Politecnico di Milano

Software Engineering II project

SafeStreets

Requirements Analysis and Specifications Document

Authors:
Mattia Calabrese
Federico Capaccio
Amedeo Cavallo

Professor:
Elisabetta Di Nitto

Octoer 23, 2019

version 1.0

Contents

1 Introduction

1.1 Purpose of this document

The purpose of a **R**equirement **A**nalysis and **S**pecifications **D**ocument is the process of discovering the purpose for which a software system was intended, by identifying stakeholders and their needs, and documenting these in a form that is amenable to analysis, communication, and subsequent implementation. [**?**] It is also concerned with the relationship of software's factors such as goals, functions and constrains to precise specifications of software behavior, and to their evolution over time and across software families.[**?**]

1.2 Scope

SafeStreets is a crowd-sourced application that intends to provide users with the possibility to notify authorities when traffic violations occur, and in particular parking violations.

The application allows users to send pictures of violations, including suitable metadata, to authorities. Examples of violations are vehicles parked in the middle of bike lanes or in places reserved for people with disabilities, double parking, and so on. In addition, the application allows both end users and municipality to mine the information that has been received, for example by highlighting the streets (or the areas) with the highest frequency of violations, or the vehicles that commit the most violations. Of course, different levels of visibility are offered to different roles.[**?**]

1.2.1 Goals

G1 Allow guest users to register to the system

G2 Allow registered users to authenticate to the system

G3 Allow users to transfer data to the system describing occurred violations, including the suitable metadata to describe the submitted violation

G4 Ensure that the chain of custody of the information provided by the users is never broken, and the information is never altered or manipulated

G5 Allow the system to retrieve data about the accidents that occur on the territory and data about issued tickets via the municipality provided service

G6 Allow the system to cross the information submitted by the users and the information retrieved from the municipality to build statistics

G7 Allow users to consult a map highlighting the streets (and the areas) with the highest frequency of violations, the identified potentially unsafe areas and view statistics about previously stored violations

G8 Allow municipality to consult the system data and receive suggestions on possible interventions via a restrict access API

1.3 Glossary

1.3.1 Definitions

System: the SafeStreets software we are to develop

Municipality: a city, a town or a village, or a small group of them

Local authorities: the local authorities of the municipality for example the local police

Guest *or* Guest user: person who access the system as non logged useraute

Logged user *or* Authenticated user: authenticated person who is interfacing with the system

User: guest user or logged user

Registration: interaction between a non registered user and the system in which the user, providing all of the information required by the system for the creation of an account, receives from the system the credentials needed to authenticate to the system

Authentication *or* Login: interaction between guests and the system that grants authenticated user's privileges to a guest user

Upload procedure: process which realizes the transfer of data between the user and the system

Restricted access API: API that can be used only by authorized person or system through an access token

GPS Coordinates: GPS coordinates are a unique identifier of a precise geographic location on the earth

Chain of Custody *or* Chain of Evidence: process of validating how any kind of evidence has been gathered, tracked, and protected on its way to a court of law. It guarantees that the data presented is "as originally acquired" and has not been tampered with and is authentic prior to admission into evidence.[**?**]

1.3.2 Acronyms

RASD: Requirements Analysis and Specification Document

API: Application Programming Interface

GPS: Global Position System

DBMS: Data Base Management System

FSM: Finite-State Machine

GIS: Geographic Information System

1.3.3 Abbreviations

m: meters (with multiples and submultiples)

w.r.t.: with respect to

i.f.f.: if and only if

e.g.: exempli gratia

etc.: et cetera

1.4 Document overview

According to the IEEE standard [**?**], this document is structured as

1. **Introduction**: it provides an overview of this entire document and product goals
2. **Overall description**: it describes general factors that affect the product providing the background for system requirements
3. **Specific requirements**: it contains all system's functional and non-functional requirements
4. **Use cases identification**: it contains the usage scenarios of the system with the use case diagram, use cases descriptions and other diagrams
5. **Appendices**: it contains the Alloy model, software and tools used, hours of work per each team member

2 Overall description

2.1 Product perspective

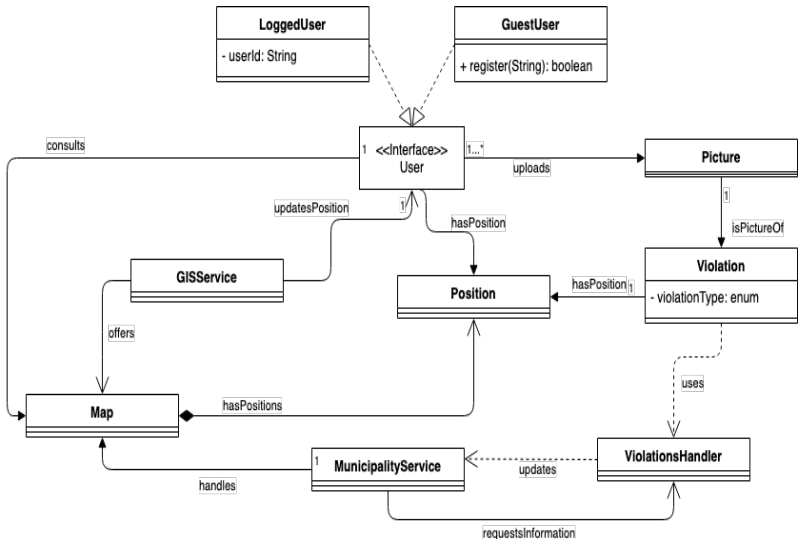

Figure 1: Class Diagram

2.1.1 System interfaces

The system we are to develop requires some external interfaces (represented in **??**) to accomplish the goals stated before.

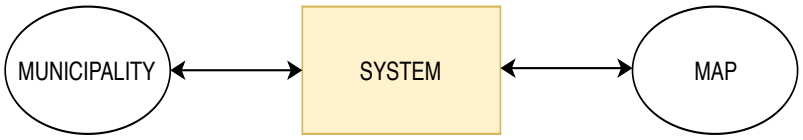

Figure 2: Overview of system interfaces

Municipality service The system will interact with municipalities. Our system will retrieve the information about the accidents that occur on the territory of the municipality and cross this information with its own data to identify potentially unsafe areas. It will also retrieve the information about issued tickets from the municipalities to build statistics, for example about the most egregious offenders, or the effectiveness of the SafeStreets initiative (e.g., by looking for trends in the issuing of tickets).

In addition, our system will expose via a *restricted access API* the stored information about the violations to the municipalities, so that the local authorities can generate traffic tickets from it, and receive suggestions for possible interventions (e.g., add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

Geographic Information System The system will interact with an external GIS. Our system will map the spatial location of stored violations and visualize the spatial relationships among them. The external GIS will map quantities, such as where the most and least number of violations occurred, to find places that meet the user requested criteria inside an area of interest. This can be accomplished mapping concentrations, or a quantity normalized by area or total number. The system can map the change in a specific geographic area to visualize statistics, or to evaluate the results of the SafeStreets initiative.

2.1.2 User interfaces

Guest user Using the interfaces of the system users can:

1. Register to the system or request an account for a specific municipality
2. Log-in to the system

Logged User Using the interfaces of the system users can:

1. Submit a violation with all the required and optional metadata
2. View a map through an external GIS with highlighted streets (or areas) with the highest frequency of violations
3. View statistics on vehicles that commit the most violations, the most egregious offenders, or the effectiveness of the SafeStreets initiative.
4. View and edit personal information

Siamo sicuri di voler fare che tutti gli utenti possono vedere queste staitis-tiche?

2.1.3 Hardware interfaces

The only hardware interfaces with which the system cooperates are the smartphones of the users and the hardware that supports the system that offers municipality's service.

2.1.4 Software interfaces

In order to reach the goals highlighted in the goals section the system need to interface with:

1. Databases and DBMSs, clearly required in order to store data about users, violations, safe/unsafe areas etc.
2. The software offered by the municipality, required in order to exchange data with its services
3. The map API offered by the GIS

2.1.5 Communication interfaces

We want to ensure the encryption of data shared with our system. This involves recording of metadata information as well as issues of access control and security for all the handling digital chain of custody.

2 Overall description

2.1 Product perspective

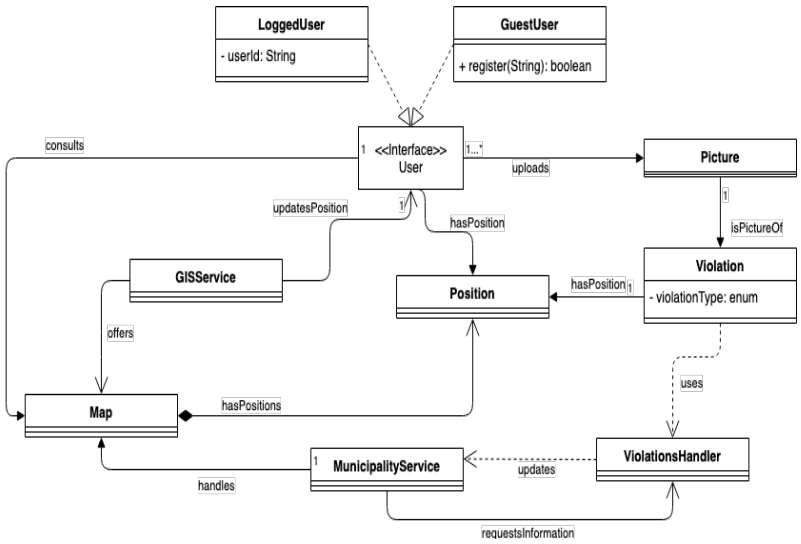

Figure 1: Class Diagram

2.1.1 System interfaces

The system we are to develop requires some external interfaces (represented in Figure 2) to accomplish the goals stated before.

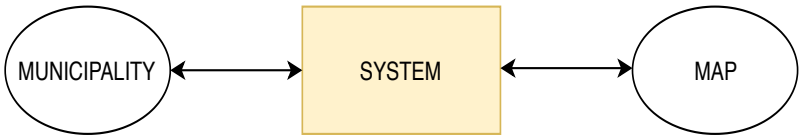

Figure 2: Overview of system interfaces

Municipality service The system will interact with municipalities. Our system will retrieve the information about the accidents that occur on the territory of the municipality and cross this information with its own data to identify potentially unsafe areas. It will also retrieve the information about issued tickets from the municipalities to build statistics, for example about the most egregious offenders, or the effectiveness of the SafeStreets initiative (e.g., by looking for trends in the issuing of tickets).

