



Politecnico di Milano  
A.Y. 2016-2017  
Software Engineering II: “PowerEnJoy”

## **R**equirements **A**nalysis and **S**pecifications **D**ocument

version 2.0  
February 5, 2017

Pozzati Edoardo (mat. 809392), Stefanetti Nadia (mat. 810622)



# Table of Contents

<b>Table of Contents</b>	<b>X</b>
<b>Requirements analysis</b>	<b>1</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Purpose	2
1.2 Actual System	2
1.3 Scope	2
1.4 Goals	3
1.5 Actors Identification	3
1.6 Stakeholders Identification	4
1.7 Documentation	4
<b>2 Overall Description</b>	<b>5</b>
2.1 Product perspective	5
2.2 Product functions	5
2.3 Users characteristics	5
2.4 Constraints	5
2.4.1 Regulatory policies	5
2.4.2 Hardware limitations	6
2.4.3 Software limitations	6
2.4.4 Parallel operation	6
2.5 Assumptions	6
2.6 Future extensions	7
2.7 The World and the Machine	8
<b>Requirements specification</b>	<b>9</b>
<b>3 Specific Requirements</b>	<b>10</b>
3.1 External Interface Requirements	10
3.1.1 User Interfaces	10
3.1.1.1 Login	10
3.1.1.2 Car localization	11
3.1.2 API interfaces	12
3.1.3 Hardware Interfaces	12
3.1.4 Software Interfaces	12

3.1.5	Communication interfaces .....	12
3.2	Functional requirements .....	12
3.2.1	[G1] Allow guests to register into the system. ....	13
3.2.2	[G2] Allow users to log in to the system and manage their accounts. ....	13
3.2.3	[G3] Allow user to view the map of the closest available cars to where he is currently located or another indicated address. ....	13
3.2.4	[G4] Allow users to reserve a single car for up to one hour before they pick it up.....	13
3.2.5	[G5] Allow users to delete an existing reservation within one hour.....	13
3.2.6	[G6] Allow users to unlock the doors and start the rental....	14
3.2.7	[G7] Allow the system to charge the user a fare for the service required. ....	14
3.2.8	[G8] Allow cars to automatically lock themselves.....	14
3.2.9	[G9] Allow the system to notify an operator to request an operation.....	14
3.2.10	[G10] Allow operators to login to the system.....	15
3.4	Scenarios .....	16
3.4.1	Scenario 1.....	16
3.4.2	Scenario 2.....	16
3.4.3	Scenario 3.....	16
3.4.4	Scenario 4.....	16
3.4.5	Scenario 5.....	16
3.4.6	Scenario 6.....	17
3.5	UML Models.....	17
3.5.1	Use Case Diagram .....	17
3.5.1.1	[G1] Allow guests to register into the system.....	18
3.5.1.2	[G2] Allow users to log into the system and manage their accounts. ....	20
3.5.1.3	[G3] Allow users to view the map of the closest available cars to where he is currently located or another indicated address. ....	22
3.5.1.4	[G4] Allow users to reserve a single car for up to one hour before they pick it up. ....	23
3.5.1.5	[G5] Allow users to delete an existing reservation within one hour. ....	24
3.5.1.6	[G6] Allow users to unlock the doors and start the rental. ....	25
3.5.1.7	[G7] Allow the system to charge the user a fare for the service required and to display it on the car. ....	25
3.5.1.8	[G10] Allow the system to notify an operator to request an operation. ....	28
3.5.2	Class Diagrams.....	29
3.5.3	State Machine Diagram .....	30
3.6	Non Functional Requirements .....	30
3.6.1	Performance Requirements.....	30
3.6.2	Software System Attributes .....	30
3.6.2.1	Reliability.....	30
3.6.2.2	Availability .....	30
3.6.2.3	Portability .....	31

3.6.2.4	Security .....	31
3.6.2.5	Maintainability .....	31
3.6.2.6	Persistence .....	31
<b>4</b>	<b>Appendix .....</b>	<b>32</b>
4.1	Alloy .....	32
4.1.1	Data Type.....	32
4.1.2	Abstract Entity .....	34
4.1.3	Fact.....	34
4.1.4	Assert .....	35
4.1.5	Predicates .....	35
4.1.6	Result .....	36
4.1.7	Generated world.....	36
	<b>Appendix A: Used Tools.....</b>	<b>IV</b>
	<b>Appendix B: Hours of work .....</b>	<b>V</b>
	<b>Appendix C: Changelog .....</b>	<b>VI</b>
	<b>Definitions.....</b>	<b>VII</b>
	<b>Acronyms.....</b>	<b>IX</b>
	<b>Bibliography.....</b>	<b>X</b>

## **Abstract**

This document represents the Requirement Analysis and Specification Document (RASD). The main purpose of this document is to give a specification of the requirements that our system must fulfil adopting the IEEE-830 standard for RASD documentation. It also describes the system in terms of functional and non-functional requirements via UML diagrams and a high-level specification of the system, shows the constraints and the limits of the software and simulate the typical use cases that will occur after the development. Finally, it presents the formal model of the specification using Alloy.

The information contained in this document are intended for stakeholders and developers of the project: for the former, this document represents a useful description to understand the project development, whereas for the developers it is quite a comfortable way to match the stakeholders' requests and the proposed solution.

## **Part I**

# **Requirements analysis**

# 1 Introduction

A general introduction and overview of the system-to-be purpose, scope and goals, along with some important information about this document.

## 1.1 Purpose

The aim of this project is to develop a digital management system called PowerEnJoy.

PowerEnJoy is a car-sharing service that exclusively employs electric-cars and allows users to easily find a car, reserve it and use it.

To encourage users' virtuous behaviours some discounts are provided and can be applied to the bill.

## 1.2 Actual System

We suppose that until now a previous version of the system does not exist and we have to create the entire application completely from scratch.

## 1.3 Scope

PowerEnJoy will provide general *functionalities*:

In order to be able to use the service, guest must *sign in* to the system by providing their credentials (a valid driving licence is mandatory) and payment information; afterwards, they receive a password that can be used to access the service.

Users can easily *search available cars* by selecting the desired location (using their current GPS position or inserting a specified address manually).

The system will allow the user to *reserve a single car*, among those compatible with the inserted data, for up to one hour before they pick it up. Within one hour from the reservation, the user can delete the reservation as needed, and the system tags the car available; otherwise, if a car is not picked-up within the hour, the system still tags it as available again, the reservation expires and the user pays a fee of 1€.

The user should be able to *notify* the system that he/she is *nearby* that specific car, so that the system *unlocks the car* and the user may enter.

Then as soon as the engine ignites, the system automatically starts *charging* the user a fixed amount per minute; the user is notified of the current charges through a screen on the car.

Finally, when the car is parked in a safe area and the user exits the car, the system stops charging the user and automatically locks the car that becomes available again.

The system should encourage *virtuous behaviours* of the users: to do that some discounts can be applied on their last ride. For example, a discount of



10% is applied if the user took at least two other passengers onto the car. Other discounts are applied: if a car is left with no more than 50% of the battery empty (20%) or if the user leaves the car at a special parking area where it can be recharged and he takes care of plugging the car into the power grid (30%).

On the other side, if the car is left at more than 3 Km from the nearest power grid station or with more than 80% of the battery empty the system *charges* 30% *more* on the last ride to compensate for the cost required to re-charge the car on-site.

## 1.4 Goals

List of the goals of PowerEnJoy application:

- [G1] Allow guests to register into the system.
- [G2] Allow users to log into the system and manage their accounts.
- [G3] Allow user to search the map for the closest available cars to where he is currently located or another indicated address.
- [G4] Allow users to reserve a single car for up to one hour before they pick it up.
- [G5] Allow users to delete an existing reservation within one hour.
- [G6] Allow users to unlock the doors and start the rental.
- [G7] Allow the system to charge the user a fare for the service required.
- [G8] Allow cars to automatically lock themselves.
- [G9] Allow the system to notify an operator to request an operation.
- [G10] Allow operators to login to the system.

## 1.5 Actors Identification

We believe that PowerEnJoy can be used by a wide range of people that want to access the system for benefit as needed. Our scope is to create an efficient, intuitive and easy-to-use system that makes all users satisfied and eager to use PowerEnJoy.

- Guest: a person who has not signed up yet. All guests can only view the login page and proceed with a new registration in order to be able to access to all the functionalities of the service.
- User: a person already registered in the system. Once logged in successfully, only users are allowed to use the service. A typical user is a person who wants to move around easily in a social and eco-friendly way.
- Operator: a PowerEnJoy employee assigned to manage the charging process in the special safe area and assigned to relocate the cars that do not have enough battery to reach the nearest power grid station.
- Customer Service Assistance: an external system that provides assistance for all user's request or notification. It can be reached through a call by the users, and it is responsible for contacting the Maintenance

Management System when a car is faulty, damaged and/or needs maintenance.

- Payment Gateway: an external system that allows and enables payment by the users.
- Maintenance Management System: an external system that manages all the problems related to faulty or broken cars received by the Customer Service.

