Politecnico di Milano, A.A. 2016/2017

Software Engineering 2: PowerEnJoy
Requirements **A**nalysis and **S**pecification **D**ocument

Binosi Lorenzo 876022

December 7, 2016

Contents

1 Introduction

1.1 Purpose

This is the Requirement Analysis and Specification Document for the PowerEnJoy car-sharing service. Its purpose is to completely describe the system, its components, functional and non-functional requirements, constraints, and relationships with the external world, and to provide typical use cases and scenarios for all the actors involved. It is also a strong baseline for project planning and cost estimization, for software evaluation and change control.

This document is written for project managers, developers, testers and systems analysts. Users are not generally interested in detailed software requirements, but they can still find it useful. It may be used in a contractual requirement.

1.2 Scope

PowerEnJoy is a digital management system for a car-sharing service that exclusively employs electric car. This car rental system provides an alternative solution to public transport, thus being not only eco-friendly, but also simple and reliable.

The service is available to whomever has a valid driving license. Anybody who wants to register must provide a copy of it, along with his credentials and payment informations. Once registered, they will be allowed to search a car in a specific area, and reserve one for up to an hour before picking it up. The system will cancel the reservation in case of them failing to arrive to the car in time, will make that specific car available again and apply a fine to the driver.

Upon arrival to the car, the driver will be able to get inside and drive it, at the cost of a per-minute fee of which he will be notified by the system through a screen on the car itself; this until the car will be left parked in a safe area, where the system will close the doors and make the car available again.

A support team is provided by an external service in case of incidents, car failures and other cases of need.

1.3 Goals

The goals of the PowerEnJoy service are the following:

- [G1] Allow a driver to rent a car.

  - [G1.1] Allow a driver to search and reserve a car.
  - [G1.2] Allow a driver to use a reserved car.
  - [G1.3] Allow a driver to park a car in a safe area, suggested by the system.
- [G2] Allow a guest to sign up to the system.
- [G3] Motivate a driver into car pooling.
- [G4] Motivate a driver to a better behavior during the rent of a car.

1.4 Definitions, acronyms, and abbreviations

RASD: Requirements Analysis and Specification Document.

System: the whole software system to be developed, comprehensive of all its parts.

Car pooling: the sharing of car journeys so that more than one person travels in a car.

User: anyone recognized by the system through credentials. In this case the only users of the system are drivers.

Driver: a user of the car-sharing service.

RDBMS: Relational DataBase Management System.

API: Application Programming Interface. A set of routines, protocols, and tools for building software applications

JVM: Java Virtual Machine. An abstract computing machine that enables a computer to run a Java program.

1.5 References

This document refers to the project rules of the Software Engineering 2 project [1] and to the RASD assignment [2].

This document follows the IEEE Standard 830-1998 [3] for the format of Software Requirements specifications.

1.6 Overview

This document is structured in three parts:

section 1: Introduction. It provides an overall description of the system scope and purpose, together with some information on this document.

section 2: Overall description. Provides a broad perspective over the principal system features, constraints, and assumptions about the users and the environment.

section 3: Specific requirements. Goes into detail about functional and nonfunctional requirements. This chapter is arranged by feature.

2 Overall Description

2.1 Product Perspective

PowerEnJoy system is a simple client-server architecture based on a back-end server application and diffent front-end client applications, supported by different operating systems.

2.1.1 User Interfaces

Guests and users can interact with the service via the web application or the mobile application. Drivers can find other service functionalities in the car application. It is necessary to provide a common and uniform look and feel among the different hardware architectures.

All interfaces shall be intuitive and user friendly. They should not require the reading of detailed documentation to be used.

2.1.2 Hardware Interfaces

All cars are provided with a dedicated embedded system connected to several plugins and a touchscreen display used for interact with the car application. Thanks to this embedded architecture the system will also be notified about the status of the car and its location, even if the main battery is completely discharged.

2.1.3 Software Interfaces

Mobile application and web application are supposed to be friendly with any device, in particular the first one must be developed for iOS and Android, and the second one will work on any operating system that support a web browser.

On the embedded system of any cars is installed a JVM that runs a Java application that provides informations to the driver and to the system.

The back-end server stores its data in a RDBMS and can run on every platform that supports the JVM. It also provides different APIs for different functions that a user, or a guest, can do through client applications.