In addition, our system will expose via a *restricted access API* the stored information about the violations to the municipalities, so that the local authorities can generate traffic tickets from it, and receive suggestions for possible interventions (e.g., add a barrier between the bike lane and the part of the road for motorized vehicles to prevent unsafe parking).

Geographic Information System The system will interact with an external GIS. Our system will map the spatial location of stored violations and visualize the spatial relationships among them. The external GIS will map quantities, such as where the most and least number of violations occurred, to find places that meet the user requested criteria inside an area of interest. This can be accomplished mapping concentrations, or a quantity normalized by area or total number. The system can map the change in a specific geographic area to visualize statistics, or to evaluate the results of the SafeStreets initiative.

2.1.2 User interfaces

Guest user Using the interfaces of the system users can:

1. Register to the system or request an account for a specific municipality
2. Log-in to the system

Logged User Using the interfaces of the system users can:

1. Submit a violation with all the required and optional metadata
2. View a map through an external GIS with highlighted streets (or areas) with the highest frequency of violations
3. View statistics on vehicles that commit the most violations, the most egregious offenders, or the effectiveness of the SafeStreets initiative.
4. View and edit personal information

Siamo sicuri di voler fare che tutti gli utenti possono vedere queste staitis-tiche?

2.1.3 Hardware interfaces

The only hardware interfaces with which the system cooperates are the smartphones of the users and the hardware that supports the system that offers municipality's service.

2.1.4 Software interfaces

In order to reach the goals highlighted in the goals section the system need to interface with:

1. Databases and DBMSs, clearly required in order to store data about users, violations, safe/unsafe areas etc.
2. The software offered by the municipality, required in order to exchange data with its services
3. The map API offered by the GIS

2.1.5 Communication interfaces

We want to ensure the encryption of data shared with our system. This involves recording of metadata information as well as issues of access control and security for all the handling digital chain of custody.

2.2 Product functions

Violation upload procedure Provide logged users with the possibility to notify authorities when traffic violations occur and, in particular, parking violations. The application allows users to send pictures of violations, including their date, time, and position, to authorities. SafeStreets then stores the information provided, completing it with suitable meta-data. In particular, when it receives a picture, it runs an algorithm to read the license plate number (the user can help with the recognition) and it stores the retrieved information with the violation, including also the type of the violation (submitted by the user) and the name of the street where the violation occurred.

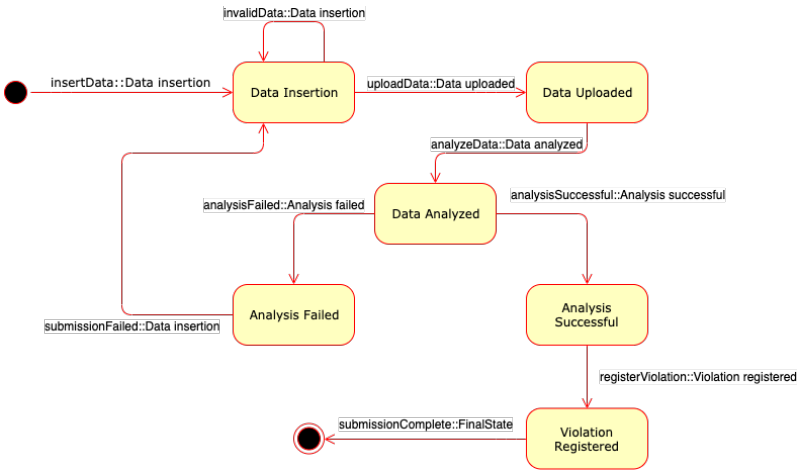

Figure 3: Violation upload process

Retrieve data from municipality Retrieve the information about the accidents that occur on the territory using the service offered by the municipalities and cross these data with SafeStreets data, to identify potentially unsafe areas. This will also allow the system to understand which violations are more likely to cause accidents in a particular zone and elaborate suggestions on possible interventions, later communicated to the municipality via a *restricted access API* provided to them.

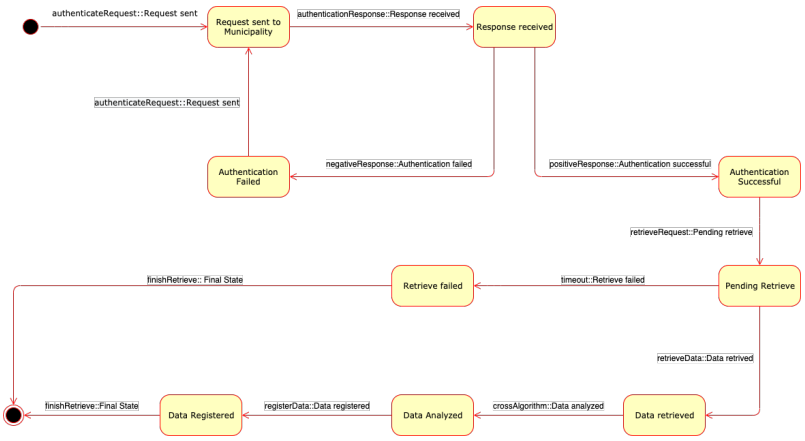

Figure 4: Retrieve data from municipality

Show information and statistics The application allows logged users to mine the information that has been received, highlighting the streets (or the areas) with the highest frequency of violations, considered unsafe areas, or the vehicles that commit the most violations. In addition, statistics about issued tickets, for example about the most egregious offenders, or the effectiveness of the SafeStreets initiative, are shown to the user if requested.

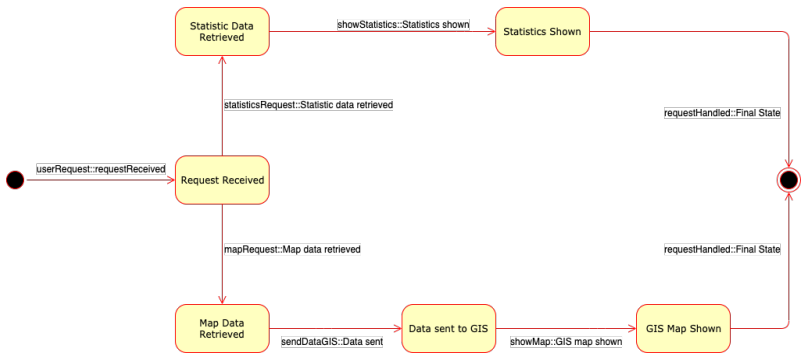

Figure 5: Show information and statistics

Restricted access API The system will expose via a *restricted access API* the stored information about the violations to the municipalities, so that the local authorities can generate traffic tickets from it and receive suggestions for possible interventions to carry out (e.g., add a barrier between the bike lane and the part of the road for motorised vehicles to prevent unsafe parking), in order to decrease the risk of those areas, so increasing their safety.

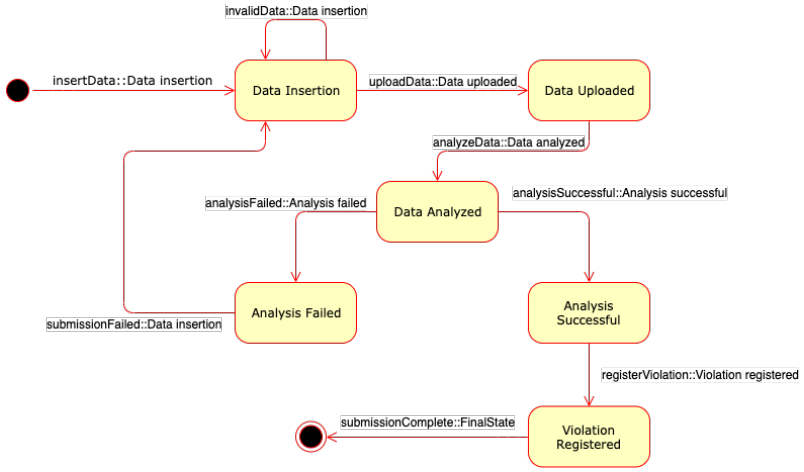

Figure 6: Restricted access API

2.3 User Characteristics

Users can use our system when they notice a violation and want to communicate it to the authorities.
Necessary conditions for the user in order to use the system are:

- He must have a smartphone with a working connection to the internet and he must be able to properly use it
- He must be in the age of majority
- He must be able to identify a violation and its type

The user agrees to these conditions during the registration to the system.

2.4 Constraints

We assume that these constraints are always met:

C1 GPS position is supposed to be accurate (max error ±5m)

C2 The quality of the picture is sufficient to recognise the plate number (min resolution 320x240)

C3 Internet connection must be strong enough to allow the upload of the picture in a reasonable amount of time (supported technologies are 3G, 4G and 5G due to the performance requirement)

2.5 Assumptions

We assume that these assumptions hold true in the domain of our system

DA1 GPS position of all users is always obtainable

DA2 Internet connection always works correctly

DA3 Municipality services are always reachable

DA4 The maps provided by the GIS are always reachable and up to date

DA5 The smartphone of the user runs iOS (9 or later) or Android (Jelly Bean or later)

R33 The system must adopt security measures to prevent malicious accesses and to protect sensible data

R34 The system must provide a secure channel to communicate with the users

R35 The system must encrypt the connection with the users in order to protect the process of providing data

?? Access the municipality archives of emitted tickets and build statistics based on the retrieved information

R36 The system must be able to retrieve data about tickets issued by the municipality

R37 The system must be able to process data retrieved from municipality

R38 The system must be able to elaborate issued tickets information to generate statistics about useful violations provided by users

3.2 Performance Requirements

The system should ensure acceptable response times in the interactions with the user, which strictly depends on the number of concurrent users and the connection speed.
The processes of providing data and loading the map of safe and unsafe areas shouldn't be too slow.

3.3 Software System Attributes

3.3.1 Availability

The system must be available 99,9% of the time (up to 8,76 hours per year of downtime). The system should be accessible 24 hours per day.

3.3.2 Security

Users personal information and payment information are encrypted and must be protected during transmission. Restricted access APIs must check that who tries to use them is actually allowed to do so.

3.3.3 Portability

The system must be also accessible by the most common mobile platforms (iOS and Android devices).

4 Use cases identification

4.1 Scenarios

Here are some scenarios that describe the usage of the system.

