Requirements Analysis and Specifications Document

Document Version 4.6

Matteo Frosi (mat. 875393)
Luca Costa (mat. 808109)

Contents

1 Introduction

1.1 Purpose of the system and brief description of the problem

PowerEnJoy is an electrical car sharing service, based on a mobile application. The targets of the service, intended as users, are people that needs to move from a place to another within a city and requires a conveyance to move (because they don't have their own or simply can't use it).
A user can make a reservation for a car, using the mobile app and his/her account, and check for the availability and status of all the cars within his/her position, identified using GPS localization, or a specific one, inserted manually by the user. As stated before, to access the service, the user must possess a private account, so a registration is needed.
The system provides the users a safe way (identification code) to access the cars, and the riding service and keeps trace of the status of all the cars.
Moreover, the system prizes or punishes a respectively good or bad behavior from the users, applying a discount or an overcharge on the cost of a ride. As example, if the user leaves the car without much battery, he/she will have to pay more than the standard cost of the ride, because the car will need to be charged and this operation has a cost. On the other hand, if a user plugs the car before ending the service, it receives a discount.
The system includes other functionality, such as GPS based maps available in every car, an emergency procedure in case an accident occur during a ride and the notification of a car status if the user requested it.

The purpose of the system, referring to both the main and each car systems, is to offer a reliable, efficient and safe service, allowing the user to benefit of the offered electric cars when needed.

1.2 Actors and stakeholders

We can clearly make a distinction between who asks for the service and who will actually use it.

- As in the Car2Go service, the stakeholder may be a corporation related to the automotive world. However we don't exclude the possibility that a city government could ask for this software, to improve the transport condition of the city or simply to lessen the environmental impact of transport.
- The clients of the service are the so called users. User is whoever needs the car sharing service benefits of it.

We can also increase the scope to include more actors, implicitly present in the service: the cars, that plays a fundamental role of interaction between the user and the system (other than being a brainless transport), and the operators of companies that provides services like technical and emergency support.

1.3 Goals

- [G1] Ensures that only the account owner can access his/her own account, if he/she is already registered.
- [G2] Allows the user to reserve an available car for up to one hour before the service starts and to take back such reservation before the picking up time expires.

- [G3] Allows the user to look for the available cars within an area that he/she specified, or in the nearby areas, for a possible reservation. If the car has a low battery charge, less than 20%, it does not appear on the user interface.
- [G4] Ensures that the user is able to trace/check the status of a car, available or already reserved, and receive push notifications about it.
- [G5] Ensures that the user is able to access, and open, the car he/she reserved, if he/she is in time.
- [G6] Ensures that the service, and the charge on the user, starts only when the engine of a car is ignited.
- [G7] Ensures that the user is informed real-time on the current cost of the service, when using it, and the possible discounts and overcharges.
- [G8] Ensures that the user knows the safe area around him/her, during the ride.
- [G9] Allows the user to press a button on the car to contact an operator in case of emergency, such as a car accident, a mechanical problem or sudden illness.
- [G10] Allows the user to leave the car but continue to use the service.
- [G11] Ensures that the user can end the service at any time.
- [G12] Ensures that the user receives a discount or overcharge when specific conditions are met.
- [G13] Ensures that the user is notified of the applied discount and overcharges once the service ends and payment is applied.
- [G14] Ensures that when the user picks up a car, it has no mechanical or electrical problem.

1.4 Goals further description

- [G1] leaves out a particular scenario. The system won't actually be able to know if the person accessing the account is the real owner. A user can just give the credential to a friend, and this friend can access and use the service as if it was the real user. Such case, is a decision, or problem, of the user and the system can't do anything to avoid or simply recognize it.
- [G3] denies the possibility to use a car when the battery is very low, less than 20% of charge. This represents the behavior of a person in the real world. Nobody would chose a car if it would not be able to reach the destination. A user would prefer a charged car, to drive for longer distances.
- [G4] derives from the idea that multiple users may want to use the same available car. Only one is able to make a reservation, but the others may be interested to check if the car is effectively taken by the reserving user. It could happen that he/she takes back the reservation or simply doesn't pick up the car within the reservation time. In such case, the car would be available once again and all the interested users will receive a notification about it.
- [G9] is one of the functional goals that provides the user a reliable way to "help" the service to work correctly. Being notified of the nearby zones to park, it will be unlikely for him/her to abandon the car midway.

- [G10] considers the real world, where a user may take the car to do shopping. The service provided should allow the user to stop and do his/her business.
- [G12] refers to the particular condition expressed in the project description. If the user takes on two ore more passengers, he/she receives a discount of 10% on the cost of the ride. If the user leaves the car with more than 50% of the battery available, a discount of 20% is applied. If the user plugs in the car before ending the service, the applied discount is 30% over the total. However, if the user leaves the car 3 km far from a safe area or with less than 20% of the battery available, he/she is overcharged of 30% of the cost of the ride.

1.5 Glossary

- User: is the person that benefits of the car-sharing service. The user is able to make reservations, check cars and ask for notification using his/her own account. To register and be able to use the service, the user must provide name, surname, phone number, e-mail, payment information, such as a credit card number, and his/her Driver License. The user is, in fact, the person who drives a car.
- Car: is one of the electric car provided by the service. The car contains various sensors which are able to detect the number of passengers, the status of the mechanical and electrical components of the car. It also possess its own system, connected to the main system of the service. This system is constantly updated on the status of the car. The car has also a screen that makes possible an interaction between the user and the car system, and a button that the user can press in case of emergency.
- Mobile application: it is the app that the user has to install on his/her mobile device in order to use the service.
- Reservation: it's the process by which the user can make a request for a car. First, the user must access his/her own account. From there, he can insert an address or chose his location, via GPS, to search for available cars from the suggested zone. Lastly, he can request a specific car to use.
- Monitoring: it it the process by which the user can chose one or more cars to be notified about its availability. The monitoring only involves cars that are not already picked up and
- GPS: global navigation satellite system that provides location and time information in all weather conditions, anywhere on or near the Earth where there is an unobstructed line of sight to four or more GPS satellites.
- Picking up time / Reservation time: it is the period of time between the time when the user reserves a car from his/her account and the unlocking of the car by the same user. The maximum reservation time, allowed by the system, is one hour.
- Reservation countdown: it is the countdown after which the user is charged of a fee if it has not picked up the car.
- Status: referred to a car, it is the set of the information that describes it. They include:
  - Location
  - Battery charge
  - Money counter

  - Components state

- Passenger: is a person who can benefit of the service without driving the car. The passenger can use a car only when an user is driving it, and it doesn't need to be registered to use the service.
- Technician: is the person dedicated to repair and fix the car. It is provided by an external company.
- Operator: is the person that communicates with the user whenever an accident happens during the service and emergency button on the used car is pressed. The operator contacts other services depending on which type of accident occurred, such as ambulance, police, firetrucks, technicians. As the latter ones, the operator is provided by an external company.
- Safe area: it is a specific area where the electric cars of PowerEnJoy service can park. The set of safe areas is pre-defined and owned by the company/society that requested the management system for the service.
- Special safe area: it is a safe area where power grid stations are installed.
- Power grid station: it is an installation that allows the recharge of an electric car.
- System: it is the new system to create. It refers to the software and hardware needed in order to make the service work. The system is composed of a database (or more than one) to store the users and cars information, and the software needed to manage users actions and the single cars.
- Car system: is the system, mostly hardware, contained in every car. It checks and elaborates the status of the car and interfaces the transport, the user and the system. The user can interact with the car thanks to an installed screen, that shows information about the service cost, a GPS based map, discounts and so on. Every car system sends information to the system via 3G connection, and viceversa.
- Discount: is an amount of money that has to be subtracted from the ride cost of a user, at the end of the service, if certain conditions (described in section 1.1, goals specification 3) are met.
- Overcharge: is an amount of money that has to be added to the ride cost of a user, at the end of the service, if certain conditions (described in section 1.1, goals specification 3) are met.
- Ride: it is the journey in the electric car. It starts when the user turns on the car and ends when the user leaves the car and closes the open doors. In the way a ride is defined, its duration coincides with the period of time where the user is charged for the service.
- Service: it refers to the whole process of reservation, ride and payment of a car, done by the user.
- Push notification/ push message: it is a notification sent to a smartphone using the mobile application.
- SMS: short message service; it is a notification sent to a mobile phone, we need a GSM gateway to use it.
- GMS gateway: device that allows SMS text messages to be sent and/or received by email, from Web pages or from other software applications by acquiring a unique identifier from the mobile phone's Subscriber Identity Module, or SIM card.
- PSP (payment service provider): offers shops online services for accepting electronic payments by a variety of payment methods including credit card.

1.6 Text assumptions

- We do not consider the case where someone can use the credentials of someone else, maybe a friend: such possibility is purely user-related and the system has no way to control it.
- We should develop a mobile application able to use the position of a device through a GPS or asking it to the user.
- Users are asked to make a reservation before using an electric car. Other car-sharing service like Car2Go allow also a use without reservation, but such case is not contemplated.
- There must be some components that allow an interaction between the user and the car, to make the first able to access the second. We assume the existence of a numeric keypad connected to the internal car system. Such keypad is also used by technicians to unlock the car during repairs and controls.
- Users must first register to gain access to the main functionality offered by the system.
- The car possesses a screen to make possible an interaction with the user and the car system.
- The ride fee calculated per minute depends on the situation. An higher fee is applied when using the car, while a lower fee is applied if the car is not parked in a safe zone and the service is still active. If the car is parked in a safe area, the service is free.
- We assume that the parking areas are provided by the PowerEnJoy owners.
- The system is able to control some functionality of the cars, such as the door lock.
- The car possesses multiple sensors that allows, every moment, to know its status, such as battery charge, decay of the mechanical components or number of passengers.
- We assume that there are other societies that collaborate with the system of PowerEnJoy. Technicians, operators, and other actors belong to such companies.

2 Overall description

2.1 Interfaces description

- Interface with GSM provider to send SMS notifications to the users.
- Interface with the external staff. Technicians and operators are provided by a different company and the system needs to interact with them.
- Interface between the user and the car system. There are two levels of interaction:
  - The user can check for general informations of a car via his/her account, on the app.
  - The user has more detailed informations on the car, the service cost and the location during the ride, interacting with the car screen.
- Interface between the user and the system, via app. The user is able to access the offered services thanks to the application.
- Interface between the car system and the system, to enable the communication of data.
- Interface between the system and the GPS map provider.
- Interface between the cars and the recharge stations, that enables the recharge of the vehicle. It is composed by a plug and sensors connected to the car system that are able to recognize if the car is plugged (and recharging).
- Interface between the cars and the people, both users and technicians. This interface refers to a device connected to the car system that is able to recognize codes and unlock the car.

2.2 Constraints

2.2.1 Regulatory policies

When registering, the system shows the user the license he/she has to accept in order to benefit from the offered services.
Treatment of personal data follows the rules described by the law (Law D. Lgs. 196/2003 in Italy).
The system also asks the permission to use and manage sensible data such as position, mail and phone number, in order to achieve its purposes, respecting in every conditions and cases the privacy law.
The system mustn't use push notifications and SMS to send SPAM and third party advertising, but can only use such means according to the offered services.

2.2.2 Possible software limitations

Problem regarding API management and integration can arise, leading to major complexities of the system and delays in the development of the system software.
The software that manages the system may not implement all the needed functionality. Such problem is mostly due to a bad development.

2.2.3 Hardware limitations

The mobile device owned by the user is subject to many problems, mostly regarding the connectivity of network and localization services:

- 3G connection, to use the app
- GPS, to use particular services of the app, such as providing the user position to find available cars
- Space for app package on the device memory
- Battery consumption

2.3 Proposed system

We will implement a client-server architecture, based on the common REST APIs, to manage the web application. A MySQL based database is needed to store the data of users and cars. A possible framework for the server is Mono, derived on the .NET Framework offered by Microsoft, but working on a large number of OS, such as iOS, Android, OSX and Microsoft. Lastly, the car interfaces with the user using some of its artifacts, such as the emergency button (User -> Car System -> System -> Operators), the car screen or the numeric keypad to access the car or to end the service.