2.2 Product Functions

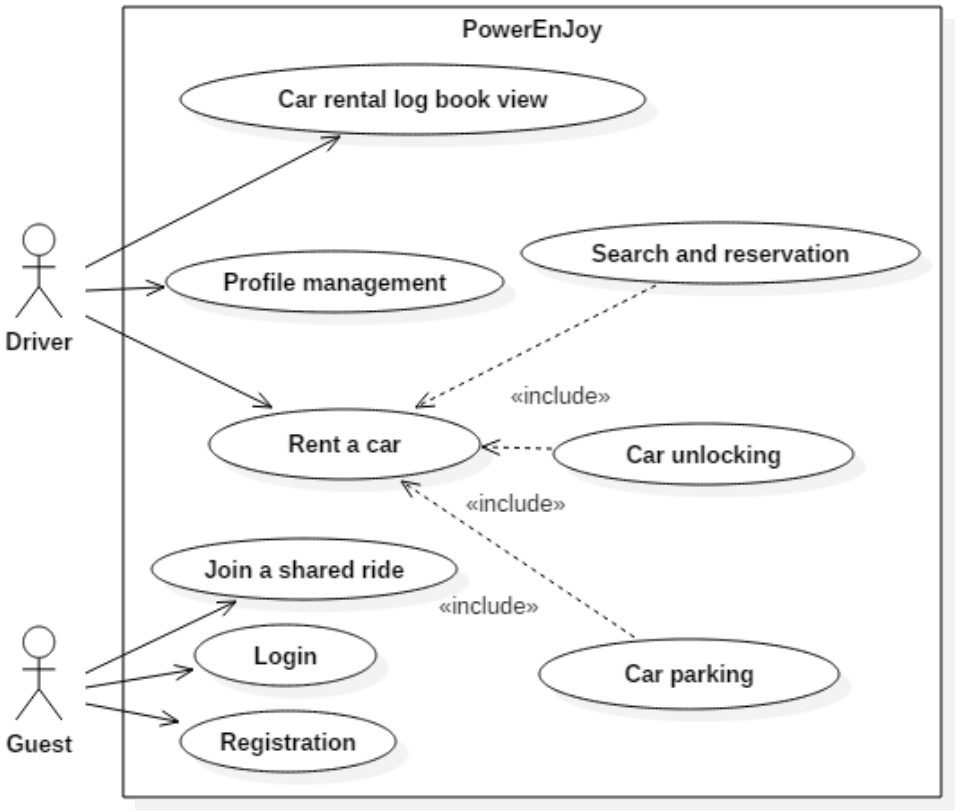

Figure 2.1: The comprehensive use-case diagram of all the functionalities implemented by the system.

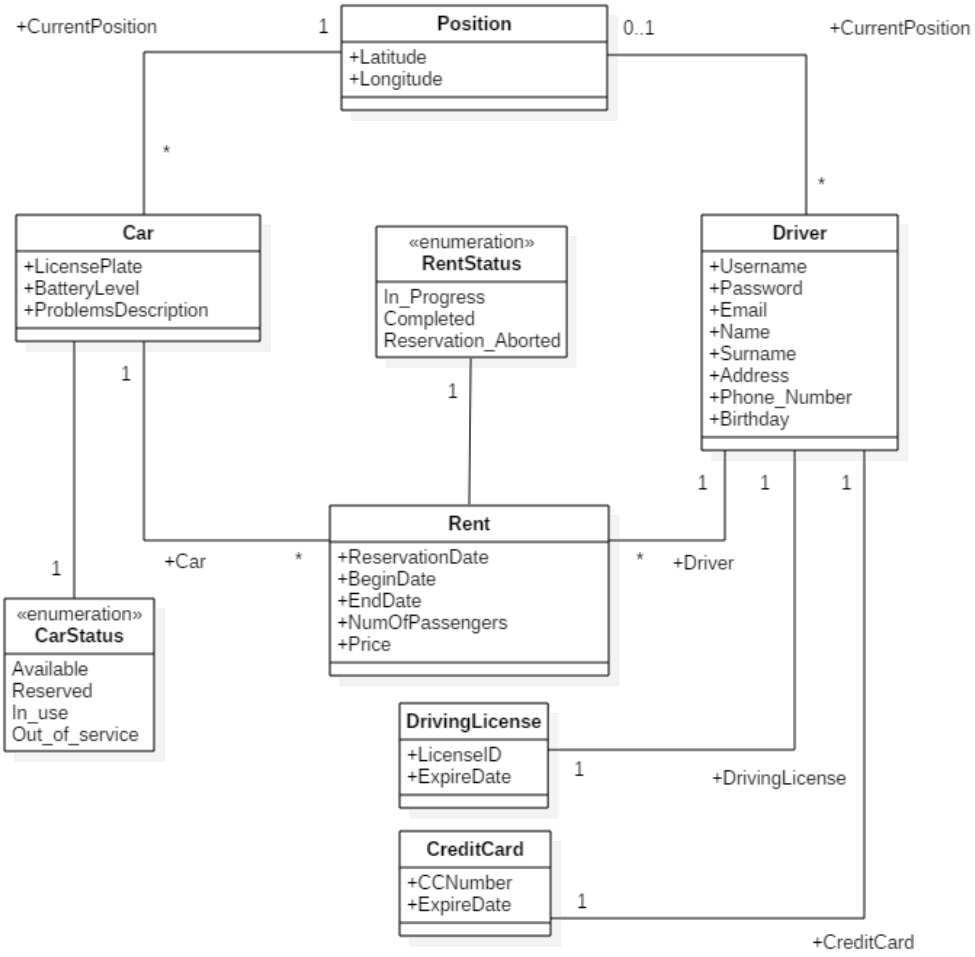

Figure 2.2: The comprehensive class diagram of the system.