4.1.1 Scenario 1

Francesco wants to have a beer with his friend Vincenzo so Francesco logs in to system and searches for cars nearby. He notices that there are two cars available next to his house, he decides to reserve the one with more battery and after few minutes he reaches it. He starts to drive until he reaches Vincenzo's house, Vincenzo gets into the car and they arrive to the beer house where they terminate the rent.

4.1.2 Scenario 2

Mirjana has been told by her friend Elisa that a new car sharing service is available in their city and so she decided to give it a try as she wants to go shopping in the city center. She registers to the system providing all information requested, she inserts the destination address and enables the money saving option. She is provided by the system with a charging station not far from the shopping center and, as it is a sunny day, she decided to take that location as destination in order to achieve a discount and reach the shopping center on foot.

4.1.3 Scenario 3

Giovanni is really interested in electric cars so he decides to use the PowerEnJoy system. He wants to go to the museum in the afternoon so he reserves a car. After one hour he is notified by the system that his reservation is expired and he is charged of a 1€ fee. When he is ready to exit his house he notices that the same car is still available so he reserves it again, reaches it and starts driving. When he arrives at the destination he sees a charging station next to the museum so he decides to leave the car there plugging the charging cable in order to get a discount.

4.1.4 Scenario 4

Dino reserves a car, he reaches it and unlock the doors. He starts driving but after few minutes the engine stops and a warning icon lights up in the dashboard. He contacts the customer care service that suggests him to reserve a different car which is nearby, the customer care operator then tags the first car as not available specifying a brief description of the fault. The maintenance service is now aware of the problem and they take care of resolving the issue: a maintenance operator is sent to the car, he unlocks it and fixes it. When the car is fully functional he tags it as available using a restricted access API of our system.

4.1.5 Scenario 5

Stefania often uses PowerEnJoy service when she is coming back home after work. Before finish her work schedule she checks with the map if there are cars

nearby in order to reserve one of them: today she can see one from the window that is not present in the map because its battery is empty. After a while she checks again the map and she notices that the car is now located in a charging station nearby and it is available for a reservation.

4.1.6 Scenario 6

Lorenzo is driving a PowerEnJoy car in order to reach Dino's house outside the safe area. As soon as he exists the safe area the car system notifies him with a message. When he is arrived he terminates the rent, the payment is executed with an extra fee and the car is tagged as not available.

4.2 Use case diagram

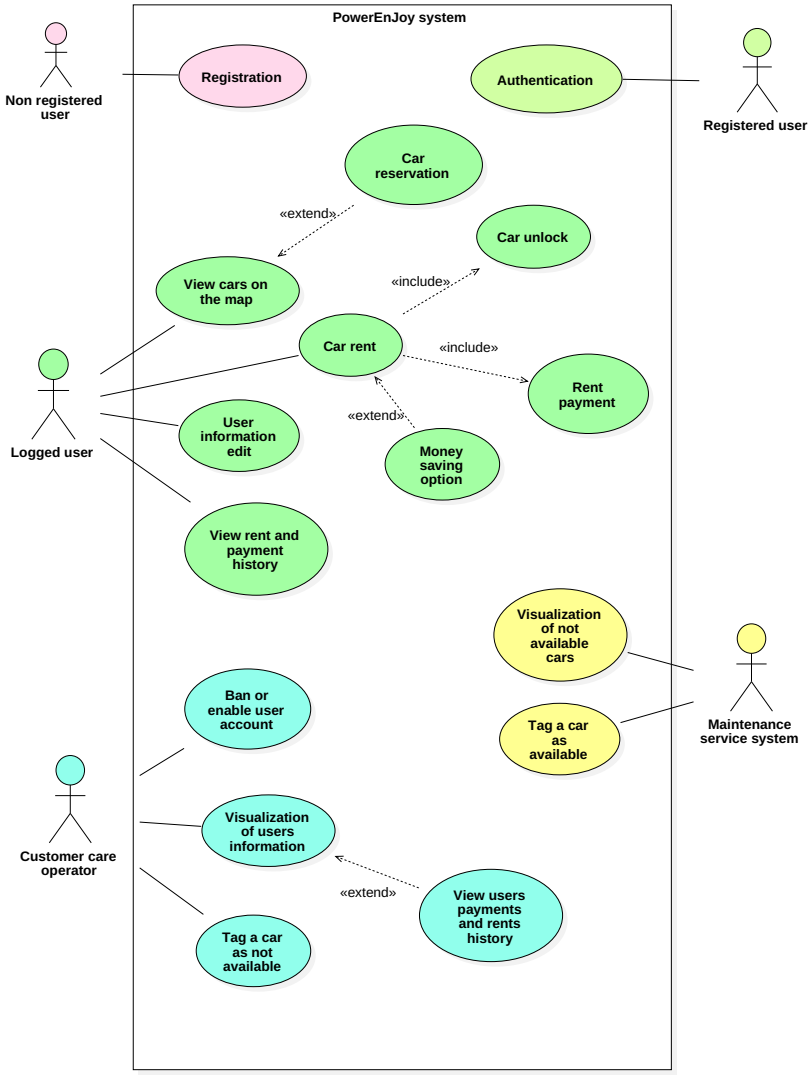

Figure 7: Use case diagram

Notes to read the diagram The use case diagram represents the possible interactions of actors with the system and the different use cases in which the actors are involved.

The "Maintenance service system" is an external software which the system-to-be needs to interact with.

We do not consider the external payment handling system an actor since it does not start any interaction with our system, it simply reacts when our system requests its services; this interaction is encapsulated as sub-procedure in the flow of events of the "Rent payment" use case.

We do not consider the car an actor for the same reason, moreover the only interactions started by the car are trigger events which are very simple interactions, which we do not consider use cases.

4.3 Use cases description

4.3.1 Registration

Name	**Registration**
Actors	Non registered user
Entry conditions	
Flow of events	1. The user asks the system to register to its services 2. The system shows the appropriate form to fill to register to the system 3. The user inserts an username to be uniquely identified by the system 4. The user inserts his own email address 5. The user inserts his name, surname, birth date and place and current domicile 6. The user inserts his driving license ID code 7. The user inserts payment information 8. The user confirms data inserted are correct e submit the form 9. The system checks the username to be unique 10. The system checks the email to be unique 11. The system checks the driving license ID to be unique 12. The system sends an email to the user with a unique link to verify the email address inserted by the user really belongs to him 13. The user clicking on the link received confirms his email address 14. The user is notified by mail the registration procedure is correctly completed and provided with a password bound to his username to access the system
Exit conditions	The user is able to authenticate to the system as *registered* user with its own credentials

Exceptions

- If the username inserted by the user is already used by another user, the system displays an error message asking the user to insert another username
- If the mail inserted by the user is already used by another user, the system displays an error message asking the user to insert another mail
- If the user notices to have entered wrong informations he could edit them at the end of the process of registration in his personal page

Table 1: *Registration* use case description

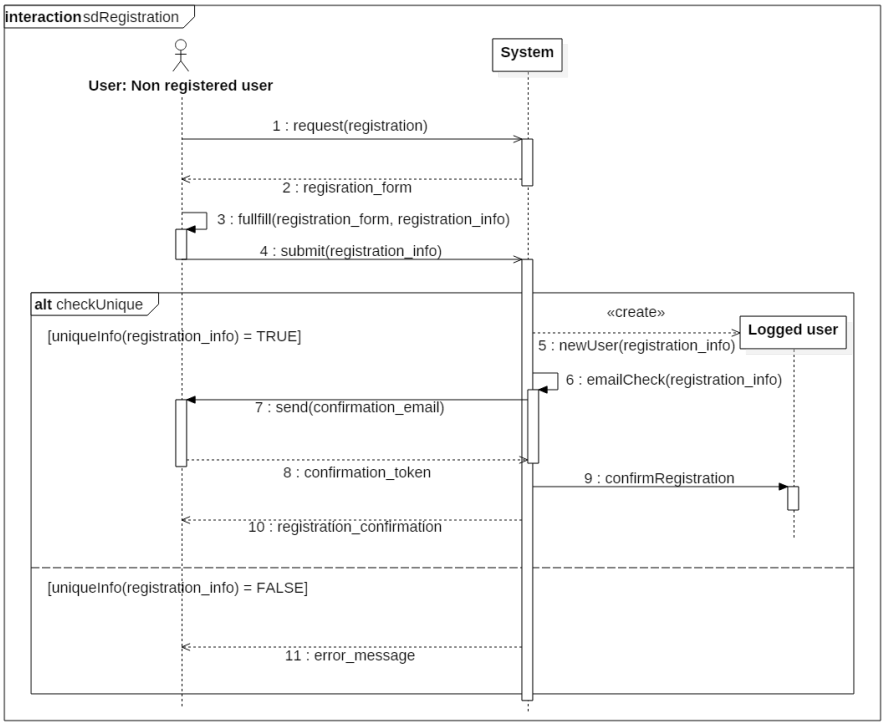

Figure 8: *Registration* sequence diagram

4.3.2 Authentication

Name	**Authentication**
Actors	Registered user
Entry conditions	The user must know his username and password
Flow of events	1. The user inserts his username and password in the appropriate form and submit it 2. The system validates the inserted credentials checking also if the user has confirmed his own email address 3. The system checks if the user is banned
Exit conditions	If the credential validation is successful and the user is not banned he is granted the proper privileges
Exceptions	• If the credential validation failed an error message is displayed • If the credential validation is successful and the user is banned a message providing assistance is displayed and the system doesn't allows the user to access to the system

