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

Requirements **A**nalysis and **S**pecifications **D**ocument

version 1.2

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

Table of Contents

Abstract

This document represent the Requirement Analysis and Specification Document (RASD). The main purpose of this document is to give a specification of the requirements that our system has to 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

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

The software house wants to offer a new car sharing service. 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 fee for the service required.
- [G8] Allow cars to automatically lock themselves.
- [G9] Allow users to contact PowerEnJoy Customer Service directly for assistance and any other notifications.
- [G10] Allow the system to notify an operator when needed.

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 Assistant: a PowerEnJoy employee that provides assistance for all user's request or notification. He/she 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 Costumer 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 ways:

- 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 portability and easiness of use.
- On-Board computer: a device already installed inside of any PowerEnJoy car. It must be user-friendly, providing an 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 administration and for management of its employees. 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 application should target natural people with a valid Italian driving license and who provide an authorized and secure credit card payment method. In addition, they must have read and accepted general legal conditions of contract.
Finally, users must be able to use a web browser and have access to internet.

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, payment methods, 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

- A previous version of the system does not exist.
- All cars are equipped with the same features.
- The number of passengers is not greater than the maximum capacity of the car.
- Sensors and devices are already installed on vehicles.
- All cars have GPS on.
- The GPS signal always gives the correct geo-localization.
- The set of safe areas is pre-defined by the management system.
- Only the cars parked in a safe area can be reserved.
- The on-board computer has an integrated map with all useful information for the user.
- Users should be registered in the system.
- Users use the car within certain boundaries.
- The user who unlocks the reserved car always enters the car.
- The user who unlocks the car is the same user who has made the reservation.
- After the registration is completed successfully, the user's information are saved in

the system’s database.

- Any fine that the company receives is paid by the company but then charged to the responsible user.
- Users always pay the charged amount directly for each ride.
- We assume that we have a fixed price per kilometre.

Part II

Requirements specification

3 Specific Requirements

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.
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.

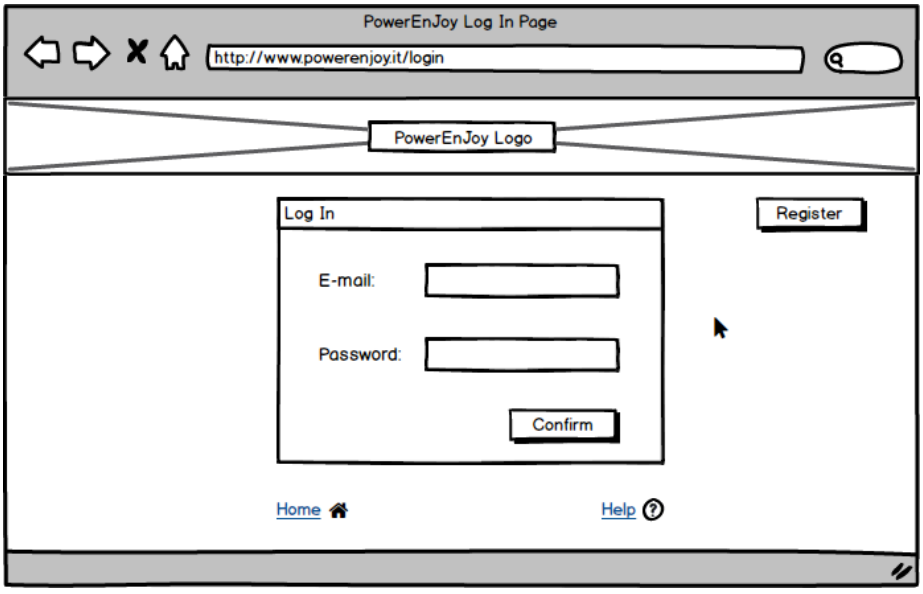


Figure 1.1: Mock-up of the web "Log In" page.

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.

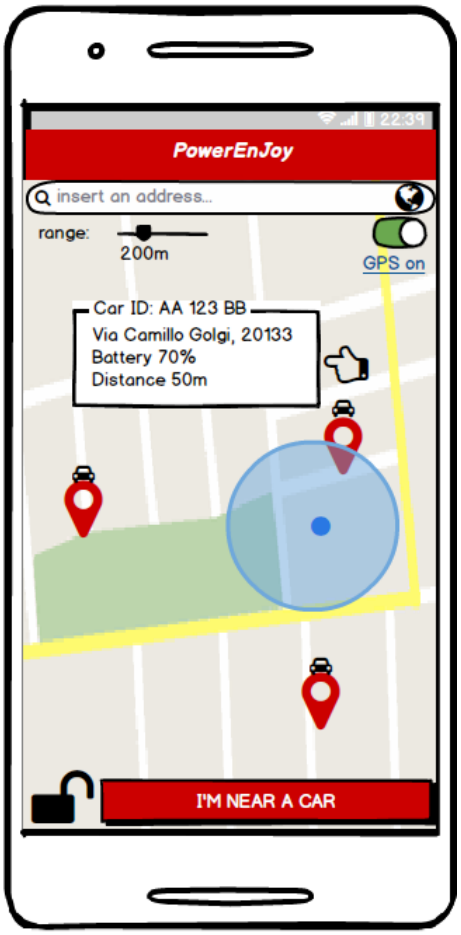


Figure 1.2: Mock-up for the localization of available cars in the mobile App.