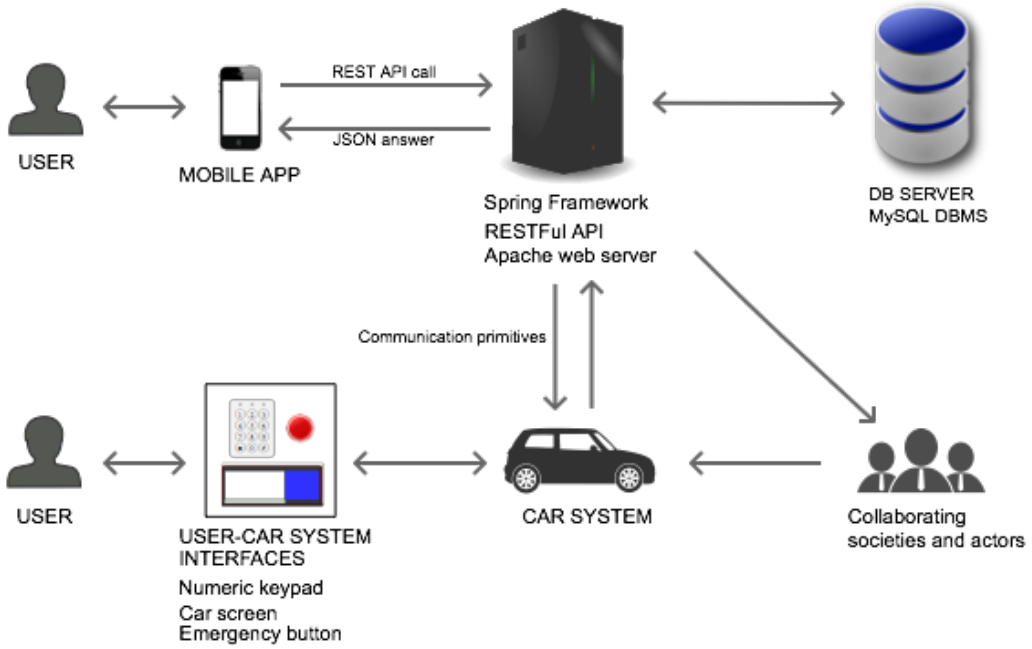


Figure 1: Architecture

2.4 Domain assumptions

- All the GPS localization services have no fault and always return the correct position of a device (car, phone, tablet...).
- If a car is locked, nobody can access it. However, technicians are able to use a code to operate on it.
- The databases on which the system relies on works without problems (no data loss, no incoherence).
- Once a user is registered, the system won't accept any other registration with his/her data. In other words, a user can register only once.
- Once a user is registered, his/her information will be saved and present in the system database.
- All the notifications that a user can receive from his/her account are immediate and always working.
- All the updates of the status of a car are immediate and correct.
- A user will never leave the car in a zone unreachable by GPS localization.
- A user will never leave the car with a door open, and neither a passenger will.
- Users will encounter no problems when accessing the car using a code.
- When registering, users will insert only credit card info as methods of payment.
- Technicians and operators are always available and provided by a third party society.
- Passwords and codes are unique.
- Discounts and overcharges are always calculated and applied in the correct way.
- Technicians and other services (police, ambulances, firetrucks) will reach the location addressed by the user/operator in the shortest possible time.
- Every car is able to detect it a mechanical/electrical problem occurs, and modify its internal status.
- No fault occurs to a car during a reservation or monitoring event (so only when the car is available).
- The set of safe areas for parking cars is pre-defined by the management system.
- The set of safe areas can change, so that new safe area are introduced or removed from the available ones.

3 Requirements

3.1 Functional requirements

- [G1] Ensures that only the account owner can access his/her own account, if he/she is already registered:
  - [R1.1] When the user registers, the system shall send him/her a mail or an SMS containing an unique code, that she/he can use to access his/her account.
  - [R1.2] When the user inserts is credential to log into his/her account, the system shall verify the correctness of the inserted data, the unique code expressed in requirement [R1.1].
- [G2] Allows the user to reserve an available car for up to one hour before the service starts and to take back such reservation before the picking up time expires:
  - [R2.1] When the user selects the option to reserve a car, the system shall verify if it is already reserved.
  - [R2.2] When the user selects the option to reserve a car, the system shall check if the reservation time is less than one hour.
  - [R2.3] If the check described in requirement [R2.2] is negative, meaning that the user expressed a reservation time longer than one hour, the system shall reject the user's request.
  - [R2.4] When the user selects the option to reserve a car, the system shall verify if it the user has already made a reservation. If he/she has, the system shall notify him/her with a message on the mobile screen.
  - [R2.5] If the reservation time expires, meaning that the user didn't reach the car in time, the system shall cancel the user reservation and charge him/her of 1 EUR. The system shall also send a push notification the user to inform him/her of the fee.
  - [R2.6] When the user makes a reservation for an available car, the system shall provide him/her an option to take back the reservation, from his/her account.
  - [R2.7] When the user picks up the car or the reservation time expires, the system shall disable the option to take back the reservation.
  - [R2.8] When the user picks up the car, the car system shall notify the system and disable the reservation countdown.
- [G3] Allows the user to look for the available cars within an area that he/she specified, or in the nearby areas, for a possible reservation. If the car has a low battery charge, such as less than 20%, it does not appear on the user interface:
  - [R3.1] The system shall be able to detect the user position, using GPS localisation of the device he/she is using (automatic localisation).
  - [R3.2] The system shall be able to recognise and locate the address given by the user (manual localisation).
  - [R3.3] The system shall display a list, interfaced to the user as a map, of the available cars with more than 20% of the battery charge, within 5km from the given location.

  - [R3.4] For every car in the list described in requirement [R3.3], the system shall provide essential information, as battery charge, estimated working distance, location, distance from the given address/current user location, maximum number of passengers.

- [G4] Ensures that the user is able to trace/check the status of a car, available or already reserved, and receive push notification about it:
  - [R4.1] Along with requirement [R3.3], the system shall also display a list of already reserved car, but not already picked up.
  - [R4.2] When the user touches a car on the map of his/her app, the system shall provide him/her two options regarding that car: reservation or monitoring, depending on the car availability. Moreover, the status of the car shall be displayed.
  - [R4.3] When the status of a car is updated - from reserved to available, from reserved to picked up and from available to reserved - the system shall notify all the users that made a monitoring request, grouped in a list, on that car.
  - [R4.4] When the user decides to not be notified on a car for which he/she previously requested a monitoring, the system shall delete the user from the list of users that wants to check the status of that car.
  - [R4.5] When a car is picked up, the system shall notify the users that made the monitoring request, as described in [R4.3] and after that, deletes the list.

- [G5] Ensures that the user is able to access, and open, the car he/she reserved, if he/she is in time:
  - [R5.1] When the user communicate via his/her account that he/she's nearby the car he/she reserved, the system shall send an SMS or a mail to the user, containing a unique code to unlock the car.
  - [R5.2] When the user communicate via his/her account that he/she's nearby the car he/she reserved, the system shall make the car unlockable only by the code descripted in requirement [R5.1].
  - [R5.3] If the user doesn't reach the car within the reservation time and he/she already received the code to open it, the system shall reset the sequence needed to open the car, making the user unable to access it.

- [G6] Ensures that the service, and the charge on the user, starts only when the engine of a car is ignited:
  - [R6.1] When the user is in the turned off car and uses its ignition key, the system shall start the car, initializing its screen and the car internal system.
  - [R6.2] When the user is in the turned off car and uses its ignition key, the system shall begin to charge the user per minute, after having initialized the internal system of the car, as stated in requirement [R6.1].

- [G7] Ensures that the user is informed real-time on the current cost of the service, when using it, and the possible discounts and overcharges:
  - [R7.1] Every minute, the car system shall increase the money counter and display its value on the screen of the car, depending on its status. Different rates are applied if the car is turne on or off while the service is active.

  - [R7.2] When the car system detects that more than two passengers are on the car, it shall display on the car screen a message about a possible 10% discount on the ride cost if the condition persists.
  - [R7.3] When the car system detects that the battery charge of the car is above 50%, it shall display on the car screen a message about a possible 20% discount on the ride cost if the condition persists.
  - [R7.4] When the car system detects that the battery charge of the car is below 20%, it shall display on the car screen a message about a possible 30% overcharge on the ride cost if the condition persists.
  - [R7.5] When a condition described in requirements [R7.2], [R7.3] and [R7.4] is not met, the car system shall display no message on the car screen about discounts/overcharges.

- [G8] Ensures that the user knows the safe areas around him/her during the ride:
  - [R8.1] When the service starts, the car system shall elaborate a detailed map of a 10km radius zone from the car location, using GPS. The map status in continuosly elaborated during the ride.
  - [R8.2] Referring to requirement [R8.1] the car system shall show the map on the car screen, along with the car status, as stated in requirement [R7.X].
  - [R8.3] When the service starts, in addition to requirement [R8.1], the system shall inform the car system of the location of the safe areas, distinguishing between normal safe areas and the ones with power grids. Only the safe areas with available parking places are shown.

- [G9] Allows the user to press a button on the car to contact an operator in case of emergency, such as a car accident, a mechanical problem or sudden ilness:
  - [R9.1] When the user presses the button, the system shall notify an operator on the location of the car and its status. The operator belongs to an outer service provider agency.
  - [R9.2] When the user presses the button, the system shall arrange a call between the user inside the car and the operator of requirement [R9.1].
  - [R9.3] If the user presses the button multiple times in a short period, less than 2 minutes, the system shall recognise only the first and start the procedure described in the requirements [R9.1] and [R9.2].
  - [R9.4] When the user presses the button, the car system shall end the service (following the ending procedure) and turn off the car.

- [G10] Allows the user to leave the car but continue to use the service.
  - [R10.1] When the user leaves and turns off the car, the system shall apply a new rate over the ride cost. When the user turns on the car once again, the old rate is applied once again and the riding service continues.
  - [R10.2] After a fixed amount of time from when the user left the car, the system shall send a push notification to the user, reminding him/her that the service is still ongoing.

- [G11] Ensures that the user can end the service at any time:
  - [R11.1] When the user leaves and turns off the car, the car system shall check if the car is located in a safe area or not, communicating with the system.

  - [R11.2] If the user has left the car in a safe area, the car system shall stop the charging counter and save the car status.
  - [R11.3] If the user has left the car in a non safe area, the car system shall continue charging the user with a different rate.
  - [R11.4] When the user has left the car, closed all the doors, and inserted into the numeric keypad the code he/she used to access the car, the car system shall communicate the car status to the system. The system shall execute the payment process, communicate the car system to lock the car.
  - [R11.5] When the car system has locked the car and informed the system, the latter shall update the car status in the main database, and tag the car available once again.

- [G12] Ensures that the user receives a discount or overcharge when specific conditions are met:
  - [R12.1] When the user uses the access code to end the service and the car system provided the car status to the system, the system shall check the battery of the used car. If more than 50% remains the system shall apply a discount of 20% on the cost of the ride with that car. If less than 20% remains, the system shall apply an overcharge of 30% on the cost of the ride with that car.
  - [R12.2] When the user uses the access code to end the service and the car system provided the car status to the system, the system shall check the number of passengers. If it is more than two the system shall apply a discount of 10% on the cost of the ride with that car.
  - [R12.3] When the user uses the access code to end the service and the car system provided the car status to the system, the system shall check if the car is plugged into a power grid. If such condition is met and the car is located in a safe-area, the system shall apply a discount of 30% on the cost of the ride with that car.
  - [R12.4] When the user uses the access code to end the service and the car system provided the car status to the system, if the car it is located at more than 3km from the nearest safe area, the system shall apply an overcharge of 30% on the cost of the ride with that car.

- [G13] Ensures that the user is notified of the applied discount and overcharges once the service ends and payment is applied:
  - [R13.1] When the system applies the payment process on the user, it shall inform the user with a push notification of the applied discount and overcharges, based on the requirements [R12.X].

