Politecnico di Milano

AA 2016-2017

Computer Science and Engineering

Software Engineering 2 Project

POWERENJOY

RASD

Requirement Analysis and Specification Document

version 1.0 - 11/13/2016

Bassetto Riccardo - 810597
Belloni Massimo - 873445

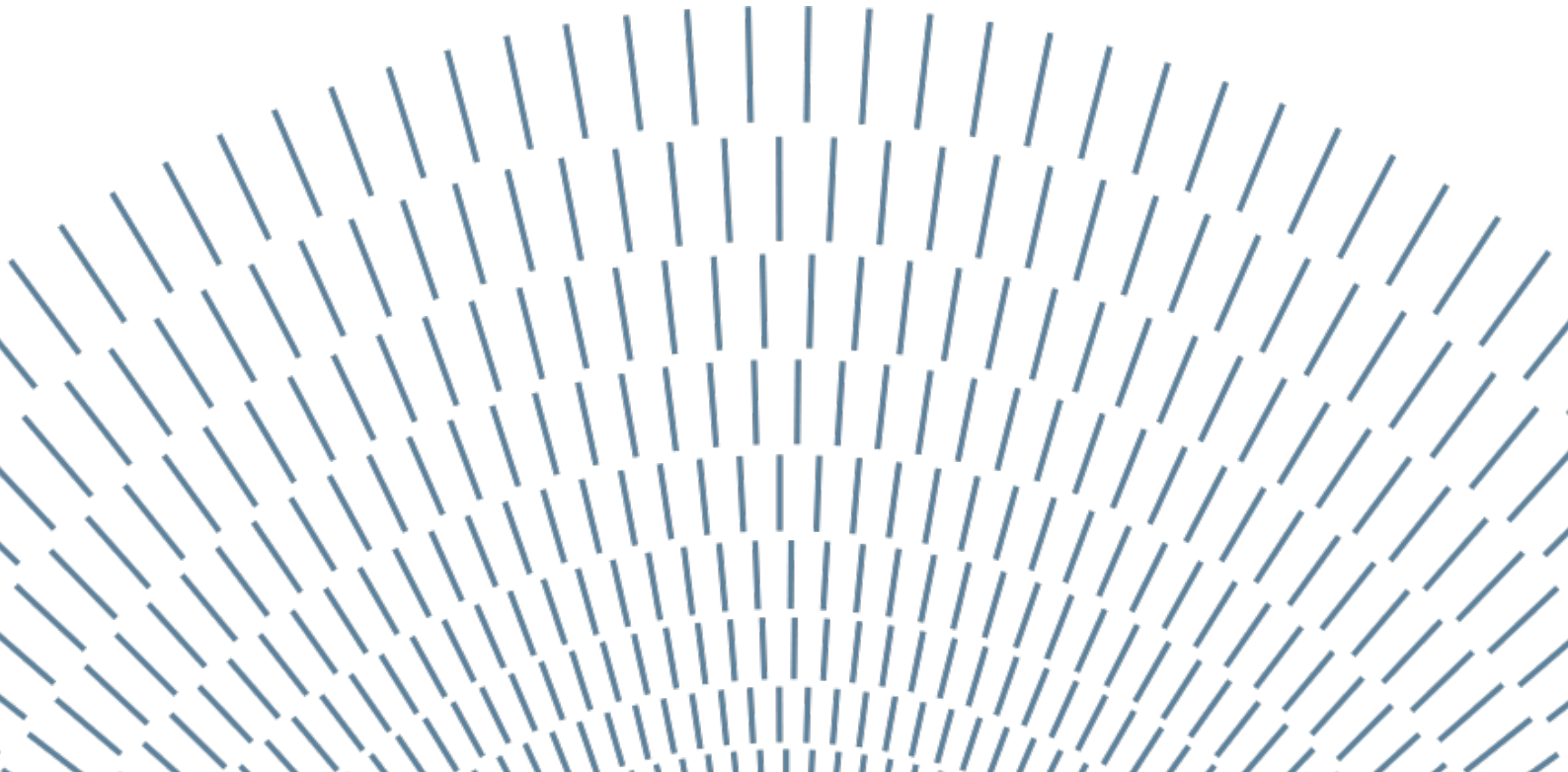


Table of Contents

1. Introduction

1.1. Purpose

This document represents the Requirement Analysis and Specification Document (RASD). Goals of this document are to completely describe the system in terms of functional and non-functional requirements, analyze the real needs of the customer in order to model the system, show the constraints and the limit of the software and indicate the typical use cases that will occur after the release. This document is addressed to the developers who have to implement the requirements and could be used as a contractual basis.

1.2. Scope

1.2.1. Description of the given problem

PowerEnJoy is car-sharing service-to-be that exclusively employs electric cars. The given problem is to design and develop a digital management system for it. Of course, the system should provide the functionalities normally provided by any car-sharing service, starting from car reservation and pickup (including a fee to be paid for "unused" reservations) and going through the whole ride management process. Cars will have to be parked in safe areas in order to terminate a ride and a very detailed discount grid will have to be implemented. A green and eco-sustainable approach to the car-sharing market is the core of the PowerEnJoy business and a lot of details about the actual implementation have been defined accordingly: if a car is parked and plugged-in in a special parking area, for instance, a discount as to be applied, or, in addition, if at least two passengers share the ride, another discount should be applied. On the contrary, bad behaviors have to be "punished": if a car is left with low battery or too far from a power grid, the user will see the price increasing. Ultimately, the system will have to be easy-to-use, reliable and highly scalable to fit perfectly with the mutable context in which it will end to be used.

1.2.2. Current System

PowerEnJoy, as already said, is a new service that is entering the market for the first time. This is a very lucky situation, allowing us to design the whole platform from scratch. There is no any current system to integrate with and to consider in our design process.

1.2.3. Goals

- [G1] Allow a Visitor to become registered User after providing credentials and payment information.
- [G2] Allow an User to find locations of available cars.

  [G2.1] Allow an User to find locations of available cars within a certain distance from his/her current location.

  [G2.2] Allow an User to find locations of available cars within a certain distance from a specified address.
- [G3] Allow an User to reserve a car.
- [G4] Allow an User to unlock the car that he/she has registered.
- [G5] Allow an User to complete and pay for a full ride with the car that he/she has registered.

  [G5.1] Allow an User to see the ride's informations through a screen on the car.
- [G6] Allow an User to get discounts on prices if he/she behaves well and respects certain conditions.

  [G6.1] If the system detects the User took at least two other passengers onto the car applies a discount of 10% on the ride.

  [G6.2] If a car is left with no more than 50% of the battery empty (at least half charged), the system applies a discount of 20% on the ride.

  [G6.3] If a car is left at special parking areas and the User takes care of plugging the car into the power grid, the system applies a discount of 30% on the ride.

  [G6.4] If a car is left at more than 3KM from the nearest power grid station, the system charges 30% more on the ride.

  [G6.5] If a car is left with more than 80% of the battery empty (less than 20% of battery charged), the system charges 30% more on the ride.
- [G7] Allow a System Manager to do operations on the system for updating and maintenance.

  [G7.1] Allow a System Manager to add a car to the system.

  [G7.2] Allow a System Manager to add or edit a safe area/special parking area.

  [G7.3] Allow a System Manager to edit fields as amount of money per minute, period of time for which a car remains opened, minimum distance of the User from the car to be considered "nearby", amount of time to plug the car into the power grid after leaving the car.

  [G7.4] Allow a System Manager to see all the cars registered into the system, or search for a specific one using its license plate.

1.3. Definitions, acronyms and abbreviations

1.3.1. Definitions

- *Reservation:* the intent of an user to pickup a car within an hour.
- *Safe area:* an area in which a car may be parked to end a ride.
- *Special parking area:* a safe area equipped with power grids.
- *Ride:* the actual service provided by PowerEnJoy. It begins when the car's engine ignites and ends when the car is parked in a safe area (not in "pit-stop mode").

- *Pit-stop button:* a button placed in the touchscreen interface that allows an user to put the ride in "pit-stop mode".
- *Pit-stop mode:* special scenario in which a car is parked in a safe area but the ride isn't terminated and the user is still charged. In this mode the reservation made obviously doesn't expire and the car isn't available.
- *Power grid timer:* the maximum amount of time that an User has to plug the car into a power grid before the ride ends and the payment is actually charged to him/her.
- *Blocked user:* an user that hasn't paid for the last ride and so won't be able to use the system services until the situation will be fixed.

1.3.2. Acronyms

- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface

1.3.3 Abbreviations

- [Gn]: n-goal.
- [Dn]: n-domain assumption.
- [Rn]: n-functional requirement.

1.4. References

- Specification Document: “Assignments AA 2016-2017.pdf”.
- GPS Performances: “http://www.gps.gov/systems/gps/performance/accuracy/".
- Alloy Dynamic Model example: “http://homepage.cs.uiowa.edu/~tinelli/classes/181/Spring10/Notes/09-dynamic-models.pdf"
- IEEE Std 830-1993 - IEEE Guide to Software Requirements Specifications.
- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications.

1.5. Overview

This RASD is composed by 6 parts and an appendix:

1. In the first part an introduction to the problem is given listing all the identified goals and providing some base informations in order to better understand the other sections of the document.
2. The second part consists of an overall description of the system in which its boundaries are identified, and the actors involved in the system’s usage lifecycle are listed. The boundaries are given providing all the necessary assumptions: both the ones required in order to better understand the customer’s specifications given and the ones that will hold into the system and now on considered as true.
3. The third part is composed by the specific requirements identified, both functional and non functional.

4. In the fourth part a list of eight scenarios is provided; each of them describes a particular situation with the system might have to cope with.
5. The fifth part is entirely composed by the UML diagrams that model the system in details.
6. Sixth part is embodied with the Alloy model of the system and includes all the relevant details; a proof of consistency and an example of the generated world are also provided.
7. The last part is accessory and contains a list of the tools used to redact this document and its contents, and a detailed report of the hours spent to do so.