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/javaee/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.

3.2 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 information 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 fee 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 then 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 users to contact PowerEnJoy Customer Service directly for assistance and any notification.

- [R1] The Costumer Service should provide assistance at any time.
- [D1] Sensor and devices are already installed in vehicles.

3.2.10 [G10] Allow the system to notify an operator when needed.

- [R1] The system shall check if a car is parked outside the special safe area.
- [R2] The system can contact an operator.
- [D1] The GPS signal always gives the correct geo-localization.
- [D2] The set of the safe areas is pre-defined by the management 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 Costumer 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

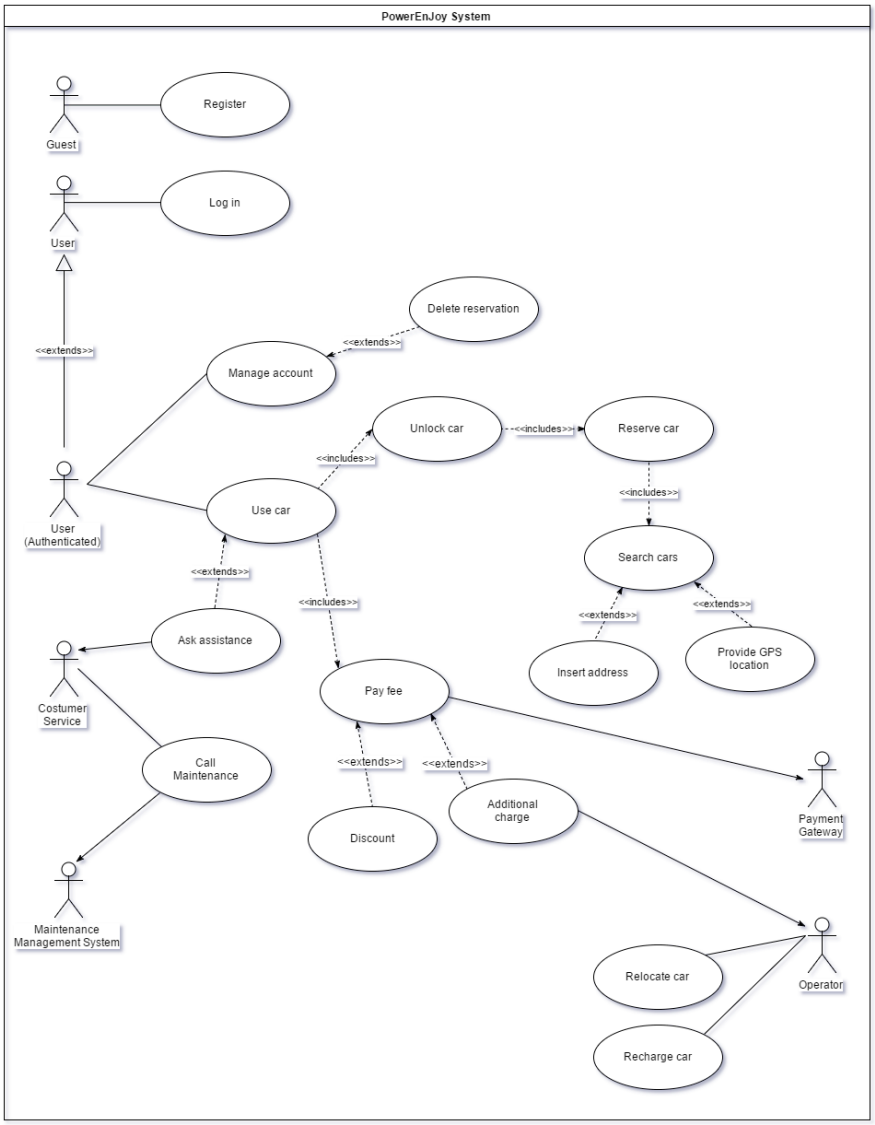


Figure 2.1: complete Use Case Diagram

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

Actor	Guest
Input Condition	NULL
Flow Events	1. Guest on the homepage clicks on "Sign Up" button to start the registration. 2. Guest fills in all the mandatory fields. 3. Guest clicks on "Confirm" button. 4. The system loads the user's homepage.
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	1. Guest is already a User. 2. One or more mandatory fields are not valid or incomplete. 3. Email chosen is already associated to another user. All exception are handled alerting the guest of the problem and application goes back to point 2 of EventFlow

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
Flow Events	1. User accesses the login page. 2. User inserts his/her email and password. 3. User clicks on "Log In".
Output Condition	PowerEnJoy shows the user's main page. The user is able to perform a reservation.
Exception	1. User's email and/or password are incorrect. User is redirected to 1. 2. User forgets to fill in both the fields. (Proceed as written above).

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	1. User decide if he/she wants to use his/her GPS position or another address. 2. Choose the range in which he/she wants to search the car.
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	1. User's GPS does not work properly. 2. The inserted address does not exist.

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	1. User clicks on a car image on the screen to see its details. 2. Click on "Confirm" to perform the reservation.
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.