- [G14] Ensures that when the user picks up a car, it has no mechanical or electrical problem:
  - [R14.1] When a car detects a problem, the car system shall inform the main system of the new status. The system shall then contact third party technicians, giving them the location of the car and a unique code to access it. Moreover, the system shall tag the car as unavailable.
  - [R14.2] When the car system notifies the main system that all the problems are solved, the system shall update the car status, reset the access code of the car and tag it as available.

3.2 Non functional requirements

3.2.1 Usability

- [R15] The system shall provide different languages that the users can chose.
- [R16] Both the mobile app and the car screen shall use different way to represent safe area and special areas with power grid stations on the maps.
- [R17] The mobile application shall be installable on different OS.
- [R18] Along with the push notifications, the mobile application shall emit a brief sound.
- [R19] The system shall provide trivial information about the service, such as discounts and overcharges conditions, rate table and contacts.

3.2.2 Performance

All the possible issues regarding the performance of the whole system (system, mobile app and car systems) are already covered in the domain assumptions and as such not considered as requirements.

- Operations are elaborated and completed without delays and in a short amount of time. With short we mean a time comparable with the times of the current systems. The discount calculation won't take hours, as example.
- The menus and screens of all the devices will be smooth and without graphical problems.

3.2.3 Interfaces mockup

Figure 2: Map search mockup

Figure 3: Registration mockup

4 Modeling

4.1 Scenario identifying

In this sub-section we present some of the possible situations that can happen.

4.1.1 Scenario 1

Lilith just finished her work and can not wait to go home. However her car had some problems in the past few days and this morning Lilith decided to have it fixed. She opens her PowerEnJoy app on her phone and logs in her personal account. From there, she searches for the available cars around her. Fortunately, Lilith finds a car 500 meters from her position. The car is available and with a high battery charge, 75%. Lilith makes the reservation, specifying that in about half of an hour she will pick up the car. It just takes 20 minutes to reach the car and Lilith uses a special option on her page to receive via SMS a special code to unlock the car. The girl picks up the car and drives, enjoying the ride. She parks the car in the nearest safe area from her home, ends the service using the previous code and walks away.

4.1.2 Scenario 2

Mitch needs to go pick up his son at the elementary school and wants to use the PowerEnJoy service. He opens the app, logs in and reserves a car. Lilith just got out of her office and needs to go shopping for some food. She lend her car to her younger brother and because she is already accustomed to the PowerEnJoy service, she immediately logs in her account and searches for an available car. However, there is only one car in the neighborhood and it is already reserved (by Mitch). Lilith is faithful that the one that reserved it will have some problems and take back the reservation, so she requests to monitor the car and receives notifications about it. Meanwhile Mitch receives a phone call from her wife, telling him that she already picked up their son. The reservation time still hasn't expired yet and Mitch is able to take back his reservation. Lilith receives then a notification on her cellphone that the car she was monitoring became available. Happy about how things turned out, she reserves the car, goes pick it up and drives to a shopping center.

4.1.3 Scenario 3

Mitch and his wife are going on a date and they are using a car offered by the PowerEnJoy service. Only Mitch needed to log in his account and make the reservation, before picking up the car, because he is the one who drives, while his wife is only a passenger. Mitch's wife is enjoying the ride but suddenly shouts to her husband to stop the car immediately: they just passed in front of a flower shop, and she loves flowers. Mitch satisfies the request of his wife, parks the car, after they have exited, he locks it with the ignition key. They pass about an hour in that shop, having fun. After that, Mitch opens the car and starts driving again, continuing the rest of the date with his wife.

4.1.4 Scenario 4

Lilith and her two sisters just got out of the gym and wants to go eat something. Hilda, one of the sisters, suggests a nice restaurant but it's pretty far from them, and they don't want to catch a bus to go there. Lilith tells then her sisters about PowerEnJoy and shows them how to make a reservation. Fortunately they find an available car just 100 meters from them. They go pick up the car and Lilith, the one who made the reservation, drives to the restaurant. Realizing that they will spend a lot of time there, Lilith decides

to end the service when exiting the car, inserting once again the code she used to access the car. During the meal, Lilith realizes that after having picked up the car she turned off her phone. Once she turns it on, she sees that not long ago, just after the end of the service, she received a notification about the cost of the ride and its discounts. However, what she sees is unexpected: she received a discount because she took two passengers, her sisters, in the car, but she also had an overcharge of 30% because she parked the car 5 km away from the nearest safe area. Hilda laughing says: "I told you it was pretty far!". Poor Lilith, looks like she won't enjoy her meal after all.

4.1.5 Scenario 5

Lilith has been working for 10 straight hours and feels very tired. She opens her PowerEnJoy application, searches for the available cars near her and reserves one. After having picked up the car using a special code received via SMS, she starts driving. Unfortunately, when waiting at a semaphore, someone collides with the car that Lilith is driving. No one is hurt, but both the cars are in a bad shape. Lilith knows what to do and quickly presses the emergency button on the car and after a few seconds she is able to speak with an operator to explain the situation. The person that picks up the emergency call is Marie, that asks Lilith information about the accident, such as location, type of problem and general information. After that, Marie contacts a society that allocates some technicians and send them to the accident point. If needed, Marie contacts also police, ambulances and firefighters. Lilith doesn't have to worry: when she pressed the button, the service ended and she doesn't have to pay more than needed. Unlucky Lilith, looks like she won't be able to relax after her hard day.

4.1.6 Scenario 6

Mitch and his wife are thinking to go on a vacation, but they are undecided about the destination. The travel agency isn't far from their home so they decide to book an electric car and use it. Mitch opens the PowerEnJoy application, searches for the available cars and find one available. After having reserved it, requested the code to access it, and picked it up, he drives, directed to the destination. Having used it a lot of times, Mitch knows the rate of the service, calculated per minute, and realizes that on his credit card there isn't enough money to pay the ride. However he also knows all the possible discounts and to spend less he parks the car in a safe zone with a power grid station and plugs it to recharge the battery. Luckily for him, he also left the car with more than 50% of the battery, so as soon as he ends the service, he receives a notification on his phone about the 30% discount due to the plugging in and 20% due to the battery level. Mitch is very happy because he only payed half of the cost, but so isn't his wife: to receive the first discount he parked 2km away from the travel agency, so they now have to go by feet.

4.1.7 Scenario 7

Lilith and her sister Hilda are talking about the services offered by PowerEnJoy. Lilith is actually a registered user and she appreciates the car sharing service, so she decided to invite her sister and make her register to benefits of PowerEnJoy. Lilith explains that in order to register, Hilda must insert in the app, under the registration section, her name, surname, number phone, mail, Driver License information (number and postcode) and the optional fiscal code (of the ID Card). Hilda inserts all the information carefully and without errors, and after less then a minute, she receives an SMS and a mail containing the password to log in to her personal area of the app.

4.1.8 Scenario 8

A car is waiting for the next user that will make a reservation and use it. However, suddenly one sensor detects that there is a problem in the battery charge, it is not stable but the monitoring goes up and down, as if the car was heavily charged and discharged simultaneously. The car system communicate this to the system, in one of the usual periodic update that it does. The system makes than a request to an external operator, providing an access code to that car. In the next hours, a technicians reaches the car and fixes it.

4.2 UML models

4.2.1 Use case diagram

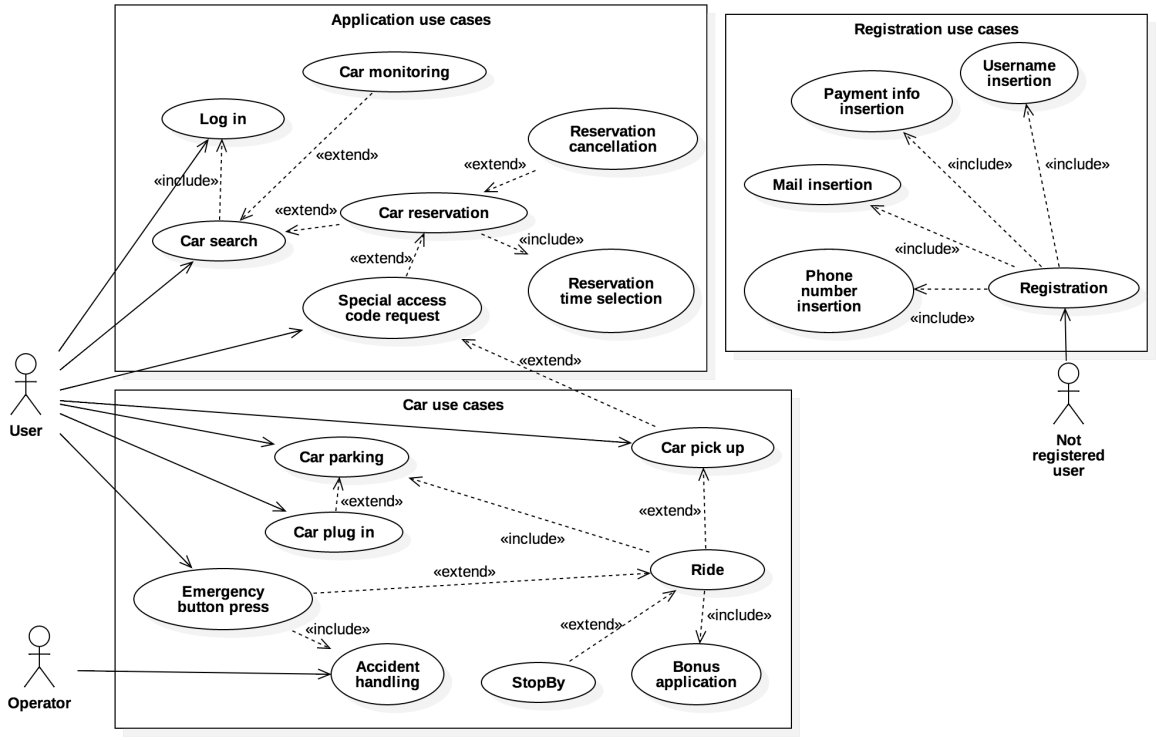


Figure 4: Use case diagram, portraying all the functionality that the user expects from the service

4.2.2 Use case descriptions

UseCase ID: 1
Name: Normal usage of the service
Actors: User (Lilith, Scenario 1)
Entry conditions: The user is already registered to the application.
Flow of events:

- Sub Use Case 1
  - **UseCase ID:** 1.1
  - **Name:** Log in

  - **Entry conditions:** None
  - **Flow of events:**
    * The user inserts the log in credentials in the dedicated area of the app.
    * The system checks for the validity of the inserted credentials.
  - **Exit conditions:** The system confirms the inserted credentials and the user identity, allowing him/her to access the whole service benefits.
  - **Exceptions:**
    * The credentials inserted by the client are not correct. In this case the system doesn't redirect the user to the main page of the app but notifies him/her with an error message on the screen.

- Sub Use Case 2
  - **UseCase ID:** 1.2
  - **Name:** Reservation procedure
  - **Entry conditions:** The user logged in correctly.
  - **Flow of events:**
    * The user searches for an available car using one of the sections of the app.
    * The system shows only the available cars, with more than 20% of the battery charged.
    * The user selects a car and inputs the time he/she thinks will be necessary to reach the car and pick it up.
    * The system adds an area allowing the user to take back the reservation.
    * When in proximity of the car (the user decides how much), the user uses a special area on the app to request a code to access the car.
    * The system sends the user an SMS or mail containing the access code.
  - **Exit conditions:** The user picks up the car using the code and the system disables the countdown and the option to take back the reservation.
  - **Exceptions:**
    * The user inserted an invalid reservation time so that his/her request is rejected.
    * The user doesn't pick up in time the car, so it has to pay a fee of 1 EUR. If the user already received the code to access the car, the system invalidates it.
    * The user takes back the reservation.

- Sub Use Case 3
  - **UseCase ID:** 1.3
  - **Name:** Ride
  - **Entry conditions:** The user picked up the car in time.
  - **Flow of events:**
    * The user turns on the car with its ignition key.
    * The system starts charging the user per minute.
    * The user drives.

