POLITECNICO
MILANO 1863

SafeStreets

Requirements Analysis and Specification Document

Giancarlo Danese - 945265
Davide Savoldelli - 928676

A.Y. 2019/2020 - Prof. Di Nitto Elisabetta

Contents

1 INTRODUCTION

1.1 Purpose

This documents represents the Requirements Analysis and Specification Document (RASD) for the SafeStreets software project. The goal of this project is to deploy a system that allows users to report traffic/street violations to authorities.
Users will have the possibility to send pictures of the violation, together with the type of violation and position of the vehicle. The software focuses particularly on parking violations, including double parking or parking in a disabled-reserved spot.

1.1.1 Goals

- [**G1**]: Allow users to register to the system with their personal data
- [**G2**]: Allow users to report traffic/parking violations
- [**G3**]: Allow users to check their reports history
- [**G4**]: Allow users to check/modify their account info
- [**G5**]: Allow authorities to register to the system with their personal data
- [**G6**]: Allow authorities to observe/request a Suggestion
- [**G7**]: Allow authorities to discard non-valid photos

1.2 Scope

SafeStreets will have an embedded algorithm which will analyze pictures of the vehicle plates sent by the user in order to recognize the vehicle. This information, together with the position of the vehicle and the type of violation that has been committed, will be stored in the software's database.
Authorities will have the chance to observe the information retrieved and mined from the database, which highlights the streets/areas in which most of the violations are committed.
Moreover, authorities can observe the type of vehicles which commit most of the violations, which type of violations occur the most and some suggestions by Safestreets on what possible interventions can be taken on those streets.

1.3 Definitions, Acronyms, Abbreviations

- `User Device`: any compatible device with the SafeStreets application, like a smartphone or a computer
- `Personal Information`: information provided by the user during the registration process. It includes name, surname, birth date, address, e-mail address, mobile number.
- `Violation Report`: the act in which users can denounce violations on the streets, by providing the system its position, a photo and by selecting a violation from a precompiled menu
- `Mobile App`: an application that can be run by mobile devices, both smartphones and smartwatches.
- `Violations Map`: a map, accessible only by authorities, which contains notifications and alerts about all the unsafe areas where several violations are committed
- `Suggestions`: an embedded algorithm will automatically suggest interventions to take in streets particularly full of violations

1.3.1 Acronyms

- RASD: Requirements Analysis and Specification Document.
- API: Application Programming Interface.
- GPS: Global Positioning System.
- PRA: Pubblico Registro Automobilistico
- AUC: Authority Unique Code
- DBP: Device-bound PIN

1.4 Revision history

- `27/11/2019` Added Web Service/Server to all the Sequence Diagrams, added the Suggestion Request sequence diagram and corrected some discrepancies/clarifications regarding the Suggestions generating processes
- `03/12/2019` Added Suggestion Requesting Use Case, added Third Party actor and corrected more discrepancies/clarifications
- `12/12/2019` Removed Third Party Actor

1.5 Reference Documents

- Specification Document "SafeStreets Mandatory Project Assignment"
- Document "5.b Structure of RASD"
- Alloy Language Reference "http://alloytools.org/download/alloy-language-reference.pdf"

1.6 Document Structure

This paper refers to the structure suggested by IEEE for RASD documents, with some modifications:

1. `Introduction`: This first section is a general description of the system's scope and goals. It also includes the revision history of the document and its references, definitions, abbreviations and acronyms used along the paper.
2. `Overall Description`: This section includes shared phenomena, requirements and domain assumptions. It also clarifies users' needs.
3. `Specific Requirements`: This section is made up of all the requirements needed for the system, both functional and non functional.
4. `Formal Analysis Using Alloy`: It includes the Alloy model of the described system.
5. `Effort Spent`: In this section you can find information about the hours spent to draft this document.
6. `References`: Here are the references about papers/documents used to support this document.

2 OVERALL DESCRIPTION

2.1 Product perspective

Here you can see a high level UML Class Diagram. It describes the systems' structure and the relationship between the parts.

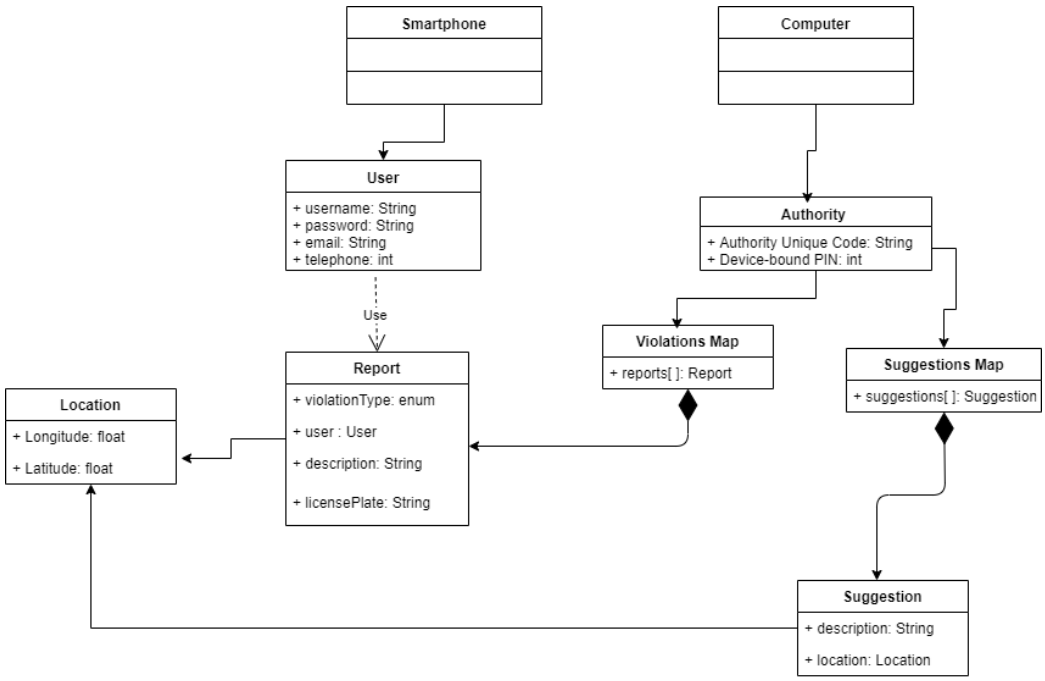


Figure 1: UML Class Diagram

Finally, below are two UML Statechart Diagrams, one for the user and the second one for authorities

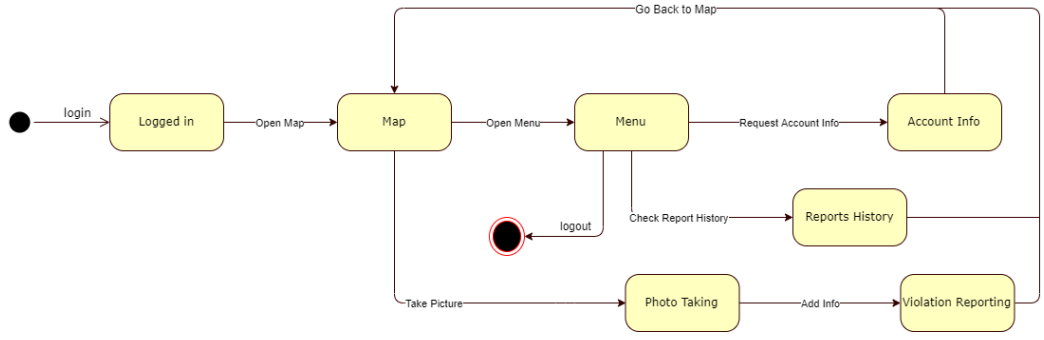


Figure 2: Users' Statechart Diagram

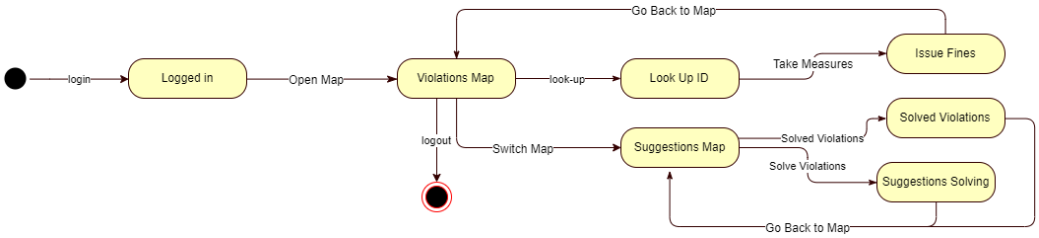


Figure 3: UML Class Diagram

2.2 Product functions

According to the goals defined in the first section, we can list the most important functions the system needs to work correctly:

- `Registration and Login`: This functionality enables users to register to the system by inserting their personal data, and to login with their chosen username and password. The system must therefore store the data in it's database and retrieve it when queried.