2. Overall Description

2.1. Product perspective

The system will be developed from scratch and, in a way that will be better described in the next paragraphs will use an external Payment System. This will be needed both to guarantee the security of the transactions in a very dangerous and complicated environment, and to simplify the overall implementation, decoupling payments management from our system.

2.2. Product functions

In the direction of defining unambiguous boundaries for the system in development, a specific list of the functions that the product will/won't offer is provided and precisely explained hereunder. Of course all the goals identified in the previous part will be offered as functions of the system in development; during their identification we coped with some on-the-border issues that won't be directly handled by the system.

2.2.1. Payment management

As already said an external Payment System will be used; it will guarantee high level of security and reliability but obviously will not ensure that all the payments will succeed (eg. expired credit cards, failed transactions, empty prepaid credit cards). These issues will be correctly notified and handled by our system but will have to be managed and solved outside of it. Defaulting users will be marked as "blocked" and won't be able to use the service until the settling of the faulty payment; the necessary procedures and operations needed in order to achieve this will have to be decided autonomously by the company (even under a legal point of view).

2.2.2. Car maintenance

All the cars used will obviously need, sooner or later, of some maintenance. The system in development won't support this in any way apart from the possibility to mark a car as "unavailable" for an indefinite amount of time.

2.2.3. Others

The concepts of "blocked" user and "unavailable" car are left on purpose a little bit fuzzy. Possibile future (and yet unknown!) issues or temporary functionalities may be handled using (or better defining) these two concepts.

2.3. User characteristics

2.3.1. Actors

- *Visitor:* a person using PowerEnJoy (through the mobile app or the website) without being registered. The only thing he/she can do is proceeding with registration.
- *Registered User / User:* a person passed through a successful registration process and now able to use all the PowerEnJoy services. He/she can login to the system and, after that, use all the platform's functionalities.
- *System Manager:* an employee of PowerEnJoy able to maintain and update the system. Registration for this kind of users is not possible and they have to be added directly during system's installation process.
- *Car:* a car involved in a ride that communicates with the system retrieving informations and updating it back with its position and ride events.
- *Passenger:* a person traveling with an User able only to trigger seat's sensors. He/she may or may not be a PowerEnJoy User, this is not important as long as he/she maintains this role.

2.4. Constraints

2.4.1. Regulatory policies

The system will ask for users' payment informations and obviously, in addition to store them safely, will use them only for fees and rides payments.
Moreover, the system will have to ask for users' permission in order to retrieve and use their positions without (at least in a first implementation) storing them. Telephone numbers and email addresses won't be used for commercial uses.

2.4.2. Hardware limitations

- Mobile App
  - iOS or Android smartphone
  - 2G/3G/4G connection
  - GPS
- Web App
  - Modern browser able to retrieve user's location

2.4.3 Interfaces to other applications

In the first release no public interfaces will be opened and third party services won’t be able to interoperate with PowerEnJoy.

2.5. Assumptions and dependencies

In the specification document presented by the customer we found some points that lacked in precision and would have led to some ambiguities. In order to better clarify those situations we decided to introduce the following assumptions.

2.5.1. Text assumptions

- Credentials that a visitor has to provide to become a registered user are: name, surname, address, email, telephone number and username. To complete the registration he/she has also to provide his/her driving license code.
- The password is sent as an SMS to the telephone number provided during the registration process in order to guarantee more security.
- In order to access to the system a system manager has to provide the username and password associated to him/her.
- A car is "available" if it isn't reserved and it has at least 20% of battery left.
- Car left with less than 20% of battery left will be recharged by a PowerEnjoy system manager.
- When a car is remotely opened remains opened for a time pre-defined by the management system.
- The amount of money per minute that has to be paid by an user is decided by the management system and acknowledged by the user when he/she makes the reservation.
- The user actually pays for the ride only when the ride is terminated. The amount of money that has to be paid by him/her obviously depends on the ride's length.
- We assume that every car has a button (from now on "pit-stop" button) that, if pushed, indicates the intention of the user to leave the car without terminating the ride.
- The required functionalities provided are unclear on the concept of "parking": a car is parked if and only if it is stopped with the engine shut off. As it is precisely specified, if an user parks the car in a safe area, the system stops charging him. Our assumption is that if an user parks in a safe area but he/she pushes the "pit-stop button" expresses the intention to not terminate the ride (see Scenario x at page y for further details). The car will be obviously closed when the user exits and can be opened by him/her in the same way as already done in the first place.
- Special parking areas are safe areas where cars can be recharged; so obviously every special parking area is a safe-area.
- Cars may be added to the system by a system manager. Since a car is added it immediately becomes available.
- Safe areas may be added or edited by a system manager.

2.5.2. Domain assumptions

- [D1] Payment information correctness may be verified using an external service that will be in charge of all the payment processes.
- [D2] If the system charges a registered user, and for some reasons depending on the external service used, the payment cannot be completed the user is tagged as "blocked" and the payment will be completed outside the S2B.
- [D3] The username must be unique.
- [D4] When the system sends a password to a new registered user this will be surely received by him/her.
- [D5] Users and cars locations are retrieved by GPS.
- [D6] Cars are equipped with sensors (GPS, sensors into the seats) and communicate over mobile network.
- [D7] Safe areas do not overlap.
- [D8] When the system shows a car in a certain position it means that it's actually there.
- [D9] The set of safe areas is never empty.
- [D10] The set of special parking areas is never empty.
- [D11] Every special parking area is a safe area.
- [D12] A safe area is a special parking area if and only if it is equipped with at least one charging station.
- [D13] A safe area, considered as a circle, is described by a coordinate and the maximum acceptable distance from the coordinate given.
- [D14] An user has a pre-defined amount of time to plug the car into the power grid after he/she has leaved the car in order to obtain the discount.
- [D15] As soon as the system stops charging an user for a ride, the system tags the car as available again.
- [D16] Sensors placed into car seats reveal that a passenger is seated if and only if a passenger is actually seated.
- [D17] Adding a car to the system consists in adding its license plate, make and model, and in allowing the onboard car system to communicate with the main system through its own credentials.

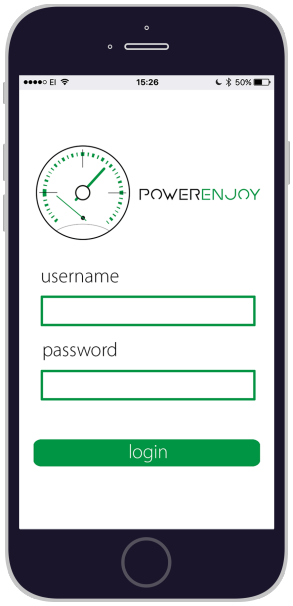
3. Specific Requirements

3.1. External interface requirements

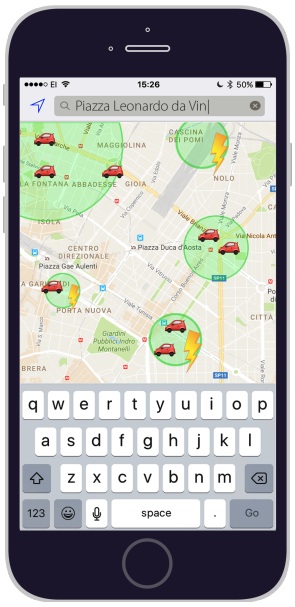
3.1.1. User interfaces

The following mockups represent a basic idea of what the mobile app will look like in the first release.

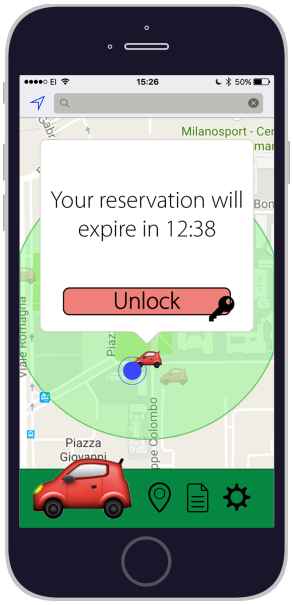
3.1.1.1. Login



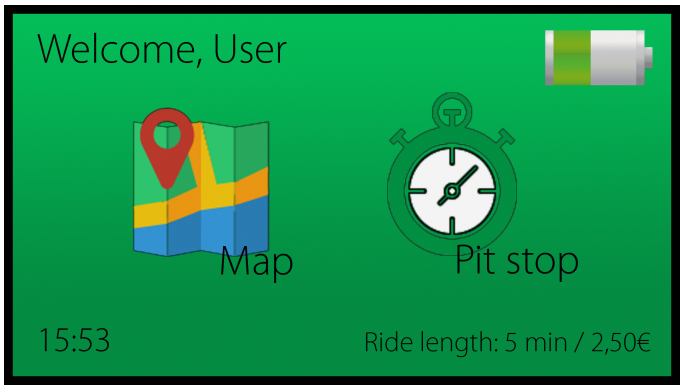
3.1.1.2. Search by address and reservation



3.1.1.3. Car unlocking



3.1.1.4. Car touchscreen



3.2. Functional requirements

3.2.1. [G1] Allow a visitor to become registered User after providing credentials and payment information.

- [R1] A visitor must be able to begin the registration process. During the process the system will ask him/her to provide credentials and payment information.
- [R2] When the correctness of payment information provided is verified the system must generate a new random password and send it to the User by SMS.

- [D1] Payment information correctness may be verified using an external service that will be in charge of all the payment processes.
- [D3] The Username must be unique.

- [D4] When the system sends a password to a new registered User this will be surely received by him/her.

3.2.2. [G2] Allow an User to find locations of available cars.