    * The user searches on the car screen an available parking spot in a displayed safe area (optional).
    * The user parks the car. The parking can be anywhere permitted by the law, not necessarily in a safe area.
  - **Exit conditions:** The turns off the car and leaves it.
  - **Exceptions:**
    * An accident may occur, so the user uses the emergency button, ending the service immediately.

Exit conditions: The user ends the service using on the car the access code thanks which he/she opened it.

UseCase ID: 2
Name: Cancellation of a reservation
Actors: User (Mitch, scenario 2)
Entry conditions: The user is already registered to the application.
Flow of events:

- The user logs in the app using his/her credentials.
- The user searches for an available car using a special section of the app.
- The user selects an available car and inputs the time he/she thinks will be necessary to reach the car and pick it up.
- Before the reservation time expires, the user selects the cancellation option from his/her app.

Exit conditions: The user cancels the reservation he/she previously made.
Exceptions:

- The credentials furnished by the client are not correct. In this case the system doesn't redirect the user to the main page of the app but notifies him/her with an error message on the screen.
- The user doesn't pick up in time the car, so it has to pay a fee of 1 EUR and cannot cancel the reservation anymore.

UseCase ID: 3
Name: Monitoring of cars
Actors: User (Lilith, scenario 2)
Entry conditions: The user is already registered to the application and has already logged in.
Flow of events:

- The user searches for the cars, available or not, within a certain zone, that could be his/her position or a specific address.
- The user selects the cars she wants to monitor.

- Every time that a car's status changes, the user receives a notification on her phone with the new status of that car.

Exit conditions: The user stops monitoring a determined car or the car is picked up, so that the system will notify every monitorer and delete the list containing them.
Exceptions:

- The credentials furnished by the client are not correct. In this case the system doesn't redirect the user to the main page of the app but notifies him/her with an error message on the screen.
- The user doesn't pick up in time the car, so it has to pay a fee of 1 EUR and cannot cancel the reservation anymore.

UseCase ID: 4
Name: Temporary stop by
Actors: User (Mitch, scenario 3)
Entry conditions: The registered user unlocks the reserved car and turns it on.
Flow of events:

- The user turns on the car and drives.
- The user decides to stop the car for a certain amount of time, still wanting to benefit of the service.
- The user extracts the key from the car after having turned it off.
- The system checks if the car is parked in a safe area or not. If it is, the halt has no further cost, otherwise the system changes the service fee (lowering it, because the user isn't consuming battery).

Exit conditions: The user unlocks the car and turns it on, to continue driving or ends the service using the access code of the car.
Exceptions:

- Before stopping by, the user realizes that the car has not enough battery to continue the trip. He/she can find a power grid or end the service.

UseCase ID: 5
Name: Bonuses application
Actors: User (Lilith and her sisters, scenario 4; Mitch, scenario 6)
Entry conditions: The registered user reserved a car, picked it up and is about to end the service.
Flow of events:

- The user parks the car, exits and uses the access code to end the service.
- The car locks the doors and resets the access code
- The system checks if the conditions to gain a discount or overcharge are met:
  - For the majority of the ride there were more than two passengers, 10% discount.

  - When the service ended, the car had more than 50% of the battery filled, 20% discount.
  - When the service ended, the car had less than 20% of the battery filled, 30% overcharge.
  - When the service enede, the car was plugged into a power grid station, 30% discount.
  - When the service ended, the car was parked more than 3km from the nearest safe area, 30% overcharge.
- The system calculates the new cost of the ride, based on the applied discounts/overcharges.

Exit conditions: The system sends push notifications to the user, informing him/her of the total cost of the ride and the applied discounts/overcharges.

UseCase ID: 6
Name: Emergency handling
Actors: User (Mitch, scenario 5)
Entry conditions: The registered user reserved a car and picked it up.
Flow of events:

- The user turns on the car and drives.
- An accident occurs, that can be a sudden illness of the user, a collision or a malfunction of the car.
- The user presses the emergency button.
- The systems redirects the user call to an available operator, that can now talk with the user.
- After having asked information about the accident, the operator contacts other services, as ambulances, police or technicians to go at the accident location.

Exit conditions: The needed employees arrive on site (where the accident occurred).
Exceptions:

- There is no accident during the drive.
- The button is not working.

UseCase ID: 7
Name: Registration
Actors: User (Hilda, scenario 7)
Entry conditions: The user downloaded and installed the app.
Flow of events:

- The user starts the app.
- The user selects the registration option from the main menu of the app.
- The user inserts data in order to complete the registration:
  - Name and surname

  - Fiscal code (optional)
  - Phone number
  - Mail address
  - Preference over the notification channel (SMS, Mail or both)
  - Driver Licence number and postcode

Exit conditions: The user receives the password and with it he/she can log in to the personal area of the mobile app.
Exceptions:

- The user is already registered.

UseCase ID: 8
Name: Fault of a car
Actors: Car (car, scenario 8)
Entry conditions: The car is waiting for the next resevation/operation and it sensorial system detects an anomaly.
Flow of events:

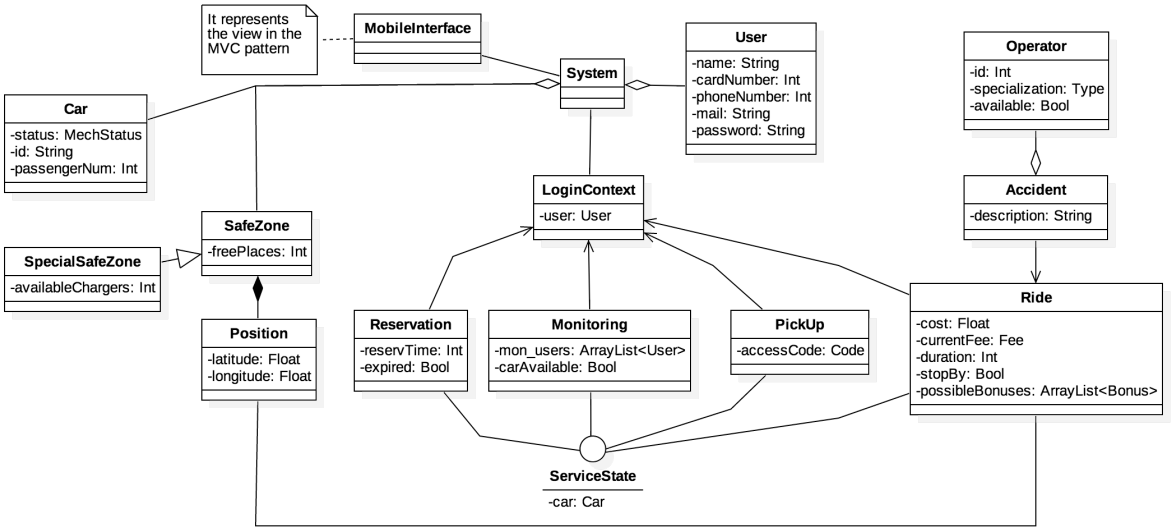
- The car system communicates to the system the problem, through the usual periodic updates
- The system tags the car as not available and updates the databese
- The system contacts an operator, providing him/her an access code to open the car.
- A technician reaches the broken car and fixes it.

Exit conditions: The technician communicates to the system that the car is now fixed.

4.2.3 Traceability matrix

Goal ID	Req ID	Use Case ID	Comments
G1	R1.1\R1.2	UC.1.1	The log in process is actually composed by the credential insertion and the system check of their validity.
G2	R2.1 to R2.8	UC.1.2, UC.2	The use case with ID 2 provides more detail about the cancellation of a reservation, while the use case with ID 1.2 describes more generically the reservation procedure.
G3	R3.3, R3.4, R4.2	UC.1.2	
G4	R4.1, R4.3, R4.4	UC.3	
G5	R5.1 to R5.3	UC.1.2	
G6	R6.2	UC.1.3	There is a possible case were the user can “cheat” and trick the system: he/she unlocks the car, starting the service but doesn’t use the car.
G9	R9.1, R9.2 and R9.4	UC.6	It’s the operator who decides what actions must be taken to handle the accident.
G10	R10.1 and R10.2	UC.4	An improvement of the sequence could be a notification sent by the system if the user is in stop for a long time (> 2 hours, as example).
G11	R11.4	UC.1	
G12	R12.1 to R12.4	UC.5	Actually, during the whole ride there is a periodic check of the conditions, to inform the user of the possible discounts that he/she will get if the condition persists. [Referring to requirements R7.X and goal G7]
G13	R13.1	UC.5	
G14	R14.1 and R14.2	UC.8	Once again, it’s repeated that the car system periodically informs the system of the car state of its component.

4.2.4 Class diagram



The shown class diagram is only an high representation of the future class model of the whole system. For now, just the most important objects and actors of the problem are represented, and their interaction. The system keeps trace of the registered users and the owned cars. Moreover, every choice of a user is saved. As example, let's consider a user that is making a reservation: the system shall know that that registered user, that has logged in the personal area, made a reservation on a certain car, which is available. Same considerations can be made for a monitoring on a car, a ride and so on, for all the possible states in which a user and the service are related.
Now, a brief introduction to possible patterns that can be associated to the class diagram.

- Obviously an MVC pattern can be usd to fulfill the Client-Server relationship, that will implement the User-System relationship (the MobileInterface class behaves as a view).
- Multiple Observers to manage the monitorings and the system update on a single car (change of availability tag).
- A Factory can be used for the code generation (based on the assumption that the structure differs for passwords, operational codes, access codes...)
- A State Pattern related to the service states for a single user, as described before in the paragraph introduction.

4.2.5 Sequence diagrams

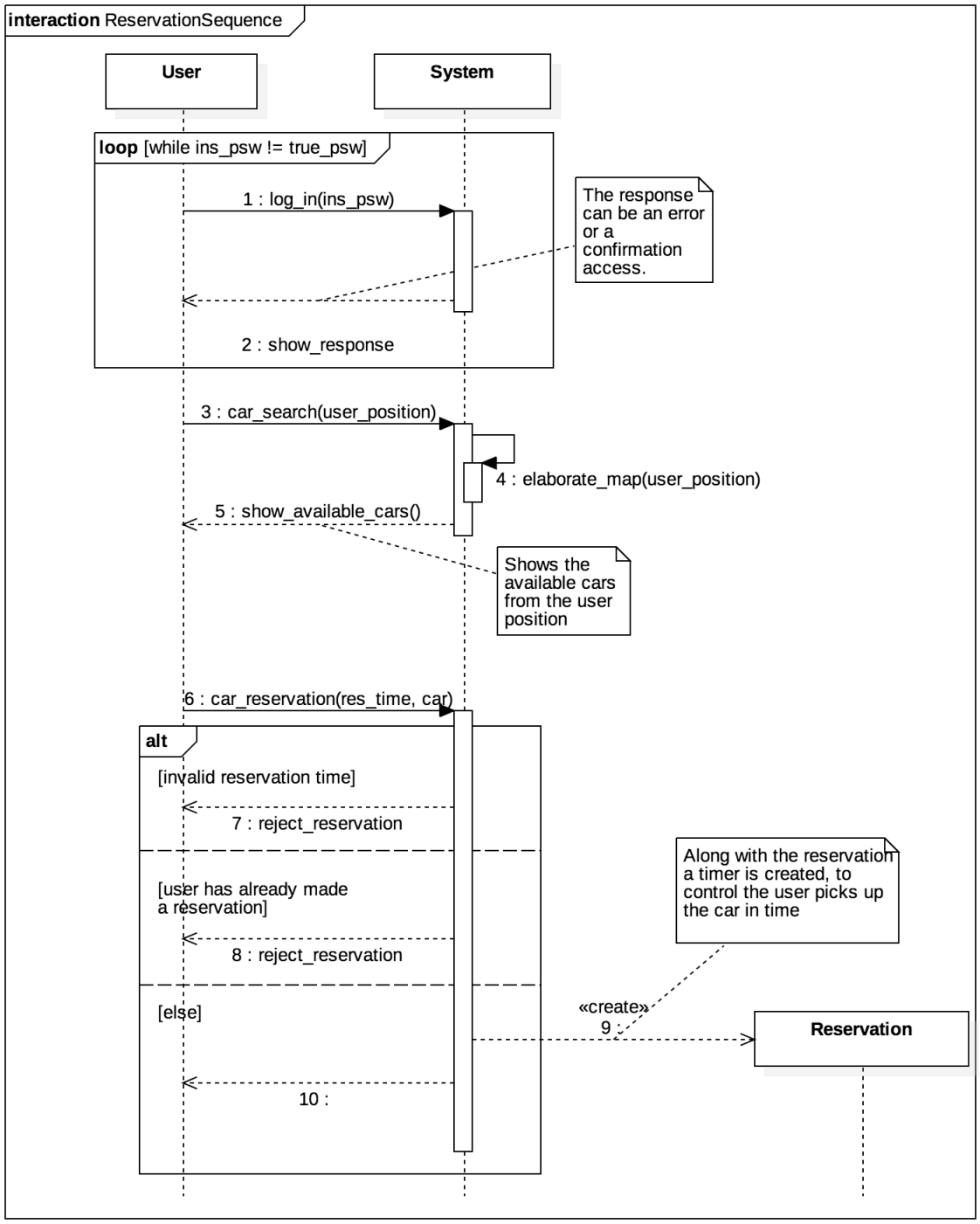


Figure 5: diagram representing the reservation process. After having logged in, from his/her app the user can search for a car. Once the system has elaborated and shown the list of cars, the user can reserve one. The reservation fails if the user has already made a reservation or inserts a reservation time longer than one hour.