## 1.6 Stakeholders Identification

- Customers: the purpose of the customer is to maximize the performances of the system and achieve the highest profit possible.  
In this project, the main stakeholder is our professor. He expects us to develop a digital management system, PowerEnjoy, which provides the functionalities normally provided by car-sharing services.  
The role of the professor is to evaluate our ability and level of comprehension of the subject.
- Producers: the project is developed and produced by us. The objective of this project is to apply in practice what we learn during lectures with the purpose of becoming confident with software engineering practices and able to address new software engineering issues in a rigorous way and as accurate as possible.

## 1.7 Documentation

We will release the following documents in order to organize our work in each phase of the development process and keep in touch with the stakeholders. Our project includes five assignments:

- RASD, the preparation of a Requirement Analysis and Specification Document, which defines our goals and assumptions and contains an overall description of the system (using scenarios and use-cases) and the models describing requirements and specifications.
- DD, the definition of the Design Document, which contains a functional description of the system using models such as UML diagrams.
- Testing report, which contains the results of the testing activity performed on another system.
- Assessment of the effort and cost required for the development of the project
- Code inspection and bug identification activity on an existing well-known open source project.

We will try to develop our project as close as possible to a real application that is ready to be launched in the market.

## **2 Overall Description**

In this chapter, the product and its requirements are described in order to provide a background for the "Requirements specification" part and make it easier to understand.

The system we are going to release will allow the user to directly interact with it through three different channels:

- Web application: it is strongly cross-platform and therefore accessible from any device with a web searching browser.
- Mobile App: it can be downloaded from any smartphone or tablet, in order to guarantee flexibility, portability and easiness of use.
- On-Board computer: a device already installed inside on any PowerEnJoy car. It should offer a user-friendly, intuitive interface.

### **2.1 Product perspective**

The application we will release is a web and mobile application which is not integrated with other existing system. The application will not be only user based, but will have an internal interface for management of its operators. Here, we will focus our attention on the description of the characteristic that will be guaranteed to the users.

### **2.2 Product functions**

The system that we are to develop should let guests register and then login in order to manage their accounts and access the service. The provided functionalities are clearly discussed in section 1.3.

### **2.3 Users characteristics**

The user that we expect will use our application is a person who wants to move around easily in a social and eco-friendly way. The service should target natural people in possess of a driving license which is valid in the country where he/she makes use of the service, and who provide an authorized and secure credit card payment method. In addition, they must have read and accepted the general legal conditions of contract.

Finally, users must be able to have access to a stable internet connection from a mobile device and a stationary one such as a laptop or a desktop PC.

### **2.4 Constraints**

#### **2.4.1 Regulatory policies**

Users must provide express consent to the geo-location of cars, to the

detection of data regarding car circulation and to the processing of personal data according to a privacy policy: data such as credentials, transactions and locations are stored and processed in order to provide a higher service quality.

#### **2.4.2 Hardware limitations**

We will not discuss the detail of the hardware of the on-board computer, as we imagine it will be already installed in any PowerEnJoy car. We suppose that they require self-diagnostic and that they should be able to allow communications with the Costumer Service.

The web application requires any device with minimum hardware requirements, such as a stable internet connection and a modern web browser.

The mobile App has to integrate information from the GPS, the Internet receptor and the Wi-Fi modules, in order to detect right locations and query the server.

#### **2.4.3 Software limitations**

The web application should support devices running any modern browser, whereas the mobile App should be developed in order to be compatible with iOs, Android and Windows phone operating system.

#### **2.4.4 Parallel operation**

The system must support heavy parallel operations, because we suppose that many different users will access the service potentially at the same time.

### **2.5 Assumptions**

- [A1] A previous version of the system does not exist.
- [A2] All cars are equipped with the same features.
- [A3] The number of passengers is not greater than the maximum capacity of the car.
- [A4] Sensors and devices are already installed on vehicles and provide correct and reliable data.
- [A5] All cars have GPS on.
- [A6] The GPS signal always gives the correct geo-localization.
- [A7] The set of safe areas is pre-defined by the management system.
- [A8] Only the cars parked in a safe area can be reserved.
- [A9] The on-board computer has an integrated map with all useful information for the user.
- [A10] Users should be registered in the system.
- [A11] Users use the car within certain boundaries.
- [A12] The user who unlocks the reserved car always enters the car.
- [A13] The user who unlocks the car is the same user who has made the reservation.

- [A14] After the registration is completed successfully, the user's information is saved in the system's database.
- [A15] Any fine that the company receives is paid by the company but then charged to the responsible user.
- [A16] Users always pay the charged amount directly for each ride.
- [A17] We assume that we have a fixed price per kilometre.
- [A18] Power grid stations are always working correctly.
- [A19] Power grid stations can exclusively be used by PowerEnjoy cars.
- [A20] Whenever a car is tagged "Out-Of-Service" and the systems sends a notification, it is always fixed by the external Maintenance System before becoming available again.
- [A21] The Maintenance System can mark "Out-Of-Service" cars as "Available" again after each maintenance intervention.
- [A22] If an operator has accepted a request, he/she will complete it successfully.

## 2.6 Future extensions

The system implementation will open the possibility of including new features, for example including an own Customer Service Care that can interact with a user. This service can request and see all user details, their rents and payment history and in general can provide direct and personal assistance to users, receiving active feedback from them.

## 2.7 The World and the Machine

For a first domain analysis of PowerEnjoy application we use “The World & Machine” model by M. Jackson & P. Zave. This approach let us identify the entities inside the domain that interact with the application (“The World”), entities to be developed (“The Machine”) and the intersection (“Shared Phenomena”) between the world and the application, that are all world informations known or managed directly by the application.

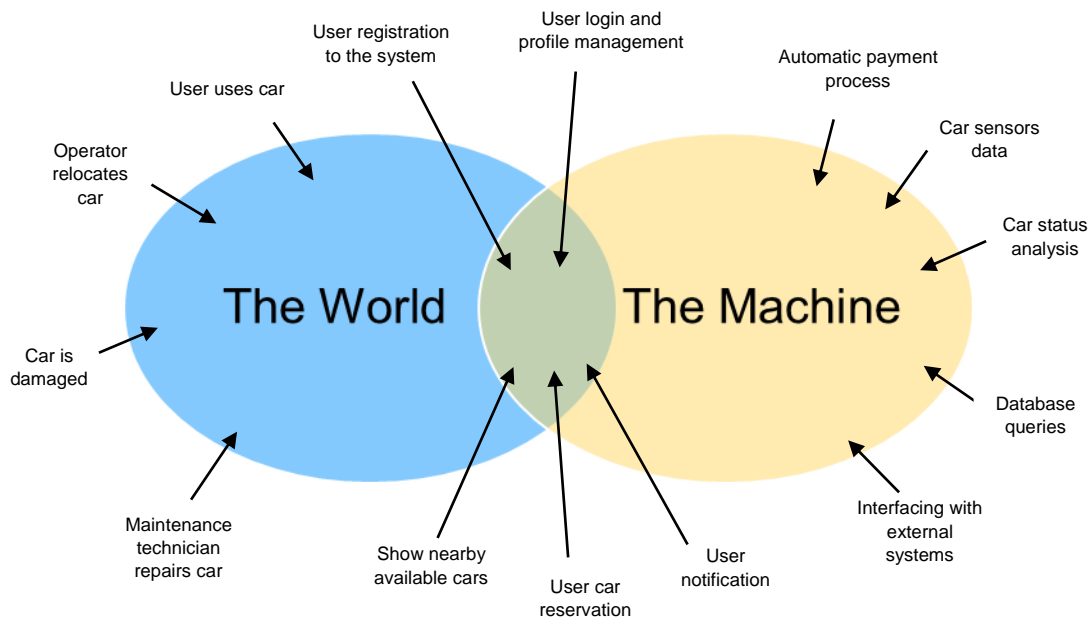


Figure 2.1: The World and the Machine model for the main functionalities to be provided by the system.

## **Part II**

# **Requirements specification**

## 3 Specific Requirements

All the software requirements are specified to a level of detail, which is sufficient to let the designers satisfy them. Both functional and non-functional requirements are mentioned.

### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces

As stated in section 2, the application's UI must be extremely user-friendly and functionally equivalent across devices.

To achieve this, the following constraints have to be satisfied by the web and mobile applications:

- The first page must always ask the user to login or register to the service;
- After the login, the system must load a user home page;
- The user's home page must offer a quick and simple overview of the main system functionalities (car reservation, show personal information area,...);
- The "Car reservation" page must show a map with a search box and a range option, in order to find all nearby available cars with respect to the specified inputs;
- The interface must offer the possibility to choose between a set of different languages;
- The user interface must dynamically adapt to the screen size;
- In order to enhance the user experience, the web application and the mobile one must use the same graphical elements.

Here some mock-ups are presented that represent an idea of the structure of the application pages, regarding the web and mobile UI.

##### 3.1.1.1 Login

The mock-up below shows the simple login page of PowerEnJoy. Here users can log in to the application and guests can access to the registration form.