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	1. User accesses the PowerEnJoy homepage. 2. User logs in into the system. 3. User selects the reservation that he wants to delete. 4. User confirms the reservation’s

	cancellation. 5. The system processes his/her request. 6. The system updates the car status. 7. The system loads the user's homepage.
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	1. User clicks on "Unlock Car" to notify he/she is nearby. 2. System unlocks the doors of the car.
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.

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

Actor	User
Input Condition	User is already in the car.
Flow Events	1. User ignites the engine. 2. The status of the car becomes "In Use". 3. The system calculates the number of passengers. 4. The system start calculating the charge. 5. User drives the car. 6. User parks in a safe area. 7. User turns the engine off. 8. The systems checks the level of the battery. 9. User exits the car. 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. 11. The system calculates the total fare, applying all the discounts and the additional charges.

	12. The amount of the fare is displayed on the user's app.
Output Condition	User automatically pays the fare to the system with his/her payment information.
Exception	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. 2. User cannot get cumulative discounts. In particular, only the greater one will be applied. 3. If a user has to pay an additional fee, no other discounts will be applied.

3.5.1.8 [G10] Allow system to notify an operator when needed.

Actor	Operator
Input Condition	A car is too far from a power grid station or it has a low battery level.
Flow Events	1. Car status is set to "Out Of Service". 2. System contacts automatically the nearest operator available. 3. Operator changes his state on "Busy". 4. Operator handles the relocation of the car or the charging process. 5. Operator returns Available.
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.

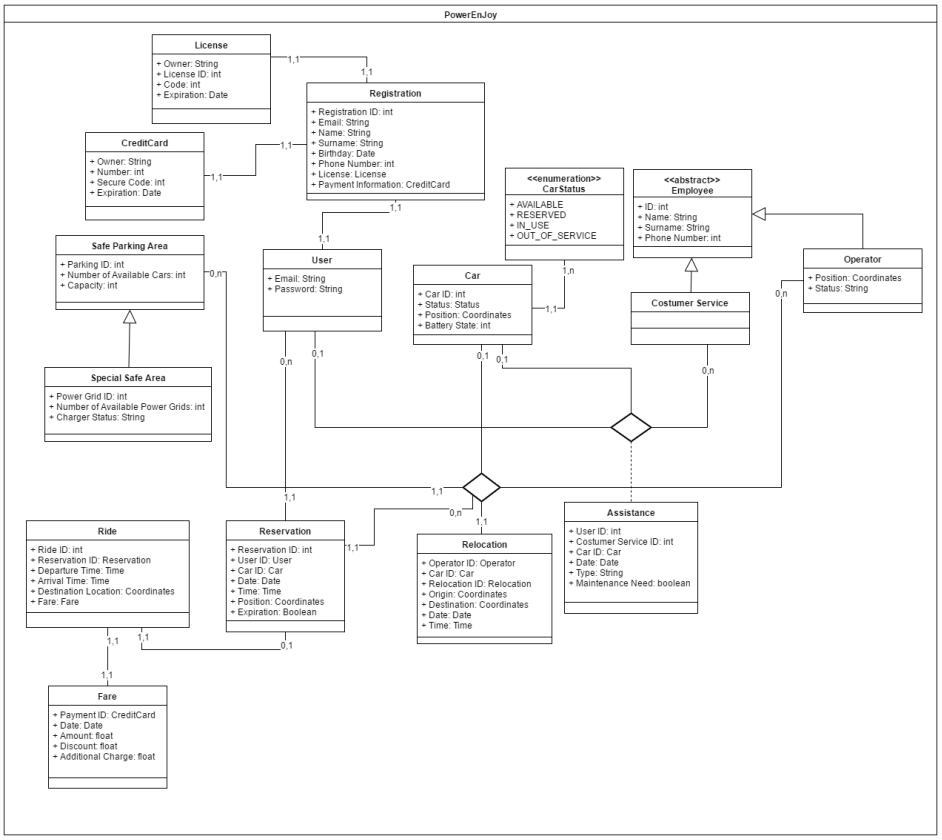
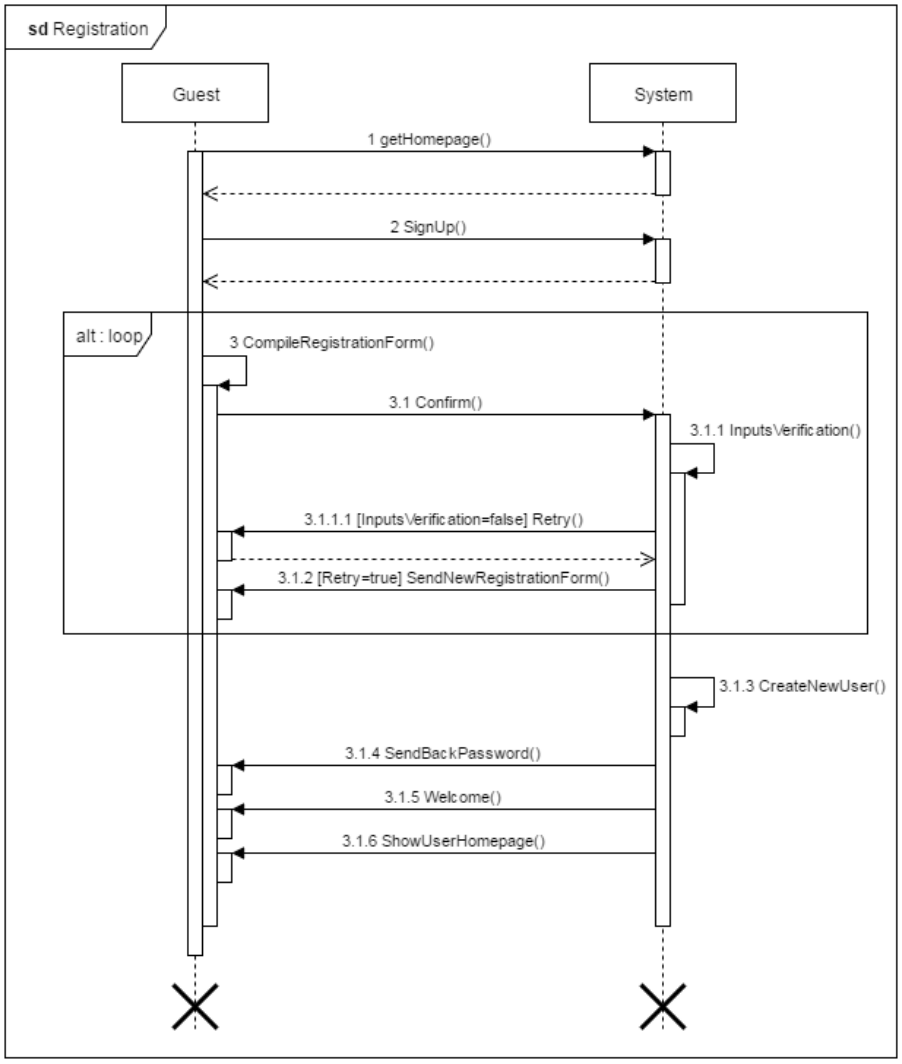