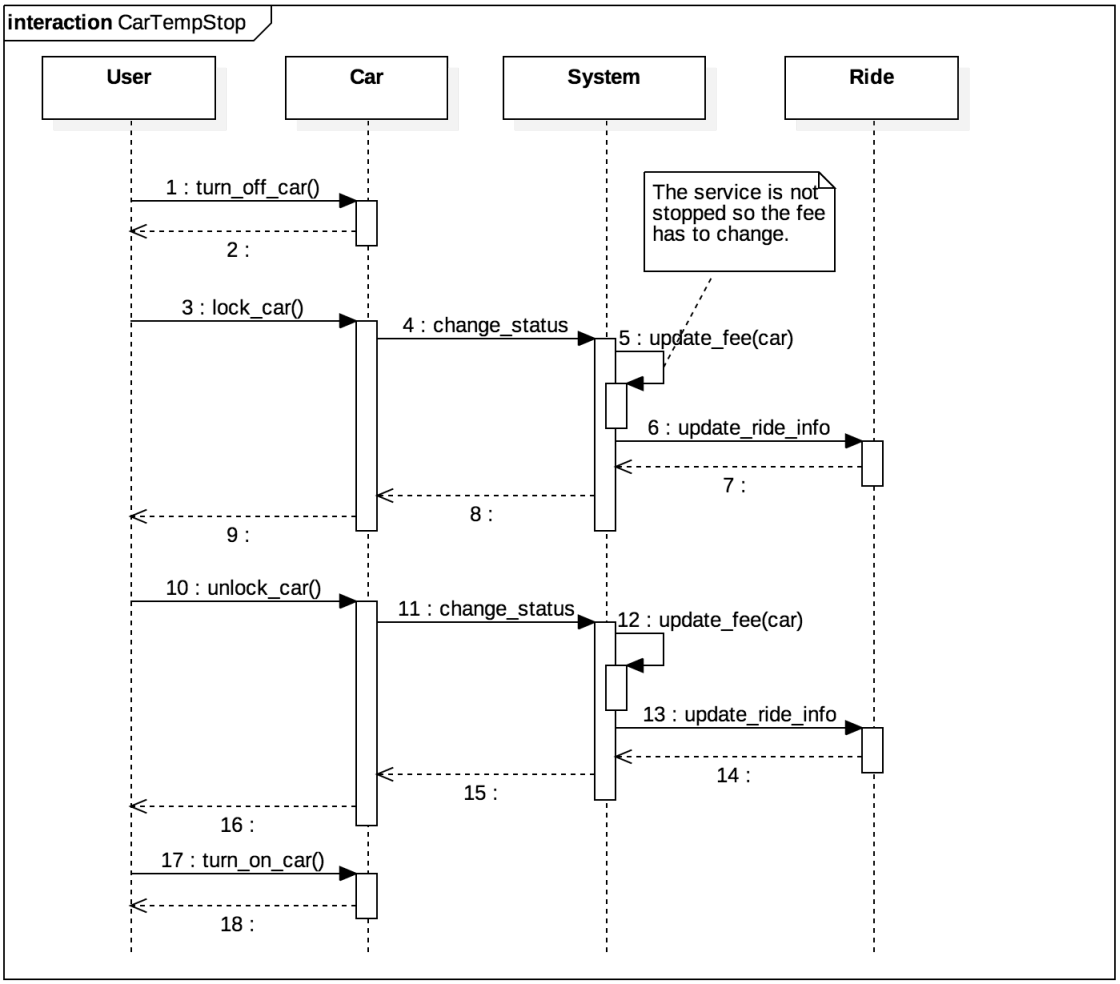


Figure 6: diagram representing the real case when the user interrupts temporarily the service. After having turned off the car and locked it, the car system communicates the change of its status to the system. The car has been turned off and locked, this means that the user is contining to benefit from the service. So, there will be only a change of fee, depending on the position of the car, and consequently the ride object associated to the user and the car. After a while the user will came back and a specular sequence of operations is made: the system changes the status of the car and updates the fee and the information of the ride. Lastly, the user turns on the car, and continues his/her drive.

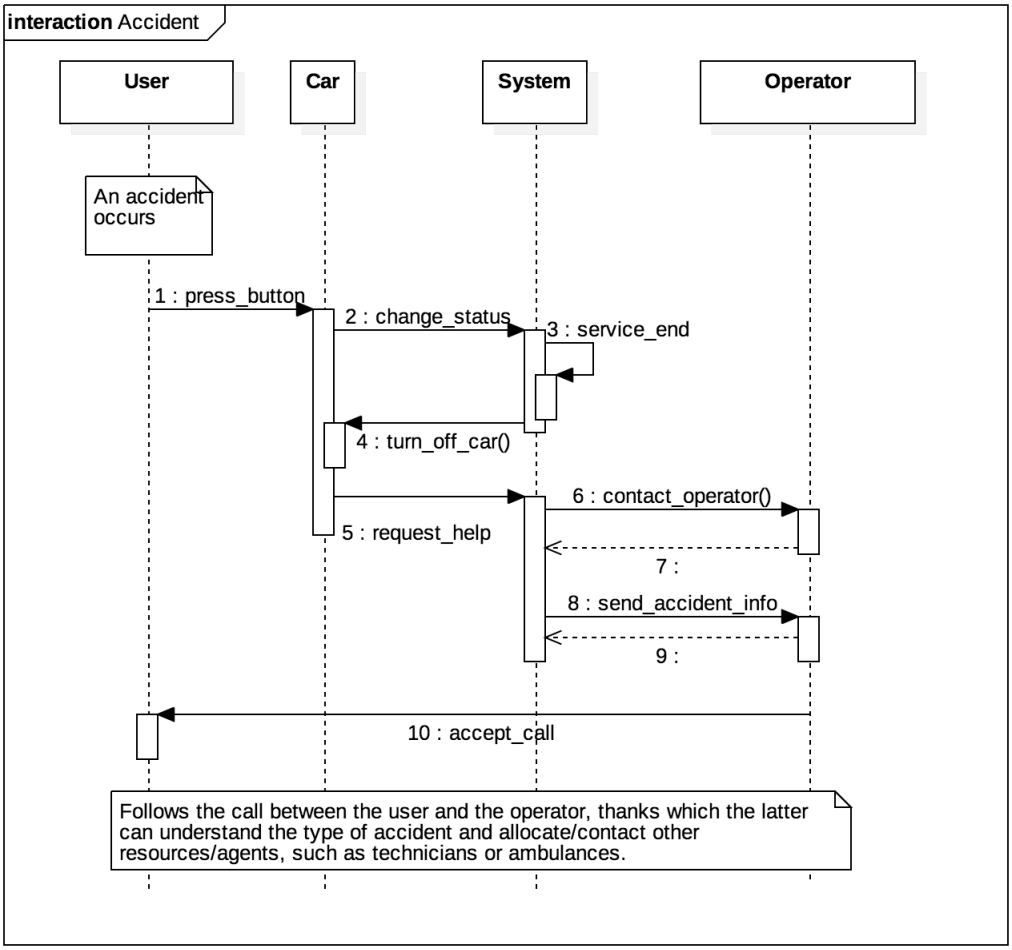
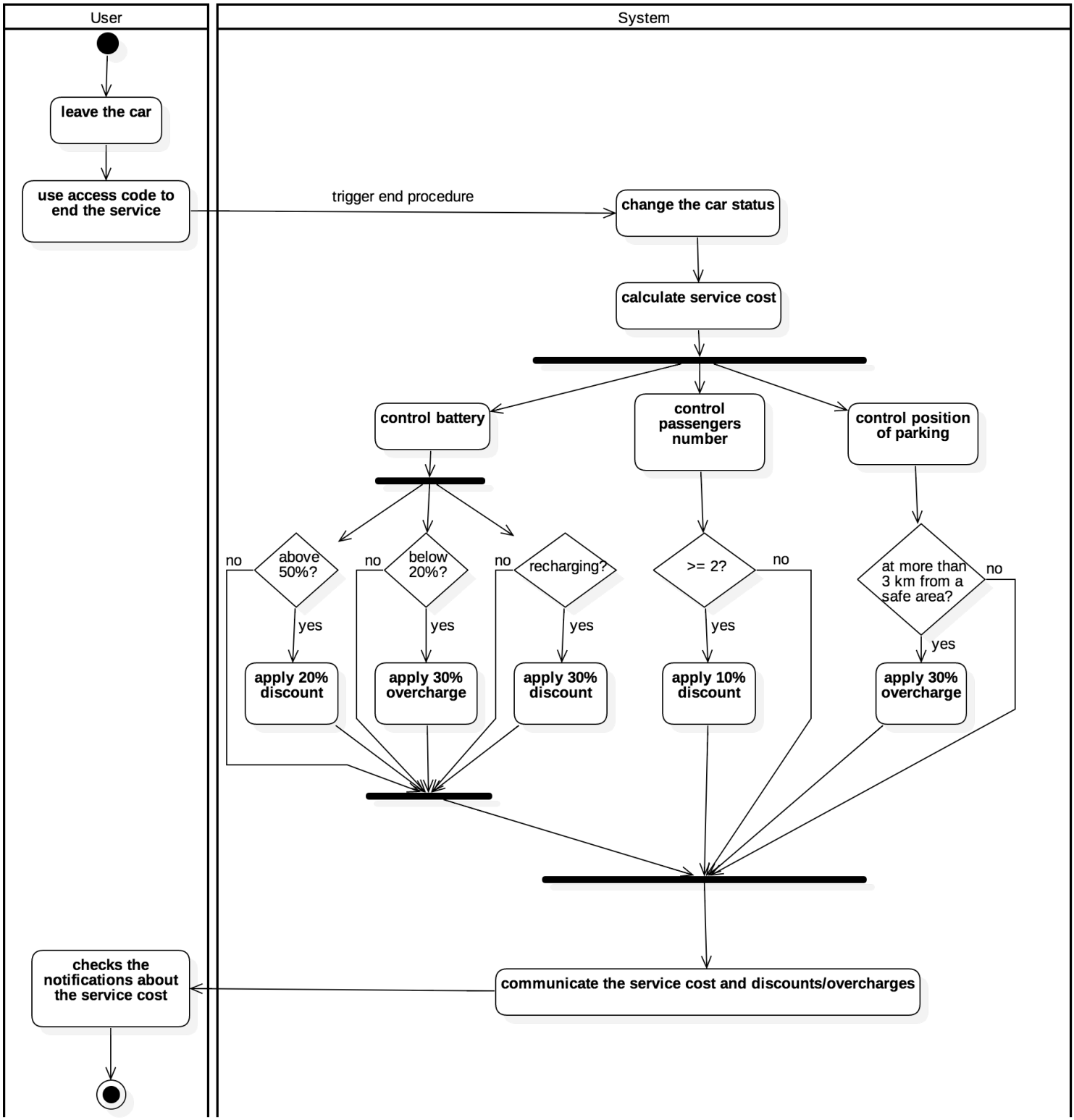