Table 2: *Authentication* use case description

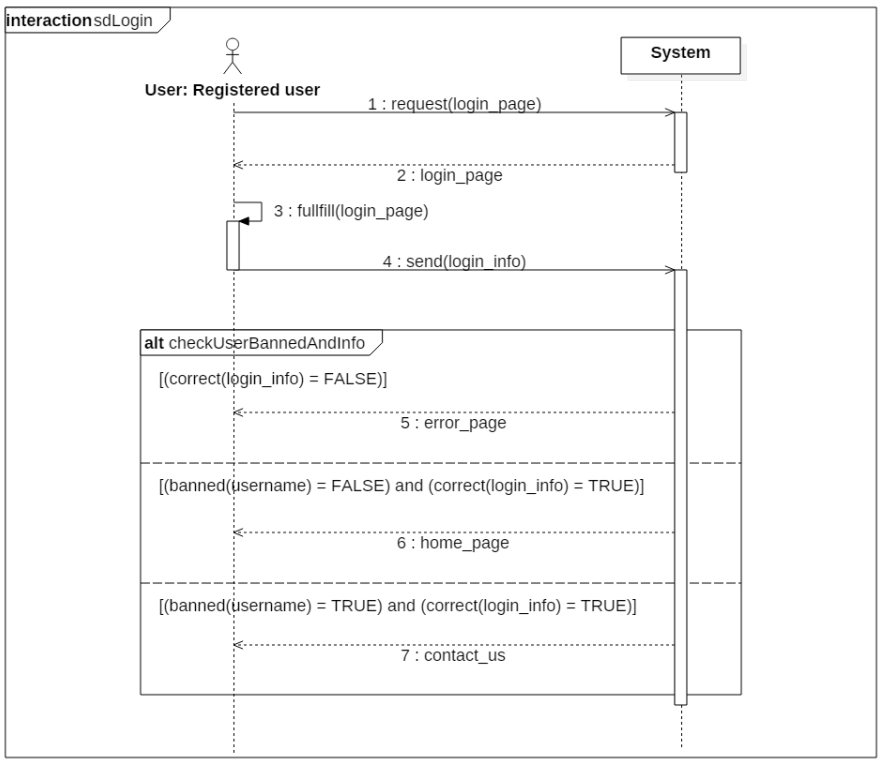

Figure 9: *Authentication* sequence diagram

4.3.3 View cars on the map

Name	**View cars on the map**
Actors	Logged user
Entry conditions	
Flow of events	1. The user chooses if he wants to use his GPS position or insert a different one manually a. The system retrieves the user's GPS position b. The user inserts a position 2. The system retrieves the position of all *Available* cars and their battery level percentage 3. The system shows a map with all available cars, charging stations position and safe areas near the position indicated 4. The user can click on a car on the map to see its battery level percentage
Exit conditions	The user can navigate a map with all available cars near the position indicated by him
Exceptions	If the position inserted by the user is not correct an error message is displayed

Table 3: *View cars on the map* use case description

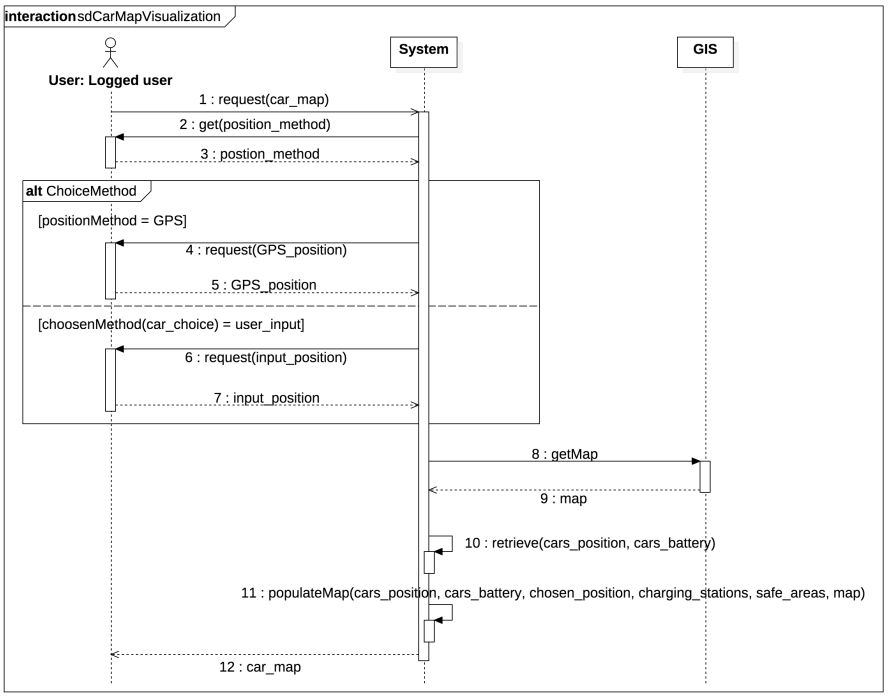

Figure 10: *View cars on the map* sequence diagram

4.3.4 Car reservation

Name	**Car reservation**
Actors	Logged user
Entry conditions	
Flow of events	1. **View cars on the map** 2. The user selects the car he wants to reserve 3. The user confirms he wants to reserve that car
Exit conditions	The system set the state of the chosen car as *Reserved* paired with the user who made the reservation
Exceptions	If the user has already reserved a car, the system shows an error message and doesn't allow him to reserve another car

Table 4: *Car reservation* use case description

4.3.5 Car unlock

Name	**Car unlock**
Actors	Logged user
Entry conditions	The user reserved car
Flow of events	1. The user asks the system to unlock the car he reserved 2. The system checks if the user's position is at most 5 meters away from the position of the car he reserved 3. The system unlocks the car with the state set as *Reserved* paired with the aforementioned user 4. The system sends a message to the user, confirming that the car is unlocked
Exit conditions	The car is unlocked and the user can pick it up
Exceptions	• If the position of the user is not at most 5 meters away from the position of the car he reserved the system displays an error message

Table 5: *Car unlock* use case description

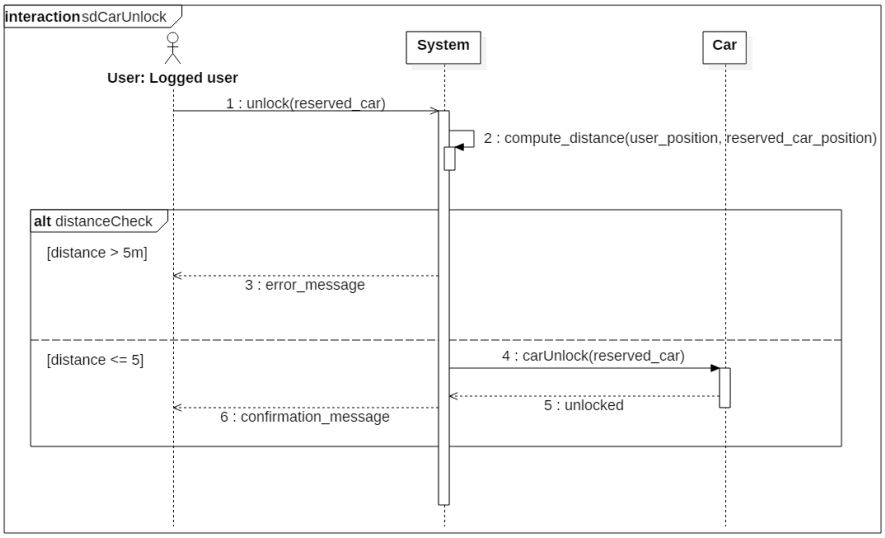

Figure 11: *Car unlock* sequence diagram

4.3.6 Car rent

Name	**Car rent**
Actors	Logged user
Entry conditions	The user is paired with the *Reserved* state of a car
Flow of events	1. **Car unlock** 2. The user ignites the car engine 3. The system sets the state of the *Reserved* car to *In Use* paired with the same user 4. During the rent the user is informed about the current charge and whether he is or not inside a safe area 5. The user leaves the car turning off the engine and closing the doors 6. The system locks the car 7. The system activates a timer to allow the user to plug the car into a charging station if it is near one of them 8. When the timer expires: 8.1 The system retrieves informations about the ride from the car: number of passengers detected during the ride, position of the car and battery level at the end of the ride and if the car is or not on charge 8.2 The system sets the car as *Available* 9. **Rent payment**
Exit conditions	The user is charged of the correct amount for the ride and at anytime could perform another rent, the car is available again
Exceptions	• If the user doesn't start the engine up to one hour after the reservation, he is charged of 1€(through a payment procedure), the car state is set as *Available* and the user is notified his reservation is expired