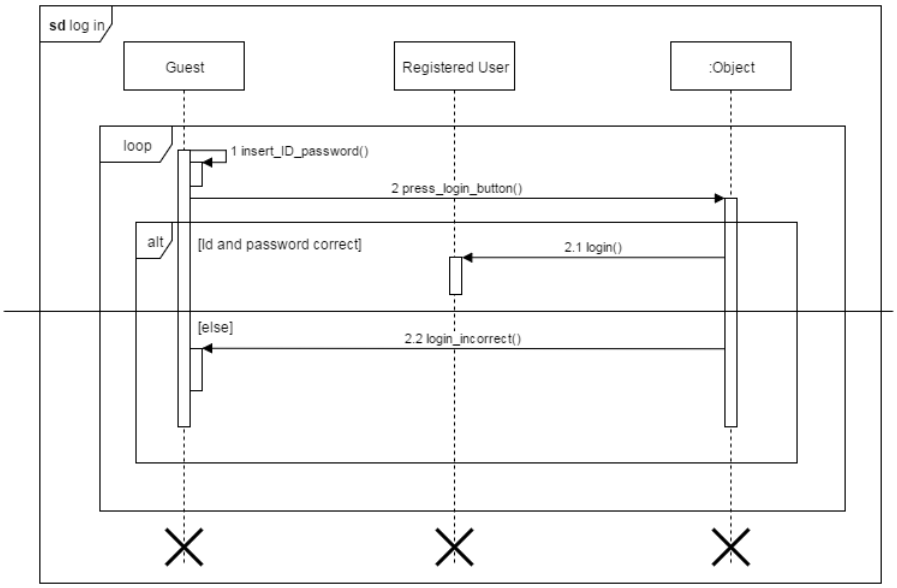
Figure 2.2: complete Class Diagram

3.5.2 Sequence Diagrams

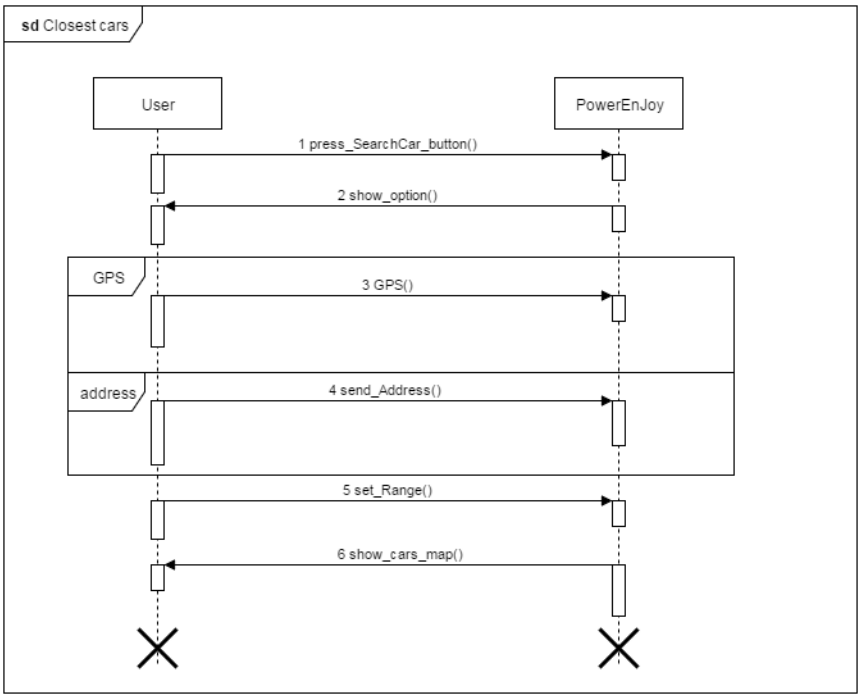
3.5.3.1 Registration:



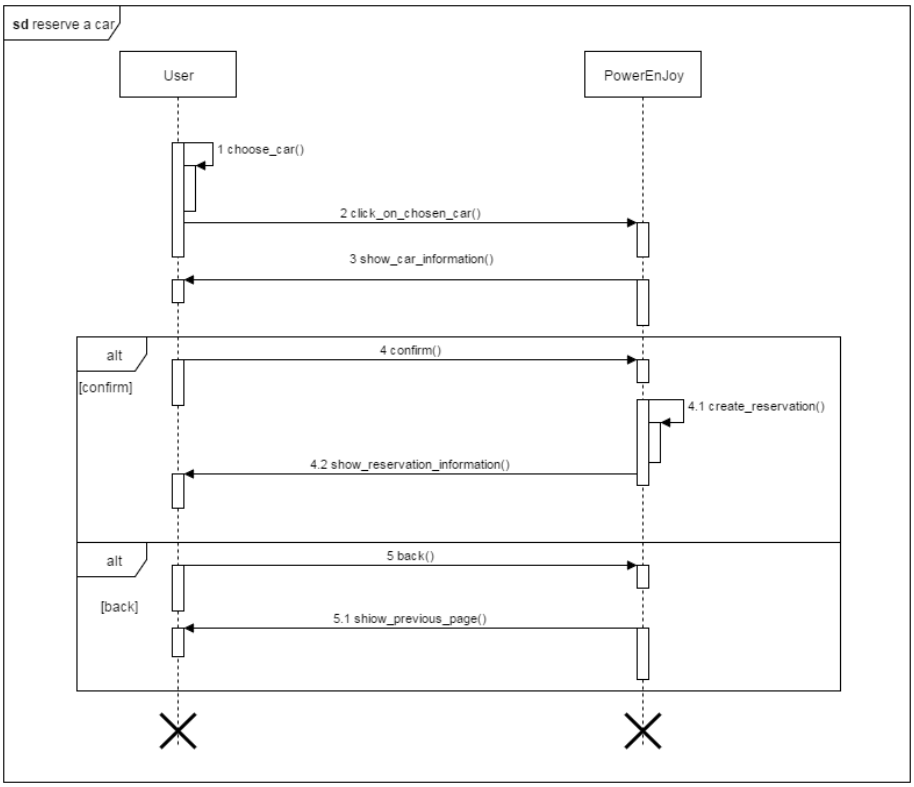
3.5.3.2 Log in:



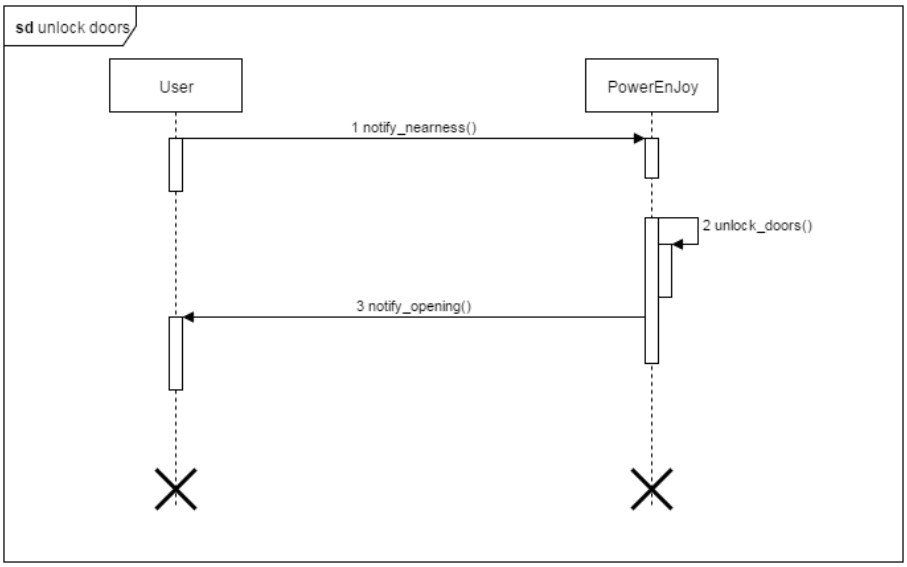
3.5.3.3 Search available cars:



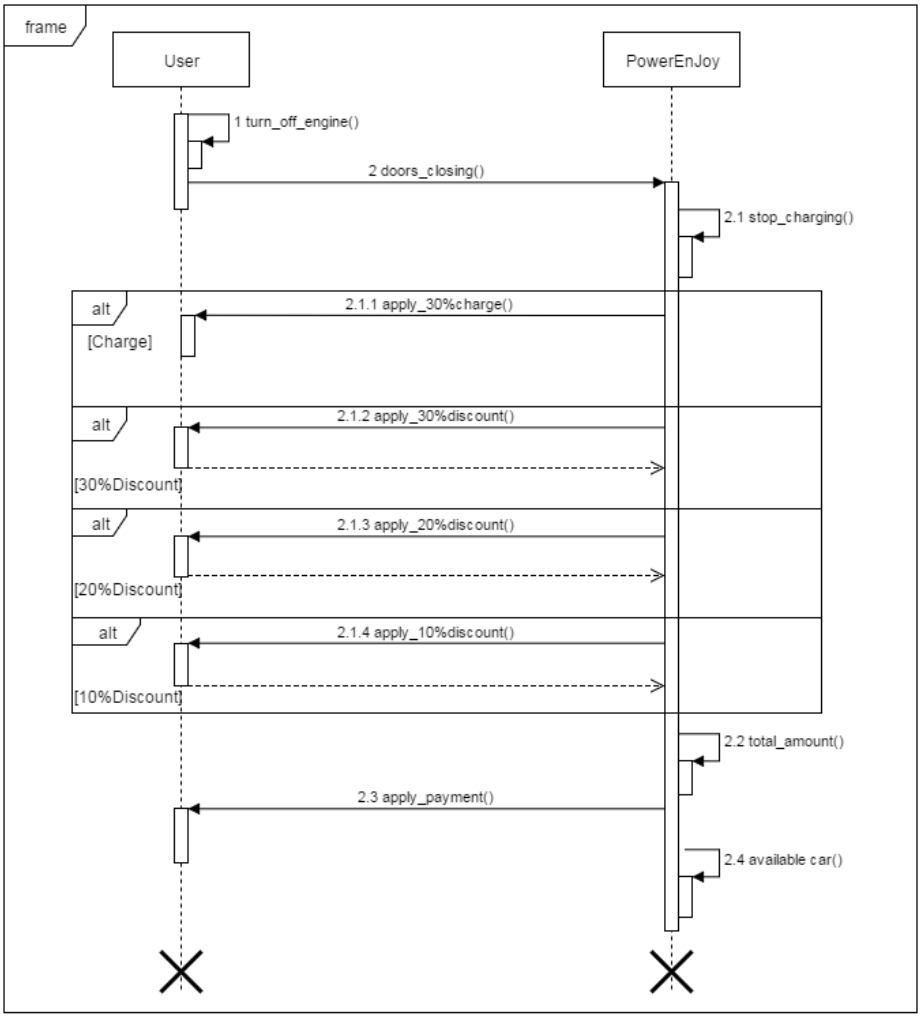
3.5.3.4 Reserve a car:



3.5.3.5 Unlock doors:

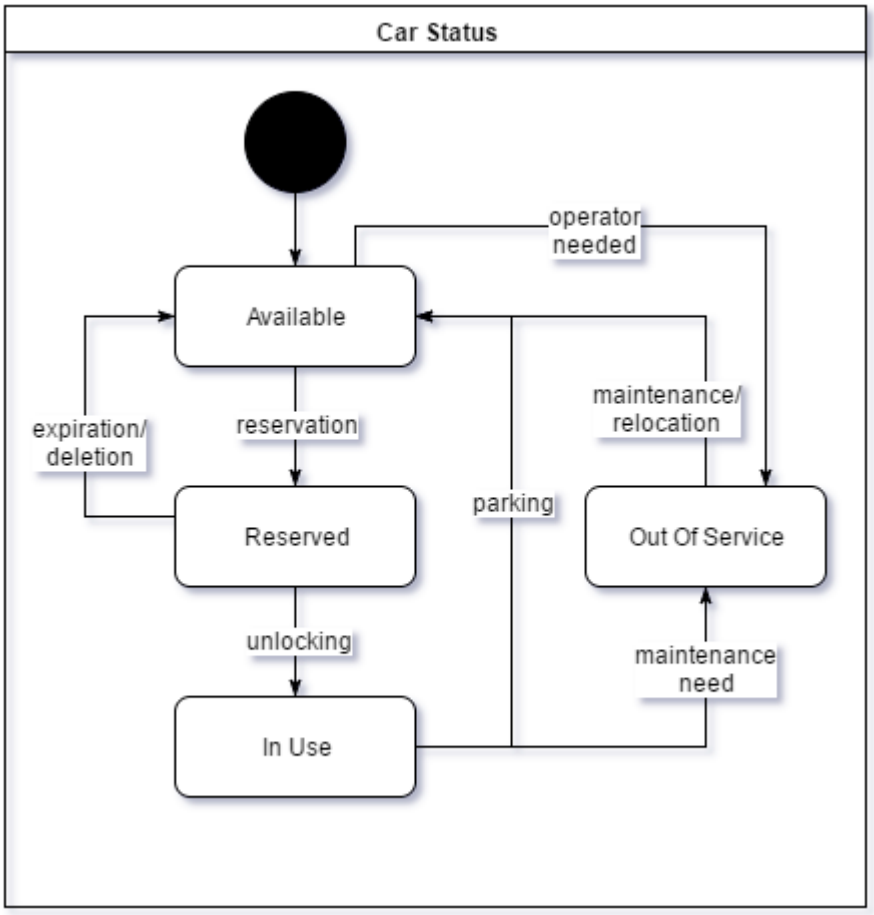


3.5.3.6 Apply discount or additional charge:



3.5.4 State Machine Diagram

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



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 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.

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

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.

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.