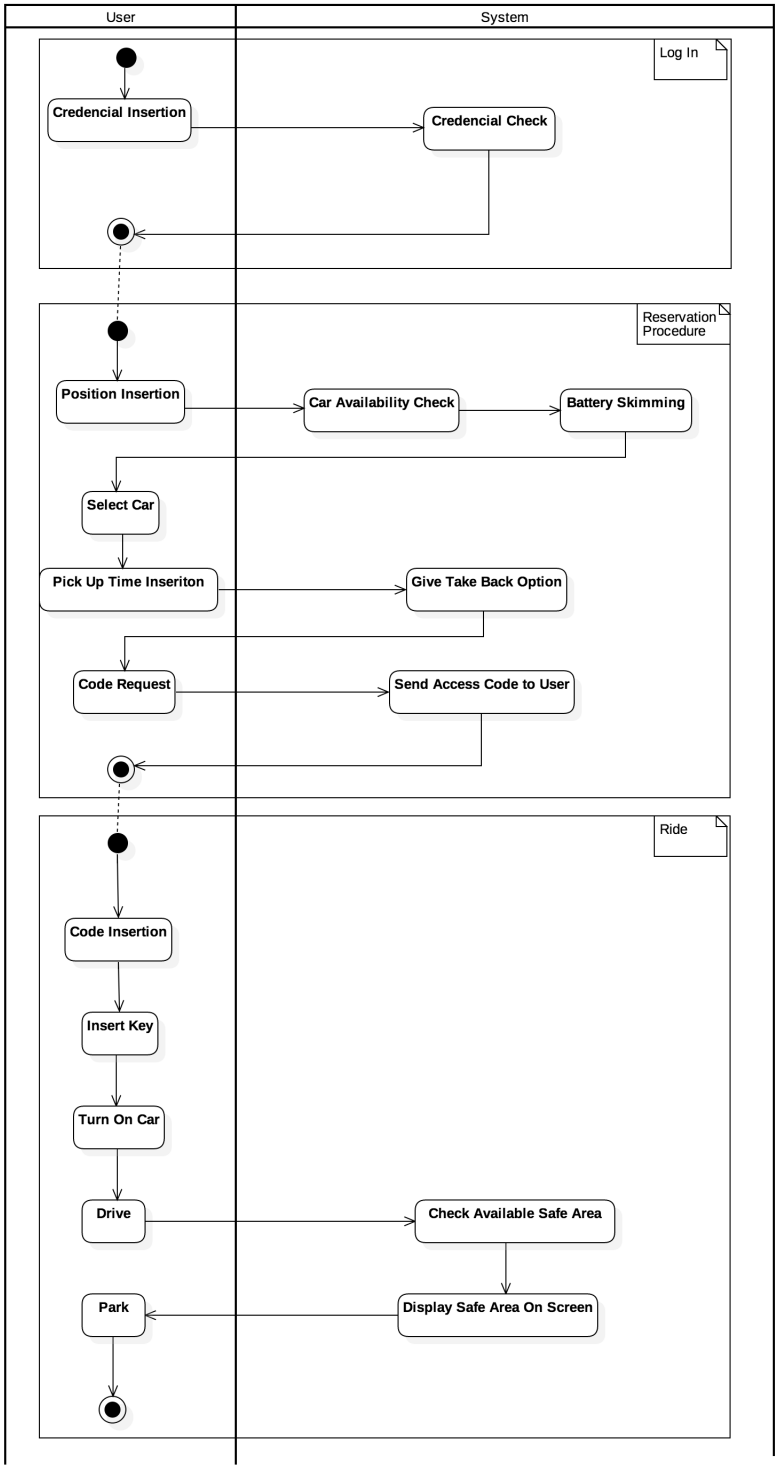
Figure 7: diagram representing the behavior of the whole system, in case of accident. When the user presses the emergency button on the car, this communicates to the system that there is a problem, and so a change of status of the car. The system turns off the car, to avoid further problems and possible user misbehavior. Then the systems contacts an operator, giving the necessary information about the accident: location, car status, and so on. The operator redirects the call directly to the user. After the conversation, which can even end abruptly (maybe the user suffered a sudden illness), the operator decides what to do and what resources have to be allocated and what other agents has to be called.

4.2.6 Activity diagrams



The diagram representing the system behavior, orientated towards a user-like vision of the problem,

when the service ends. After the user has left the car and ended the service using the access code to unlock the car before picking it up after the reservation, the system begins the payment and notification procedure. After having changed the car status, that involves a major number of activities - such as car locking, change of the car tag, check of the car condition with possible interaction with external operators, database update -, the system proceeds calculating the real cost of the ride, that is the standard cost (SC = RideTime*StdFee + PauseTime*SpFee) on which all the possible bonuses are applied. This is followed by the effective payment of the ride, handled by the PSP of the system. This is not included in the diagram because from the point of view of the user, the system just have to handle the payment, doesn't care on how it does. However, the user needs to know if the payment was successful, so the system sends him/her push notifications to inform him/her of the effective cost of the ride and the applied discounts and overcharges.
The diagram shown below represents the sequence of activities during a normal service, described in the use case model 1.


User
System
Log In
Credencial Insertion
Credencial Check
Reservation Procedure
Position Insertion
Car Availability Check
Battery Skimming
Select Car
Pick Up Time Inseriton
Give Take Back Option
Code Request
Send Access Code to User
Ride
Code Insertion
Insert Key
Turn On Car
Drive
Check Available Safe Area
Park
Display Safe Area On Screen

4.2.7 State diagram

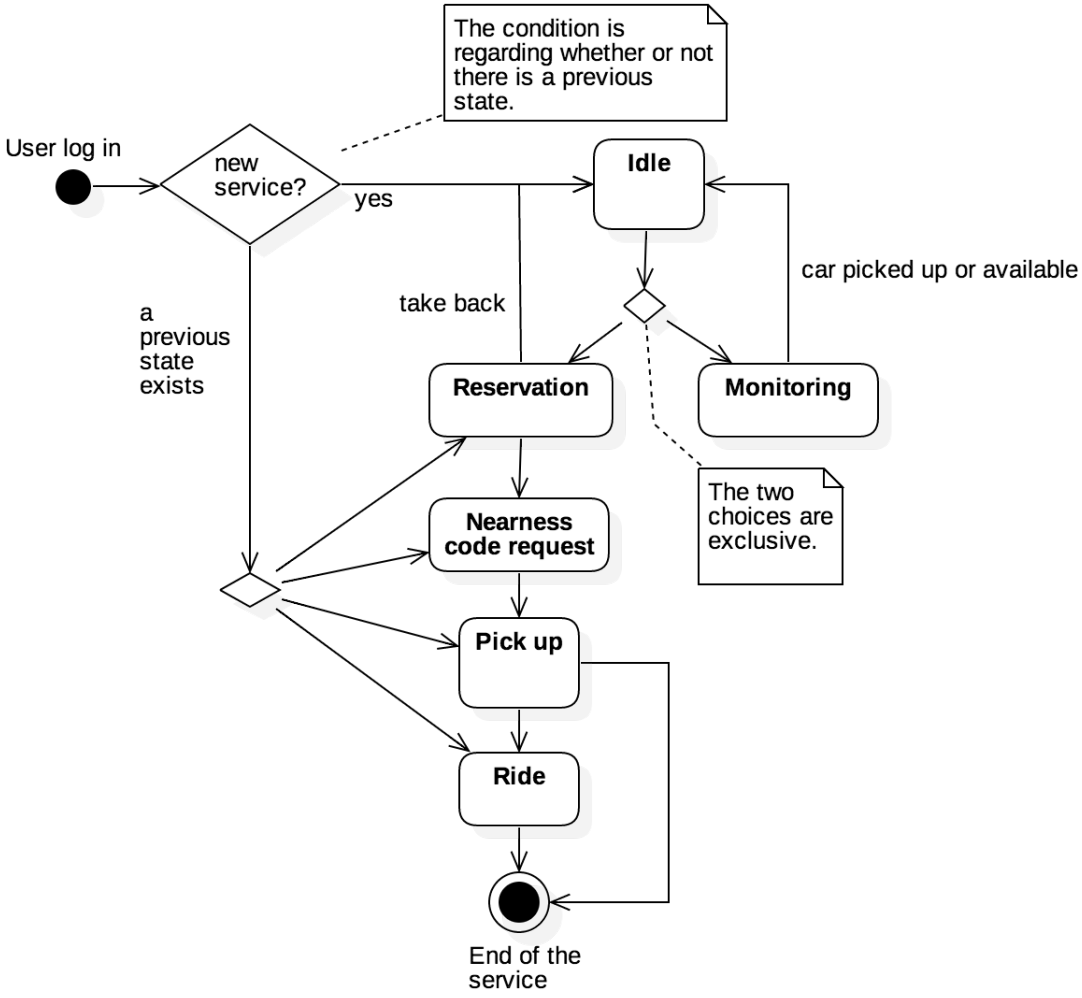


Figure 8: diagram representing the change of state of the service for a single user. In a real situation, the log in state of a user doesn't stay alive for the eternity, but after a fixed amount of time expires. Thus, the system needs to remember the last state of a user before the logout. As stated more times in this document, once logged in a user can reserve a car or monitor multiple cars. After the reservation follows the request of a code to access the car, the unlock and access to the car and the ride, that is a complex composed state (not shown in figure). Obviously, if the user doesn't pick up the car in time, there is no ride and the service ends.

