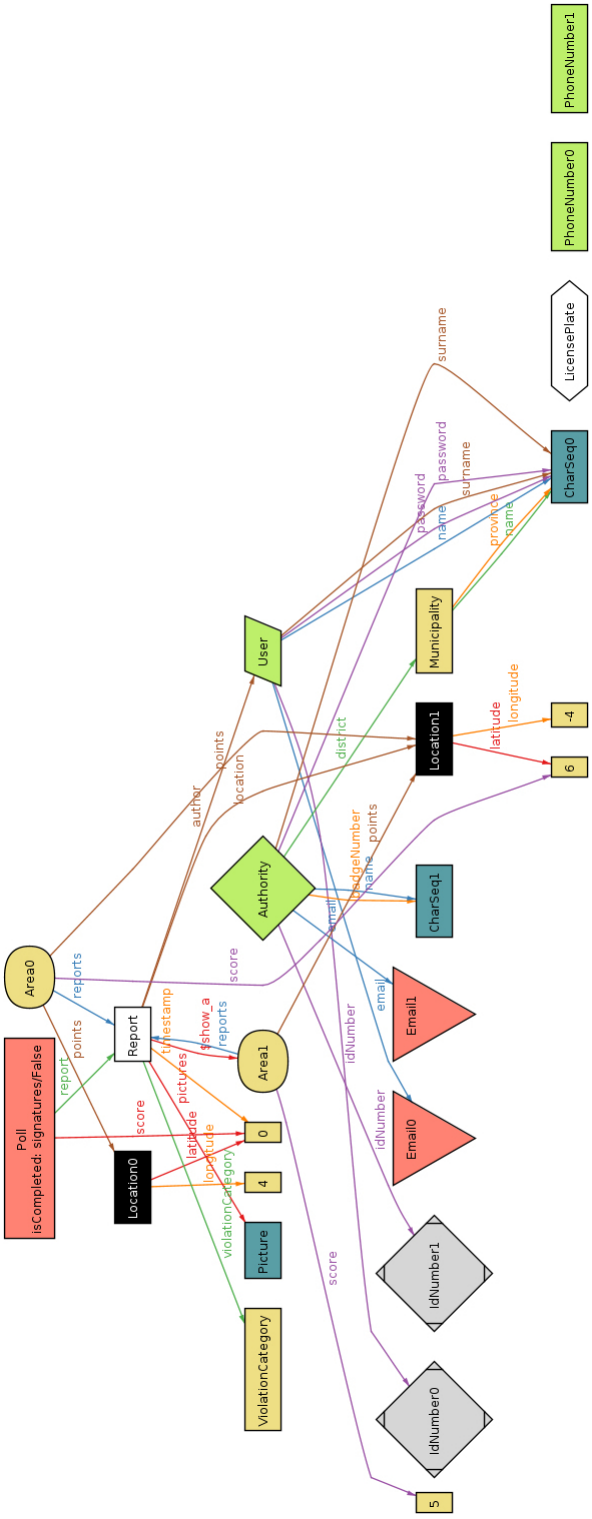

Figure 4.1: *Example Alloy World*

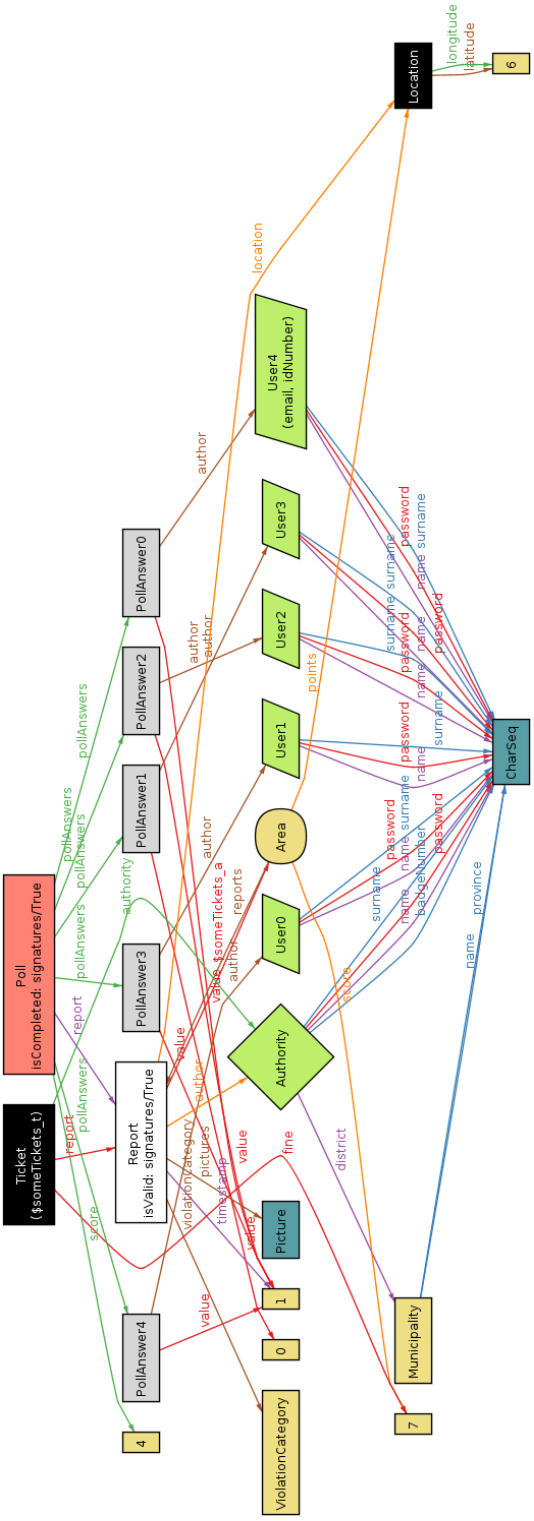

Figure 4.2: *Visualizing a Ticket instance*

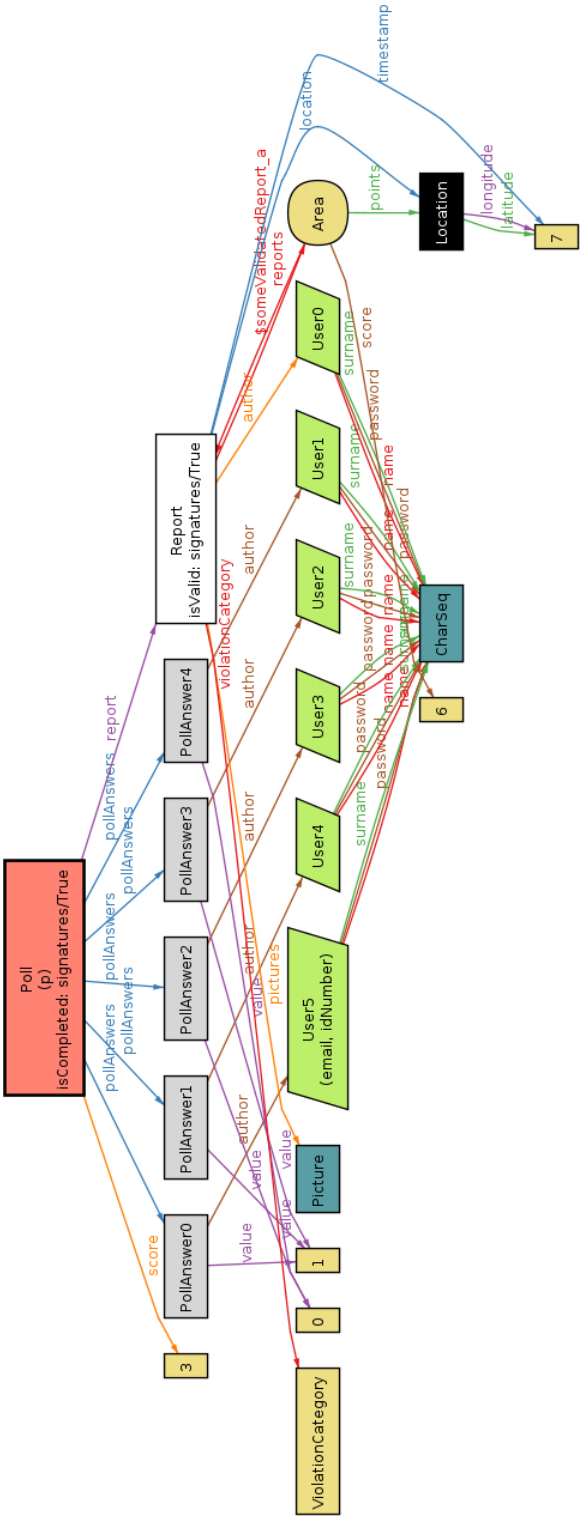

Figure 4.3: *Visualizing a Validated Report instance*

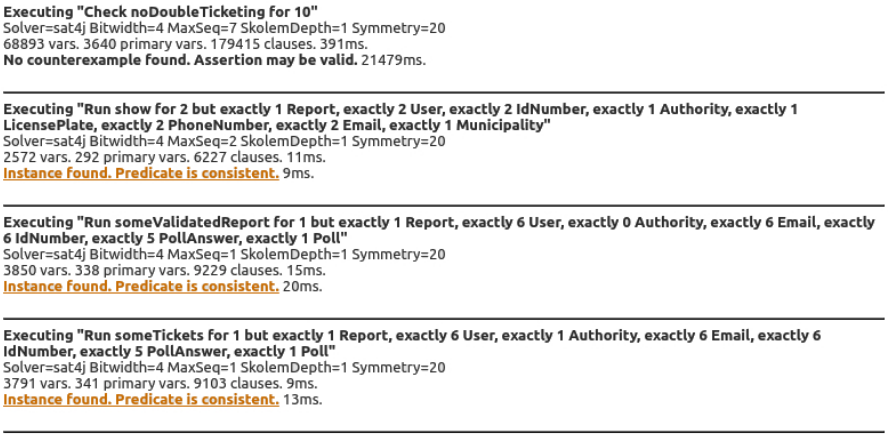

Figure 4.4: *Alloy execution results*

Chapter 5

Effort spent

Carlo Dell'Acqua

Task	Time spent (hours)
Project setup	2
Functional requirements	5
Design constraints	0.5
General Review	3
Introduction	1
App Images	3.5
Alloy	8.5
UML	1
Traceability Matrix	1
Group Sessions	4

Adriana Ferrari

Task	Time spent (hours)
Project Setup	2
Functional Requirements	4
Design Constraints	0.5
Product Perspective	0.5
Product Functions	0.5
General Review	2
Alloy	6
Domain Assumptions	1
User Characteristics	0.5
Dependencies and Constraints	1