- [R3] A registered User must be able to login to the system using his/her credentials.
- [R4] The system must be able to provide the list of available cars within a certain area, encoded as the coordinates of an origin point and the maximum distance from it, from the current position of an User.
- [R5] The system must be able to provide the list of available cars within a certain area, encoded as the coordinates of an origin point and the maximum distance from it, from an address specified by an User.

- [D5] Users and cars locations are retrieved by GPS.
- [D6] Cars are equipped with sensors (GPS, sensors into the seats) and communicate over mobile network.
- [D8] When the system shows a car in a certain position it means that it's actually there.

3.2.3. [G3] Allow an User to reserve a car.

- [R3] A registered User must be able to login to the system using his/her credentials.
- [R6] The system must be able to permit to a User to reserve a single car among the available cars for up to one hour before picking it up.
- [R7] If a reserved car is not picked up within one hour from the reservation the system tags it as available, the reservation expires and the User that made the reservation pays a fee of 1 EUR.
- [R8] Reservations made by Users last for one hour, after that period the car is tagged as available again.
- [R9] The system must charge a registered User for 1 EUR if him/her doesn't pick up the car he/she has reserved.

3.2.4. [G4] Allow an User to unlock the car that he/she has registered.

- [R3] A registered User must be able to login to the system using his/her credentials.
- [R10] An User must be able to notify the system that he/she is nearby a car that he/she has registered.
- [R11] When the system is notified by an User of his/her nearness to a reserved car it has to verify the User's location and possibly unlock the car.

- [D5] Users and cars locations are retrieved by GPS.
- [D8] When the system shows a car in a certain position it means that it's actually there.

3.2.5. [G5] Allow an User to complete and pay for a full ride with the car that he/she has registered.

- [R12] As soon as the car's engine ignites the system starts charging the User for a given amount of money per minute.
- [R13] The system must notify the User about current charges trough a screen on the car.
- [R14] When the car is parked in a safe area and the "pit-stop" button is not activated the system stops charging the User and the current ride must be terminated.
- [R15] When a ride is terminated the system stops charging the User and calculates ride's length.
- [R16] The system must keep track of a ride's length in minutes.
- [R17] The system locks the car automatically when the User exits.
- [R23] After a specified amount of time after ride's termination the system calculates the ride price according to the ride's length and to the discounts possibly achieved and charges him/her of the total due.

- [D9] The set of safe areas is never empty.

3.2.6. [G6] Allow an User to get discounts on prices if he/she behaves well and respects certain conditions.

- [R15] If the sensors placed into car seats reveals more than two passengers' presence the system has to apply a discount of 10% on the ride.
- [R19] If a "parked-in-a-safe-area-car" is left with the battery at least half charged (>= 50%), the system applies a discount of 20% on the ride.
- [R20] If a car is plugged into the power grid of a special parking area the system applies a discount of 30% on the ride.
- [R21] If the location of a parked car is more than 3KM far from a power grid station, the system charges 30% more on the ride.
- [R22] If a parked car is left with less than 20% of battery charged, the system charges 30% more on the ride.

- [D16] Sensors placed into car seats reveal that a passenger is seated if and only if a passenger is actually seated.

3.2.7. [G7] Allow a System Manager to do operations on the system for updating and maintenance.

- [R24] A System Manager must be able to login using his/her credentials.
- [R25] A System Manager must be able to add a new car providing its license plate, make and model, and creating new credentials.
- [R26] A System Manager must be able to add a new safe area or edit an existing one providing its coordinate and maximum acceptable distance.

- [R27] A System Manager must be able to edit fields as amount of money per minute, period of time for which a car remains opened, minimum distance of the User from the car to be considered "nearby", amount of time to plug the car into the power grid after leaving the car.
- [R28] A System Manager must be able to see all the cars registered into the system, or search for a specific one using its license plate.

- [D13] A safe area, considered as a circle, is described by a coordinate and the maximum acceptable distance from the coordinate given.
- [D17] Adding a car to the system consists in adding its license plate, make and model, and in allowing the onboard car system to communicate with the main system through its own credentials.

3.3. Non-functional requirements

3.3.1. Performance

The system has to be able to respond to a possibly great number of simultaneous requests and more generally to a great number of request throughout the day. Basing the analysis on competitors' results an average of 200000 (two hundred thousands) subscribed users and 10000 (ten thousands) reservations per day may have to be expected and so correctly managed.

3.3.2. Reliability

The system must guarantee a 24/7 service. Very small deviations from this requirement will be obviously acceptable.

3.3.3 Security

Users credentials and payment informations will be stored. Security of the data and of the communications user-system and car-system is a primary concern.

3.3.4 Scalability

Sharing economy and specifically car sharing, are very promising market sectors. The system that has to be built have to consider this horizon, guaranteeing an high level of scalability both on cars number and available safe areas.

3.3.5. Accuracy

Accuracy of the informations provided (positions of available cars, power grid stations, ecc) has to be the best possible. All the sensors used must provide positions' data with an error lower than 10 meters (GPS technology is theoretically able to provide locations with an error less or equal to 7.8 meters [1.4. References]).

4. Scenarios

4.1. Scenario 1

Tim asks to his best friend John if he wants to join him for dinner this evening. Unfortunately John's car is temporarily out of service and so Tim advises him to try PowerEnJoy. John picks up his smartphone and downloads the PowerEnJoy app. Then, John provides his informations to the system and receives back the password for the access. One hour before leaving home to reach his friend, John searches for an available car nearby through the application and once he has found one, proceeds to the reservation. Eventually, John picks up his reserved car and drives to Tim's house for dinner.

4.2. Scenario 2

Elon today is going to have breakfast at the local Starbucks store fifteen minutes of walk far from his house, and then he is going to pick up a PowerEnJoy car to go to work. Elon provides his credentials to the application and after logging in, he types the address of the Starbucks store looking for an available car nearby. Elon finds one and reserves it. Once finished his breakfast he picks the car up to reach his workplace.

4.3. Scenario 3

Matt has invited his brother to join him for lunch in their favorite fast food. To reach it Matt decides to reserve a PowerEnJoy with his smartphone during the business meeting he is attending. Unfortunately, the meeting lasts longer than expected and Matt can't pick up the reserved car paying a 1 EUR fee. Matt needs to run now to get to his brother in time. Luckily, the burned calories, unlike the 1 EUR fee, may be replenished with a hearty lunch.

4.4. Scenario 4

Emma needs to go to the grocery store to pick up some food for the next few days. Her boyfriend has recommended her a new app called PowerEnJoy with which she can take an electric car for a quick ride. Emma has already downloaded the app so goes directly into the registration process. After doing that she reserves an available car and goes off to the store. Once arrived, she presses the "pit-stop button" on the touchscreen to ensure that the car remains reserved. When she has finished her errand, she goes back to the parking spot and drives back to her house with the same car.

4.5. Scenario 5

Kate and two friends of her need to find a way to reach the place where their favorite band is going to have a concert tonight. This afternoon Kate has seen an advertisement of this new car sharing service called PowerEnJoy. After signing in, she reserves a car nearby her house and gets ready to going out. During the ride Kate picks up her two friends and together they get to the concert spot. As Kate has taken with her two more passengers the system will apply a discount of 10% on the total price of the ride.

4.6. Scenario 6

Ben and Max want to go to the cinema tonight to see the movie based on the last novel of Dan Brown (which, by the way, will be very disappointing). Ben already knows PowerEnJoy and is a faithful user of the service. In the evening Max reaches his friend's house, and together they pick up an available car immediately after having reserved it through the website using Ben's account. Once arrived to the cinema, the guys park the car in a special parking area nearby, and despite there is 55% of battery left they plug it into a power grid. Because of that Ben will receive a 20% discount for the amount of battery left in the car, and an additional 30% off for taking care of plugging the car into a power grid. Good job guys.

4.7. Scenario 7

Casey needs to reach the other side of the town to see a friend of him that will be there only for the day. Casey uses his smartphone to reserve an available car nearby his office starting the ride few minutes later. Once arrived (a little bit late, to be fair) he parks the car in a safe area 5.5 KM far from the nearest power grid and with the 15% of battery level left. Casey will have to pay 60% more (30% for the low battery level and 30% for the parking spot).

4.8. Scenario 8

Lukas, a PowerEnJoy's system manager, needs to add a new available car and two new safe areas into the system. Lukas logs into the platform providing his credentials and starts with adding the license plate and with configuring the credentials of the new car that has been left by one of his colleagues in the main road of the town. Then he proceeds to the safe areas insertion, providing the coordinates and the maximum acceptable distances for both of them.

5. UML modeling

5.1. Use case descriptions

5.1.1. Visitor registration

ACTORS	Visitor
GOALS	[G1]
INPUT CONDITIONS	There are no entry conditions.
EVENTS FLOW	1. The Visitor on the home page clicks on the “Sign In” button to start the registration process. 2. The Visitor fills all the mandatory fields and provides his/her payment information. 3. The Visitor clicks on the “Confirm” button. 4. The system saves the data. 5. The system sends an SMS to the new User with the password.
OUTPUT CONDITIONS	The Visitor successfully ends the registration process and become a new User. From now on he/she can log in to the application providing his/her credentials and start using PowerEnJoy.
EXCEPTIONS	1. The Visitor is already an User. 2. The Visitor inserts not valid informations in one or more mandatory fields. 3. The Visitor chooses an username that has already been taken by another user. 4. The Visitor chooses an email that has been associated with another user. All exceptions are handled notifying the issue to the Visitor and taking back the Event Flow to the point 2.

5.1.2. User login

ACTORS	User
GOALS	[G1]
INPUT CONDITIONS	The User is already on the home page.
EVENTS FLOW	1. The User inserts his/her credentials into the “username” and “password” fields. 2. The User clicks on the “Log In” button in order to access. 3. The system redirects the User to the map and searches for available cars in the nearby automatically.
OUTPUT CONDITIONS	The User is successfully redirected to the map.
EXCEPTIONS	1. The User inserts a not valid username. 2. The User inserts a not valid password. All exceptions are handled notifying the issue to the Visitor and taking back the Event Flow to the point 2.