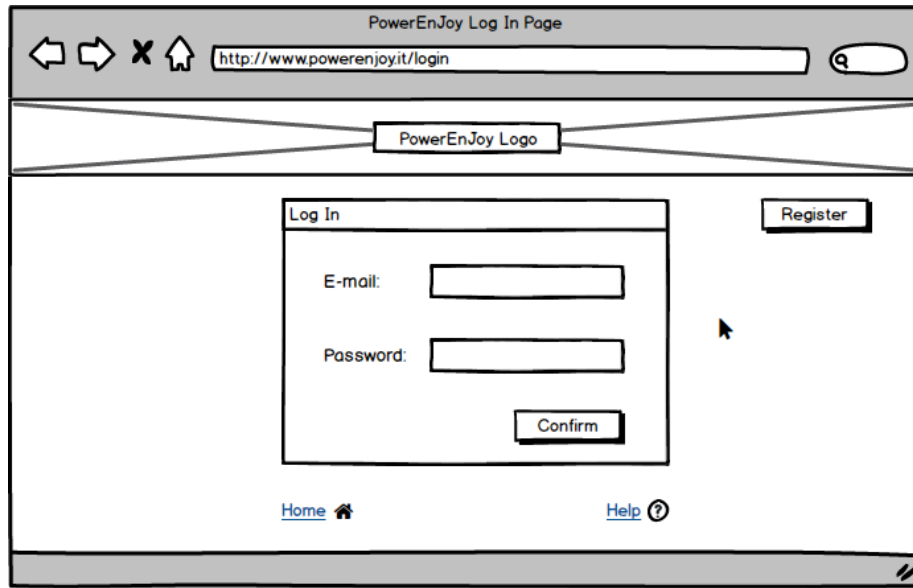


Figure 3.1: Mock-up of the web "Log In" page.

### 3.1.1.2 Car localization

This mock-up shows how the user is able to browse the available cars nearby his GPS or his provided address.



Figure 3.2: Mock-up for the localization of available cars in the mobile App.

### 3.1.2 API interfaces

For the localization part of the PowerEnJoy application, we use Google Geolocation API. As well described on the website, applications that want to perform geolocation must support the W3C Geolocation standard. For more information see the website:

<https://developers.google.com/maps/documentation/javascript/geolocation?hl=it/>.

### 3.1.3 Hardware Interfaces

This project does not support any hardware interfaces.

### 3.1.4 Software Interfaces

- Database Management System (DBMS):
  - Name: MySQL.
  - Version: 5.7.16
  - Source: <http://www.mysql.com/>
- Java Virtual Machine (JVM).
  - Name: JEE
  - Version: 7
  - Source: <http://www.oracle.com/technetwork/java/javasee/tech/index.html>
- Application server:
  - Name: Glassfish.
  - Version: 4.1.1
  - Source: <https://glassfish.java.net/>
- Operating System (OS).
  - Application must be able to run on any SO which supports JVM and DBMS specified before.

### 3.1.5 Communication interfaces

The clients communicate with the server via HTTPS protocol by using TCP and default port 443. Moreover, a safe and stable connection is also established between the server and the payment handler whenever the user performs a payment.

## 3.2 Functional requirements

Looking at the goals of the project described in section 1.4, we can derive the functional requirements that PowerEnJoy has to implement.

### **3.2.1 [G1] Allow guests to register into the system.**

- [R1] Guest must not be already registered into the system.
- [R2] The license and the payment information provided in the registration form must be valid.
- [R3] Email address used for registration must be formally correct.
- [R4] User cannot sign up more than once.
- [R5] Guest can just see login page.
- [R6] Guest can only access to registration form.
- [R7] The system shall give back a unique password that must be used to log-in.
- [D1] After the registration is completed successfully, the user's credentials are saved in the system's database.

### **3.2.2 [G2] Allow users to log in to the system and manage their accounts.**

- [R1] Email and password inserted during login process must be correct.
- [R2] Wrong credentials will not grant access to the service.
- [R3] Guests cannot access to the service before registration.
- [D1] User must be registered in the system.

### **3.2.3 [G3] Allow user to view the map of the closest available cars to where he is currently located or another indicated address.**

- [R1] User must be logged in the application.
- [R2] User must provide an address (either manually or with the GPS) and a searching range.
- [R3] The system shall provide the location of all the available cars around the selected range.
- [R4] The system shall provide the information about the cars.
- [R5] The system shall implement a map of the world considered.
- [R6] A best-optimized distribution of cars should be guaranteed.
- [D1] All cars are equipped with the same features.
- [D2] All cars have GPS on.
- [D3] The GPS signal always gives the correct geo-localization.

### **3.2.4 [G4] Allow users to reserve a single car for up to one hour before they pick it up.**

- [R1] User must be already registered and logged in the application.
- [R2] The system shall know the state of cars.
- [R3] The system shall know if a user has reserved a car.
- [R4] The system shall change the state of a car, when a car is being reserved.
- [R5] The system shall know the time when a user performs a reservation.
- [R6] The system shall tag a car available again if a car is not picked-up within an hour from the reservation.
- [R7] The system should notify the user with details of the booked car.
- [R8] User should perform real-time car reservation.
- [D1] User can reserve a single car per session.
- [D2] Reservation time is included between 00.00 and 23.59.

### **3.2.5 [G5] Allow users to delete an existing reservation within one hour.**

- [R1] User must be already registered and logged in the application.
- [R2] The system shall know if a user chooses to delete a reservation.
- [R3] The system shall know the time when a user has done a reservation.

- [R4] User must confirm deleting process.
- [R5] Deleting process is not reversible, previous reservation data will be lost.
- [R4] The system shall change the state of a car, when a car is being unreserved.
- [D1] Users can delete only an existing reservation.

### **3.2.6 [G6] Allow users to unlock the doors and start the rental.**

- [R1] The system shall unlock a car when user, who has reserved that specific car, is nearby.
- [R2] User shall notify the system he is nearby.
- [D1] The user who unlocks the car is the same user who has made the reservation.
- [D2] The user who unlocks the reserved car always enters the car.

### **3.2.7 [G7] Allow the system to charge the user a fare for the service required.**

- [R1] The system shall charge the user at the end of the ride using the payment information provided during the registration.
- [R2] The system shall start calculating the charge as soon as the engine ignites.
- [R3] The system shall stop calculating the charge as soon as the car is parked in a safe area and the user exits the car.
- [R4] The system should apply a discount of 10% on the ride if it detects that the user took at least two other passengers onto the car.
- [R5] The system should apply a discount of 20% on the ride if the car is left with no more than 50% of the battery empty.
- [R6] The system should apply a discount of 30% on the last ride if the car is left at a special parking area where it can be recharged and the user takes care of plugging the car into the power grid.
- [R7] The system should charge 30% more in the last ride if a car is left at more than 3 Km from the nearest power grid station or with more than 80% of the battery empty.
- [R8] The system shall apply a fee of 1 EUR if a car is not picked-up within an hour from the reservation.
- [R9] The system shall apply discounts up to 5 minutes from the end of the ride.
- [D1] We assume that we have a fixed price per kilometre.
- [D2] The set of the safe areas is pre-defined by management system.
- [D3] The number of passengers cannot be greater than the maximum capacity of the car.
- [D4] The system can check the number of passengers thanks to sensors already installed on the cars.
- [D5] Any fine that the company receives is paid by the company but then charged to the responsible user.
- [D6] Users always pay the charged amount directly for each ride.

### **3.2.8 [G8] Allow cars to automatically lock themselves.**

- [R1] The system shall lock the car when a user has parked the car in a safe area and exits the car.
- [D1] The set of the safe area is pre-defined by management system.

### **3.2.9 [G9] Allow the system to notify an operator to request an operation.**

- [R1] The system shall check if a car is parked outside the special safe area.
- [R2] The system can contact an operator.

- [R3] The system shall check if a car needs to be relocated.
- [D1] The GPS signal always gives the correct geo-localization.
- [D2] The set of the safe areas is pre-defined by the management system.
- [D3] If an operator has accepted a request, he/she will complete it successfully.
- [D4] Power grid stations are always working correctly.

**3.2.10 [G10] Allow operators to login to the system.**

- [R1] Email and password inserted during login process must be correct.
- [R2] Wrong credentials will not grant access to the service.
- [D1] Operators are already registered in the system.

### **3.4 Scenarios**

#### **3.4.1 Scenario 1**

Laura wants to go to Duomo so she decides to use the PowerEnjoy Service but she is not registered. Laura accesses the homepage and signs up filling in all the mandatory fields (Name, Surname, Birthday, Email, Phone Number, License and Credit Card). After that, she receives an email with a new password generated by the system. Now she can log in with her email and password and check the closest available cars using her GPS. Laura selects one car at 1 kilometre. When Laura arrives, she notifies the system to unlock the car, then she drives to Duomo and parks the car in a safe area. The on-board computer displays her that she has to pay 4,50 EUR. When Laura leaves the car, she automatically pays with the credit card's credential that were inserted in the registration module.

#### **3.4.2 Scenario 2**

Marco, Chiara and Emanuele study at Politecnico di Milano and one morning they decide to use the PowerEnjoy Service instead of taking the subway. When Marco tries to find the closest car, he notices that his phone's GPS doesn't work, so he must insert the address manually. He finds two cars in a 500 metres range so he checks the conditions of both. One of them has less than 30% of battery so he decides to reserve the other one. When Marco, Chiara and Emanuele get in the car, the sensors detect the presence of two passengers, so Marco will have a discount of 10%. At the end of the ride Marco pays 2,70 EUR instead of 3,00 EUR.