- `Violation Reporting`: The system must provide users with an in-app photo sending functionality where users can send pictures of the vehicles' plate, select the type of violation from the dropdown menu and let their GPS track down their position.

- `Reports History`: The system must allow users to access their reports history, where all those users' previous reports are stored with their data (type, location etc.)

- `Violations/Suggestions Map`: The software must cross its report positions with the municipality to identify unsafe areas and highlight them on a map together with their suggestions.

- `Violation Solving`: The system must enable authorities to access the map of violations and as soon as measures are taken, check them as "solved".

2.3 User characteristics

- `Registered User`: A person who registered to SafeStreets, sharing his personal data. He can login to the system by providing credentials and exploit all the services.
- `Authorities`: A Law Enforcement agent who has direct access to the users' reports on their web app and can issue warnings, fines and measures.

2.4 Assumptions, dependencies and constraints

- [**D1**]: Users' must provide valid credentials when registering to the system
- [**D2**]: The license plate must be clearly visibile in the photo
- [**D3**]: Users' device must support the GPS technology
- [**D4**]: Users' device must support Mobile Applications
- [**D5**]: Users' device must be connected to the internet
- [**D6**]: Users' device must have a camera
- [**D7**]: Users' must always take pictures of vehicles only
- [**D8**]: Users' can't load photos from their device
- [**D9**]: The verification e-mail/SMS will be received by the user
- [**D10**]: Authorities must have access to the internet
- [**D11**]: Authorities must be provided with their authority identification
- [**D12**]: Authorities must always check the validity of the photos taken

3 SPECIFIC REQUIREMENTS

3.1 External Interface Requirements

3.1.1 User Interfaces

These mockups represent an idea of how the user's interface will be shown and easily accessed, spanning from the simple registration and login to the violation report function.

You can also notice how SafeStreets provides the authorities' with a Web App which is separated both in functions and in design from the normal mobile app destinated to users.