5.1.3. User reserves an available car after GPS localization

ACTORS	User
GOALS	[G2] [G3]
INPUT CONDITIONS	The User has to be already logged into the system
EVENTS FLOW	1. The User looks at the map that shows available cars nearby. 2. The User chooses one of the available cars. 3. The User clicks on the “Reserve” button on the reservation page. 4. The system removes the car from the set of available cars and starts the timer for the reservation. 5. The system redirects the User to a confirmation page.
OUTPUT CONDITIONS	The User has successfully reserved a car.
EXCEPTIONS	1. The User clicks on the “Reserve” button but meanwhile the car he/she want to reserve has already been reserved by someone else. This exception is handled redirecting the User to an error page in which will be explained the issue occurred.

5.1.4. User reserves a car after searching it by address

ACTORS	User
GOALS	[G2] [G3]
INPUT CONDITIONS	The User has to be already logged into the system.
EVENTS FLOW	1. The User inserts the address of the position where he wants to search for an available car. 2. The system shows on the map the chosen area. 3. The User chooses one of the available cars. 4. The User clicks on the “Reserve” button on the reservation page. 5. The system removes the car from the set of available cars and starts the timer for the reservation. 6. The system redirects the User to a confirmation page.
OUTPUT CONDITIONS	The User has successfully reserved a car.
EXCEPTIONS	1. The User clicks on the “Reserve” button but meanwhile the car he/she want to reserve has already been reserved by someone else. This exception is handled redirecting the User to an error page in which will be explained the issue occurred.

5.1.5. User unlocks a car

ACTORS	User
GOALS	[G4]
INPUT CONDITIONS	The User has already reserved the car.
EVENTS FLOW	1. The User, once he/she is near to the reserved car, notifies his/her condition to the system. 2. The system receives the notification from the User. 3. The system verifies that the User is actually in the right place. 4. The system opens the car. 5. The User gets into the car.
OUTPUT CONDITIONS	The User has successfully opened the car.
EXCEPTIONS	1. The User is not in the cars's neighborhood. This exception is handled reporting the problem on the screen of the User's smartphone.

5.1.6. User stops a ride

ACTORS	User
GOALS	[G5]
INPUT CONDITIONS	The User is driving the car.
EVENT FLOWS	1. The User stops the car in a safe area. 2. The User gets out of the car. 3. The system starts the timer for plugging the car into a power grid. 4. When the countdown is finished the system calculates the total price for the ride and actually charges the User. 5. The User is notified of the successful payment.
OUTPUT CONDITIONS	The User has successfully finishes his/her ride.
EXCEPTIONS	

5.1.7. User enters in pit-stop mode

ACTORS	User
GOALS	[G5]
INPUT CONDITIONS	The User is driving the car.
EVENTS FLOW	1. The User stops the car. 2. The User presses the “pit-stop button”. 3. The system receives from the car the notification of the User’s intention. 4. The system shows on the car’s screen the confirmation message. 5. The User gets out of the car.
OUTPUT CONDITIONS	The User has successfully put the car in “Pit-stop” mode.
EXCEPTIONS	1. The User clicks on the “pit-stop button” while he/she is still driving. This exception is handled showing on the car’s screen an error message.

5.1.8. User plugs a car into a power grid

ACTORS	User
GOALS	[G6]
INPUT CONDITIONS	The User already stopped the car and, he/she is already out and the system already started the power grid timer.
EVENTS FLOW	1. The User plugs the car into a power grid. 2. The system recognizes that the car is charging. 3. The system applies a 30% off on the total price for the ride.
OUTPUT CONDITIONS	The car is charging and the User has a discount on the price.
EXCEPTIONS	

5.1.9. System manager login

ACTORS	System Manager
GOALS	[G7]
INPUT CONDITIONS	There are no entry conditions.
EVENTS FLOW	1. The System Manager opens the home page of his private interface. 2. The System Manager inserts his/her credentials into the “username” and “password” fields. 3. The System Manager clicks on the “Log In” button in order to access. 4. The system redirects the System Manager to the map of all the cars registered into the system.
OUTPUT CONDITIONS	The System Manager is successfully redirected to the map.
EXCEPTIONS	1. The System Manager inserts a not valid username. 2. The System Manager inserts a not valid password. All exceptions are handled notifying the issue to the System Manager and taking back the Event Flow to the point 2.

5.1.10. System manager searches for a car

ACTORS	System Manager
GOALS	[G7]
INPUT CONDITIONS	The System Manager has to be already logged into the system.
EVENTS FLOW	1. The System Manager selects the option “Search”. 2. The System Manager inserts the license plate of the new car. 3. The system shows all the information about that car (license plate, model, color, battery level) and its position.
OUTPUT CONDITIONS	The System Manager successfully finds the car.
EXCEPTIONS	1. The System Manager inserts a not valid license plate. This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 2.

5.1.11. System manager edits system fields

ACTORS	System Manager
GOALS	[G7]
INPUT CONDITIONS	The System Manager has to be already logged into the system.
EVENTS FLOW	1. The System Manager selects the option “Edit fields”. 2. The System Manager inserts the updated values in the fields that he/she wants to modify. 3. The System Manager presses the “Confirm” button. 4. The system redirects the System Manager to the home page.
OUTPUT CONDITIONS	The system has successfully updated the fields.
EXCEPTIONS	1. The System Manager inserts a not valid input in one or more fields. This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 2.

5.1.12. System manager adds a new car to the system

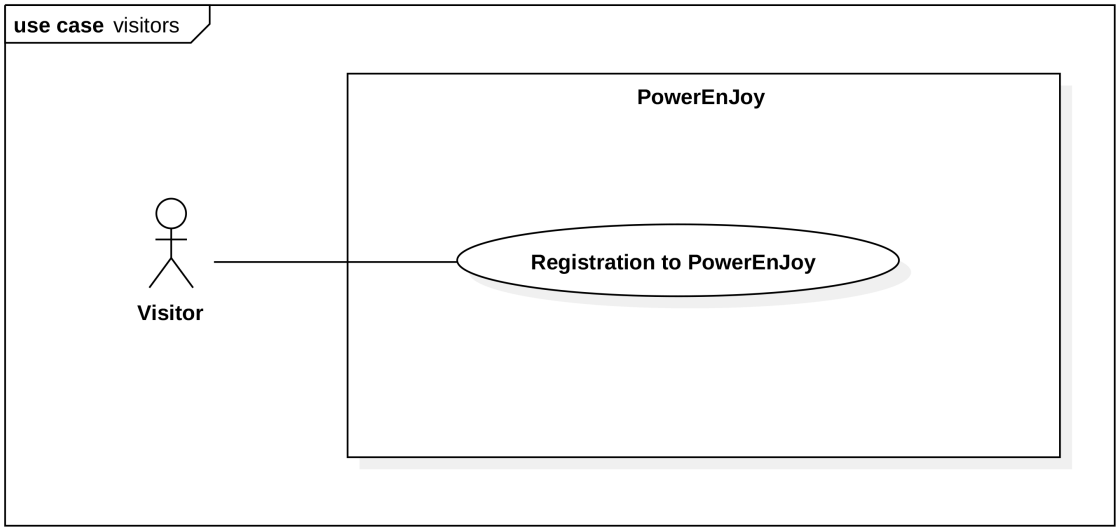
ACTORS	System Manager
GOALS	[G7]
INPUT CONDITIONS	The System Manager has to be already logged into the system.
EVENTS FLOW	1. The System Manager selects the option “Add a car”. 2. The System Manager inserts the license plate of the new car. 3. The System Manager presses the “Confirm” button. 4. The system generates new credential for the added car.
OUTPUT CONDITIONS	The new car is successfully added to the set of available cars.
EXCEPTIONS	1. The System Manager inserts a not valid license plate. This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 2.