#### **3.4.3 Scenario 3**

Paolo is invited by his friend at dinner. After logging in, he reserves a car. 5 minutes later his friend calls him saying that he is sick and won't be hosting the party, so Paolo has to delete the reservation. Since the hour hasn't expired yet, he doesn't pay any fee.

#### **3.4.4 Scenario 4**

Giulia and his boyfriend want to go to the theatre. Giulia books a car but when she parks it, one light bulb goes off. Giulia notifies the Customer Service that promptly changes the availability of the car to "Out of Service" and handles its reparation to the Maintenance System. Giulia also leaves the car with more than 50% of battery so she gets a discount of 20% on the ride. She pays 1,36 EUR instead of 1,70 EUR.

#### **3.4.5 Scenario 5**

Elisa is invited at a barbeque. She reserves a car at 3 km from her. As she is going out, it starts raining so the barbeque is cancelled. Since she forgets to delete the reservation, the system applies a fee of 1 EUR to Elisa because she hasn't picked the car up within one hour

### 3.4.6 Scenario 6

Mario wants to visit Brera's Pinacoteca. He finds more than one car in a 300 metres range, he books it but 30 minutes later he searches again and finds a car closer to him and so he wants to book this one. Mario deletes the first reservation and creates a new one selecting that closer car. At the end of the ride, he parks in a special safe area and he plugs the car into the power grid. The system applies a discount of 30% because he plugged it in in less than five minutes after exiting the car. He pays 3,50 EUR instead of 5,00 EUR.

## 3.5 UML Models

### 3.5.1 Use Case Diagram

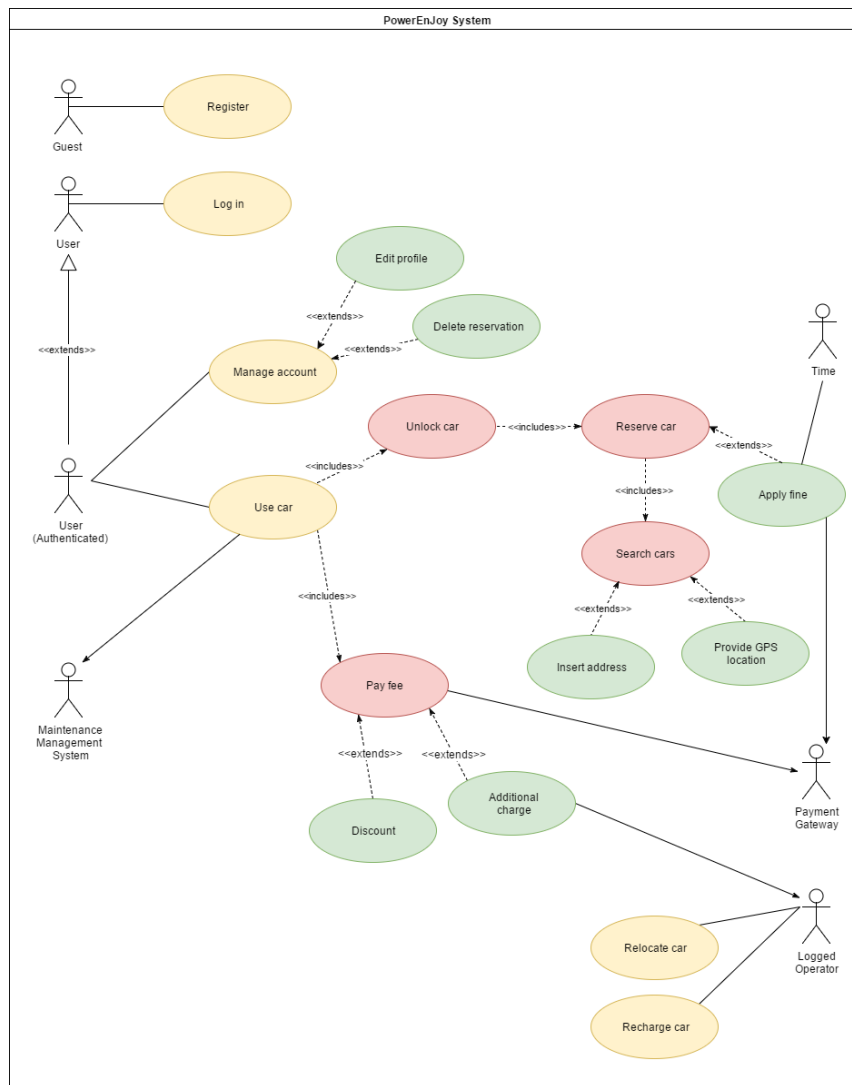


Figure 3.3: complete Use Case Diagram of all the functionalities provided by the system. Note that all directed arrows from the system functions towards actors imply a notification.

### 3.5.1.1 [G1] Allow guests to register into the system.

Actor	Guest
Input Condition	NULL
Flow Events	<ol style="list-style-type: none"><li>1. Guest on the homepage clicks on "Sign Up" button to start the registration.</li><li>2. The registration form is loaded and guest fills in all the mandatory fields.</li><li>3. Guest clicks on "Confirm" button.</li><li>4. The system loads the user's homepage.</li></ol>
Output Condition	Guest successfully becomes a User. He/she receives an email with his/her password. From now on, he/she can log in to the application using email and password. The system saves the user in the DB.
Exception	<ol style="list-style-type: none"><li>1. Guest is already a User.</li><li>2. One or more mandatory fields are not valid or incomplete.</li><li>3. Email chosen is already associated to another user.</li></ol> <p>All exception are handled alerting the guest of the problem and application goes back to point 2 of EventFlow.</p>



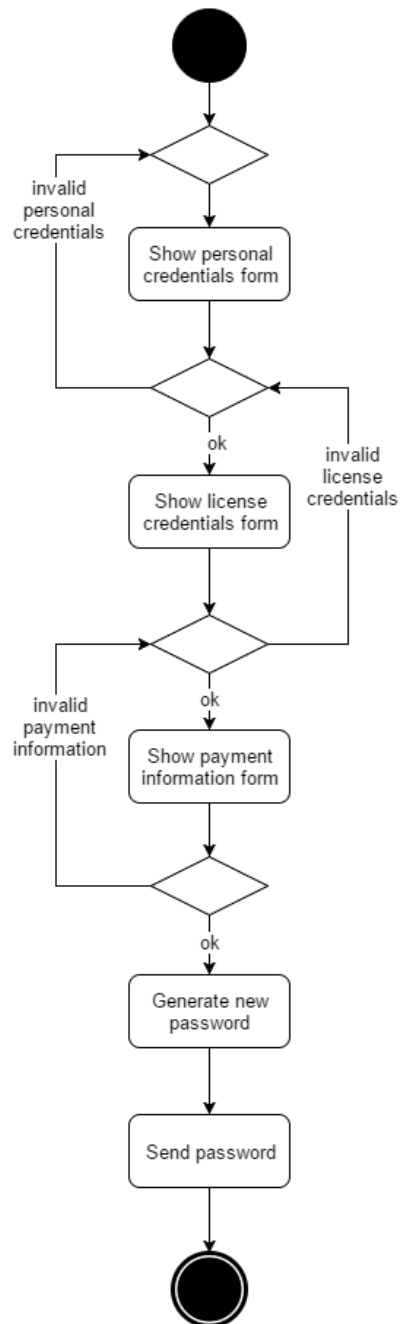


Figure 3.4: Activity diagram of the registration process.

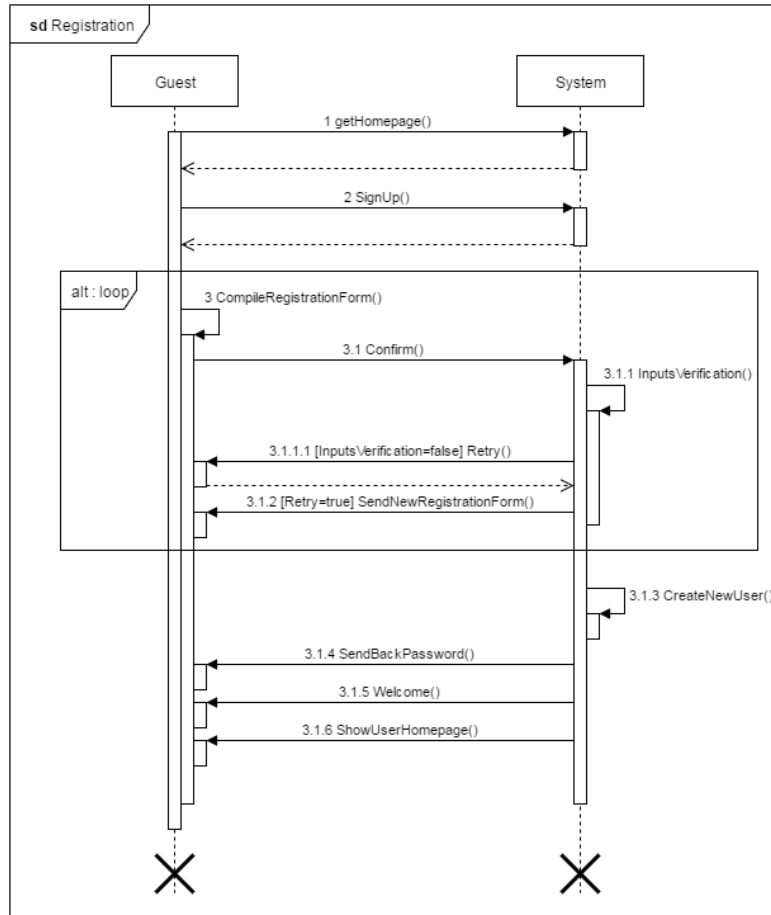


Figure 3.5: *Registration* sequence diagram.