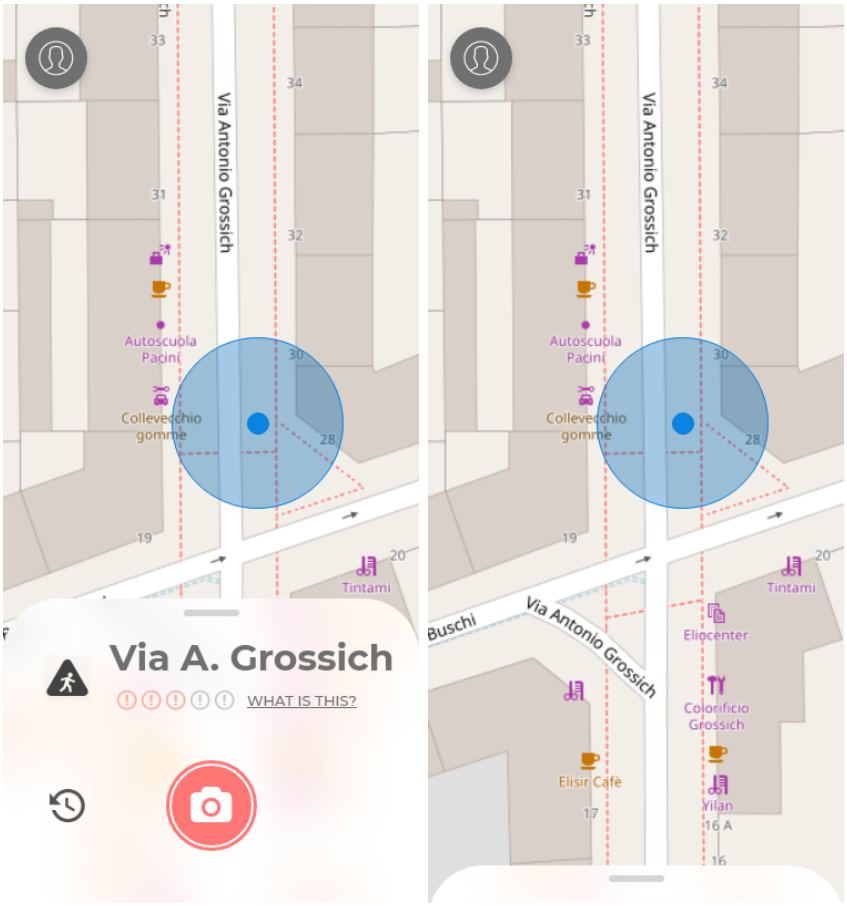


Figure 4: The Homepage with its photo button

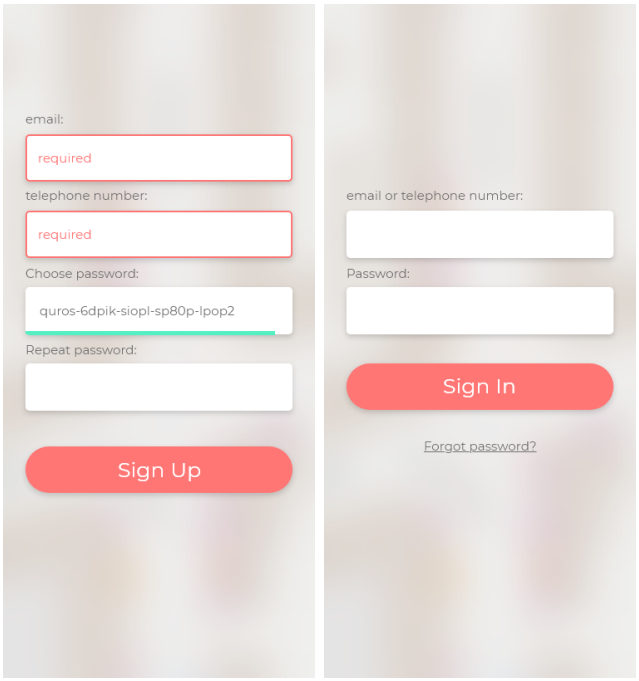


Figure 5: The registration and login pages where you input your credentials

Figure 6: Snapshots from the photo-taking screen and the confirmation photo

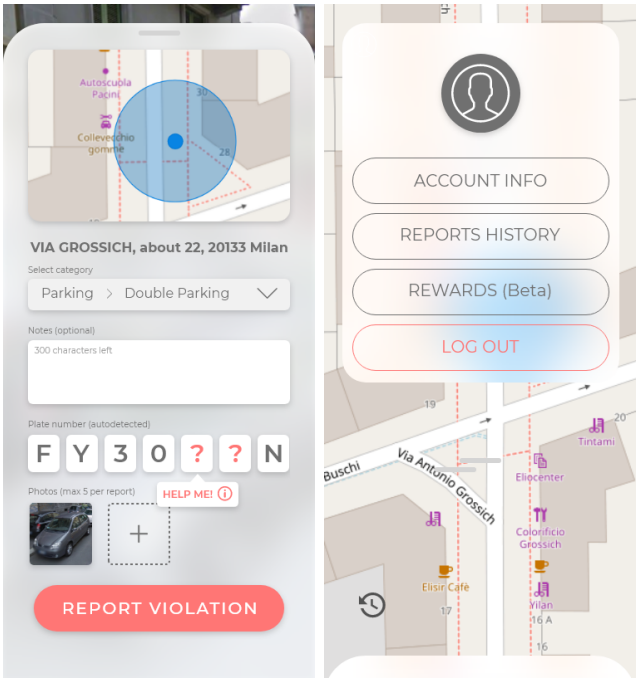


Figure 7: The Violation Reporting functionality and the menu

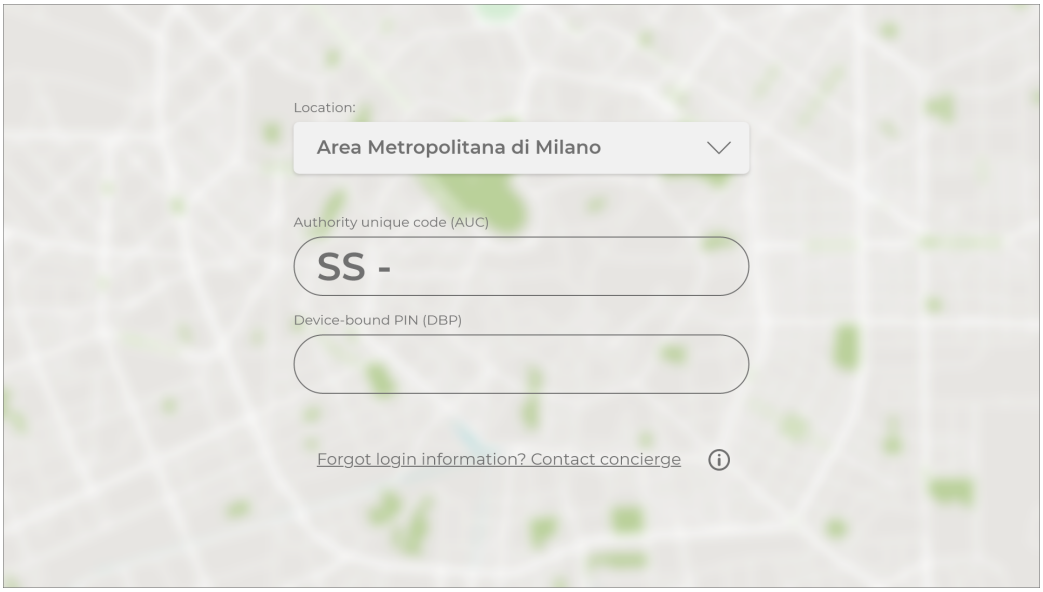


Figure 8: The login page for authorities

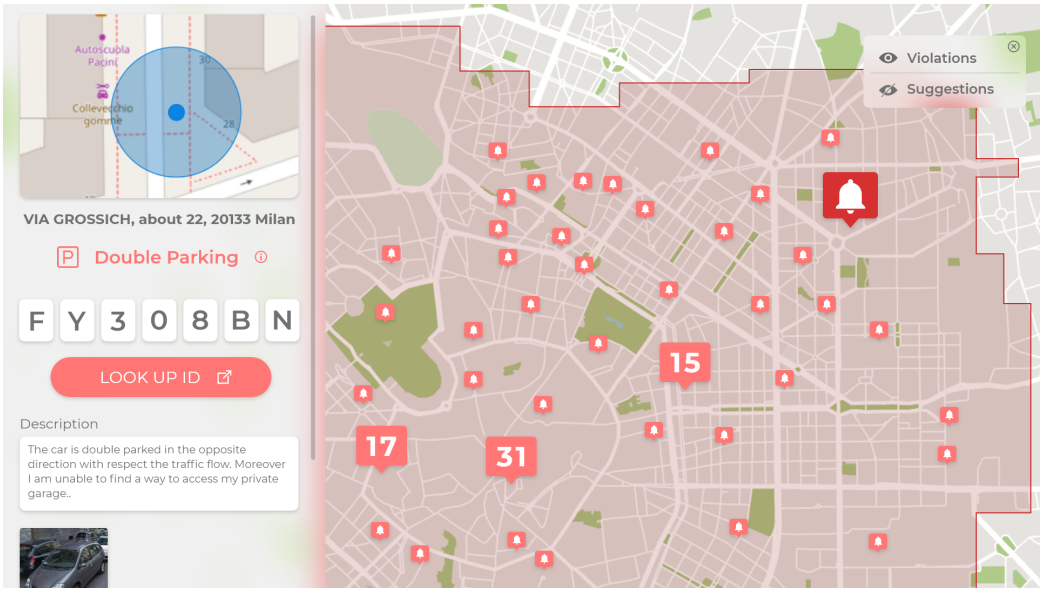


Figure 9: The Violations Map where authorities can solve violations or look-up license plates

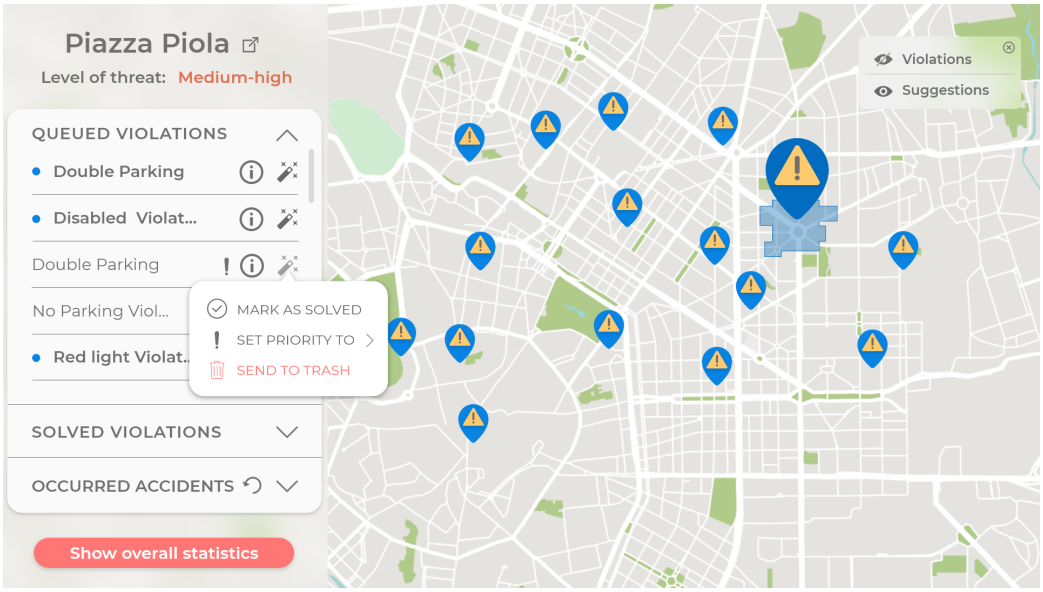


Figure 10: The Suggestions Map where authorities can observe the suggestions Safestreets provided them

3.1.2 Hardware Interfaces

The application must run over the Internet, therefore all the hardware must connect to the network and there will be an hardware interface for the system, both server and client side.

- Server-side: e.g. Modem, WAN - LAN, Ethernet Cross-Cable.
- Client-side: e.g. Wi-Fi 802.11ac+ antenna, 3G/4G antenna.

3.1.3 Software Interfaces

SafeStreets will provide an API for the user, together with the already Web Application Interface for authorities.
More specifically, it will provide an API that allows users to report violations and check their reports history, and an API for authorities which enables them to check the Violations/Suggestions Map and take measures.

3.1.4 Communication Interfaces

Exploits all the basic network information transporting protocols, such as TCP and UDP, sockets and RMI

3.2 Scenarios

Scenario 1 - User Registration

Marco decides to contribute to his cities' street security by registering to SafeStreets. He opens the app, and he inserts his personal data, such as name, surname and e-mail. He then decides a username and a password to access the app. He is now ready to report violations and secure the city.

Scenario 2 - Authority Registration

Giovanni is a policeman who wants to register as an authority on the system. He makes a special request, sending his personal information and his authority identification to the system which checks it's validity and sends back to Davide an Authority Unique Code (AUC), which he will need to login, and a Device-bound PIN (DBP), which is bound to special devices in dotation to authorities only. He can now check the users' reports, issue fines and check the Unsafe Areas Map.

Scenario 3 - Violation Report

Roberto is a guy who lives in a street constantly full of double-parked cars. Tired of this situation, he decides to report them to the authorities throughout SafeStreets. He takes a picture of the car with the license plate fully visible, he lets the app track down his position and he selects a type of violation from the dropdown menu.