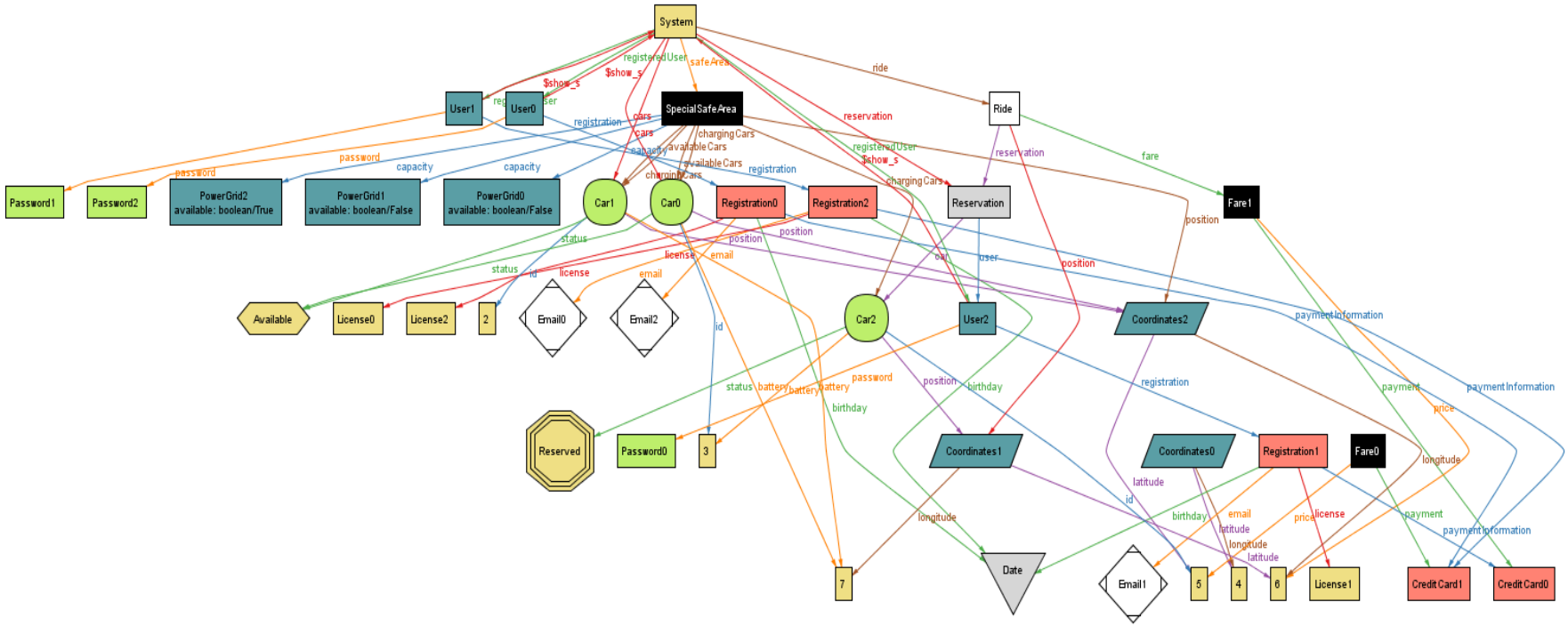
```
//---PREDICATES----//

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
```