### 3.5.1.2 [G2] Allow users to log into the system and manage their accounts.

Actor	User
Input Condition	User is already registered into the System and wants to log in.
Flow Events	<ol style="list-style-type: none"> <li>1. User accesses the login page.</li> <li>2. User inserts his/her email and password.</li> <li>3. User clicks on "Log In".</li> <li>4. User enters his/her personal information area.</li> <li>5. The system loads the page.</li> </ol>

Output Condition	PowerEnJoy shows the user's main page (or his/her personal area if selected). The user is able to perform a reservation, or check his/her profile, the pending reservation, the rides history as well as the completed payments.
Exception	<ol style="list-style-type: none"> <li>1. User's email and/or password are incorrect. User is redirected to 1.</li> <li>2. User forgets to fill in both the fields. (Proceed as written above).</li> <li>3. In the user's area there are no pending reservation or rides history.</li> </ol>

[Goal 2 - continue]

Actor	User
Input Condition	The logged user wants to edit his/her profile.
Flow Events	<ol style="list-style-type: none"> <li>1. User enters his/her personal area.</li> <li>2. User selects "Edit Profile".</li> <li>3. The system loads the "Edit Profile" page.</li> <li>4. User enters the new information.</li> <li>5. User confirms the desired changes.</li> </ol>
Output Condition	The system updates the user's personal information.
Exception	If the new information are invalid, the system does not save the modification and reloads the "Edit Profile" page.

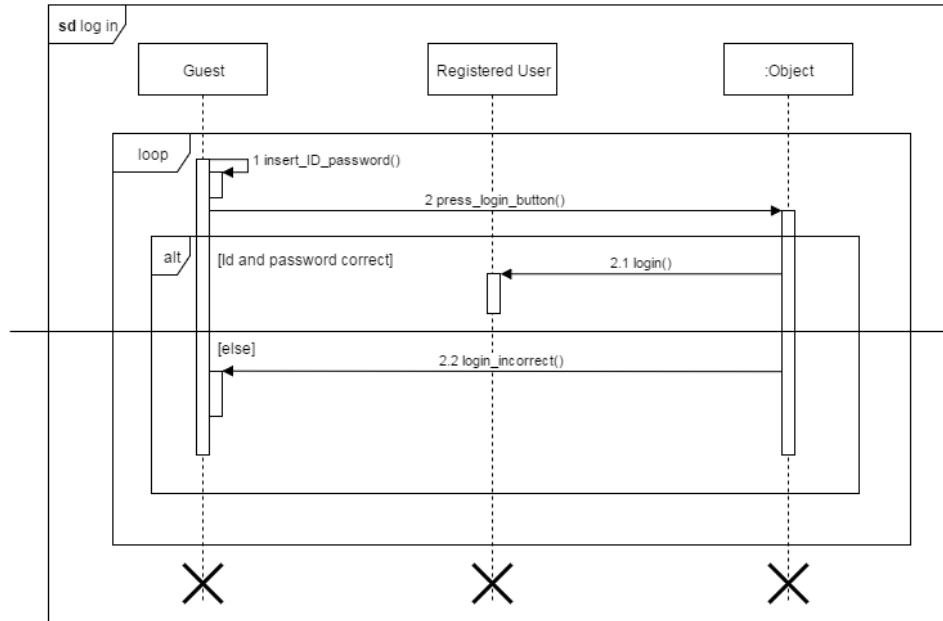


Figure 3.6: *Login* sequence diagram.

**3.5.1.3 [G3] Allow users to view the map of the closest available cars to where he is currently located or another indicated address.**

Actor	User
Input Condition	User is already logged in into the system.
Flow Events	<ol style="list-style-type: none"> <li>1. User decide if he/she wants to use his/her GPS position or another address.</li> <li>2. Choose the range in which he/she wants to search the car.</li> </ol>
Output Condition	PowerEnJoy shows all the available cars in the chosen range with their details. User are able to select one car and start the reservation.
Exception	<ol style="list-style-type: none"> <li>1. User's GPS does not work properly.</li> <li>2. The inserted address does not exist.</li> </ol>

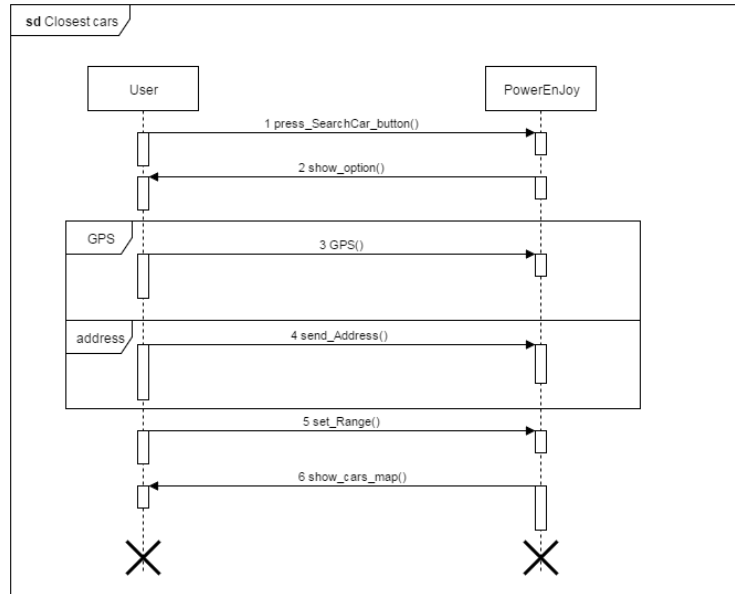


Figure 3.7: *Search available cars* sequence diagram.

#### 3.5.1.4 [G4] Allow users to reserve a single car for up to one hour before they pick it up.

Actor	User
Input Condition	User is already searching a car.
Flow Events	<ol style="list-style-type: none"> <li>1. User clicks on a car image on the screen to see its details.</li> <li>2. Click on "Confirm" to perform the reservation.</li> </ol>
Output Condition	The car status becomes "Reserved". User sees the timer before the reservation expires. User is able to notify he/she is nearby to unlock the car. User can delete the reservation until one hour.
Exception	While the user is browsing the available cars, one of them gets reserved by another user.

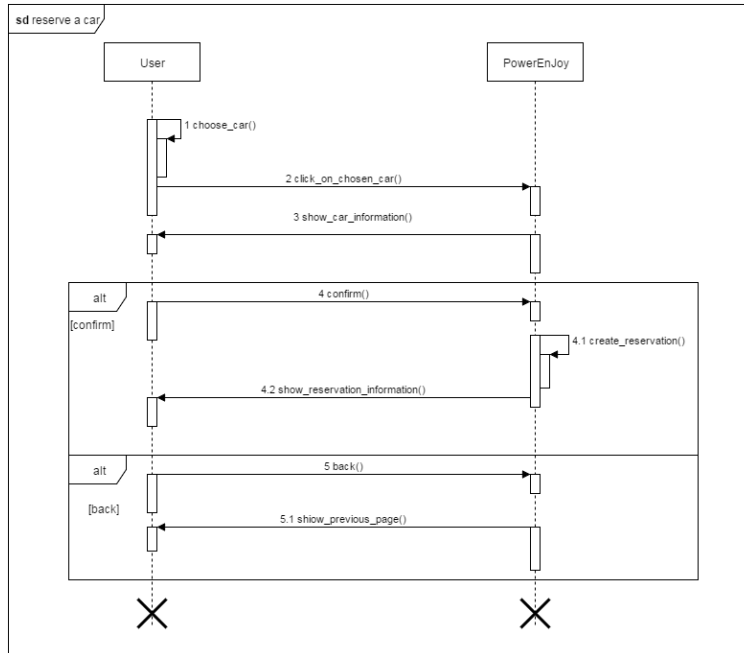


Figure 3.8: *Reserve a car* sequence diagram.

### 3.5.1.5 [G5] Allow users to delete an existing reservation within one hour.

Actor	User
Input Condition	User has previously reserved a car and wants to delete the reservation.
Flow Events	<ol style="list-style-type: none"> <li>1. User accesses the PowerEnjoy homepage.</li> <li>2. User logs in into the system.</li> <li>3. User selects the reservation that he wants to delete.</li> <li>4. User confirms the reservation's cancellation.</li> <li>5. The system processes his/her request.</li> <li>6. The system updates the car status.</li> <li>7. The system loads the user's homepage.</li> </ol>
Output Condition	User is in his/her homepage and the reservation is deleted. The car status becomes "Available".
Exception	While the user is deleting the reservation, the reservation itself expires.

### 3.5.1.6 [G6] Allow users to unlock the doors and start the rental.

Actor	User
Input Condition	User has already confirmed the reservation.
Flow Events	<ol style="list-style-type: none"> <li>1. User clicks on "Unlock Car" to notify he/she is nearby.</li> <li>2. System unlocks the doors of the car.</li> </ol>
Output Condition	The car is open. User is able to entry the car and start the rental.
Exception	The timer marks one hour and the reservation expires before the user is able to tell the system he/she is nearby.