5.1.13. System manager adds a new safe area

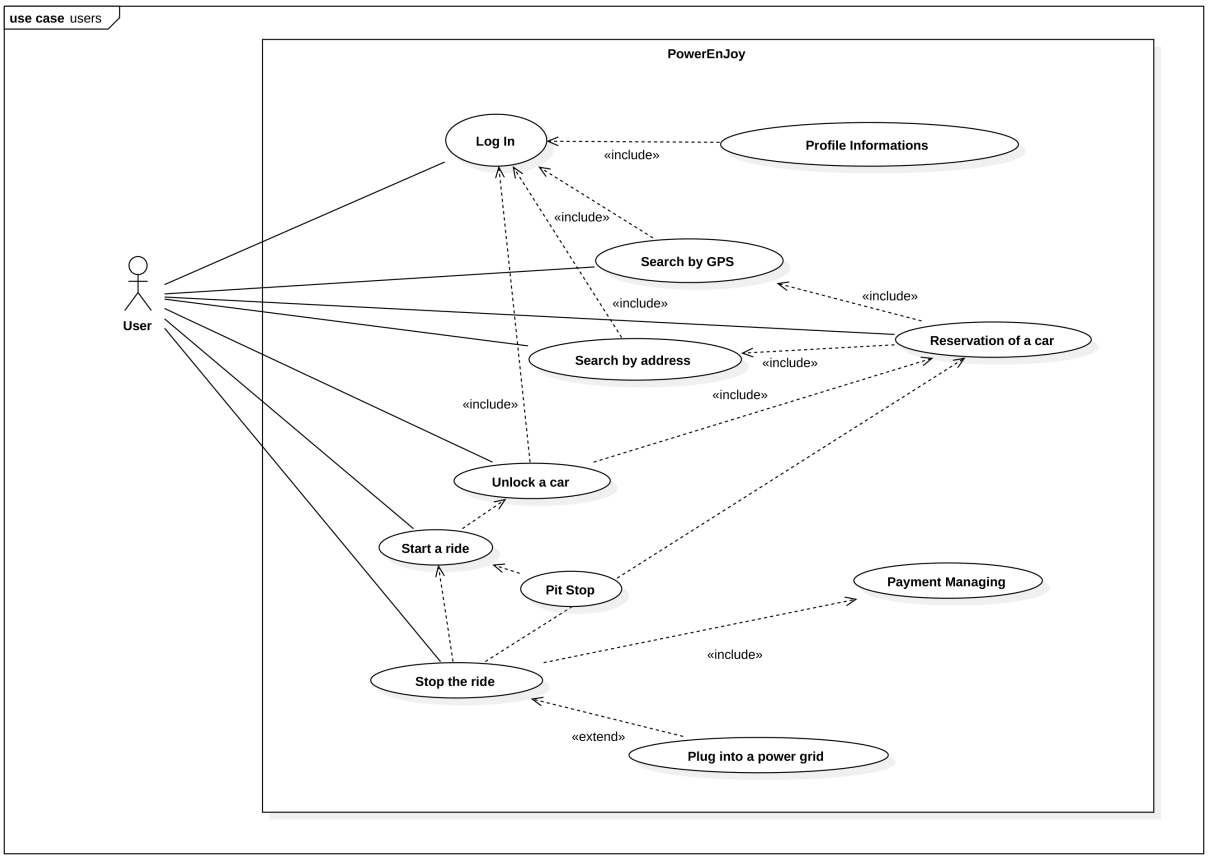
ACTORS	System Manager
GOALS	[G7]
INPUT CONDITIONS	The System Manager has to be already logged into the system.
EVENTS FLOW	1. The System Manager selects the option “Add a safe area”. 2. The System Manager inserts the coordinates and the maximum acceptable distance for the new area. 3. The System Manager checks the “Special Parking Area” checkbox if in the new safe area there is at least one power grid. 4. The System Manager presses the “Confirm” button. 5. The system adds the new safe area to the set of available safe areas. 6. The system redirects the System Manager to the home page.
OUTPUT CONDITIONS	The new safe area is successfully added to the set of available areas.
EXCEPTIONS	1. The System Manager inserts a not valid input in one or more fields. This exception is handled notifying the issue to the System Manager and taking back the Event Flow to the point 2.