Table 6: *Car rent* use case description

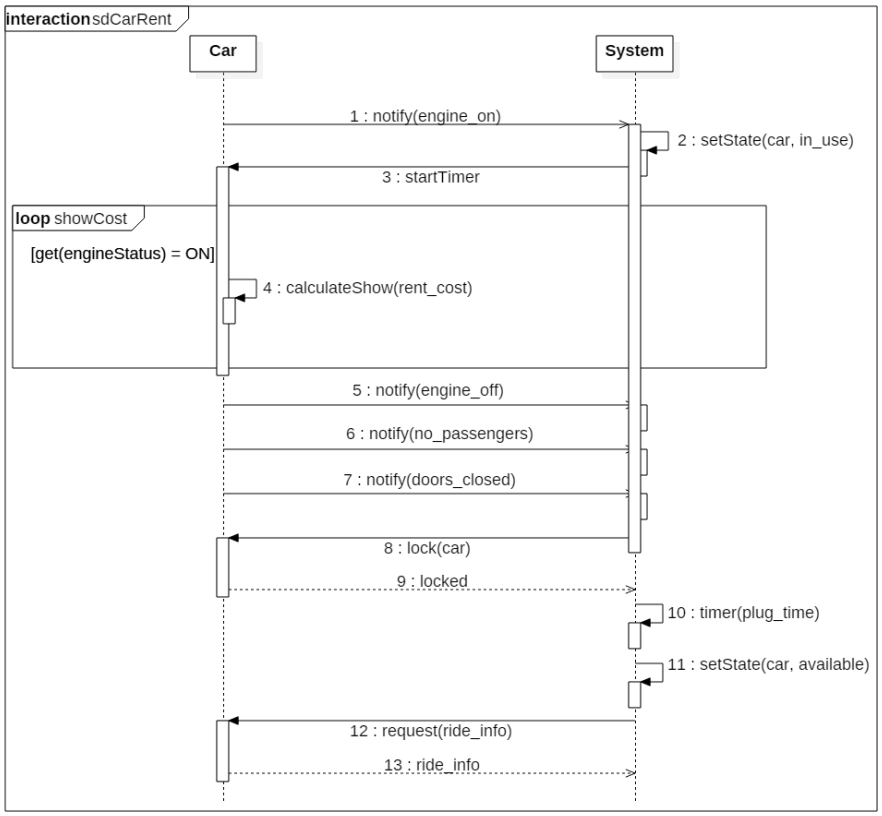

Figure 12: *Car rent* sequence diagram

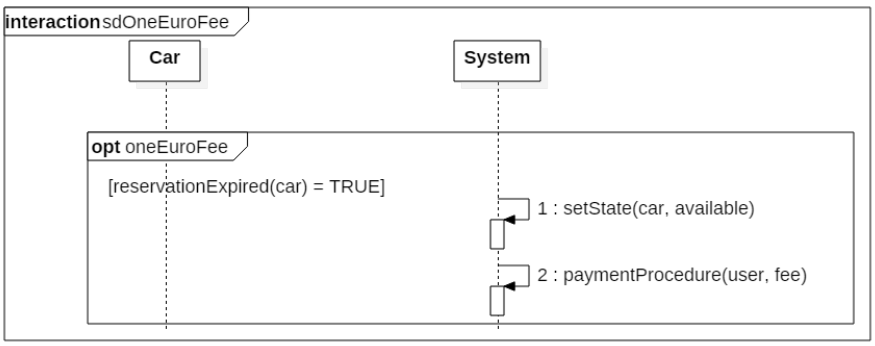

Figure 13: *One Euro fee* sequence diagram

The overall status of a car can be represented by the FSM in **??**

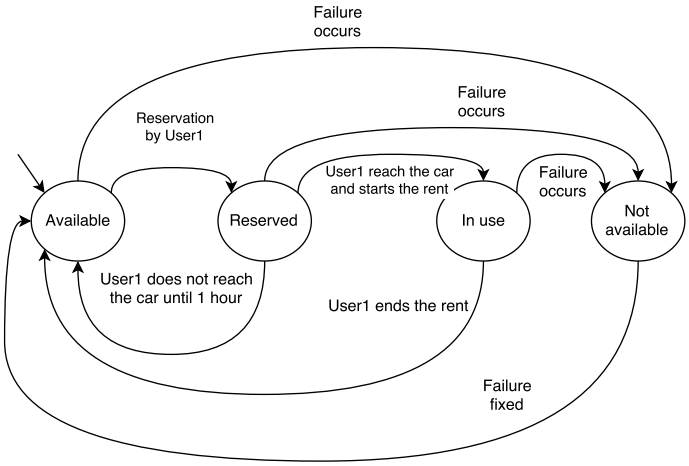

Figure 14: Car status FSM

Notes to read the diagram The *Not Available* state includes the cases in which the car is either broken or a user left it with a critical battery level and not on charge.

The system changes the state of a car from *Available* to *Not Available* when its battery level is critical and the car is not on charge (see **??**).

Even if the car is left with no battery left, it is still able to communicate with the system, so the rent can end normally and the maintenance service will take care the car (see **??**).

4.3.7 Rent payment

Name	**Rent payment**
Actors	Logged user
Entry conditions	The user must have completed a rent shutting off the engine and exiting the car. The system has retrieved information about the ride from the car.
Flow of events	1. The system checks if the car position is or is not inside a safe area 2. The system checks if the car has detected more than one passenger during the rent 3. The system checks the car battery percentage 4. The system checks if the car is plugged on a charging station 5. The system checks the distance of the car from the nearest charging station 6. The system calculates the cost of the ride based on the rent time 7. The system determines the applicable discounts/extra fee applying it to the cost of the ride 8. The system starts a payment procedure with user's payment information using an external service 9. The system waits a response from the external payment service 10. The system logs data about the rent and the payment 11. The system notifies the user about the result of the payment procedure and on discount/extra fees applied

Alternative flow	Flow of events as specified upon from *1* to *7* 8 a. The system detects the user has enabled the *money saving option* 8 b. The system checks if the car is currently on charge on the charge station determined by the system at the begin of the rent 8 c. The system determines the applicable discounts/extra fee applying it to the cost of the ride eventually also taking in account the *money saving option* discount if the car is currently on charge on the charge station determined by the system at the begin of the rent Flow of events as specified upon from *9* to *12*
Exit conditions	The user is charged of the correct amount for the ride
Exceptions	• If the payment procedure is not correctly completed the user is banned, rent information is stored, the payment suspended and the user is informed to contact the customer service.

Table 7: *Rent payment* use case description

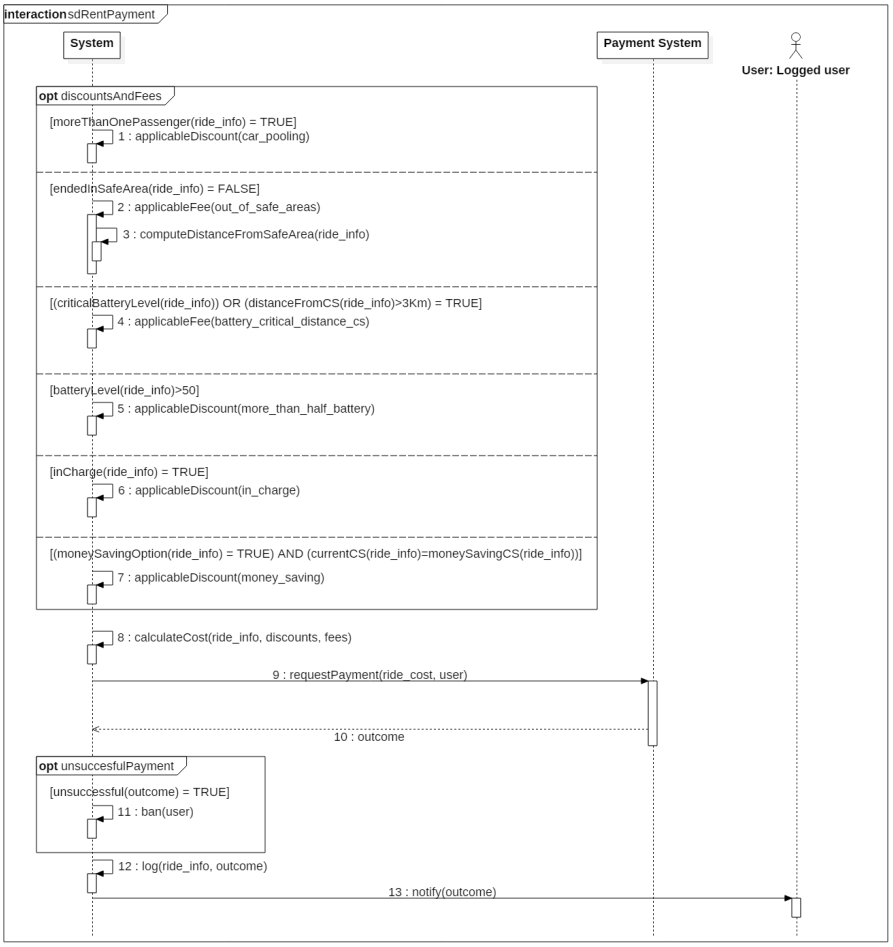

Figure 15: *Rent payment* sequence diagram

4.3.8 Money saving option

Name	**Money saving option**
Actors	Logged user
Entry conditions	The user should have enabled the *money saving option*
Flow of events	1. **Car Reservation** 2. The system asks the user to insert his destination 3. The user inserts his destination 4. The system searches for charging stations near the destination position inserted by the user with available plugs 5. The system chooses a charging station in order to ensure a uniform distribution of cars in the city and taking in account the destination of the user 6. The system informs the user about the charging station to reach in order to obtain the discount 7. **Car Rent** (**Car Reservation** already done)
Exit conditions	• If the user has left the car plugged in the charging station suggested by the *money saving option* he has obtained the correct discount • The user can any time perform another rent • Car is again available
Exceptions	• If the user doesn't leave the car in the charging station suggested by the *money saving option* he doesn't obtain the related discount

Table 8: *Money saving option* use case description

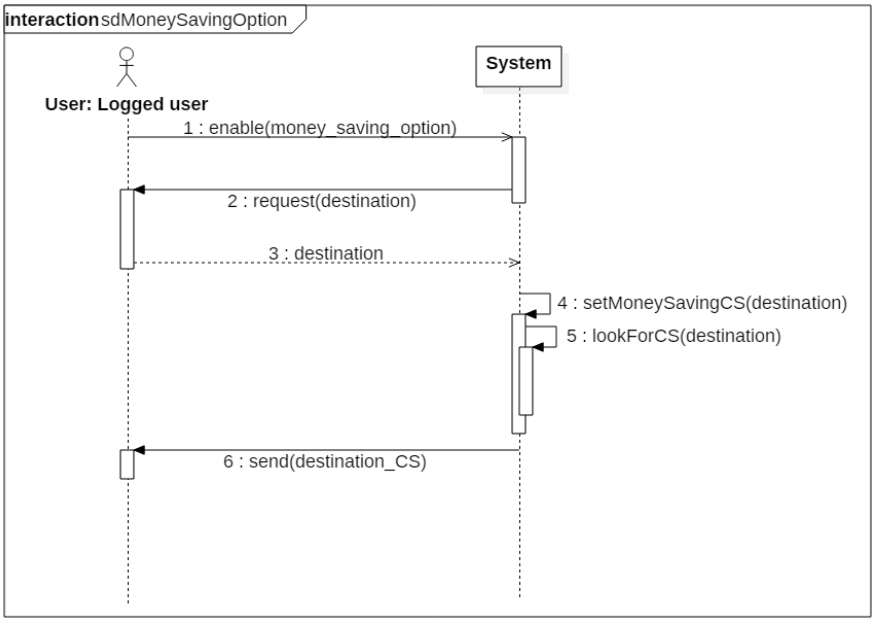

Figure 16: *Money saving option* sequence diagram

4.3.9 Visualization of not available cars

Name	**Visualization of not available cars**
Actors	Maintenance service system
Entry conditions	Maintenance service system must know the access token to be identified by the system
Flow of events	1. The maintenance service system asks for the list of car with state set as *Not Available* sending the request paired with the access token 2. The system checks the access token 3. The system retrieves the list of car with state set as *Not Available* along with the identifier used by the system to identify each car, the GPS position of each car, the description of the problem of each car and the software key to access each car 4. The system sends the information to the maintenance service system
Exit conditions	The maintenance service system receives the list of cars with state set as *Not Available*
Exceptions	• If the access token sent by the maintenance service system is not recognized, the system sent to the maintenance service system an error message