5 Alloy

5.1 Script of the model

```
open util/ boolean

/* ——- User and car definition ———–*/
sig User {}
sig Car {
    position: Position,
    battery: Int
} {
    battery > 0 and battery < 15
}

// Two car cannot share the same position...they would be overlapping!
fact uniquePositions {
    all disj c1, c2: Car | c1.position != c2.position
}
/* ———————————————–*/

/* ———— System definition ————–*/
one sig System {
    users: set User,
    cars: set Car,
    reservations: set Reservation,
    monitorings: set Monitoring,
    codeRequests: set CodeRequest,
    rides: set Ride,
    safeAreas: set SafeArea
} {
    #cars >0
    #safeAreas > 0
}

// The system contains all of the service objects
fact systemContainsAll {
    all u: User | u in System.users
    all c: Car | c in System.cars
    all r: Reservation | r in System.reservations
    all m: Monitoring | m in System.monitorings
    all c: CodeRequest | c in System.codeRequests
    all r: Ride | r in System.rides
}
/* ———————————————–*/
```

```
/* ———– Service states ————*/
sig Reservation {
    user: one User,
    car: one Car
}

fact reservationUniqueness {
    // A user cannot make more than one reservation
    all disj r1, r2: Reservation | r1.user != r2.user
    // A car cannot be reserved by more than one user
    all disj r1, r2: Reservation | r1.car != r2.car
}

sig Monitoring {
    mon_users: set User,
    mon_car: Car
} {
    #mon_users > 0
}

// There are no multiple monitorings on the same car
fact oneMonitorForCar {
    all disj m1, m2: Monitoring | m1.mon_car != m2.mon_car
}

// Monitoring and reservations only on the permitted cars
fact permittedCarsMonitorReservation {
    all m: Monitoring | m.mon_car in System.cars
    all r: Reservation | r.car in System.cars
}

// A user cannot be reserving a car and monitoring others
fact monitorOrReservation {
    all r: Reservation, m: Monitoring | #(r.user & m.mon_users) = 0
}

// Asserts that not every user has made a reservation or is monitoring a car
assert someUserIdle {
    some u: User | no r: Reservation, m: Monitoring | r.user = u and u in m.mon_users
}
/* ———————————————–*/

/* —- Code request after a reservation —-*/
sig CodeRequest {
    user: one User,
```

```
    accessCode: one Code
}

sig Code{}

// All the codes to access a car are unique
fact codeUniqueness {
    no disj c1, c2: CodeRequest | c1.accessCode = c2.accessCode
}

/* A user can request for an access code only if he/she has already made a reservation. Moreover, a user
cannot make more than one reservation */
fact codeRequestFollowsReservation {
    all c: CodeRequest | one r: Reservation | c.user = r.user
    all disj c1, c2: CodeRequest | c1.user != c2.user
}

// All the codes of the world are generated only for a code request
fact noUnusedCode {
    #CodeRequest = #Code
}
/* ——————————————–*/

/* ———— Ride ————-*/
sig Ride {
    user: User,
    car: Car,
    cost: Int,
    currentFee: Fee,
    rideDuration: Int,
    possibleBonuses: set Bonus,
    carStopped: Bool
} {
    rideDuration > 0
}

// A user cannot be in more than one ride and so does a car
fact rideSingularity {
    all disj r1, r2: Ride | r1.user != r2.user and r1.car != r2.car
}

// When the user is riding a car, the reservation and monitorings users are cancelled
fact onceRidingReservationAndMonitoringExpire {
    all r: Ride, u: User, c: Car | r.user = u and r.car = c
    implies
```

```
    (no res: Reservation | res.user = u or res.car = c)
    and
    (no m: Monitoring | u in m.mon_users or m.car = c)
}

fact rideFeeConstraint {
    all r: Ride | (r.carStopped = True and r.car in SafeArea.parkedCars) implies r.currentFee = SafeAreaFee
    all r: Ride | (r.carStopped = True and r.car not in SafeArea.parkedCars) implies r.currentFee = StopFee
    all r: Ride | r.carStopped = False implies r.currentFee = StandardFee
}

// Even the code request should expire after the ride begins
assert rideImpliesNoCodeRequest {
    all r: Ride | no c: CodeRequest | r.user = c.user
}
/* ——————————————–*/

/* ——— Accident description ——–*/
sig Accident {
    unfortunateRide: one Ride,
    assignedOperators: set Operator
} {
    #assignedOperators > 0
}

sig Operator{}

// An accident can occur only during a ride
fact noAccidentWithoutRide {
    all a: Accident | a.unfortunateRide.currentFee = SafeAreaFee
}

/* ———– Fee description ———-*/
abstract sig Fee{
    fee: Int
}

one sig StopFee extends Fee{
    stopMultiplier: Int
}

one sig SafeAreaFee extends Fee {
    safeMultiplier: Int
}
```

```
one sig StandardFee extends Fee {
    standardMultiplier: Int
}

/* When stopping and leaving the car in a safe zone, the fee is zero. When leaving the car there is a small fee due to the continue use of the service. However, this fee is smaller than the one used during a ride because there is no battery consumption */
fact feeConstraints {
    StopFee.stopMultiplier > SafeAreaFee.safeMultiplier and
    StandardFee.standardMultiplier > StopFee.stopMultiplier and
    SafeAreaFee.safeMultiplier = 0
}
/* ——————————————–*/

/* —– Position and areas description —-*/
sig Position{
    latitude: Int,
    longitude: Int
}

sig SafeArea{
    parkedCars: set Car,
    availablePositions: set Position
}{
    #availablePositions > 0
}

sig SpecialSafearea extends SafeArea{
    availableChargers: Int
} {
    availableChargers > 0 && availableChargers < 10
}

// It is a third person/society that owns parking areas and can sell/buy it to/from our society
sig Stakeholder {
    owned_areas: set SafeArea
} {
    #owned_areas > 0
}

// Safe areas is a set contained within the system and the stakeholders
fact safeAreaDomain {
    no s: SafeArea | s in Stakeholder.owned_areas and s in System.safeAreas
    SafeArea = Stakeholder.owned_areas + System.safeAreas
}
```

```
// A position that is in a safe area cannot appear in another safe area
fact allSafeAreaHaveDisjPositions {
    all s1, s2: SafeArea | s1!=s2 implies #(s1.availablePositions & s2.availablePositions) = 0
}

// The positions of the parked cars in a safe area are some of the positions available in such area
fact parkedCarPositionInSafeAreaPositions {
    all s: SafeArea | s.parkedCars.position in s.availablePositions
}

// There isn't any car parked in the stakeholders areas
fact noServiceInStakeholderAreas {
    no c: Car | c.position in Stakeholder.owned_areas.availablePositions
}

// There should not be any position in a stakeholder area, given the previous assumptions
assert noCarInStakeholderSafeAreas {
    no c: Car | c in Stakeholder.owned_areas.parkedCars
}

// Predicates the addition of a safe area to the system, sold by some stakeholder
pred addSafeArea[sa: SafeArea, s,s': System, st,st': Stakeholder] {
    sa in st.owned_areas and sa not in s.safeAreas
    implies
    st'.owned_areas = st.owned_areas - sa and s'.safeAreas = s.safeAreas + sa
}

// Predicates the subtraction of a safe area from the system, bought by some stakeholder
pred delSafeArea[sa: SafeArea, s,s': System, st,st': Stakeholder] {
    sa not in st.owned_areas and sa in s.safeAreas
    implies
    st'.owned_areas = st.owned_areas + sa and s'.safeAreas = s.safeAreas - sa
}

// Predicates the parking of a car in a safe area
pred parkingCar[c: Car, s,s': SafeArea] {
    c not in SafeArea.parkedCars and c.position in s.availablePositions
    implies
    s'.parkedCars = s.parkedCars + c
}

// Predicates the leaving of a car from a parking zone
pred leavingCar[c: Car, s,s': SafeArea] {
    c in s.parkedCars and c.position in s.availablePositions
```

```
    implies
    s'.parkedCars = s.parkedCars - c and c not in SafeArea.parkedCars and c.position in s'.availablePositions
}
/* ——————————————————*/

/* —— Some bonuses description ——*/
abstract sig Bonus {}

one sig HighBattery extends Bonus {
    discount = 10
}

one sig LowBattery extends Bonus {
    overcharge = 15
}

/* If during a ride a car has more than 10 battery (50%) the user may get a discount if the condition persists. On the other side if during a ride a car has less than 5 battery (20%), the user may incur in a 30% overcharge if the condition persists */
fact possibleBonusesPerRide {
    all r: Ride | r.car.battery > 10 implies (HighBattery in r.possibleBonuses and #r.possibleBonuses = 1)
    all r: Ride | r.car.battery < 5 implies LowBattery in r.possibleBonuses and #r.possibleBonuses = 1 all r: Ride | (r.car.battery >= 5 and r.car.battery <= 10) implies #r.possibleBonuses = 0
}

/* A car should not have both bonuses related to the battery percentage */
assert noRideHasBothBatteryBonuses {
    no r: Ride | HighBattery in r.possibleBonuses and LowBattery in r.possibleBonuses
}
/* ——————————————————*/

/* Utilities constraints.
    - Sum of #Ride and #Reservation < #User
    - Sum of #Ride and #Reservation < #Car
    - #CodeRequest < #Reservation */
fact {
    #User = 6
    #Reservation = 2
    #CodeRequest = 1
    #Monitoring < 3 && #Monitoring > 0
    #Ride = 2
    #SafeArea = 2
    #Operator < 3
    #Accident = 1
```

```
    #Stakeholder = 1
    some r: Reservation | r.car in SafeArea.parkedCars
    some p: Position | p not in SafeArea.availablePositions
    some r: Ride | r.car.position in System.safeAreas.availablePositions
    some r: Ride | LowBattery in r.possibleBonuses
    some c: Car | c.battery < 5 some c: Car | c.battery > 10
    #System.safeAreas.parkedCars = 2
}

pred show[] {}

//Utility assertion assert
someResOutArea {
    some r: Reservation | r.car.position not in SafeArea.availablePositions
}

// World controls and checks
check someResOutArea
check rideImpliesNoCodeRequest
check someUserIdle
check noRideHasBothBatteryBonuses
check noCarInStakeholderSafeAreas
run addSafeArea for 7 but 5 Int
run delSafeArea for 7 but 5 Int
run parkingCar for 7 but 5 Int
run leavingCar for 7 but 5 Int
run show for 7 but 5 Int
```

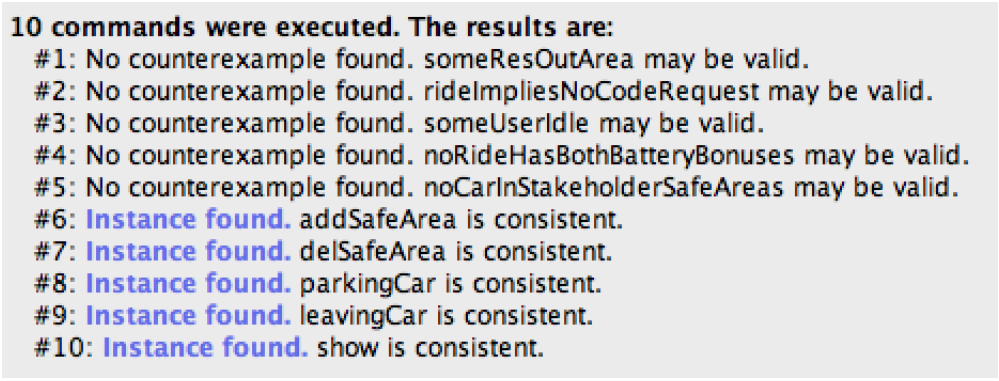
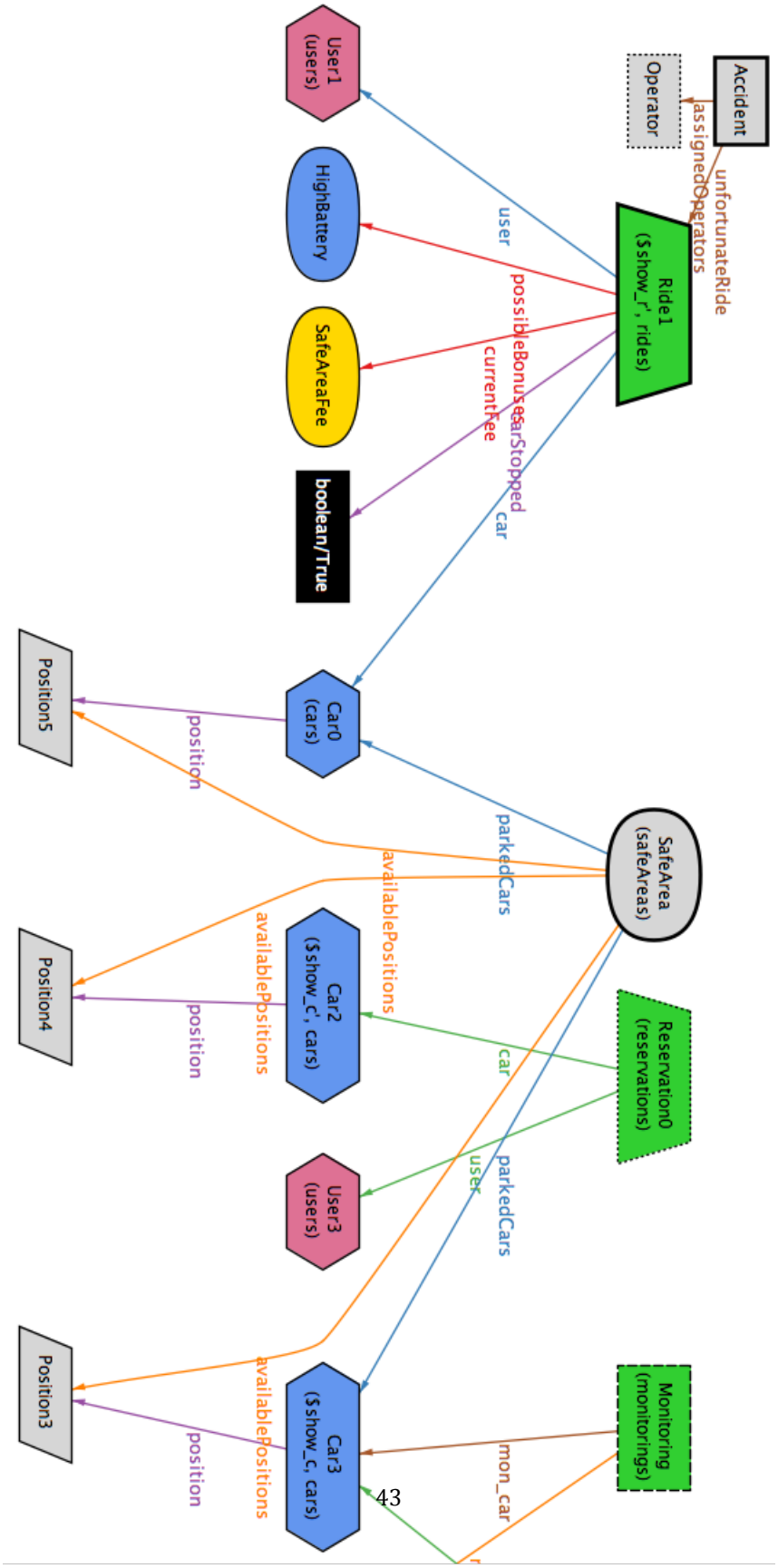