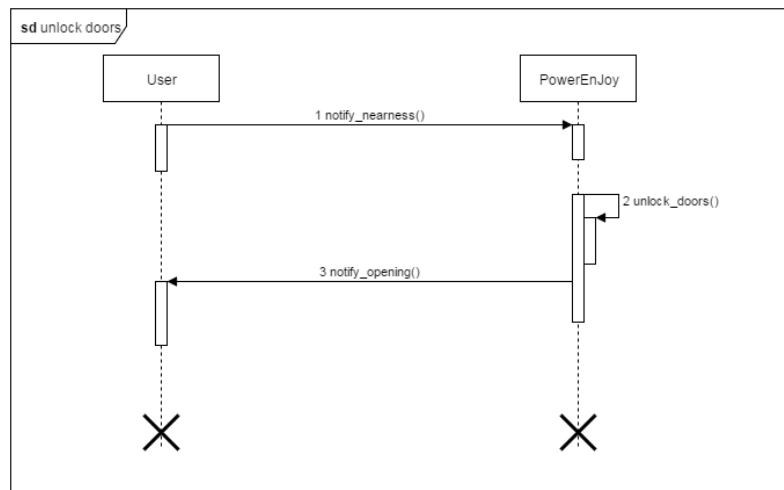


Figure 3.9: *Unlock doors* sequence diagram.

### 3.5.1.7 [G7] Allow the system to charge the user a fare for the service required and to display it on the car.

Actor	User
Input Condition	User is already in the car.
Flow Events	<ol style="list-style-type: none"> <li>1. User ignites the engine.</li> <li>2. The status of the car becomes "In Use".</li> <li>3. The system calculates the number of passengers.</li> <li>4. The system start calculating the charge.</li> <li>5. User drives the car.</li> </ol>

	<ol style="list-style-type: none"> <li>6. User parks in a safe area.</li> <li>7. User turns the engine off.</li> <li>8. The systems checks the level of the battery.</li> <li>9. User exits the car.</li> <li>10. The system checks if the car is located at more than 3 Km from the nearest power grid station. Otherwise, if the system detects that the car is located in a special safe are, it waits 5 minutes and checks if the car is plugged into the power grid.</li> <li>11. The system calculates the total fare, applying all the discounts and the additional charges.</li> <li>12. The amount of the fare is displayed on the user's app.</li> </ol>
Output Condition	User automatically pays the fare to the system with his/her payment information.
Exception	<ol style="list-style-type: none"> <li>1. If the user ends the ride outside a safe area, we assume that an additional charge will be applied. In this particular case, the status of the car will be immediately switched to "Out Of Service". The system will notify an operator to relocate the car in the nearest safe area.</li> <li>2. User cannot get cumulative discounts. In particular, only the greater one will be applied.</li> <li>3. If a user has to pay an additional fee, no other discounts will be applied.</li> </ol>



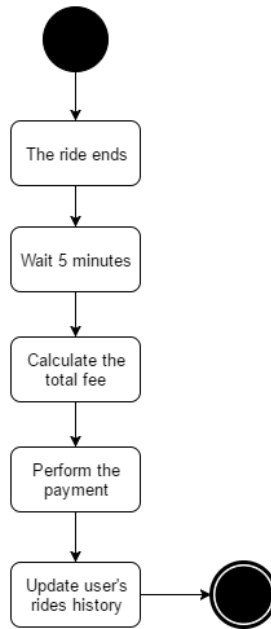


Figure 3.10: *Compute fare* activity diagram.

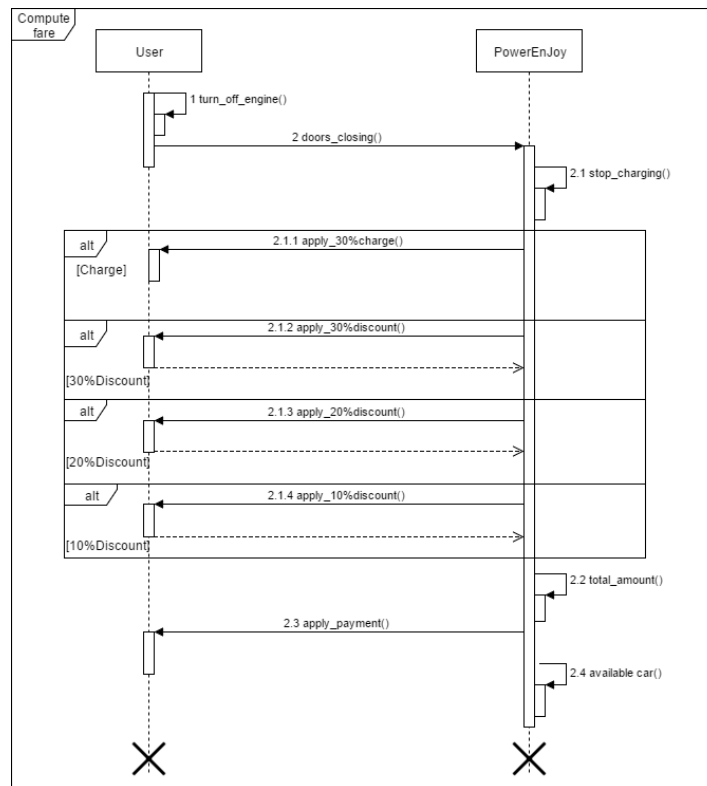


Figure 3.11: *Compute fare* sequence diagram.

**3.5.1.8 [G10] Allow the system to notify an operator to request an operation.**

Actor	Operator
Input Condition	A car is too far from a power grid station or it has a low battery level.
Flow Events	<ol style="list-style-type: none"><li>1. Car status is set to “Out Of Service”.</li><li>2. System displays the car and notifies all operators.</li><li>3. Operator selects the required operation.</li><li>4. The selected car is removed in the list of required operations.</li><li>5. Operator handles the relocation of the car or the charging process.</li></ol>
Output Condition	The car is set “Available” again.
Exception	NULL

### 3.5.2 Class Diagrams

Here is presented the complete class diagram. This diagram will be updated during the developing process especially by adding all needed methods.

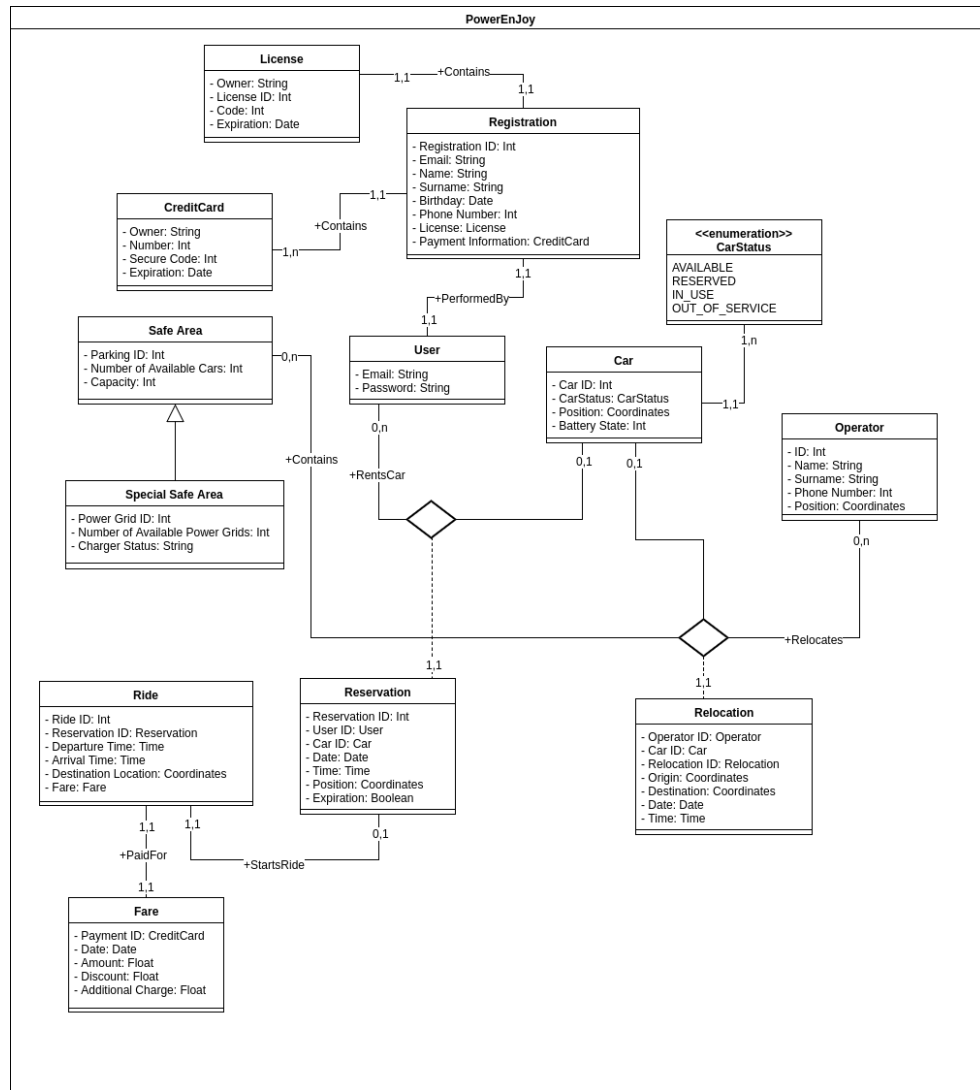


Figure 3.12: complete Class Diagram.