Scenario 4 - Suggestion Observation

Ciro is in his office watching the Unsafe Areas map of his city. In particular, he notices that there's an area in his city full of reports and violations of cars parking in the bike lane and therefore occupying it. He also notices how SafeStreets' hint is to place a protection barrier parallel to the edge of the bike lane in order to prevent cars from parking on them. So he decides to take those measures following SafeStreets suggestions

3.3 Functional Requirements

[G1]: Allow users to register to the system with their personal data

- [D1]: Users' must provide valid credentials when registering to the system
- [D4]: Users' device must support Mobile Applications
- [D5]: Users' device must be connected to the internet
- [D9]: The verification e-mail/SMS will be received by the user

- [R1]: The system must allow users to provide their credentials and personal data
- [R2]: The system must let the user verify his account with his e-mail or by SMS
- [R3]: The system must verify there are no other registered users with the same e-mail or telephone number
- [R4]: In order to register successfully, the system must oblige the user to accept the data privacy policies and conditions

[G2]: Allow users to report traffic/parking violations

- [D2]: The license plate must be clearly visibile in the photo
- [D3]: Users' device must support the GPS technology
- [D5]: Users' device must be connected to the internet
- [D6]: Users' device must have a camera
- [D7]: Users' must always take pictures of vehicles only
- [D8]: Users' can't load photos from their device

- [R5]: The system must give the user the possibility to take pictures
- [R6]: The system must retrieve the users' position correctly

- [R7]: The system must recognize the license plate
- [R8]: The system must give the user the chance to select a violation type from the dropdown menu
- [R9]: The system musn't give the user the possibility to load photos from their device
- [R10]: If the users' wants to, the system must let them write an optional description.

[G3]: Allow users to check their reports history

- [D4]: Users' device must support Mobile Applications
- [D5]: Users' device must be connected to the internet

- [R11]: The system must show to the user all the violations he reported in the past

[G4]: Allow users to check their account info

- [D4]: Users' device must support Mobile Applications
- [D5]: Users' device must be connected to the internet

- [R12]: The system must show to the user his account information and personal data when requested

[G5]: Allow authorities to register to the system with their personal data

- [D10]: Authorities must have access to the internet
- [D11]: Authorities must be provided with their authority identification

- [R13]: The system must allow authorities to provide their credentials and personal authority identification

- [R14]: The system must verify there are no other registered authorities with the same identification
- [R15]: In order to register successfully, the system must oblige the authority to accept the data privacy policies and conditions

[G6]: Allow authorities to observe/request a Suggestion

- [D10]: Authorities must have access to the internet
- [D12]: Authorities must always check the validity of the photos taken

- [R16]: The system must allow authorities to request a suggestion from SafeStreets
- [R17]: The system must allow authorities access the Suggestions Map and see which suggestions have been posted for each area

[G7]: Allow authorities to discard non-valid photos

- [D10]: Authorities must have access to the internet
- [D12]: Authorities must always check the validity of the photos taken

- [R18]: The system must allow authorities to discard pictures which don't represent a vehicle
- [R19]: The system must allow authorities to discard pictures of vehicles which aren't committing any violation

3.4 Use Cases

3.4.1 Visitor Registration

NAME	Visitor registers to SafeStreets
ACTOR	Visitor
ENTRY CONDITIONS	The visitor has installed the app on his/her mobile device
EVENTS FLOW	1. The visitor fills all the mandatory fields in the Registration Form with his credentials 2. The visitor clicks on the "Register" button 3. The system checks the datas' validity 4. The system sends an SMS/e-mail as confirmation 5. The system saves the users' personal data
EXIT CONDITIONS	The visitor has successfully registered to SafeStreets
EXCEPTIONS	1. The e-mail is already registered 2. The mobile number is already registered 3. Some mandatory fields are not filled

3.4.2 User Login

NAME	User login to SafeStreets
ACTOR	User
ENTRY CONDITIONS	The user is registered to the system and is on the login page
EVENTS FLOW	1. The user enters his/her e-mail or telephone number 2. The user enters the password 3. The users clicks on the "Login" button 4. The system checks the credentials validity
EXIT CONDITIONS	The visitor has successfully logged in to SafeStreets
EXCEPTIONS	1. The User is not registered 2. The e-mail is wrong 3. The mobile number is wrong 4. The password is wrong 5. Some mandatory fields are not filled

3.4.3 Authority Registration

NAME	Authority registers to SafeStreets
ACTOR	Authority
ENTRY CONDITIONS	The authority has installed the web app on his/her computer
EVENTS FLOW	1. The authority sends his ID document and his authority identification number 2. The system checks the datas' validity 3. The system sends back to the authority an AUC and DBP 4. The system saves the authorities' personal data
EXIT CONDITIONS	The authority has successfully registered to SafeStreets
EXCEPTIONS	1. The AUC is already registered 2. The DBP is already registered 3. Some mandatory fields are not filled

3.4.4 Authority Login

NAME	Authority login to SafeStreets
ACTOR	Authority
ENTRY CONDITIONS	The authority is registered to the system and is on the login page
EVENTS FLOW	1. The authority enters his/her AUC 2. The authority enters his/her DBP 3. The authority clicks on the "Login" button 4. The system checks the credentials validity
EXIT CONDITIONS	The authority has successfully logged in to SafeStreets
EXCEPTIONS	1. The authority is not registered 2. The AUC is wrong 3. The DBP is wrong 4. Some mandatory fields are not filled

3.4.5 Account Information

NAME	Account Information
ACTOR	User
ENTRY CONDITIONS	The user is logged in and is on the menu
EVENTS FLOW	1. The user clicks on the "Account Info" button 2. The checks his account information 3. Optionally, the user can decide to modify one or more credentials 4. If the user decided to modify his info, the system checks the new datas' validity
EXIT CONDITIONS	The user has successfully checked/modified his account information
EXCEPTIONS	1. The user is not logged in 2. The user tries to modify his info with new wrong credentials

3.4.6 Violation Reporting

NAME	Violation Reporting
ACTOR	User
ENTRY CONDITIONS	The user is logged in and is on the map page
EVENTS FLOW	1. The user clicks on the "Photo" button 2. The user takes the picture of the license plate of the car committing the violation 3. The user chooses the violation type from the dropdown menu 4. The user lets the GPS retrieve his position 5. Optionally, he can add a comment 6. The user clicks the "Report Violation" button
EXIT CONDITIONS	The user has successfully reported a violation
EXCEPTIONS	1. The user is not logged in 2. The users' device doesn't have a camera 3. The users' camera is not working 4. The user doesn't take the picture of a car 5. The license plate is not clearly visible 6. The users' GPS is not working 7. The users' GPS is deactived

3.4.7 Reports History

NAME	Reports History
ACTOR	User
ENTRY CONDITIONS	The user is logged in and is on the menu
EVENTS FLOW	1. The user clicks on the "Reports History" button 2. The user scrolls his/her previous reports 3. The user clicks on the chosen reports' details
EXIT CONDITIONS	The user has successfully seen his old reports' details
EXCEPTIONS	1. The user is not logged in 2. The user never reported a violation

3.4.8 Violation Solving

NAME	Violation Solving
ACTOR	Authority
ENTRY CONDITIONS	The authority is logged in and is on the violations map
EVENTS FLOW	1. The authority clicks on the "Queued Violations" dropdown menu 2. The authority scrolls the menu until he reaches the interested violation 3. The authority checks the violations validity 4. The authority looks up the vehicles' owner 5. The authority decides to take measures (issuing a fine) 6. The authority marks the violation as "solved"
EXIT CONDITIONS	The authority has successfully marked the violation as solved
EXCEPTIONS	1. The authority is not logged in 2. The authority isn't on the violations map 3. There aren't any queued violations 4. The authority finds an invalid violation