Table 9: *Visualization of not available cars* use case description

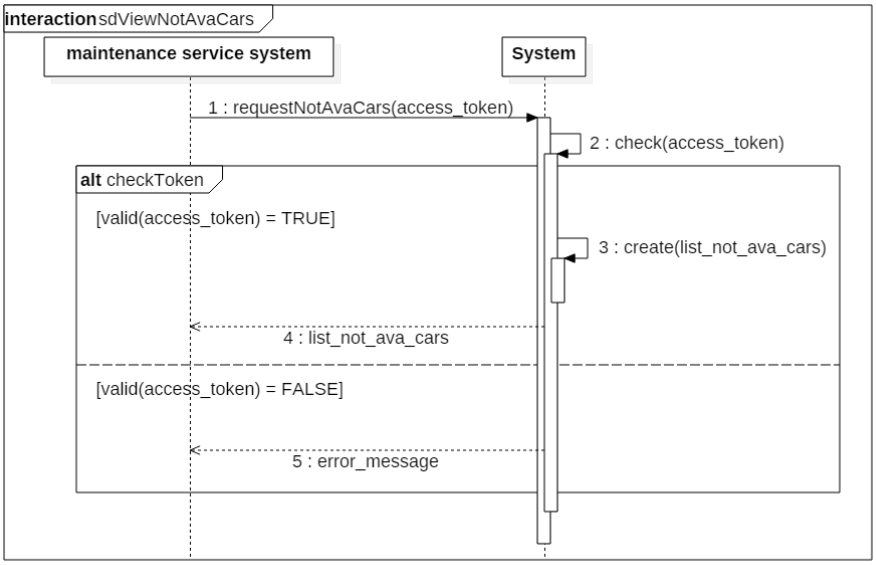

Figure 17: *Visualization of not available cars* sequence diagram

4.3.10 Tag a car as available

Name	**Tag a car as available**
Actors	Maintenance service system
Entry conditions	Maintenance service system must know the access token to be identified by the system
Flow of events	1. The maintenance service system asks to tag a car as Available sending the car identifier paired with the access token 2. The system checks the access token sent by the maintenance service 3. The system checks the identifier received corresponds to a car with state set as *Not Available* 4. The system checks if the car identified by the identifier received is locked 5. The system set the state of the car identified by the aforementioned identifier as *Available* 6. The system sends to the maintenance service system a confirmation message the car state has been set as *Available*
Exit conditions	The car state is set as *Available*
Exceptions	• If the access token sent by the maintenance service system is not recognized, the system sends to the maintenance service system an error message • If the car identifier sent by the maintenance service system is not recognized or doesn't correspond to a car set as *Not Available*, the system sends to the maintenance service system an error message • If the car identifier sent by the maintenance service system corresponds to a car not locked, the system sends to the maintenance service system an error message

Table 10: *Tag a car as not available* use case description

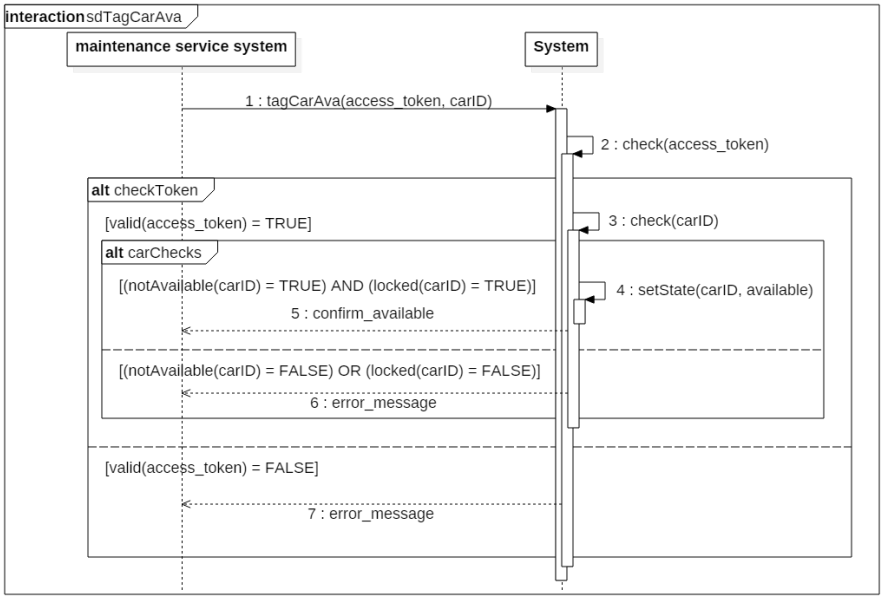

Figure 18: *Tag a car as available* sequence diagram

4.3.11 Visualization of users information

Name	Visualization of users information
Actors	Customer care operator
Entry conditions	
Flow of events	1. The customer care operator inserts the username or the mail of a registered user 2. The system checks if the username or the mail correspond to a user registered to the system 3. The system retrieves user's data (name, surname, birth date and place, current domicile and driving license information) along with information about the car state the user is actually paired with 4. The system shows to the customer care operator the info about the user
Exit conditions	The customer care operator can view the information required about the user
Exceptions	• If no users are found according to the parameters inserted by the customer care operator the system shows an error message

Table 11: *Visualization of users information* use case description

4.3.12 View users payments and rents history

Name	**View users payments and rents history**
Actors	Customer care operator
Entry conditions	
Flow of events	1. **Visualization of users information** 2. The customer care operator asks to view user's payments and rents history 3. The system retrieves the list of user's payments (successful and unsuccessful) 4. The system retrieves the list of user's rents 5. The system shows to the customer care operator user's payments and rents history
Exit conditions	The customer care operator can view the information required about the user
Exceptions	

Table 12: *View users payments and rents history* use case description

4.3.13 Mark or unmark a user as banned

Name	**Mark or unmark a user as banned**
Actors	Customer care operator
Entry conditions	
Flow of events	1. The customer care operator inserts the username of a registered user 2. The customer care operator asks to ban or to enable the registered user paired with the inserted username • If the operator wants to mark a user as *banned* he must insert a brief description of reasons why 3. The system checks if the username corresponds to a user registered to the system 4. The system marks or unmarks the user paired with the username as *banned*
Exit conditions	The state of the user is updated
Exceptions	• If the username inserted by the customer car operator is not recognized, the system shows an error message

Table 13: *Mark or unmark a user as banned* use case description

4.3.14 Tag a car as not available

Name	**Tag a car as not available**
Actors	Customer care operator
Entry conditions	
Flow of events	1. The customer care operator inserts the identifier of the car 2. The customer care operator asks to mark the car as *Not Available* 3. The customer care operator inserts a brief description of why the car state must be set as *Not Available* 4. The system checks the car identifier 5. The system set the state of the car identified by the aforementioned identifier as *Not Available* paired with the description 6. The system shows a confirmation message the car has been tagged as *Not Available*
Exit conditions	The state of the car is setted as *Not Available*
Exceptions	• If car identifier sent by the customer care operator is not recognized, the system displays an error message

Table 14: *Tag a car as not available* use case description

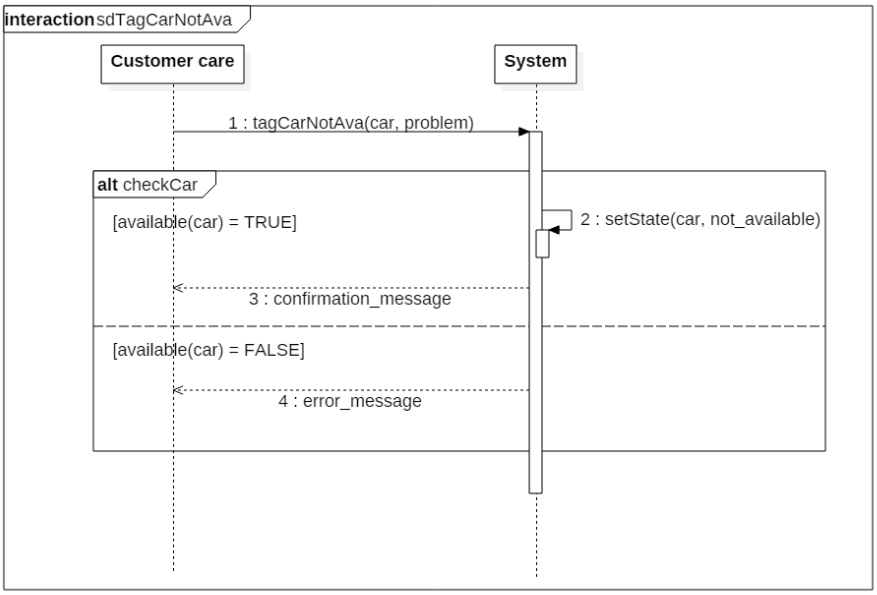

Figure 19: *Tag a car as Not Available* sequence diagram

4.4 UML class diagram

Based on collected scenarios and on the identified use cases we have developed the following requirements-level class diagram[**?**]. To ensure a better readability class attributes are not represented.