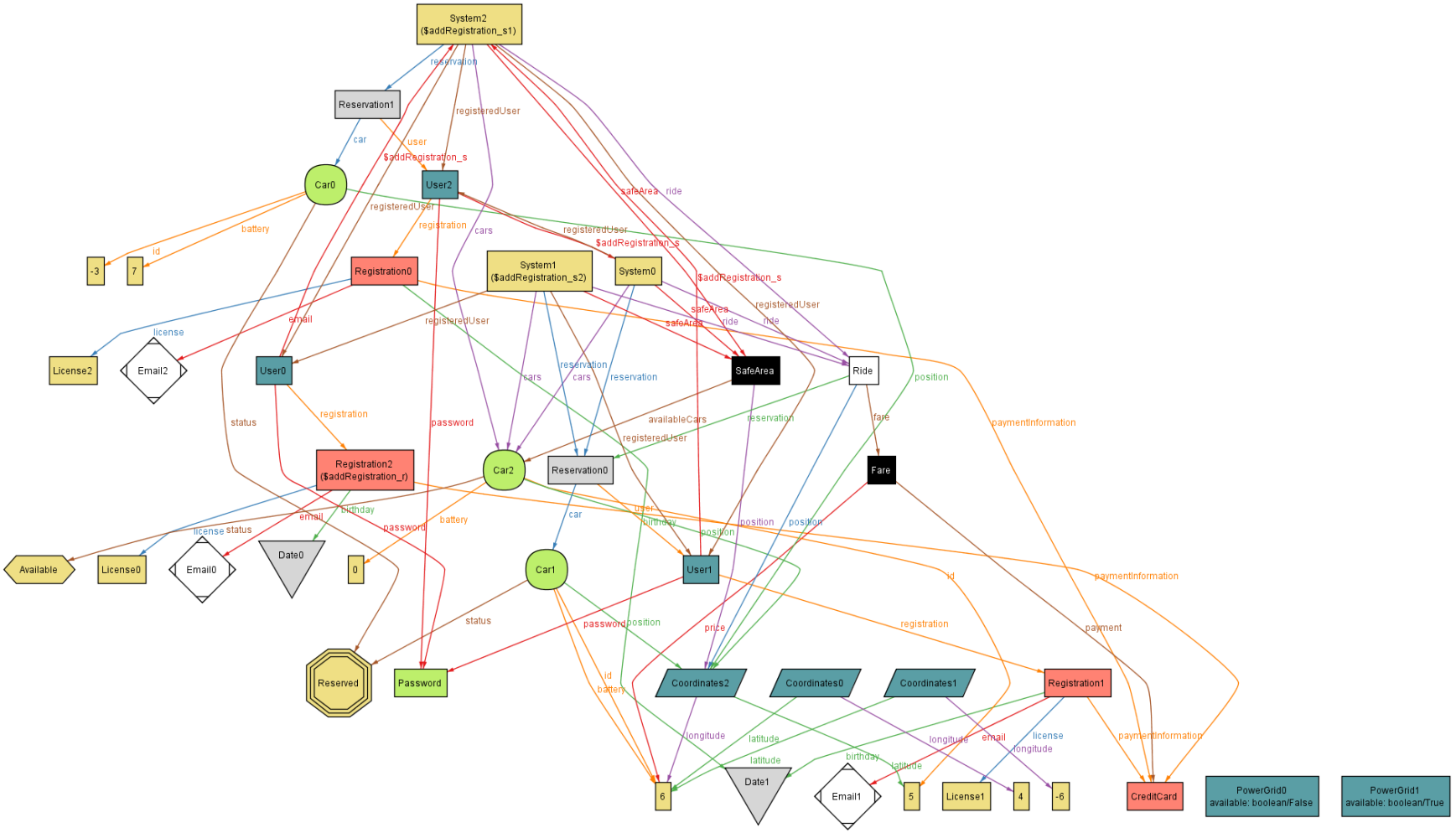
4.1.6 Result

This screenshot of the Alloy Analizer 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.


System
registeredUser
safeArea
$show_s
$show_s
ride
User1
User0
SpecialSafeArea
Ride
reservation
registration
cars
cars
chargingCars
capacity
availableCars
capacity
chargingCars
registration
registeredUser
$show_s
chargingCars
reservation
fare
password
password
capacity
capacity
Password1
Password2
PowerGrid2
available: boolean/True
PowerGrid1
available: boolean/False
PowerGrid0
available: boolean/False
Car1
Car0
Registration0
Registration2
Reservation
Fare1
position
position
position
status
status
id
license
email
license
email
position
user
Available
License0
License2
2
Email0
Email2
Car2
User2
Coordinates2
paymentInformation
id
status
battery
battery
password
position
birthday
birthday
registration
payment
paymentInformation
price
Reserved
Password0
3
Coordinates1
Coordinates0
Registration1
Fare0
longitude
latitude
id
longitude
birthday
email
latitude
price
license
payment
paymentInformation
longitude
latitude
7
Date
Email1
5
4
6
License1
CreditCard1
CreditCard0


System2
($addRegistration_s1)
reservation
registeredUser
Reservation1
car
user
$addRegistration_s
Car0
User2
safeArea
ride
registeredUser
registration
cars
registeredUser
$addRegistration_s
battery
id
-3
7
Registration0
System1
($addRegistration_s2)
System0
$addRegistration_s
safeArea
ride
registeredUser
email
registeredUser
license
License2
Email2
User0
reservation
cars
cars
reservation
SafeArea
Ride
position
status
registration
password
availableCars
reservation
fare
paymentInformation
registeredUser
Registration2
($addRegistration_r)
Car2
Reservation0
Fare
birthday
email
password
battery
user
birthday
position
position
position
license
status
Available
License0
Email0
Date0
0
Car1
User1
id
paymentInformation
car
status
password
position
price
registration
payment
Reserved
Password
id
battery
Coordinates2
Coordinates0
Coordinates1
Registration1
longitude
latitude
latitude
birthday
latitude
longitude
email
license
longitude
paymentInformation
6
Date1
Email1
5
License1
4
-6
CreditCard
PowerGrid0
available: boolean/False
PowerGrid1
available: boolean/True

Appendix A: Used Tools

A.1 Microsoft Word 2013

To redact and format and redact this document.

A.2 *g*it

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: ~35 hours
- Stefanetti Nadia: ~35 hours
- Total work time: 70 hours

Glossary

- Cars: set of vehicles provided by the PowerEnJoy company.
- Car status: an attribute that specifies whether a car is available, reserved, in use or out of service.
- Credentials: set of information that a guest provides in the registration form (name, surname, date of birth, e-mail and phone number).
- 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).
- Power grid station: a component that allows users to charge the battery of the cars as needed.
- Ride: the main action performed by the user from when it enters the car and ignites the engine to when the car is parked in a safe area and the user exits the car.
- Safe parking area: an area within the Service Coverage Area owned by the car sharing company and reserved only for their cars.
- 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: our software product as a whole.

Acronyms

- RASD: Requirements Analysis and Specification Document.
- GPS: Global Positioning System.
- DB: Data Base.
- DBMS: Data Base Management System.
- API: Application Programming Interface.
- OS: Operating System.
- JVM: Java Virtual Machine.
- JEE: Java Enterprise Edition.

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