3.4.9 Suggestion Observing

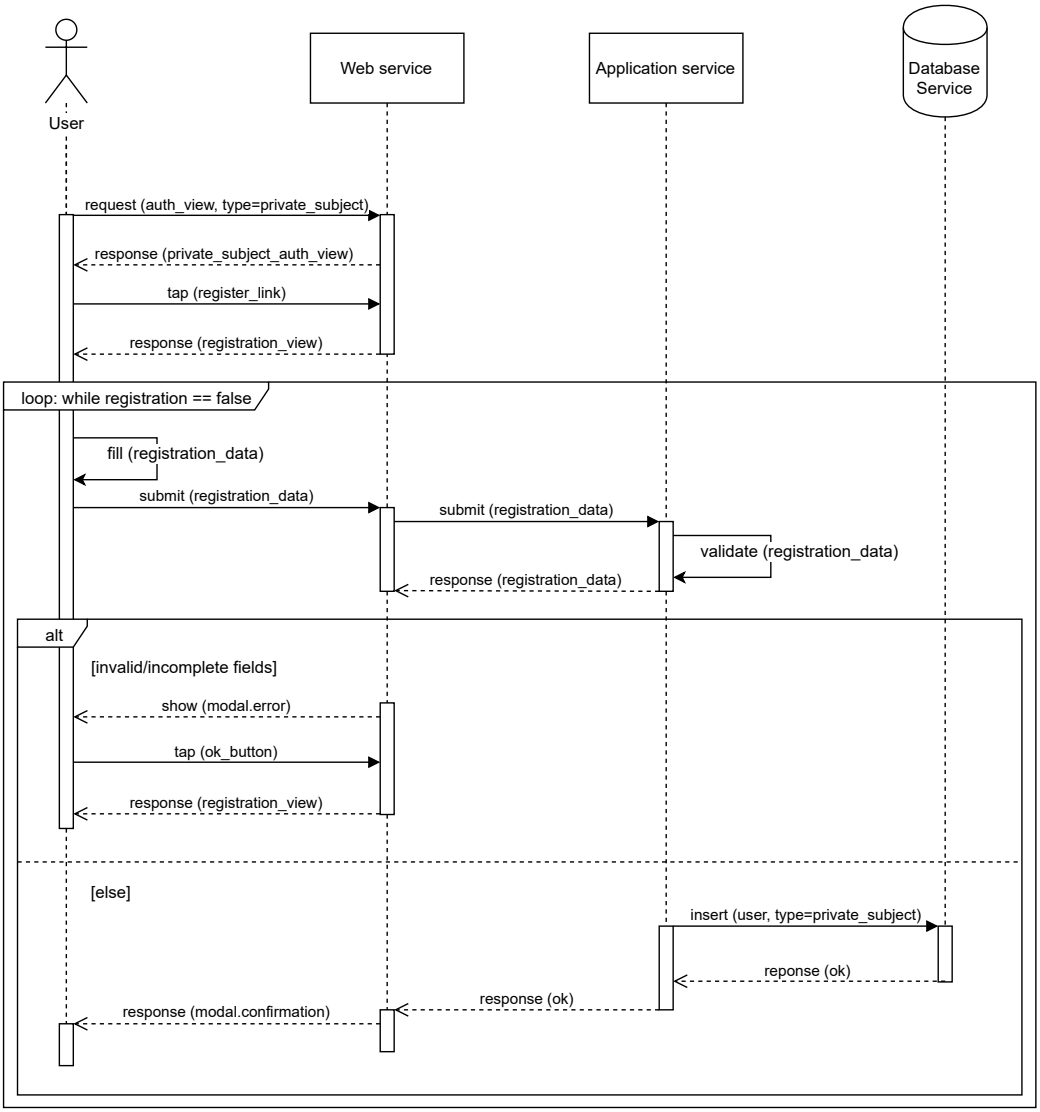
NAME	Suggestion Observing
ACTOR	Authority
ENTRY CONDITIONS	The authority is logged in and is on the suggestions map
EVENTS FLOW	1. The authority clicks on the show Suggestions button 2. The authority clicks on the blue alert of the interested area 3. The authority observes which interventions SafeStreets' algorithms have suggested 4. The authority can call back the municipality and order them to take the suggested measures
EXIT CONDITIONS	The authority has successfully observed a suggestion
EXCEPTIONS	1. The authority is not logged in 2. The authority isn't on the suggestions map 3. There aren't any suggestions in that area

3.4.10 Suggestion Requesting

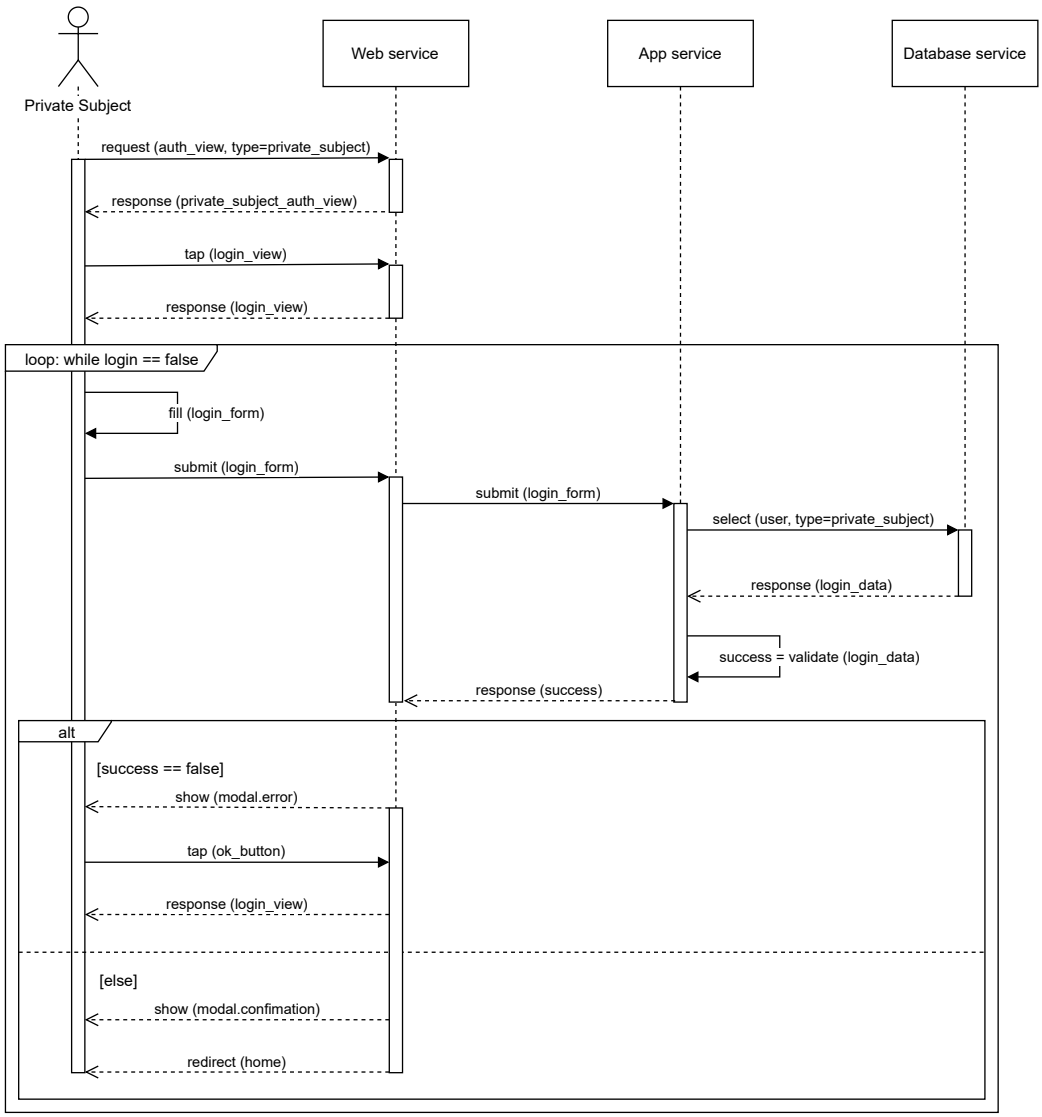
NAME	Suggestion Requesting
ACTOR	Authority
ENTRY CONDITIONS	The authority is logged in and is on the suggestions map
EVENTS FLOW	1. The authority clicks on the Request Suggestion button 2. The authority waits for the request to be forwarded to municipality 3. SafeStreets crosses its own data with the one municipality provided them 4. SafeStreets' algorithms generate a suggestion and sends it back to the authority
EXIT CONDITIONS	The authority has successfully requested a suggestion
EXCEPTIONS	1. The authority is not logged in 2. The authority isn't on the suggestions map 3. There aren't any suggestions to generate for that area