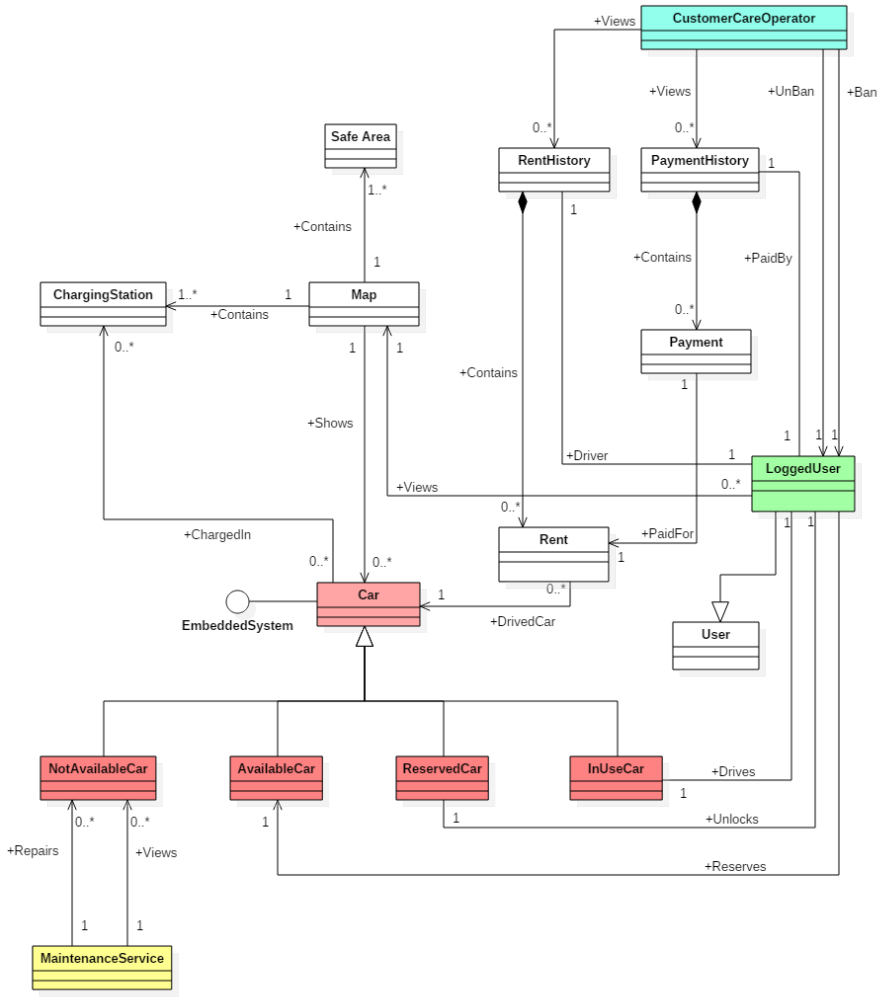

Figure 20: UML class diagram

Appendices

A Alloy model

A.1 Source code

```
open util/boolean

sig Car{
  batteryLevel: one BatteryLevelPercentage,
  status: one CarStatus,
  usedBy: lone LoggedUser,
  reservedBy: lone LoggedUser,
  numberOfPassengers: NOPType,
  onCharge: one Bool,
  engineOn: one Bool
}

//Car statuses
abstract sig CarStatus{}
one sig Available extends CarStatus{}
one sig Reserved extends CarStatus{}
one sig InUse extends CarStatus{}
one sig NotAvailable extends CarStatus{}

//Battery level percentage: should be a percentage 0-100%
abstract sig BatteryLevelPercentage{}
one sig Lower20Full extends BatteryLevelPercentage{}
one sig More50Full extends BatteryLevelPercentage{}
one sig From20to50Full extends BatteryLevelPercentage{}

//Number of passengers, we assume to deal with 5 passengers
    cars
abstract sig NOPType{}
one sig Zero extends NOPType{}
one sig One extends NOPType{}
one sig Two extends NOPType{}
one sig Three extends NOPType{}
one sig Four extends NOPType{}
one sig Five extends NOPType{}

abstract sig User{}
sig LoggedUser extends User{
  //personal information
  //other parameters
  banned: one Bool
}

sig ChargingStation{
  charging: set Car
}

```

```
46 //A RentMade models a rent made in the past, so, for example
       , in the world created
47 //the user who made the rent can be banned or the car used
       for the rent can be NotAvailable
48
49 //If a RentMade corresponds to a reservation expired the
       corrispondent fee is assigned but
50 //others parameters regarding the end of the rent are set to
        default acceptable values
51
52 //leftMSOstation: true iff money saving option enabled and
       auto left on charge
53 //in the station determined by MSO
54
55 //Choice of discount to be applied is not modeled
56 sig RentMade{
57   userRent: one LoggedUser,
58   carRent: one Car,
59   endPosition: one PositionWrtPowerGrid,
60   endSafeArea: one Bool,
61   reservationExpired: one Bool,
62   endBatteryLevel: one BatteryLevelPercentage,
63   onChargeAtTheEnd: one Bool,
64   passengersDuringTheRide: one NOPType,
65   discountApplicableRent: set Discount,
66   additionalFeeRent: set Fee,
67   leftMSOstation: one Bool
68 }
69
70 abstract sig PositionWrtPowerGrid{}
71 one sig More3kmPowerGrid extends PositionWrtPowerGrid{}
72 one sig Lower3kmPowerGrid extends PositionWrtPowerGrid{}
73
74 //M1PD = MoreThan1PassengerDiscount
75 //BHFD = BatteryHalfFullDiscount
76 //CCD = CarOnChargeDiscount
77 //MSOD = MoneySavingOptionDiscount
78 abstract sig Discount{}
79 one sig M1PD extends Discount{}
80 one sig BHFD extends Discount{}
81 one sig CCD extends Discount{}
82 one sig MSOD extends Discount{}
83
84 //REF =ReservationExpiredFee
85 //OSAF =OutSafeAreaFee
86 //A3BCF =Away3kmOrCSBatteryCriticalFee
87 abstract sig Fee{}
88 one sig REF extends Fee{}
89 one sig OSAF extends Fee{}
90 one sig A3BCF extends Fee{}
91
92 //A user can be only in one car at a given time
93 fact OneUserCanBeInOneCarAtSameTime{
```

```
94    no disjoint c1,c2:Car | c1.usedBy = c2.usedBy and c1.
        usedBy != none
95  }
96
97  //A user can reserve only one car at a given time
98  fact ACarReservedByOnlyOneUser{
99    no disjoint c1,c2:Car | c1.reservedBy = c2.reservedBy and
        c1.reservedBy != none
100 }
101
102 //A car in use cannot be reserved
103 fact ACarInUseCannotBeReserved{
104     all c:Car | c.usedBy != none implies c.reservedBy = none
105 }
106
107 //A user cannot use one car and reserve another car at a
      given time
108 fact NoUsersCanUseAndReserveDifferentCars{
109   no disjoint c1,c2:Car | c1.usedBy = c2.reservedBy and c1.
        usedBy != none and c2.reservedBy != none
110 }
111
112 //Cars set as Available cannot be used or reserved at a
      given time
113 fact AvaialableCarsCantBeReservedOrUsed{
114   no c:Car | c.status = Available and (c.usedBy != none or c
        .reservedBy != none)
115 }
116
117 //Cars set as Not Available cannot be used or reserved at a
      given time
118 fact NotAvaialableCarsCantBeReservedOrUsed{
119   no c:Car | c.status = NotAvailable and (c.usedBy != none
        or c.reservedBy != none)
120 }
121
122 //Reserved statuts must be paired with only one user
123 fact ReservedStatusMustBePairedWithOneUser{
124   all c:Car | c.status = Reserved implies (c.reservedBy !=
        none and c.usedBy = none)
125 }
126
127 //In Use statuts must be paired with only one user
128 fact InUseStatusMustBePairedWithOneUser{
129   all c:Car | c.status = InUse implies (c.reservedBy = none
        and c.usedBy != none)
130 }
131
132 //Car with battery percentage lower than 20 percent full
      must be set as Not Available
133 fact CarWithBatteryPercentageLower20FullNotAvailable{
134   all c:Car | (c.batteryLevel = Lower20Full and c.onCharge =
        False) implies c.status = NotAvailable
135 }
```

```
136
137 //A car not in use can not detect number of passengers
        greater than zero
138 fact PassengersOnlyOnInUseCars{
139   no c:Car | c.status != InUse and c.numberOfPassengers !=
        Zero
140 }
141
142 //A car In Use must detect at least one passenger
143 fact AtLeastOnePassengerOnInUseCars{
144   no c:Car | c.status = InUse and c.numberOfPassengers =
        Zero
145 }
146
147 //A car In Use has the engine turned on
148 //Note that a In Use car can have the engine turned off
149 fact OnlyInUseCarEngineOn{
150   all c:Car | c.engineOn = True implies c.status = InUse
151 }
152
153 //A car In Use can not be on charge
154 fact InUseCarNotOnCharge{
155   all c:Car | c.status = InUse implies c.onCharge = False
156 }
157
158 // A car is charging when connected to a charging station
159 fact CarIsChargingWhenConnected{
160   all s:ChargingStation, c:Car | c in s.charging implies c.
        onCharge = True
161   all c:Car | some s:ChargingStation | c.onCharge = True
        implies c in s.charging
162 }
163
164 // At most one charging station connected to a car
165 fact NoMoreOneCSForOneCar{
166   all disjoint s1,s2:ChargingStation | s1.charging & s2.
        charging = none
167 }
168
169 //Banned users cannot deal with cars
170 fact NoBannedUsersDealingWithCars{
171   no u:User | some c:Car | u.banned = True and ( c.usedBy =
        u or c.reservedBy = u )
172 }
173
174 //A REF is applicable if the reservation is expired
175 //No other fee are applicable if the reservation is expired
176 fact ReservationExpiredFeeApplicable{
177   all r:RentMade | r.reservationExpired = True iff (REF in r
        .additionalFeeRent and #r.additionalFeeRent = 1)
178   no r:RentMade | r.reservationExpired = False and REF in r.
        additionalFeeRent
179 }
180
```

```
181 //A reservation expired could not be outside safe area
182 fact NoReservationExpiredOutsideSafeArea{
183   no r:RentMade | r.reservationExpired = True and r.
      endSafeArea = False
184 }
185
186 //No discount are applicable if a reservation is expired
187 fact NoDiscountOrFeeIfReservationExpires{
188   all r:RentMade | r.reservationExpired = True implies r.
      discountApplicableRent = none
189 }
190
191 //No passengers can be detected during the ride if the
      reservation is expired
192 fact NoPassengersIfReservationExpires{
193   all r:RentMade | r.reservationExpired = True iff r.
      passengersDuringTheRide = Zero
194 }
195
196 //M2P discount must be applied iff there are at least two
      passengers detected during the ride
197 fact M1PDiscountAppliable{
198   all r:RentMade |( r.passengersDuringTheRide != Zero and r.
      passengersDuringTheRide != One)
199    iff M1PD in r.discountApplicableRent
200 }
201
202 //BHF discount must be applied iff the car is left with more
       than 50 percent of battery
203 //at the end of the rent
204 fact BHFDiscountAppliable{
205   all r:RentMade | r.endBatteryLevel  = More50Full
206    iff BHFD in r.discountApplicableRent
207 }
208
209 //CC discount must be applied iff the car is left on charge
      at the end of the ride
210 fact CCDiscountAppliable{
211   all r:RentMade | r.onChargeAtTheEnd  = True
212    iff CCD in r.discountApplicableRent
213 }
214
215 //If a car is left on charge at the end of the ride it is
      located inside a safe area
216 fact AllCharginStationInSafeArea{
217   all r:RentMade | r.onChargeAtTheEnd  = True implies r.
      endSafeArea = True
218 }
219
220 //If a car is left in the charge station determined by the
      MSO at the end of the
221 //ride, it is on charge at the end of the ride
222 fact IfLeftMSOStationIsOnCharge{
```

```
223   all r:RentMade | r.leftMSOstation = True implies r.
      onChargeAtTheEnd  = True
224 }
225
226 //MSO discount must be applied iff the car is left in the
      charging station
227 //determined by the MSO
228 fact MSODiscountAppliable{
229   all r:RentMade | r.leftMSOstation = True iff MSOD in r.
      discountApplicableRent
230 }
231
232 //OSA fee must be applied iff the car is left outside a safe
       area at the end of the rent
233 fact OSAFeeMustBeAdded{
234   all r:RentMade | r.endSafeArea = False
235     iff OSAF in r.additionalFeeRent
236 }
237
238 //A3BC fee must be applied if the car is left more than 3km
      away from the nearest
239 //charging station or with battery percentage lower than 20
      percent
240 fact A3BCFeeMustBeAdded{
241   all r:RentMade | r.endPosition  = More3kmPowerGrid
242     implies  A3BCF in r.additionalFeeRent
243   all r:RentMade | r.endBatteryLevel = Lower20Full
244     implies  A3BCF in r.additionalFeeRent
245   all r:RentMade | A3BCF in r.additionalFeeRent
246     implies (r.endPosition  = More3kmPowerGrid or r.
      endBatteryLevel = Lower20Full)
247 }
248
249 //A3BC fee cannot be applied if the car is left on charge
250 fact NoA3BCFeeIfOnCharge{
251     no r:RentMade | r.onChargeAtTheEnd  = True and A3BCF in
      r.additionalFeeRent
252 }
253
254 //If CC discount is applied the end car position can not be
      more than 3km away
255 //from the nearest power grid
256 fact NoCCDMoreThan3km{
257   no r:RentMade | CCD in r.discountApplicableRent and r.
      endPosition = More3kmPowerGrid
258 }
259
260 // Assertions
261 //Can not exists reserved car with engine turned on
262 assert NoReservedCarWithEngineOn{
263   no c:Car | c.engineOn = True and c.status = Reserved
264 }
265 check NoReservedCarWithEngineOn
266
```

```
267 //Can not exists a car in charge with engine turned on
268 assert NoCarInChargeWithEngineOn{
269   no c:Car | c.engineOn = True and c.onCharge = True
270 }
271 check NoCarInChargeWithEngineOn
272
273 //If a car is left on charge at the end of the rent the
        outside safe area fee (OSAF)
274 //can not be applied because alla charging stations are
        inside a safe area
275 assert NoOSAFIfOnChargeAtTheEndRent{
276   no r:RentMade |  r.onChargeAtTheEnd  = True and OSAF in r.
        additionalFeeRent
277 }
278 check NoOSAFIfOnChargeAtTheEndRent
279
280 //If CCD is applied A3BCF can not be applicable and
        viceversa
281 assert NoCCDAndA3BCF{
282   no r:RentMade | CCD in r.discountApplicableRent and A3BCF
        in r.additionalFeeRent
283 }
284 check NoCCDAndA3BCF
285
286 //If CC discount is applied the end car position can not be
        more than 3km away
287 //from the nearest power grid
288 assert NoMSODMoreThan3km{
289   no r:RentMade | MSOD in r.discountApplicableRent and r.
        endPosition = More3kmPowerGrid
290 }
291 check NoMSODMoreThan3km
292
293 //If MSOD is applied A3BCF can not be applicable and
        viceversa
294 assert NoMSODAndA3BCF{
295   no r:RentMade | MSOD in r.discountApplicableRent and A3BCF
         in r.additionalFeeRent
296 }
297 check NoMSODAndA3BCF
298
299 pred show{#charging > 2 some u:LoggedUser | u.banned = True}
300 run show for 10 but exactly 2 ChargingStation, exactly 4 Car
        , exactly 4 LoggedUser, exactly 2 RentMade
```