### 3.5.3 State Machine Diagram

The following State Machine Diagram will give a vision of the management of the status of cars.

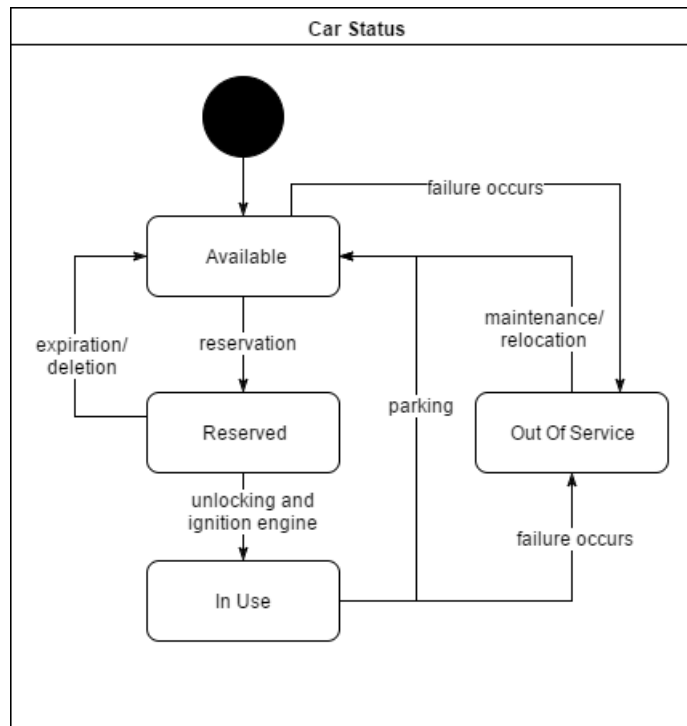


Figure 3.13: *Car status* state chart.

## 3.6 Non Functional Requirements

### 3.6.1 Performance Requirements

Performance must be acceptable to guarantee a good grade of usability. The user interface has to be user-friendly. It has to be simple, intuitive and well organized. A typical guest has not a previous knowledge of the system and we shall do it as easy to use as possible.

### 3.6.2 Software System Attributes

#### 3.6.2.1 Reliability

The software product does not have any reliability factors because in case of malfunction it will cause minor inconveniences.

#### 3.6.2.2 Availability

The system should be available 24/7. The application should be accessible online anytime, providing that users have a stable internet connection. High availability is required since there are monetary issues related to the systems availability. The system's high availability will also ensure users' satisfaction and loyalty.

#### **3.6.2.3 Portability**

Our system should be very portable because of the variety of our target users. The web application should support devices running any modern browser, whereas the mobile App should be developed in order to be compatible with iOS, Android and Windows phone operating system. The application should be developed in order to allow updates that will always guarantee compatibility with the latest software versions.

The application can be installed in any operation system that supports the JVM and its dependent components.

#### **3.6.2.4 Security**

We need to apply security protocols to ensure a correct access to the data. Every access to the database has to be filtered in order to avoid an incorrect management and the inconsistency of the data. The system must provide secure storage of the email, password and payment information of its users. This can be achieved by using any cryptographic techniques.

The connection with the system has to respect the https protocol, which ensures the quality and the privacy of the connection.

For internet access, the following requirements are mandatory:

- Confidentiality: sensitive data must be encrypted if any (credit card payments).
- Safety: credit card data must not be kept at a local database.
- Data integrity: data sent across the network cannot be modified by a tier.
- Auditing: every sensitive action can be logged.
- Non-repudiation : gives evidence a specific action occurred.

#### **3.6.2.5 Maintainability**

The database must be backed up periodically, so that new registration and its mandatory data are not lost in case of malfunction.

#### **3.6.2.6 Persistence**

Data persistence will be addressed using a relational database and J2EE s Object Relational Mapping capability will be reused.

## 4 Appendix

### 4.1 Alloy

The complete alloy file (.als) could be find on our git repository. The following alloy model presented is created using the class diagram. We try to divide the code dividing signature from fact, assert and predicates. In the last part, there are the generated word.

#### 4.1.1 Data Type

This is the definition of data type.

open util/boolean

//----DATA-----//

```
sig System{
    cars: some Car,
    registeredUser: some User,
    reservation: some Reservation,
    safeArea: some SafeArea,
    ride: some Ride
}
{
    cars.status = Available
}
```

```
sig Car {
    id: Int,
    status: one CarStatus,
    position: one Coordinates,
    battery: Int
}
{
    battery >=0 and battery<=7
}
```

```
sig Coordinates{
    latitude: Int,
    longitude: Int
}
```

```
sig Email, Password{}
```

```

sig User {
    registration: one Registration,
    password: one Password,
}

sig Date, Time, License, CreditCard{}

sig Registration {
    birthday: one Date,
    email: one Email,
    license: one License,
    paymentInformation: one CreditCard
}

sig SafeArea{
    availableCars: set Car,
    position: one Coordinates
}

sig PowerGrid{
    available: one Bool
}

sig SpecialSafeArea extends SafeArea{
    capacity: set PowerGrid,
    chargingCars: set Car
}
{
    #capacity>0 and #chargingCars<=#capacity
}

sig Reservation {
    car: one Car,
    user: one User
}
{
    car.status=Reserved
}

sig Fare {
    price: one Int,
    payment: one CreditCard
}
{
    price>=0
}

sig Ride {
    reservation: one Reservation,
    fare: one Fare,
    position: one Coordinates
}

```

#### 4.1.2 Abstract Entity

This is the definition of abstract entity.

```
//----ABSTRACT-DATA-----//

abstract sig CarStatus{}

sig Available extends CarStatus{}

sig Reserved extends CarStatus{}

sig InUse extends CarStatus{}

sig OutOfService extends CarStatus{}
```

#### 4.1.3 Fact

This is the fact part that defines the constraints of the class.

```
//-----FACTS-----//

fact lowBatteryCarInSafe{
    all c:System.cars | all s:SpecialSafeArea| (c.battery<4 and
c.status=Available) implies c.position=s.position
}

fact IDCarsAreUnique {
    all c1,c2: Car | (c1 != c2) => c1.id != c2.id
}

fact noDupEmail{
    no disj u1,u2: User | u1 != u2 and u1.registration.email =
u2.registration.email
}

fact noSameRegistration{
    all r1,r2: Registration | r1!=r2 implies (r1.email!=r2.email and
r1.license!=r2.license)
}

fact noUserOutOfSystem {
    all u:User | some s: System | u in s.registeredUser
}

fact availableCarsAreInSafe{
    all c: System.cars | all s:SafeArea | c.status = Available implies
c.position=s.position
}

fact positionIsConsistance{
    all c: Car | all s: SafeArea | c in s.availableCars implies c.position =
s.position
}
```



```

}

fact areaPositionAreUnique{
    all a1,a2: SafeArea | (a1 != a2) => a1.position != a2.position
}

fact noSameReservation{
    all r1,r2: Reservation | r1!=r2 implies (r1.car!=r2.car and
r1.user!=r2.user)
}

fact reservationToOneRide {
    all r1,r2: Ride | r1!=r2 implies r1.reservation!=r2.reservation
}

fact userReservedCarAlreadyUse {
    all r: Ride |
r.reservation.user.registration.paymentInformation=r.fare.payment
}

fact carPositionMove {
    all r: Ride | r.reservation.car.position= r.position
}

```

#### 4.1.4 Assert

In this last code part is presented the assert used to verify the model.

```

//----ASSERTS----//

assert noDoubleCarsInReservation{
    no disj c1,c2:Car, r:Reservation | r.car=c1 and r.car=c2 and c1=c2
}
check noDoubleCarsInReservation for 3

assert noDoubleUserInReservation{
    no disj u1,u2:User, r:Reservation | r.user=u1 and r.user=u2 and u1=u2
}
check noDoubleUserInReservation for 3

```

#### 4.1.5 Predicates

This is the predicates used with the previous assert to verify the model.

```

//---PREDICATES---//

pred show() {
    #System=1
    #Reservation>0
    #Ride>0
}
run show for 3

```

```

pred addRegistration (r: Registration, s1,s2: System){
    r not in s1.registeredUser.registration implies s2.
    registeredUser.registration=s1.registeredUser.registration+r
}
run addRegistration for 3

pred addReservation (r: Reservation, s1,s2: System){
    r not in s1.reservation implies s2.reservation=s1.reservation+r and
    r.car.status=Reserved
}
run addReservation for 3

pred deleteReservation(r: Reservation, s1,s2: System){
    r in s1.reservation implies s2.reservation=s1.reservation-r and
    r.car.status=Available
}
run deleteReservation for 4

pred addRide(r: Ride, s1,s2: System){
    (r not in s1.ride and r.reservation in s1.reservation) implies s2. ride=s1.
    ride+r
}
run addRide for 3

pred deleteRide(r: Ride, s1,s2: System){
    r in s1.ride implies s2.ride=s1.ride-r and s2.reservation=s1.reservation-
    r.reservation and r.reservation.car.status=Available
}
run deleteRide for 4