3.5 Sequence Diagrams

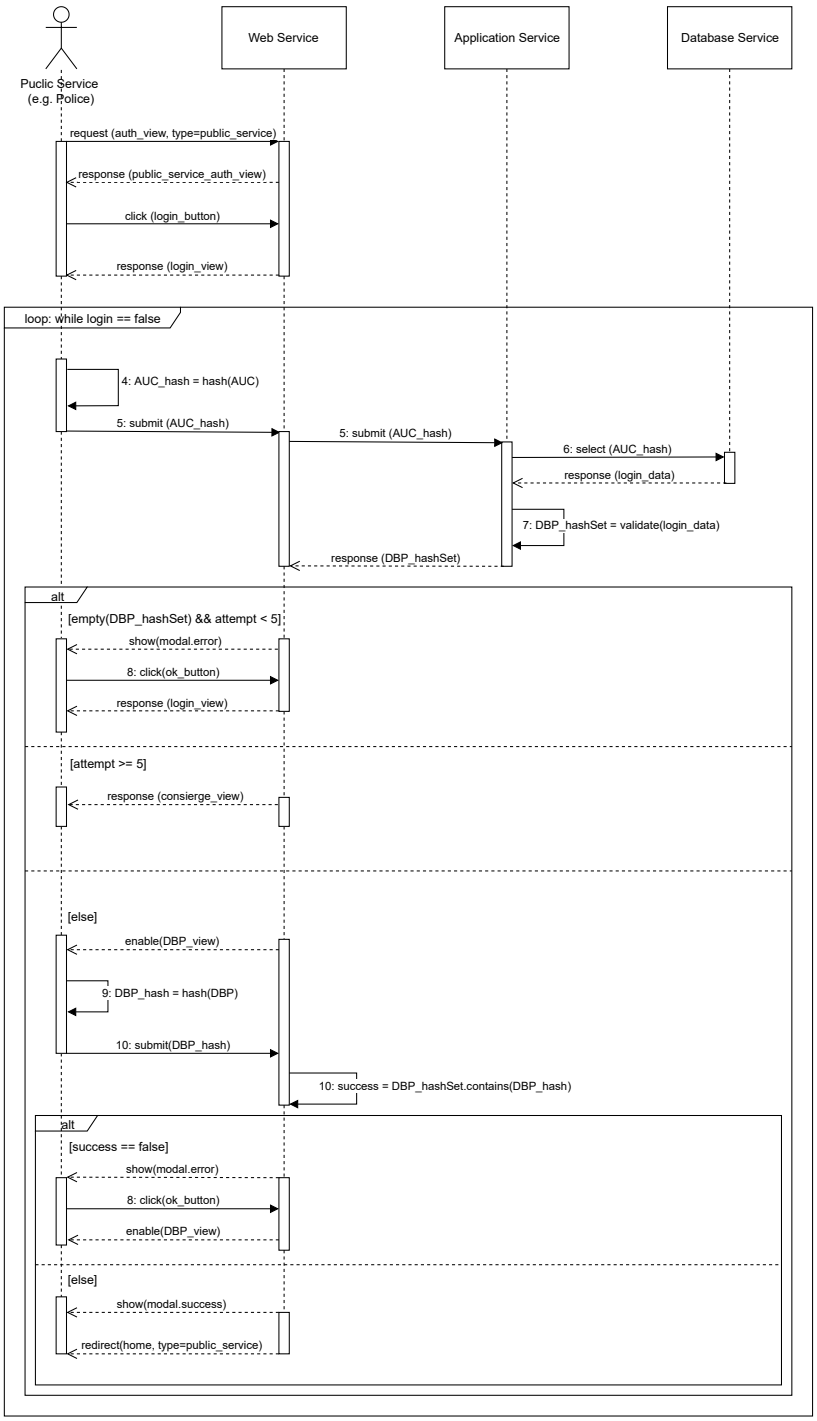
3.5.1 User Registration



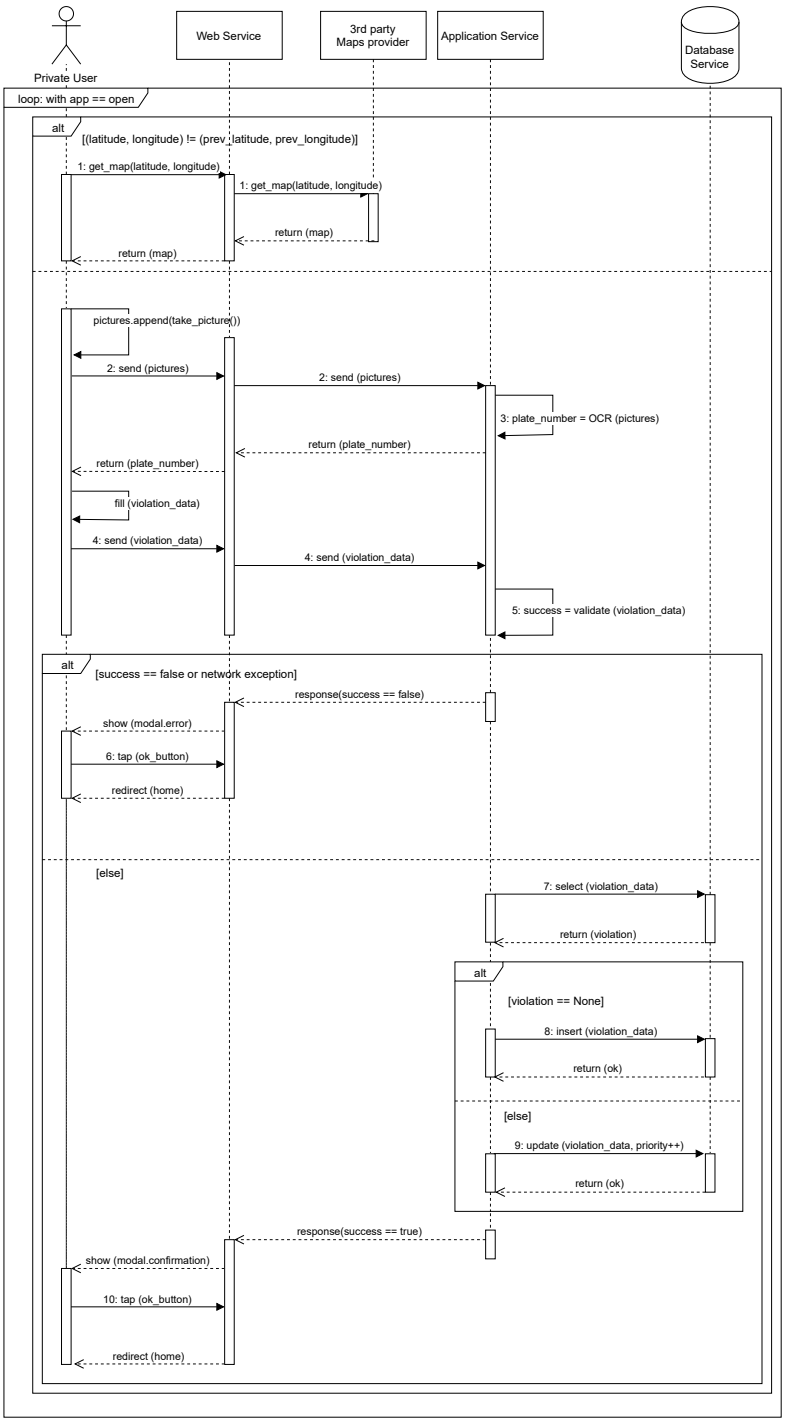
3.5.2 User Login



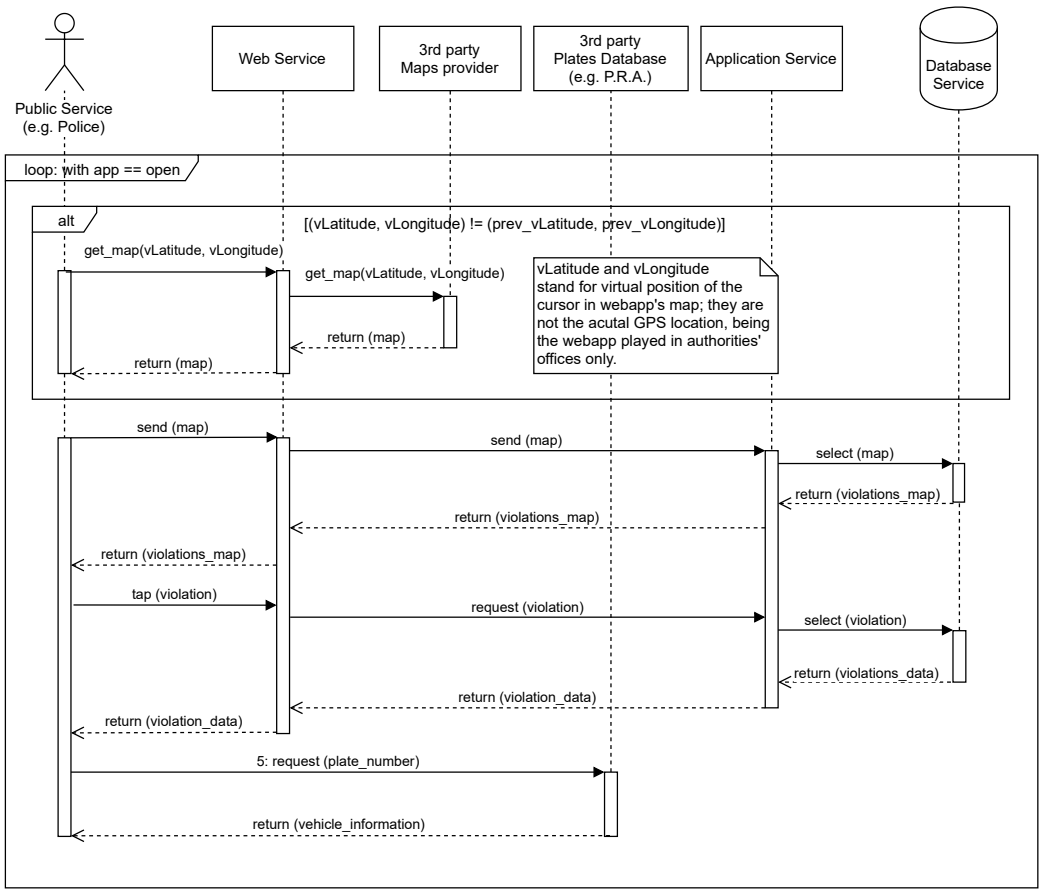
3.5.3 Authority Login



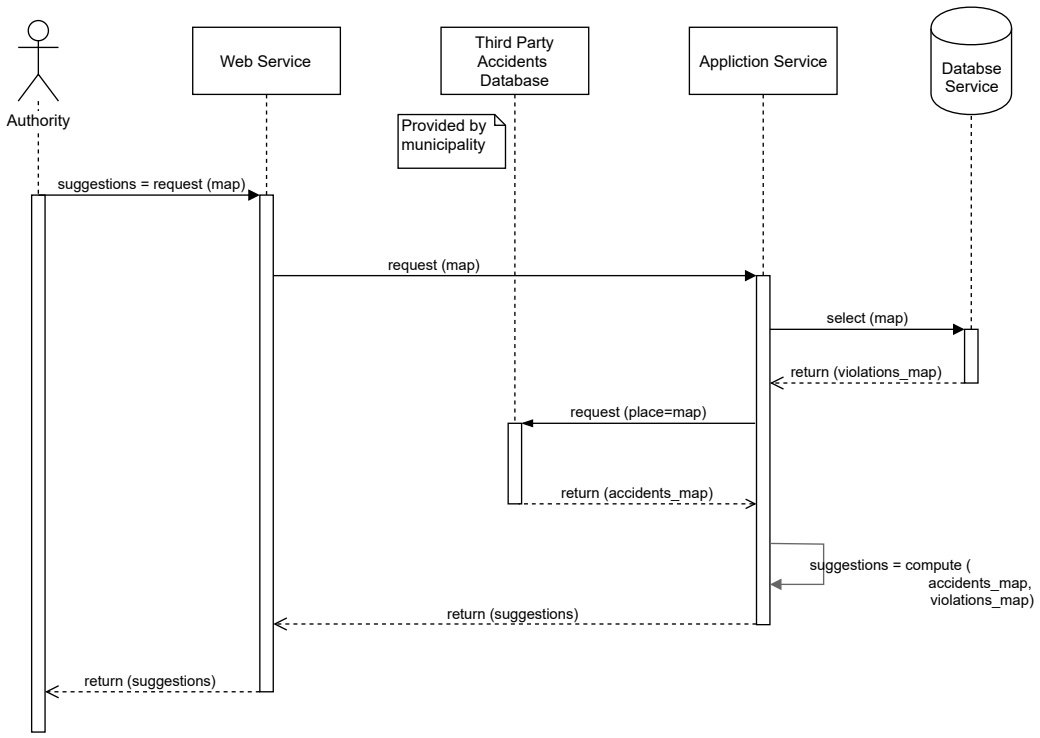
3.5.4 Violation Reporting



3.5.5 Violation Solving



3.5.6 Suggestion Request



3.6 Performance Requirements

The system must be able to handle a big quantity of reports with several images attached throughout the day. In order to improve the performance of the system, SafeStreets should rely on lightweight TCP connections.
For what concerns the Violations/Suggestions Map, the map scrolling must be fluid and responsive, the violation alerts clear and the login process for both users and authorities must be fast and reliable

3.7 Design Constraints

3.7.1 Standards compliance

Since the system processes sensitive data, the software complies to the General Data Protection Regulation (GDPR), a european regulation law on privacy policies and data protection for all the individuals living within the European Union (EU)

3.7.2 Hardware limitations

- `Mobile App`:
  - Smartphones: iOS/Android, Internet Connection, GPS, camera.
- `Web App`: modern web browser (e.g. Google Chrome / Safari), specifically intended for authorities only, enabled by the above said communication protocols and APIs.

3.8 Software System Attributes

3.8.1 Reliability

The system must be active 24/7

3.8.2 Availability

Whenever requested (a user wants to report a violation immediately or a user wants to register), the system must respond correctly and rapidly to the queries

3.8.3 Security

Since personal information is stored, the system guarantees not to divulgate them to unauthorized third parties, with respect to the GDPR

3.8.4 Maintainability

Enforced by the usage of specific design patterns and the provided hardware/software requirements, the software must be easy to fix and modifiy (or maintain) in the future

3.8.5 Portability

The software is generally projected for every type of mobile device, such as tablets and smartphones, and every type of OS, such as iOS or Android. Specifically for authorities, the Web App is compatible with every known browser

4 FORMAL ANALYSIS USING ALLOY

4.1 Purpose of the Model

Through this model we want to formalize some fundamental aspects of SafeStreets

- All the users that register to the system, must have a unique e-mail/telephone number and their data must be saved into the database.
  The same must happen for authorities, they all must have a unique AUC and/or DBP

- All the reports must be unique and no more than one report for the same violation must be in the queued violations set.
  In case the same violation is reported more than once, that reports' priority will increase