5.2. Use case diagrams

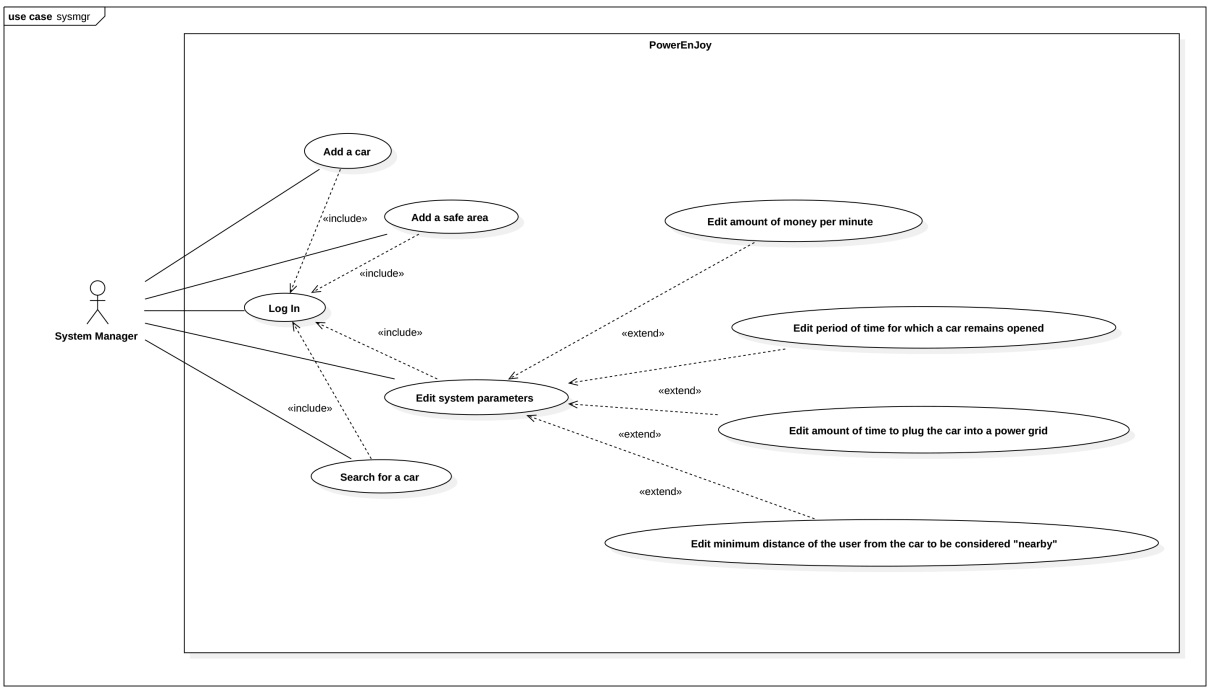
5.2.1. Visitor



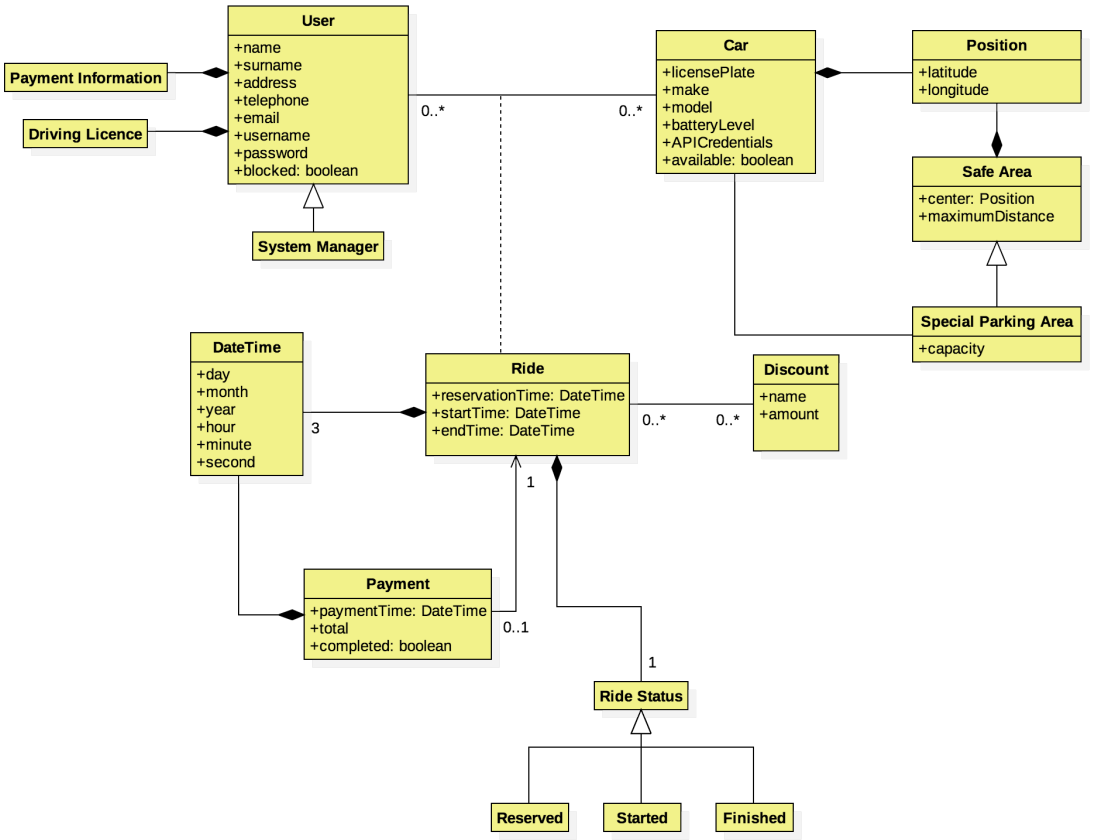
5.2.2. User



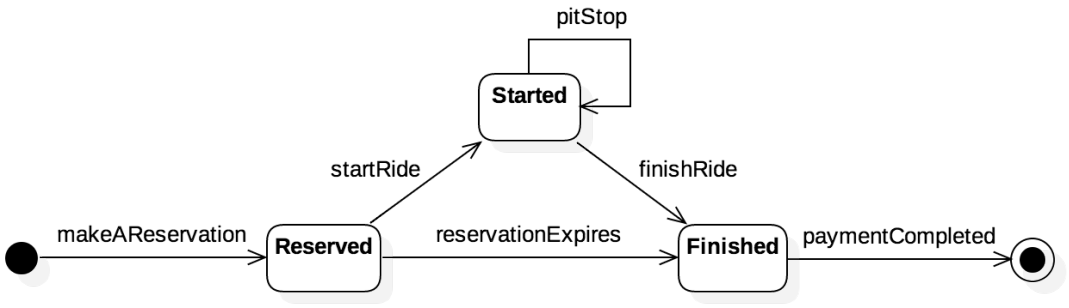
5.2.3. System manager



5.3. Class diagram

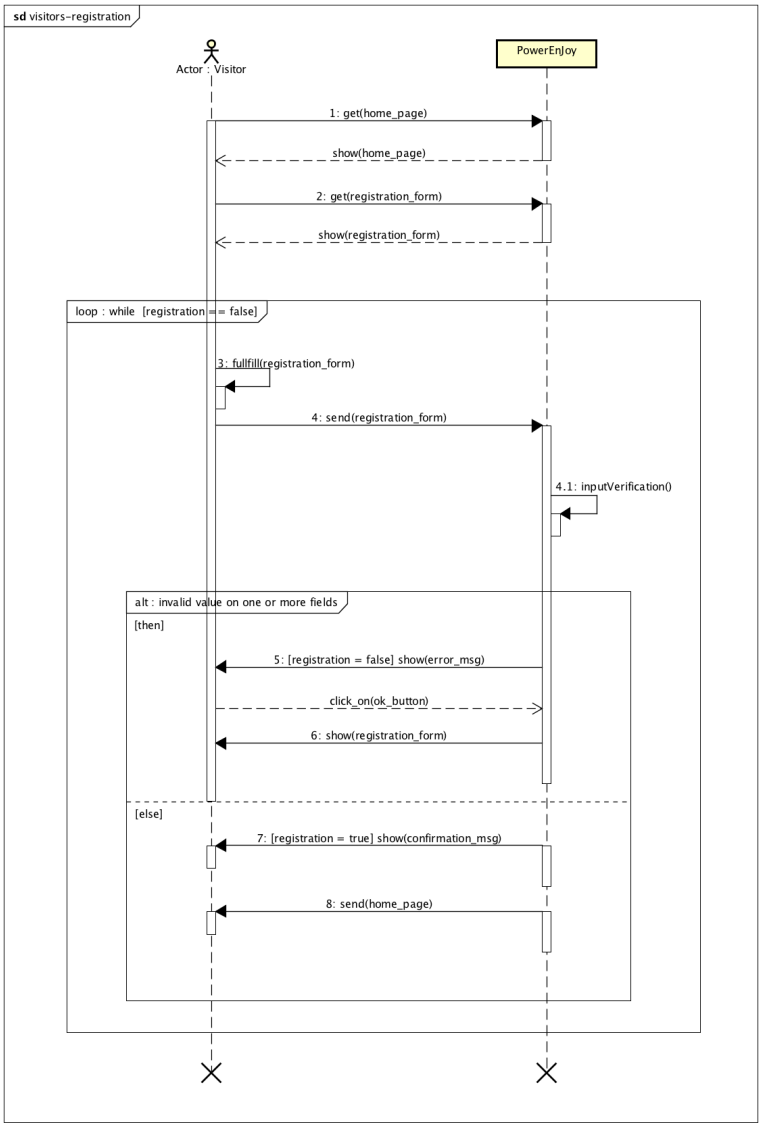


5.4. Statechart diagram

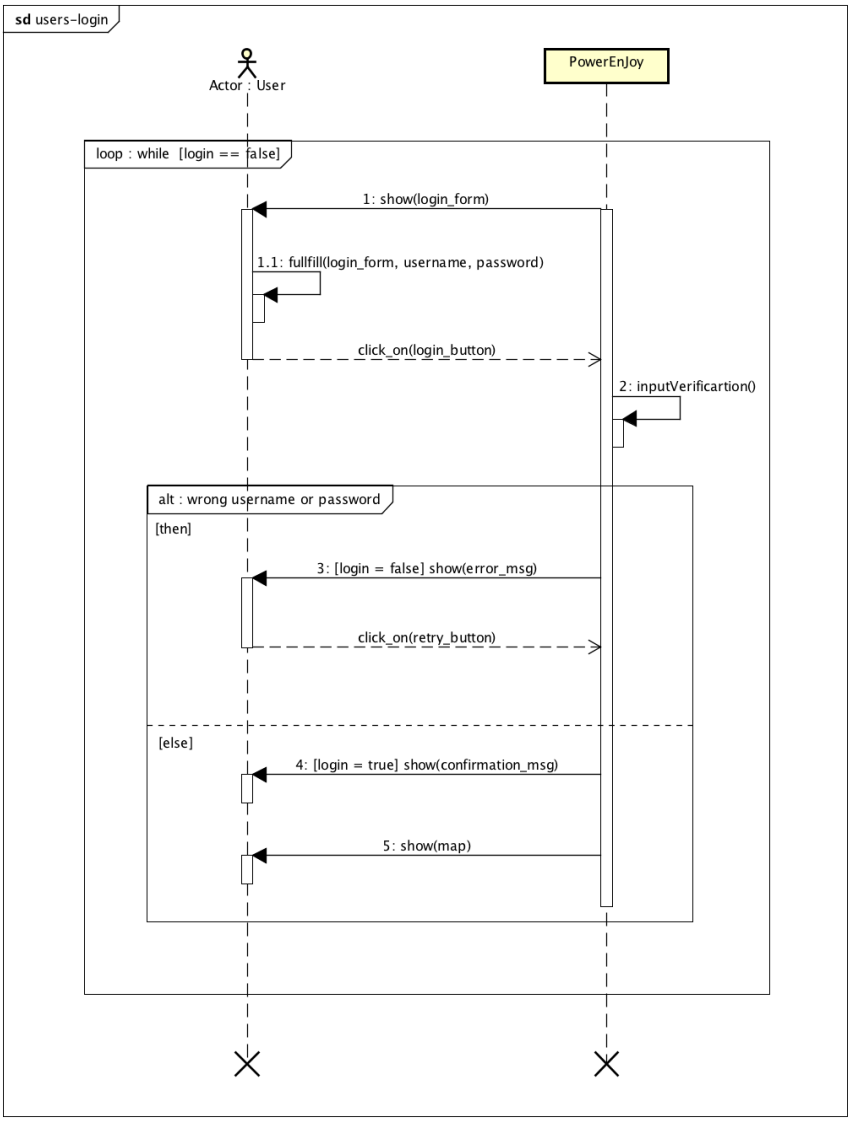


5.5. Sequence diagrams

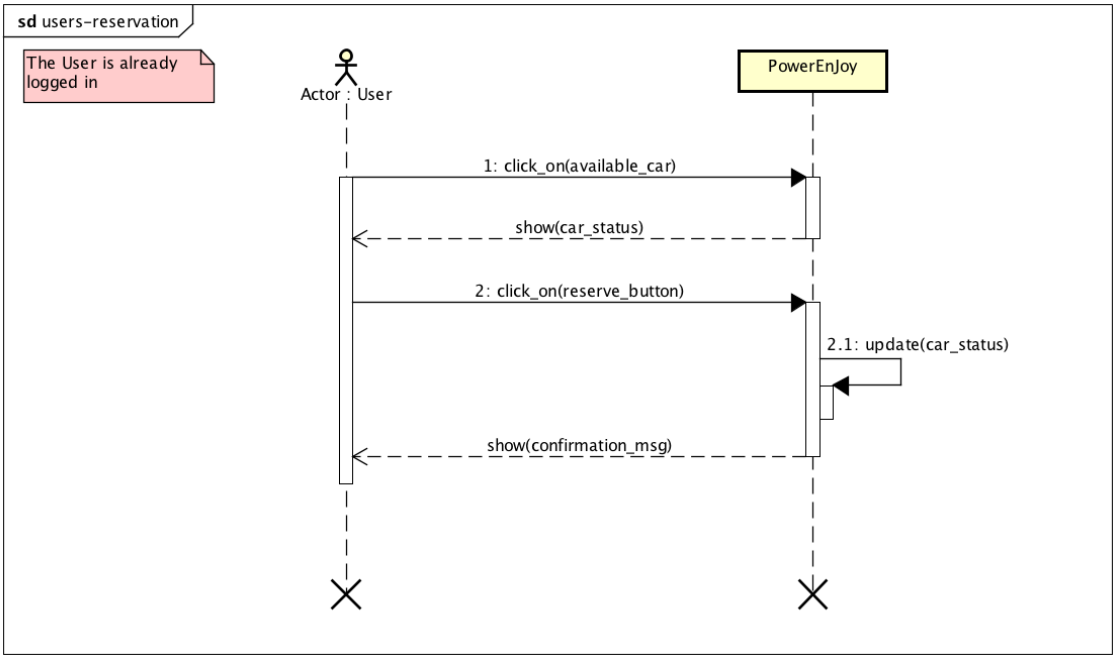
5.5.1. Visitor registration



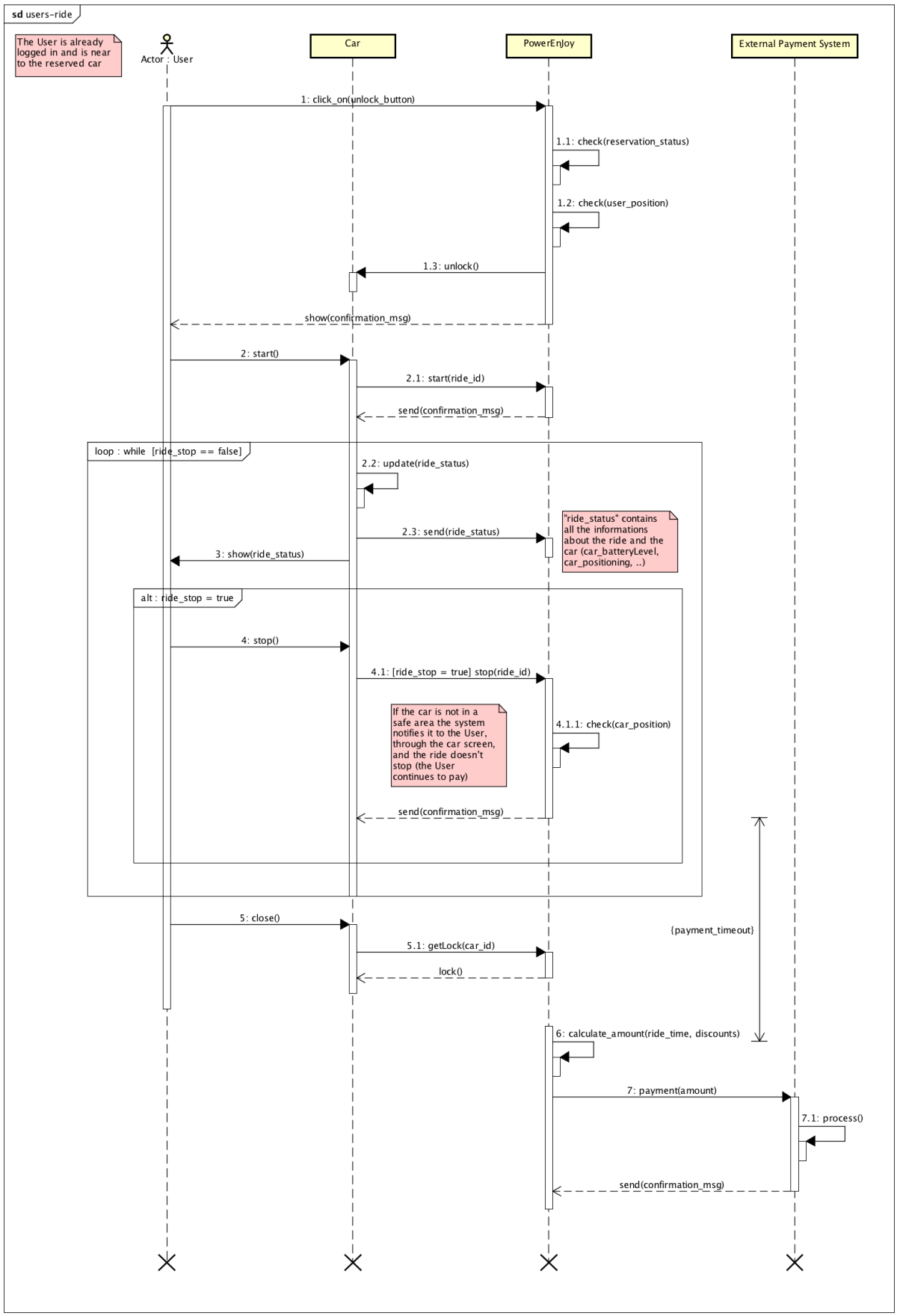
5.5.2. User login



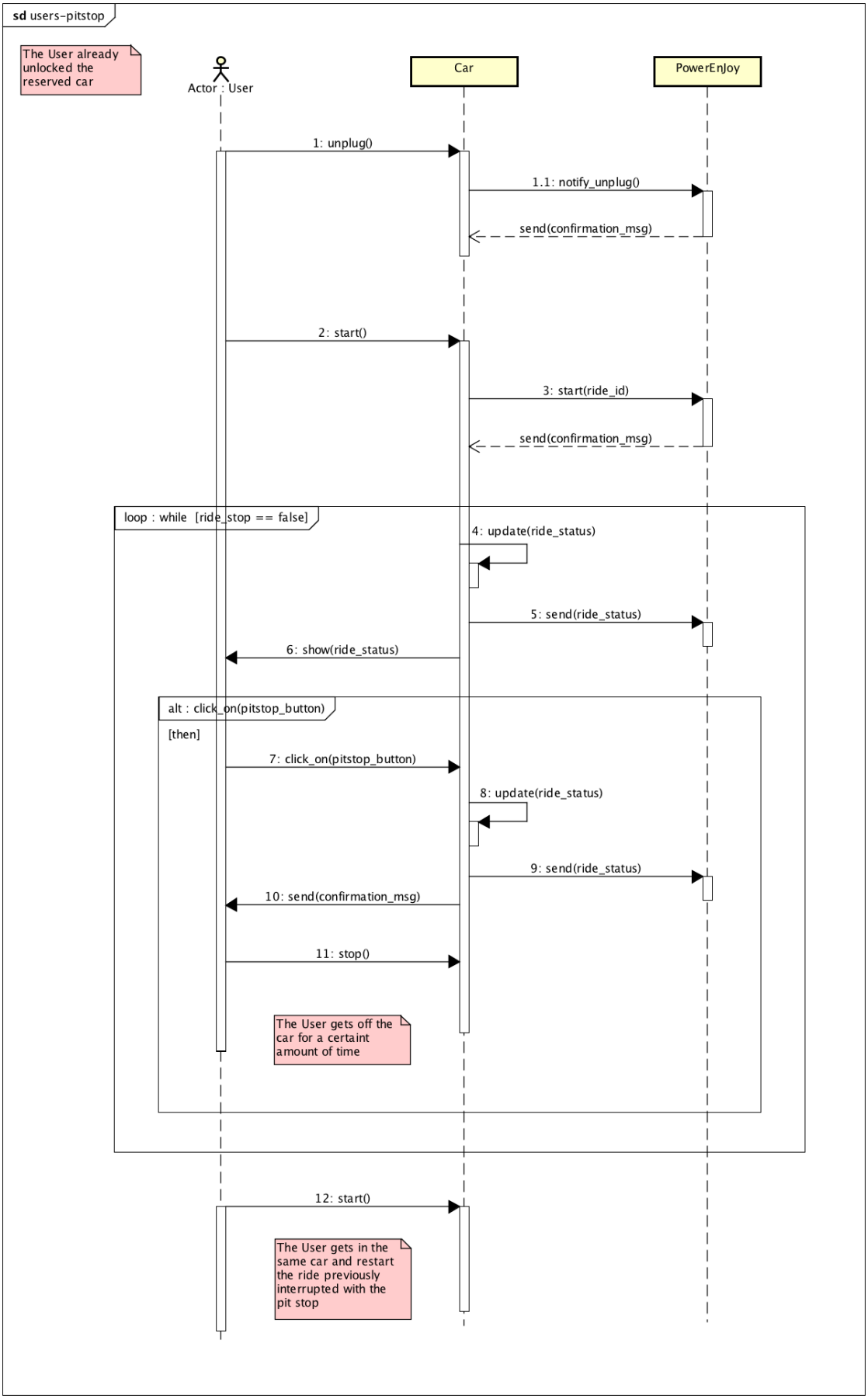
5.5.3. User reservation



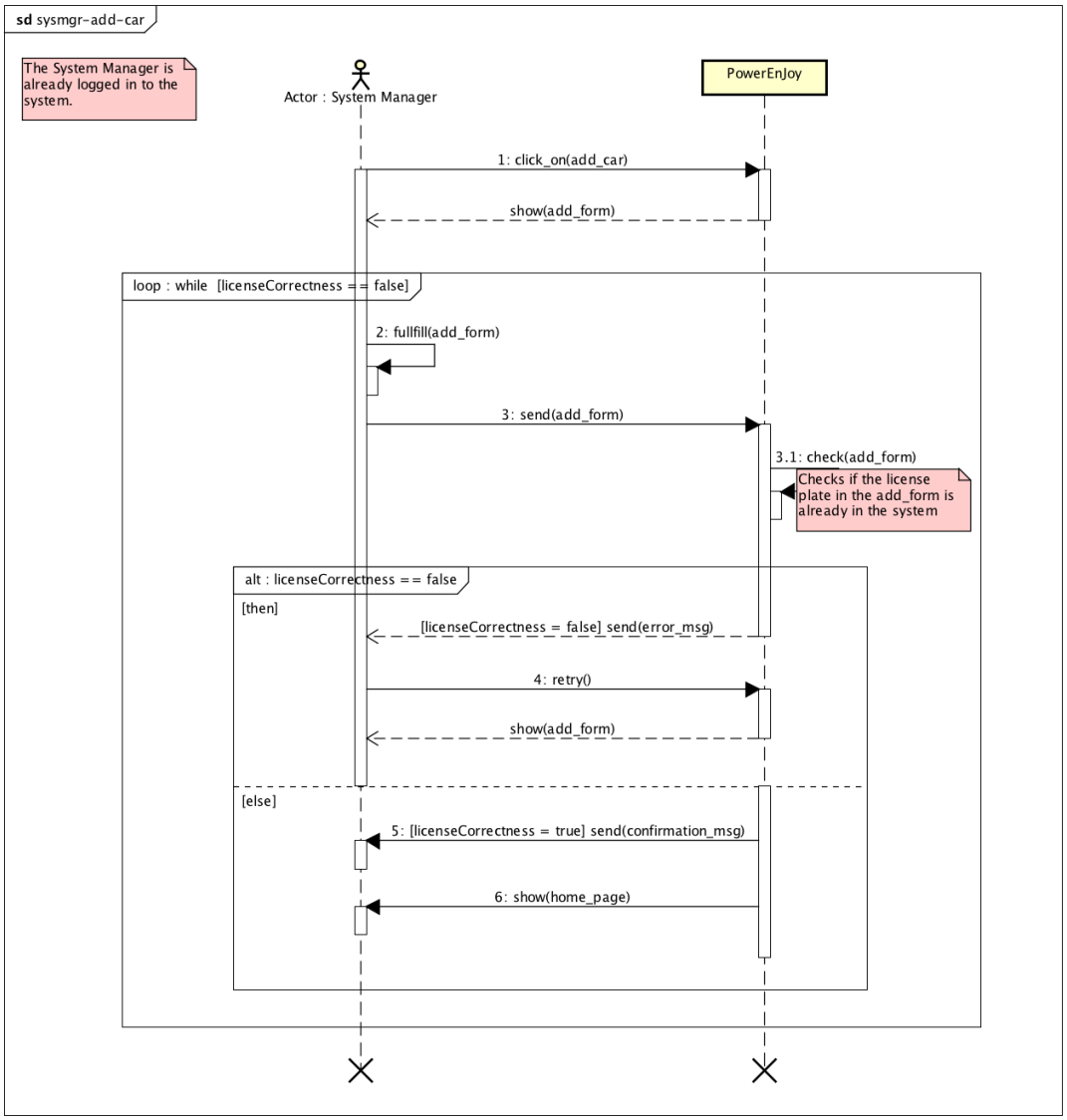
5.5.4. User in a Ride



5.5.5. User enters in pit-stop mode



5.5.6. System manager adds a new car to the system



6. Alloy modeling

6.1. Signatures

```
open util/time
open util/integer
open util/boolean


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

sig User{
	username: one String,
	blocked: Bool one -> Time
}

sig Car{
	licensePlate: one String,
	batteryLevel:  Int one -> Time,
	position: Position one ->Time,
	available: Bool one -> Time
} {
	all level: batteryLevel.Time | level >= 0 and level <= 100
}

sig Discount{
	amount: Int
} {
	amount > 0 and amount <= 100
}

abstract sig RideStatus {}
one sig Reserved extends RideStatus{}
one sig Started extends RideStatus{}
one sig Finished extends RideStatus{}

sig Ride{
	user: one User,
	car: one Car,
	rideStatus: RideStatus lone -> Time,
	discounts: set Discount
}

sig Payment{
	relatedRide: one Ride,
	paymentTime: one Time,
	amount: one Int,
	completed: one Bool
} {
	amount > 0
	-- A Payment must be raised only if the relatedRide is Finished
	all t: Time | gte[t,paymentTime] => relatedRide.rideStatus.t = Finished
}

sig SafeArea{
	center: one Position,
	maximumDistance: one Int
} {
	maximumDistance > 0
}

sig SpecialParkingArea extends SafeArea{
	capacity: one Int,
	pluggedCars: Car set -> Time
} {
	capacity > 0
	all t: Time | capacity >= #pluggedCars.t
}
```