```

#### 4.1.6 Result

This screenshot of the Alloy Analyzer software that shows the consistence of the model in all part.

```

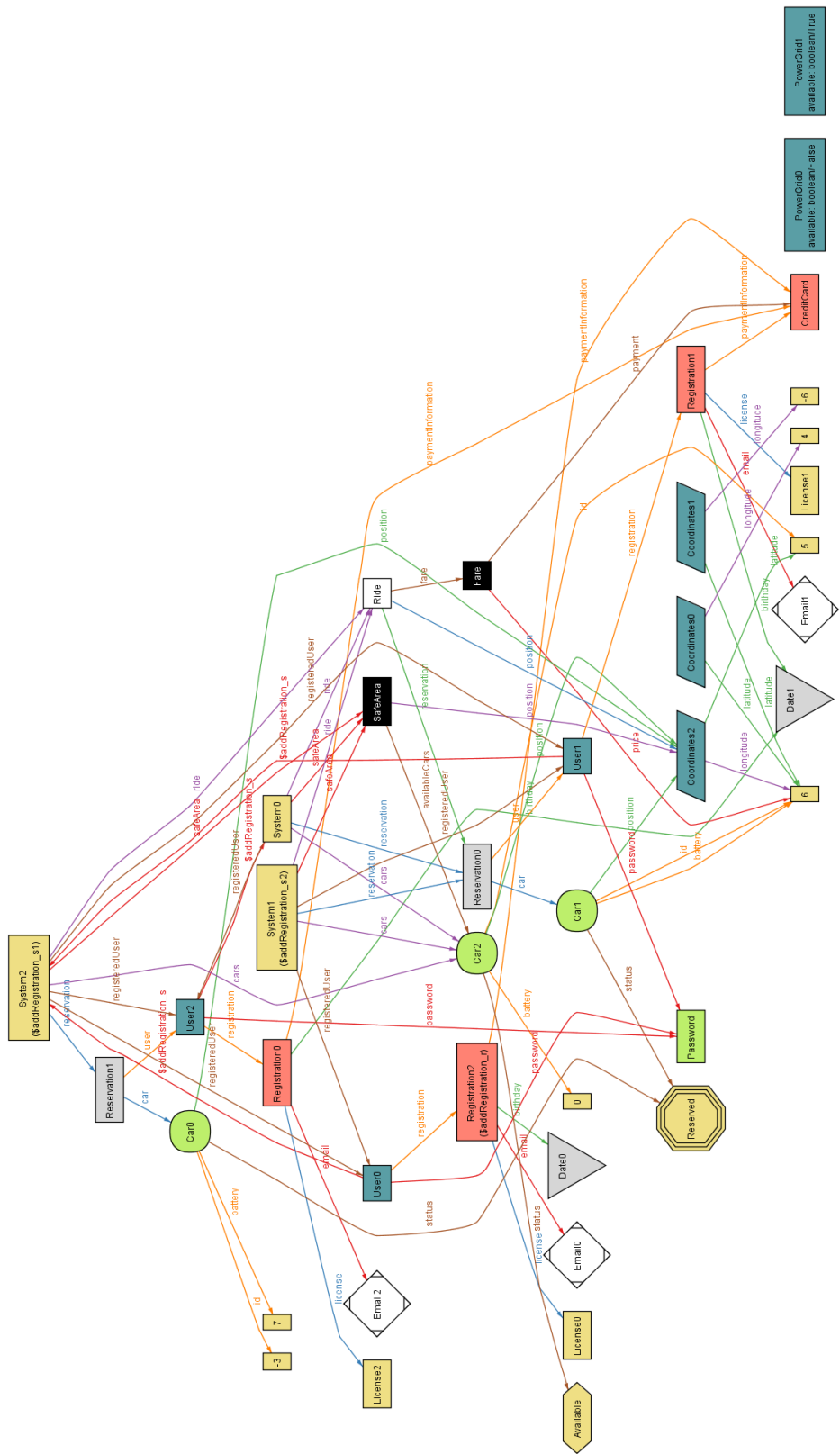
8 commands were executed. The results are:
#1: No counterexample found. noDoubleCarsInReservation may be valid.
#2: No counterexample found. noDoubleUserInReservation may be valid.
#3: Instance found. show is consistent.
#4: Instance found. addRegistration is consistent.
#5: Instance found. addReservation is consistent.
#6: Instance found. deleteReservation is consistent.
#7: Instance found. addRide is consistent.
#8: Instance found. deleteRide is consistent.

```

#### 4.1.7 Generated world

Here is presented the generated world using the alloy verification software. All diagrams are the predicate show() for 3 case. For more case the model will be more and more complex and difficult to read.





# Appendix A: Used Tools

## A.1 Microsoft Word 2013

To redact and format and redact this document.

## A.2 *git*

To submit this document in the online repository.

## A.3 Dropbox

Used as version control system in order to lead development.

## A.4 Balsamiq Mockups 3

To design web and mobile application mock-ups.

## A.5 Draw.io

To create and design Use Cases Diagrams, Sequence Diagrams, Class Diagrams and State Machine Diagrams.

## A.6 Alloy Analyzer 4.2

To prove the consistency of our model.

# Appendix B: Hours of work

This is the time spent by each group member in order to redact this document:

- Pozzati Edoardo: ~40 hours
- Stefanetti Nadia: ~40 hours
- Total work time: 80 hours

# Appendix C: Changelog

- **v1.0**
- **v2.0**
  - fixed index layout and corrected some typos
  - added “Time” as actor in the use case diagram
  - moved sequence diagrams from subsection 3.5.2 to 3.5.1
  - added activity diagrams for completeness and better explanations for the functionalities
  - changed Costumer Service Assistance as external system and removed Costumer Service and Assistance entities from the class diagram
  - improvements in class diagram: function names to better readability
  - added subsection “The World and the Machine”
  - removed goal “Allow users to contact PowerEnJoy Customer Service directly for assistance and any other notifications”
  - removed Costumer Service from use case diagram
  - added goal “Allow operators to login to the system”
  - added assumption A18, A19, A20, A21, A22
  - added subsection “Future extensions”

# Definitions

- **Cars or vehicles:** set of electric vehicles owned by the PowerEnjoy company.
- **Car status:** a tag that specifies whether a car is available, reserved, in use or out of service.
  - **Available:** it means that the car is parked in a Safe Area and is ready to be reserved by a user.
  - **Reserved:** it means that the car has been reserved by a user, and the reservation has not yet expired or been deleted.
  - **In use:** it means that the car is been driven by a user, who has ignited the engine.
  - **Out-Of-Service:** it means that the car is either broken down or is out of battery charge, and therefore it cannot be reserved.
- **Credentials:** set of information that a guest provides in the registration form (name, surname, date of birth, e-mail and phone number).
- **Fare:** the total amount of price to be charged to the user after when the ride is completed.
- **Fee:** 1€ charge for reservation's expiry.
- **Login:** the process between a user and the system that grants authenticated user's privileges.
- **Password:** a code of 8 characters provided by the system to a user after his successful registration process. It is made up randomly of at least one capital letter (A-Z), one lowercase letter (a-z) and one number (0-9). This password must be used during the user's log in.
- **Payment information:** set of information that a guest provides in the registration form about his credit card (owner, number, secure code and expiration date).
- **Pending reservation:** a reservation of a car that is not older than an hour, meaning that it is not expired yet and the user that performed it can still claim and use the reserved car.



- **Power grid station:** a component that allows users to charge the battery of the cars as needed.
- **Registration:** interaction between a guest and the system in which the former, providing his/her credentials, payment information and license, receives the generated password needed to login to the system.
- **Ride:** the main action performed by the user from when he/she enters the car and ignites the engine to when the car is parked and the user and the passengers (if any) exit the car.
- **Safe Area:** a predefined area within the Service Coverage Area owned by the car sharing company and reserved only for their cars. It marks the boundaries within which the user can park a car and conclude the ride so that the system stops charging him/her, without any additional fine.
- **Service Coverage Area:** an area with a 15 Km radius in which the service is provided.
- **Special safe area:** a safe parking area with power grid stations.
- **System:** the PowerEnJoy software system-to-be as a whole.
- **Vehicles:** see cars.

# Acronyms

- API: Application Programming Interface.
- DB: Data Base.
- DBMS: Data Base Management System.
- GPS: Global Positioning System.
- JEE: Java Enterprise Edition.
- JVM: Java Virtual Machine.
- m, km: meter, kilometer *International System of Units (SI)* units of measure.
- OS: Operating System.
- RASD: Requirements Analysis and Specification Document.
- UML: Unified Modeling Language, the modelling language used in this document.
- W3C: World Wide Web Consortium. It is the main international standards organization for theWorld WideWeb, founded and led by Tim Berners-Lee.

# Bibliography

- [1] IEEE Std 830-1998 IEEE *Recommended Practice for Software Requirements Specifications*
- [2] Luca Mottola and Elisabetta Di Nitto, *Software Engineering 2: Project goal, schedule and rules*, 2016