- All reports which stay in the queue for more than 7 days, will be deleted

4.2 Alloy Model

```
--------SIGNATURES--------
sig User {
                email: one String,
                telephone: one Int,
                password: one String,
                id: one Int
}

sig Authority{
                auc: one String,
                dbp: one Int,
                id: one Int
}

sig Violation {
                id: one Int,
                plate: one String,
                time: one String,
                location: one String,
}

sig Report {
                violation: one Violation,
                priority: one Int,
                elapsedDays: one Int
} {
                priority > 0
}
```

```
sig Database {
                -- Queued: it represents the set of unverified
                -- reports that come from users

                -- Confirmed: it represents the set of Reports that
                -- have been checked from the authority.

                queued: set Report,
                confirmed: set Report,
                suggestions: set Suggestion
} {
                -- The cardinality of queued set is greater or equal to
                -- the one of confimed because reports can be discarded

                #queued >= #confirmed
}

sig UserDatabase {
                users: set User
}

sig AuthorityDatabase {
                authorities: set Authority
}

sig Suggestion {
                id: one Int,
                location: one String
}
```

```
------------------FACTS------------------

fact NoDifferentViolations {

		-- if two violations have occurred
		-- on the same location in the same time and
		-- belong to the same plate then they must have
		-- the same id and vice versa.

		one v1, v2: Violation |
				(
						v1.plate = v2.plate and
						v1.time = v2.time and
						v1.location = v2.location
				) iff
				(
						v1.id = v2.id
				)
}

fact NoDifferentEmailsOrTelephone{

		--if two emails or two telephones
		--are equal, then it must be
		-- the same user

		one u1, u2: User |

		(
				u1.email = u2.email or
				u1.telephone = u2.telephone
		) iff
		(
				u1.id = u2.id
		)
}


fact NoDifferentAucOrDbp{

		-- if two auc or two dbp
		-- are equal, then it must be
		-- the same authority and vice versa

		one a1, a2: Authority |
		(
				a1.auc = a2.auc or
				a1.dbp = a2.dbp
		) iff
		(
				a1.id = a2.id
		)
}
```

```
fact AuthorityExistance {

		-- if an authority is registered then
		-- he must be in the authority database

		one a: Authority, db: AuthorityDatabase |
		(a in db.authorities) iff (a.id>0)
}


fact UserExistance {

		--if a user is registered then
		--he must be in the user database

		one u: User, db: UserDatabase |
		(u in db.users) iff (u.id>0)
}

fact SuggestionExistance {

		--if a suggestion is raised then
		--he must be in the suggestion database

		one s: Suggestion, db: Database |
		(su in db.suggestions) iff (s.id>0)
}


fact NoSuggestionIfNoReports {

		-- If queued does not contain any reports of
		-- violations occurred in a specified location
		-- then Suggestion of that location must not be raised.

		all r: Report, s: Suggestion | one db: Database |
				(
						r.violation.location = s.location and
						r not in db.queued
				) implies
						s not in db.suggestions
}

fact DisjointQueuedAndConfirmed {

		-- A report cannot be both in confirmed and
		-- queued sets at the same time.

		one r: Report, db: Database |
				r in db.queued iff r not in db.confirmed
}
```

```
----------ASSERTIONS----------

assert NoDuplicatedReports {

        -- All different reports in queued or confirmed
        -- inboxes must have different related violations

        one db: Database |
                (
                        all r1, r2: Report |
                                r1 in db.queued and r2 in db.queued and
                                r1.violation.id != r2.violation.id and
                                r1 != r2
                )
        and
                (
                        all r1, r2: Report |
                                r1 in db.confirmed and r2 in db.confirmed and
                                r1.violation.id != r2.violation.id and
                                r1 != r2
                )
}

check NoDuplicatedReports for 10

assert NoMoreThan7Days {

        -- if one report is not moved from queued to confirmed within 7 days
        -- then it is deleted from the database.

        all r: Report, d: Database |
        (r.elapsedDays >= 7) implies (r not in d.queued)
}

check NoMoreThan7Days for 10

assert PriorityIncrease {

        -- if one report added to the database is already present into it,
        -- then the actual report's priority is increased.

                all r1,r2: Report, d: Database |
        (r1 in d.queued and r2 in d.queued and r1.id = r2.id) implies
        (r1.priority > r2.priority)
}

check PriorityIncrease for 10
```

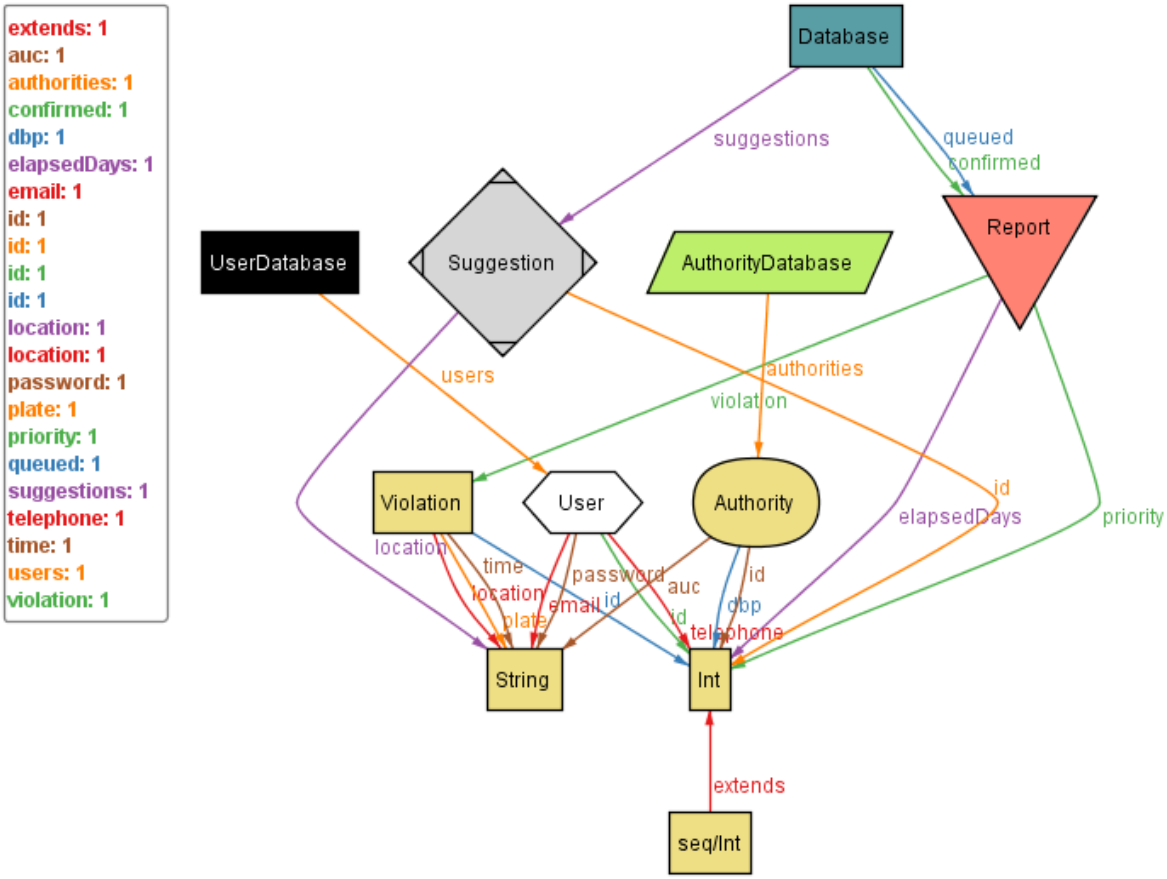
```
--------------------PREDICATES----------------------

pred createUser[u: User, em: String, tel: Int, pass: String, i: Int] {
                        u.email = em
                        u.telephone = tel
                        u.password = pass
                        u.id = i
}

pred createAuth[a: Authority, au: String, dev: Int, i: Int] {

                        a.auc=au
                        a.dbp=dev
                        a.id=i
}
```

```
Executing "Check PriorityIncrease"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4417 vars. 411 primary vars. 11329 clauses. 282ms.
  No counterexample found. Assertion may be valid. 16ms.


Executing "Check NoDuplicatedReports for 10"
  Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20
  28685 vars. 1690 primary vars. 69130 clauses. 480ms.
  No counterexample found. Assertion may be valid. 24ms.

Executing "Check NoMoreThan7Days"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4179 vars. 408 primary vars. 10464 clauses. 72ms.
  No counterexample found. Assertion may be valid. 0ms.

Executing "Check PriorityIncrease"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  4417 vars. 411 primary vars. 11329 clauses. 37ms.
  No counterexample found. Assertion may be valid. 0ms.

3 commands were executed. The results are:
  #1: No counterexample found. NoDuplicatedReports may be valid.
  #2: No counterexample found. NoMoreThan7Days may be valid.
  #3: No counterexample found. PriorityIncrease may be valid.
```

4.3 Generated World

4.3.1 Description

Here you can see there are two different type of "clients", users and authority, each with its own personal database.
The Report class, which links to the Violation one, is unfolded in the Database which contains several sets of all the violations and their corresponding reports.



5 EFFORT SPENT

- Giancarlo Danese

Day	Subject	Hours
18/10/2019	Purpose, Scope	2
23/10/2019	Domain Assumptions, User Characteristics	1.5
28/10/2019	Definitions, Acronyms, Product Functions	1
30/10/2019	Scenarios, Domains	2
02/11/2019	Functional Requirements	2
04/11/2019	Use Cases	2
06/11/2019	Alloy	2
07/11/2019	Alloy	2
08/11/2019	Alloy	2
09/11/2019	RASD Revision	1

- Davide Savoldelli

Day	Subject	Hours
18/10/2019	Goals, Product Functions	1
23/10/2019	Document Structure	2
28/10/2019	User Interfaces, Mock-ups	3
30/10/2019	Goals, Requirements	1
02/11/2019	Constraints, Interfaces	2
04/11/2019	Sequence Diagrams	2
06/11/2019	Alloy	2
07/11/2019	Alloy	2
08/11/2019	Alloy	2
09/11/2019	RASD Revision	1

6 REFERENCES

6.1 Tools

- **Draw.io**: `https://www.draw.io/`
- **TeXworks**: `http://www.tug.org/texworks/`
- **Github**: `https://github.com/`