6.2. Facts

```
fact usernameIsUnique{
	no disjoint u1,u2: User | u1.username = u2.username
}

fact licensePlateIsUnique{
	no disjoint c1,c2: Car | c1.licensePlate = c2.licensePlate
}

fact atLeastOneSafeArea{
	#SafeArea > 0
}

-- Ride static behavior
fact rideStateChart{
	-- A ride is always created as "Reserved"
	all r: Ride | one t': Time | r.rideStatus.t' = Reserved
	all r: Ride, t: Time |
		-- Once a ride is "Finished" it can't change status again
		(r.rideStatus.t = Finished =>
			all t': Time | gte[t',t] => r.rideStatus.t' = Finished)
		and
		-- Once a ride is "Started" it cannot get back to "Reserved"
		(r.rideStatus.t = Started =>
			all t': Time | gte[t',t] => r.rideStatus.t' != Reserved)
}

-- A Car can be involved in one "active" Ride per time
fact oneReservationPerTimePerCar {
	no disjoint r1,r2: Ride |
		r1.car = r2.car and
			some t: Time |
				r1.rideStatus.t = Started and r2.rideStatus.t != Finished
				or
				r2.rideStatus.t = Started and r1.rideStatus.t != Finished
}

-- An User can be involved in one "Started" Ride per time
fact oneStartedPerUser {
	no disjoint r1,r2 : Ride |
		r1.user = r2.user and
			some t: Time |
				r1.rideStatus.t = Started and r2.rideStatus.t = Started
}

fact finishedRideMustBePaid {
	all t: Time, r: Ride | r.rideStatus.t = Finished =>
		one p: Payment | p.relatedRide = r
}

-- A Car can be plugged into at most one power grid at time
fact atMostOnePowerGrid {
	all t: Time, c: Car | lone s: SpecialParkingArea |
		c in s.pluggedCars.t
}
```

6.3. Dynamic model

```
pred isUserInARide[u: User, t: Time]{
	one r: Ride | r.rideStatus.t = Started and r.user = u
}

pred isCarAvailable[c: Car, t: Time]{
	-- A car is actually available iff has at least 20 battery left,
	-- is not involved in any "active" Ride and is available to usage.
	c.available.t = True
	c.batteryLevel.t >= 20
	no r: Ride | r.car = c and
		(r.rideStatus.t = Reserved or r.rideStatus.t = Started)
}

pred makeAReservation[u: User,c: Car,t: Time,r': Ride]{
	// preconditions
	isCarAvailable[c,t]
	not isUserInARide[u,t]
	// postconditions
	r'.user = u
	r'.car = c
	not isCarAvailable[c,t.next]
	not isUserInARide[u,t.next]
	r'.rideStatus.(t.next) = Reserved
}

pred startARide[r: Ride,t: Time]{
	// preconditions
	not isCarPluggedIn[r.car,t]
	not isUserInARide[r.user,t]
	r.rideStatus.t = Reserved
	// postconditions
	r.rideStatus.(t.next) = Started
	isUserInARide[r.user,t.next]
}

pred endARide[r: Ride,t: Time]{
	// preconditions
	not isCarAvailable[r.car,t]
	isUserInARide[r.user,t]
	r.rideStatus.t = Reserved || r.rideStatus.t = Started
	// postconditions
	r.rideStatus.(t.next) = Finished
	isCarAvailable[r.car,t.next]
	not isUserInARide[r.user,t.next]
}
```

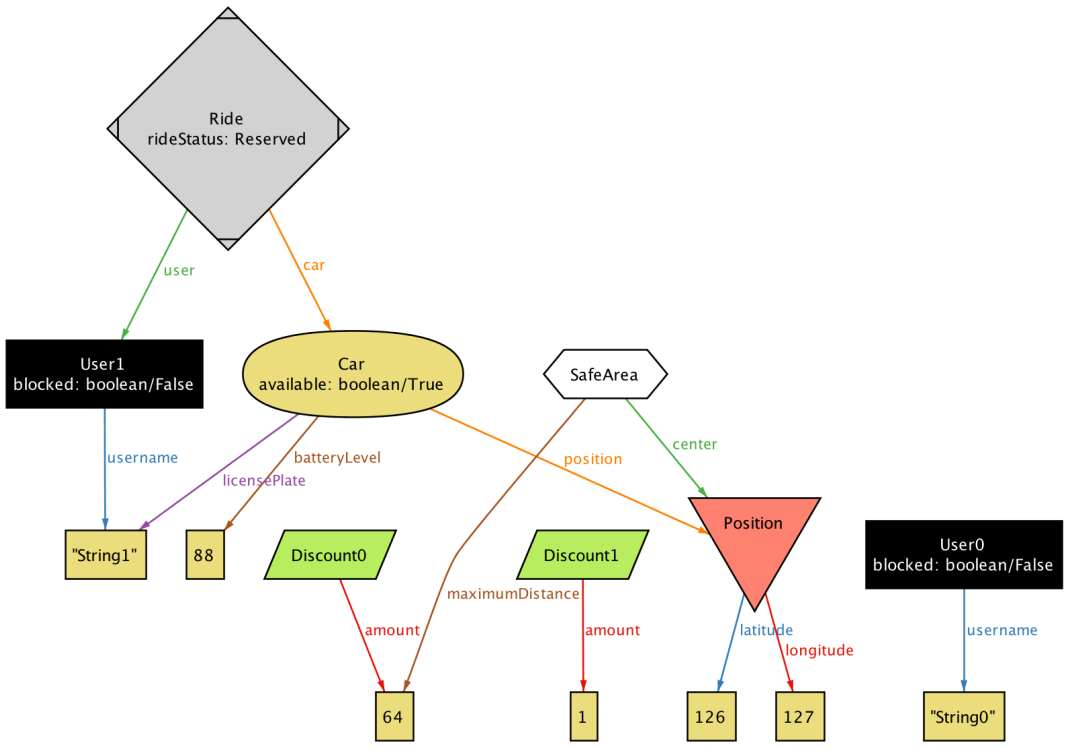
6.4. Results

```
run makeAReservation for 5 but 8 Int, exactly 5 String
run startARide for 5 but 8 Int, exactly 5 String
run endARide for 5 but 8 Int, exactly 5 String
run {#Ride>0} for 2 but 8 Int, exactly 2 String
```

6.4.1. Proof of consistency

```
Executing "Run makeAReservation for 5 but 8 int, exactly 5 String"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  146357 vars. 13370 primary vars. 453526 clauses. 1134ms.
  Instance found. Predicate is consistent. 329ms.

Executing "Run startARide for 5 but 8 int, exactly 5 String"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  126883 vars. 13360 primary vars. 381275 clauses. 1383ms.
  Instance found. Predicate is consistent. 468ms.

Executing "Run endARide for 5 but 8 int, exactly 5 String"
  Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20
  146281 vars. 13360 primary vars. 453602 clauses. 1058ms.
  Instance found. Predicate is consistent. 306ms.

Executing "Run run$4 for 2 but 8 int, exactly 2 String"
  Solver=sat4j Bitwidth=8 MaxSeq=2 SkolemDepth=1 Symmetry=20
  44203 vars. 3672 primary vars. 139554 clauses. 680ms.
  Instance found. Predicate is consistent. 276ms.

4 commands were executed. The results are:
  #1: Instance found. makeAReservation is consistent.
  #2: Instance found. startARide is consistent.
  #3: Instance found. endARide is consistent.
  #4: Instance found. run$4 is consistent.
```

6.4.2. Generated world



7. Appendix

7.1. Used tools

- Apple Pages 6.0.5
- Brackets 1.7.0
- StarUML 2.7.0
- GIMP 2.8.18
- Astah Professional 7.1.0
- Alloy Analyzer 4.2_2015-02-22

7.2. Hours of work

7.2.1. Bassetto Riccardo

DATE	TASK	HOURS
17/10/2016	Specifications and goals	1
18/10/2016	Goals and assumptions	2
20/10/2016	Scenarios	3
21/10/2016	Use cases	4
22/10/2016	Use cases	4
24/10/2016	Use cases diagrams	3
27/10/2016	Sequence diagrams	2
30/10/2016	Sequence diagrams	3
03/11/2016	RASD	3
04/11/2016	RASD	4
	TOTAL:	**29**

7.2.2. Belloni Massimo

DATE	TASK	HOURS
17/10/2016	Specifications and goals	0,5
18/10/2016	Goals and assumptions	2
19/10/2016	Requirements	2
21/10/2016	RASD	4,5
22/10/2016	Class diagram	2,5
25/10/2016	Alloy	2
27/10/2016	Various	1
28/10/2016	UI	4
02/11/2016	Alloy	1
03/11/2016	RASD	2
04/11/2016	Alloy and RASD	8
	TOTAL:	**29,5**