External Interface Requirements	0.5
Traceability Matrix	2
References	1
Group Sessions	4

Angelica Sofia Valeriani

Task	**Time spent (hours)**
Software System Attributes	2
Performance Requirements	1
Domain Assumptions	1
External Interface Requirements	1
User Characteristics	1
Scenarios	3
Use cases	3
General review	1
Definitions, acronyms, abbreviations and document structure	0.5
Traceability Matrix	1
Sequence Diagrams	5
UML	2.5
State Diagrams	2
Graphical review	0.5
Alloy review	1
Revision History	0.5
Group Sessions	4

References

[1] LaTeX for writing the document
[https://www.latex-project.org/]

[2] Alloy Latex Highlighter, a LaTeX package to highlight alloy syntax that we modified to render improve the visualization of some operators
[https://github.com/Angtrim/alloy-latex-highlighting]

[3] Title Pic, a LaTeX package that allows customization of the cover page
[http://tug.ctan.org/tex-archive/macros/latex/contrib/titlepic/titlepic.sty]

[4] draw.io to draw UML diagrams and sequence diagrams
[https://www.draw.io/]

[5] Alloy Analyzer to generate Alloy worlds and run alloy predicates
[https://alloytools.org/]

[6] tocbibind, a LaTeX package to bind the bibliography to the table of contents
[https://ctan.org/pkg/tocbibind]

[7] GitHub for version control
[https://github.com]

[8] Visual Studio Code the IDE we used
[https://code.visualstudio.com/]

[9] Wikibooks we read the section about LaTeX to learn how to structure the document
[https://en.wikibooks.org/wiki/LaTeX/]

[10] Stack Exchange for further doubts about LaTeX
[https://tex.stackexchange.com/]

[11] Ionic to generate mockups
[https://ionicframework.com/]

[12] GIMP to edit mockups
[https://www.gimp.org]