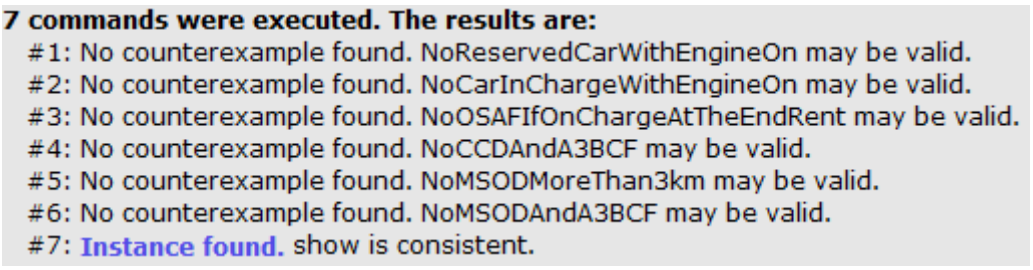

Figure 21: Alloy execution result

A.2 Generated worlds

Note that in **??** LoggedUser3 has been banned *after* completing RentMade0.

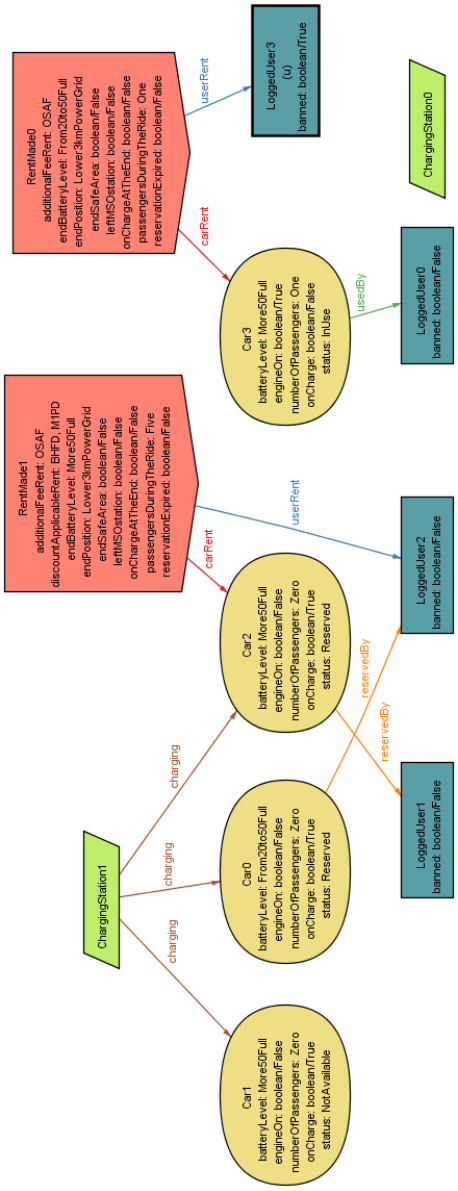

Figure 22: First alloy generated world

Note that in **??** RentMade1 is actually a reservation expired of Car3 made by LoggedUser2. He now has reserved Car1.

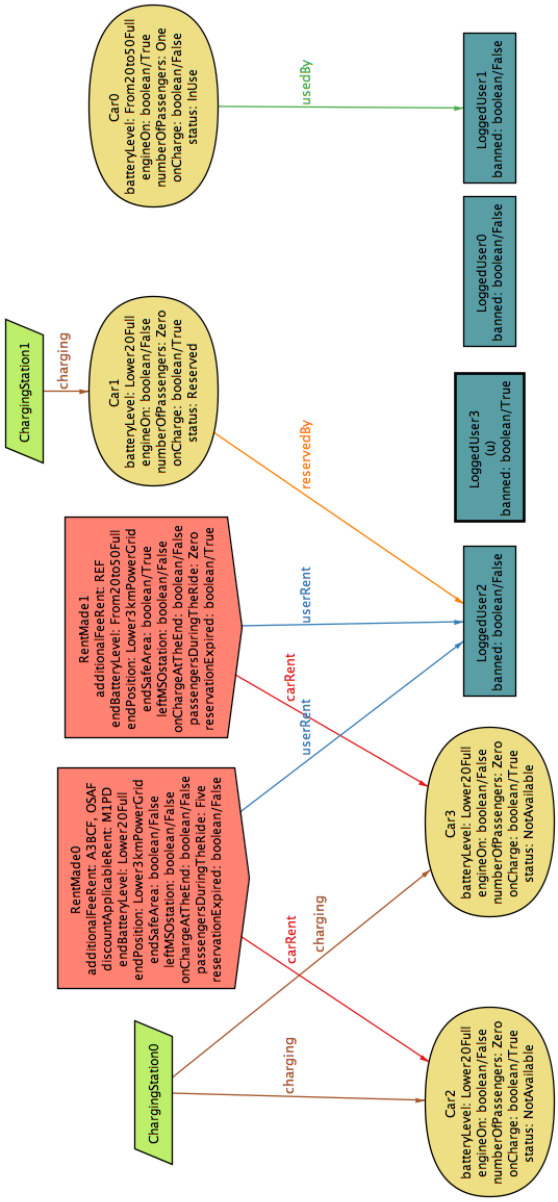

Figure 23: Second alloy generated world

B Software and tools used

For the development of this document we used

- LaTeX as document preparation system
- GitHub as version control system
- Draw.io for graphs

C Hours of Work

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

- **Section 1 - Introduction**
  - Amedeo Cavallo - 2 hours
  - Mattia Calabrese - 1 hour
  - Federico Capaccio - 1 hour

D Changelog

- **v1.0** October 23, 2019
  - Initial RASD document structuring and redaction
  - Introduction (Purpose and Scope sections)

References

[1] B. Nuseibeh, S. Easterbrook, *Requirements Engineering: A Roadmap*, 2000

[2] P. Zave, *Classification of Research Efforts in Requirements Engineering*, ACM Computing Surveys, 1997

[3] E. Di Nitto, L. Mottola, *Software Engineering 2 Assignment*, AA 2019-2020

[4] A. Stone, "Chain of custody: How to ensure digital evidence stands up in court," September 2015

[5] IEEE Std 830:1993, *IEEE Recommended Practice for Software Requirements Specifications*, 1993

B Software and tools used

For the development of this document we used

- LaTeX as document preparation system
- GitHub as version control system
- Draw.io for graphs

C Hours of Work

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

- **Section 1 - Introduction**
  - Amedeo Cavallo - 2 hours
  - Mattia Calabrese - 1 hour
  - Federico Capaccio - 1 hour

D Changelog

- **v1.0** October 23, 2019
  - Initial RASD document structuring and redaction
  - Introduction (Purpose and Scope sections)

References

[1] B. Nuseibeh, S. Easterbrook, *Requirements Engineering: A Roadmap*, 2000

[2] P. Zave, *Classification of Research Efforts in Requirements Engineering*, ACM Computing Surveys, 1997

[3] E. Di Nitto, L. Mottola, *Software Engineering 2 Assignment*, AA 2019-2020

[4] A. Stone, "Chain of custody: How to ensure digital evidence stands up in court," September 2015

[5] IEEE Std 830:1993, *IEEE Recommended Practice for Software Requirements Specifications*, 1993