Figure 9: alloy analisys over the model

5.2 Generated world



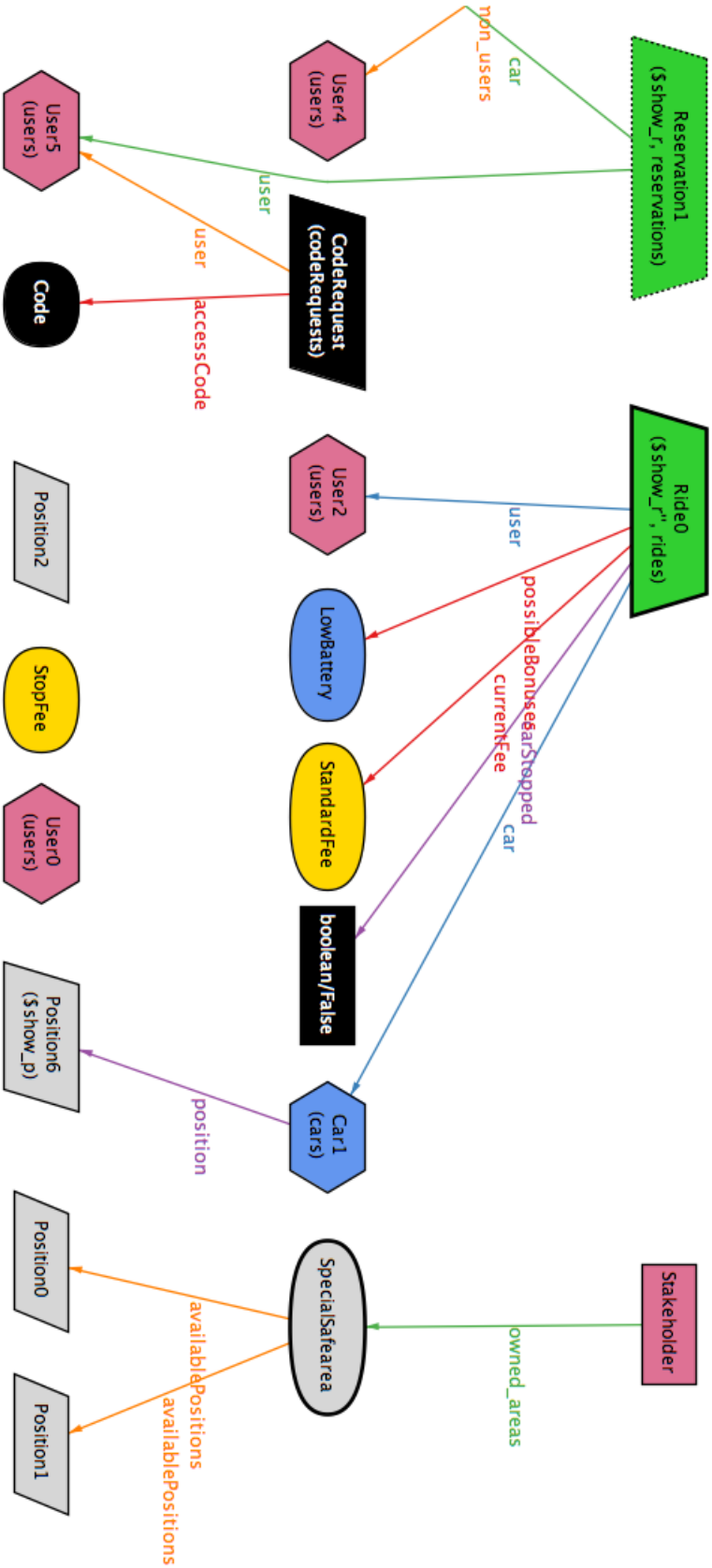


Figure 10: generated world

Here follows a brief description of the generated world and what is represented in it. The first thing to say is that some elements are hidden, because otherwise the dimension and complexity of the view would have been difficult to interpret and simplify. The hidden elements are the System, that here only serves as a container of other elements, such as cars, users, reservations and so on, the Integers used to describe positions, batteries and available places in safe areas and Strings.
Let's analyze, from left to ride, the captured state of this generated world:

- User1 had an accident on Car0 that is parked in SafeArea, so an operator is assigned to survey the car.
  - Scenario 1: the car doesn't turn on and the user is communicating the problem
  - Scenario 2: the user suddenly feels an illness and requests help
  - Scenario 3: when parking the user crashes towards a pillar or another car
- User3 made a reservation on Car2 parked in SafeArea
- User4 is monitoring the reserved Car3 that in SafeArea
- User5 reserved Car3 and being in proximity of the car, requested the access code to unlock it.
- User2 is driving Car1 (in Ride0) and is passing on Position6
- User0 isn't doing anything, probably searching for a car or being idle
- An external agent, Stakeholder, owns an area that in future could be bought by the system as a safe area

6 Document further information

6.1 References

In writing the document, the following sites/materials have been consulted:

- Assignments AA 2016-2017.pdf
- http://www.monperrus.net/martin/alloy-quick-ref.pdf, a document containing Alloy basics
- http://alloy.mit.edu/alloy/, for a deeper documentation about Alloy
- https://www.ics.uci.edu/~alspaugh/cls/shr/alloy.html, for informations about Alloy relations and operators
- https://www.car2go.com/US/en/austin/, to analyse the similarities with a service much similar to the problem we studied
- https://www.smashingmagazine.com/2011/04/taking-credit-card-payments-online-whats-involved/, which contains a description on how a site can implement an online payment (mostly read because of possible future implementations)
- RASD sample from Oct. 20 lecture.pdf
- Requirements Analysis Document Template.pdf
- RAD-V1.10.docx
- guidelines_v211.pdf
- IEEE standard on requirement engineering.pdf

6.2 Used tools

In creating the RASD document, the following tools have been used:

- Github, for version controller
- LyX, to write the document and converting in pdf
- StarUML, for UML diagrams and image export
- Alloy Analizer 4.2, to prove the consistency of the model and view some scenarios
- https://ninjamock.com, for interface mockups
- Pencil, for the system architecture sample
- Hunspell, for the spell check of the document

6.3 ChangeLog

[29/10/2016] [**Version 1.0**] :: Initial description of goals, domain assumptions and requirements.
[30/10/2016] [**Version 1.1**] :: Added first part of the glossary and made some language adjustments.
[30/10/2016] [**Version 1.2**] :: Completed the glossary.
[30/10/2016] [**Version 1.3**] :: Added an introduction to the document. Updated the indexing of requirements and goals.
[30/10/2016] [**Version 2.0**] :: Added constraints and corrections to the Requirement section.
[31/10/2016] [**Version 2.1**] :: Changed some goals. It is at this point supported a real world situation where a user may stop the car to do something but still need the service to continue. Other corrections about requirements consistency were made.
[31/10/2016] [**Version 2.2**] :: Improved the glossary and added Text assumptions and some Non functional requirements, such as Usability and Performance requirements.
[01/11/2016] [**Version 3.0**] :: Added other Text assumptions, system architecture diagram, a mockup of the mobile app and multiple scenarios. Major change regarding the UML section and in particular the Use case diagram. Started the use case description considering the use case diagram and the previously introduced scenarios.
[01/11/2016] [**Version 3.1**] :: Added some use cases description and mockup of the registration page of the mobile app.
[02/11/2016] [**Version 3.2**] :: Added use case descriptions and minor corrections.
[03/11/2016] [**Version 3.3**] :: Added first version of the class diagram.
[05/11/2016] [**Version 3.4**] :: Added sequence diagrams and an activity diagram. Also added the UML project document to the folder ("RASD.mdj").
[06/11/2016] [**Version 3.5**] :: Added a state diagram, describing the service form a user point of view, but still related to the class diagram and thus to the software implementation. Added also a description for all the UML diagrams, Class diagram excluded.
[08/11/2016] [**Version 4.0**] :: Added alloy model.
[09/11/2016] [**Version 4.1**] :: Minor changes in alloy model. Addition of one generated world, based on such model (broken in two pages). Addition of a scenario and a use case description (about [G14], considered until now only in the requirements).
[10/11/2016] [**Version 4.2**] :: Improved description of the class diagram. Started the traceability table and the refining process of use cases (bigger use cases now will be composed of multiple sub use cases).
[10/11/2016] [**Version 4.3**] :: Grammar correction and minor changes of the glossary.
[10/11/2016] [**Version 4.4**] :: Added the final sections of the documents (references, hours of work).
[11/11/2016] [**Version 4.5**] :: Revision of the document and minor corrections over grammar and coherence.
[27/11/2016] [**Version 4.6**] :: Added the ChangeLog and minor changes due to the DD writing (such as the change of the server framework in the Proposed system section, 2.3.

6.4 Hours and description of work

Matteo Frosi

18-10-2016: 1.30 hours (meeting to discuss about an informal analysis of the problem)
20-10-2016: 0.45 hours (meeting to discuss about an informal analysis of the problem)
24-10-2016: 2.00 hours (informal analysis of project description writing)
25-10-2016: 0.30 hours (informal analysis of project document revision + meeting about RASD defini-

tion)
29-10-2016: 7.30 hours (description of goals, domain assumption and requirements)
30-10-2016: 8.00 hours (document shaping, glossary and introduction additions and further analysis of requirements)
31-10-2016: 7.00 hours (improvement of various document sections, addition of Non functional requirements)
1-11-2016: 8.00 hours (scenarios, interface mockups and UML diagrams additions)
2-11-2016: 2.00 hours (addition of some use case descriptions and minor corrections)
3-11-2016: 2:00 hours (addition of the first version of the class diagram)
4-11-2016: 2.30 hours (addition of state diagram and diagrams descriptions)
from 5 to 8-11-2016: 16.00 hours (addition of alloy model)
9-11-2016: 3.00 hours (minor changes in alloy model, addition of the world generated with alloy model and of a scenario about [G14] requirements)
10-11-2016: 2.00 hours (addition of document final sections and descriptions)
11-11-2016: 2.00 hours (review of the document + traceability matrix)
Total work hours: 64.45 hours

Luca Costa

18-10-2016: 1.30 hours (meeting to discuss about an informal analysis of the problem)
20-10-2016: 0.45 hours (meeting to discuss about an informal analysis of the problem)
24-10-2016: 1.00 hours (informal analysis of project description revision)
25-10-2016: 0.30 hours (informal analysis of project document revision + meeting about RASD definition)
29-10-2016: 2.30 hours (description of goals and requirements)
1-12-2016: 2.30 hours (discussion about requirement minor changes)
2-11-2016: 1.30 + 3.30 hours (grammar correction + scenarios analysis
3-11-2016: 2.30 hours (use case description)
4-11-2016: 4.00 hours (class diagram building)
5-11-2016: 2.30 hours (diagrams description)
from 6 to 9-11-2016: 3.00 hours (alloy check and coherence evaluation)
11-11-2016: 2.15 hours (addition of an activity diagram describing use case 1)
Total work hours: 25.30 hours + periodic revision of the document