The following lists reassume what anyone can do interacting with the system.

- Guests can:
  - create an account (sign up to the system)
  - log in
- Drivers can:
  - edit profile informations
  - delete their account

- rent a car
- access to the car rental log book

2.3 User Characteristics

The only users of the system are drivers. Drivers must provide a valid driving license and a valid payment method in order to register to the service.

It's granted that drivers have access to an internet connection.

2.4 Constraints

2.4.1 Regulatory policies

Any driver must follow current traffic and regulamentation laws of the area where the system is operating its car-sharing service. Driving licenses from other countries must be compatible with the laws of the country where the service is operating.

The system must ask the driver for the permission to acquire, store and process personal data and web cookies.

2.4.2 Interfaces to other services

PowerEnJoy needs to communicate with a service to solve all those issues that can hit a car or that can happen while using it. These problems may be:

- breaking of one or more parts of the car
- battery power lower than 25%
- accidents
- faulty communication between system and car

The service assigned to this kind of problems, receives all reports that fall in the previously specified cases, and solves them as soon as possible, reporting the results back to PowerEnJoy.

2.4.3 Hardware limitations

The service requires an Internet connection fast enough to guarantee a fast response from the server and hardware architectures that can run properly the client side applications (web and mobile apps).

2.4.4 Reliability requirements

The system must have a minimum availability of 99%.

2.4.5 Parallel operations

The system must support parallel operations from different drivers that may require access to the database.

2.5 Assumptions and Dependencies

It's assumes that:

- All drivers have access to a stable internet connection.
- All cars are the same model, so that any of them will have the same number of seats
- The number of cars is sufficient to satisfy the demand in each area.
- All the areas where the system is operating are covered by a reliable 3G/4G connection.
- The system is operating in an area composed by a city or agglomerated cities, of the same country, closed each other.

2.6 Future Extensions

The system will be implemented foreseeing the possibility of further extensions, for example:

1. If a car is left at special parking areas where they can be recharged and the driver take care of pluggin the car into the power grid, the system applies a discount of 30% on the last ride.
2. 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, the system charges 30% more on the last ride to compensate to compensate for the cost required to re-charge the car on-site.
3. If the driver enables the money saving option, he/she can input his/her final destination and the system provides informations about the station where to leave the car to get a discount. This station is determinate to ensure a uniform distribution of cars in the city and depends both

on the destination of the driver and on the availability of power plugs at the selected station.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User interfaces

User interfaces must satisfy the following UI constraints:

- Web application
    1. Web pages must adhere to the W3C standards. In particular, the software shall conform to the HTML 5 [4], CSS [5] standards.
- Mobile application
    1. The iOS version must adhere to the iOS Human Interface Guidelines [6].
    2. The Android version must follow Android design guidelines [7].
- Common to web and mobile applications:
    1. The client applications must have an UI that is accessible to disabled people.
    2. The interface must offer the possibility to choose the language used at all times.
    3. The first screen (homepage) must ask the guest to log in or sign up in order to begin operations.
    4. Once logged, the homepage must show the car search page. It also includes one button for log out and buttons that open different pages. These pages are:
        - reservation page; where the driver can find the functionality in order to unlock the car. The page also provides informations about the reserved car.
        - profile management page; where the driver can manage his/her profile informations or update his/her driving license and payment method.
        - car rental log book page; where the driver can find information about his/her past rentals.
    5. UI controls and views must be suitable for the input interface and the screen size.

- Car application
  1. radio and GPS navigator applications must be implemented.
  2. always notifies the driver about his/her current charges.
- Server back-end
  1. The server back-end must be configurable by means of a configuration text file.

3.1.2 Hardware interfaces

The embedded system of any cars must be provided of

- a 7" touchscreen display
- a GPS device
- sensors that check if all the parts of the car are working correctly
- sensors that check how many passengers are on the car
- a 4G router for a stable Internet connection
- a secondary battery used only if the main battery is discharged. The car can't use this battery

3.1.3 Software interfaces

The required software products used by the back-end are:

- MySQL 5.7[1]
- Java SE 8[2]

The required software product used by the car application is:

- Java Embedded[3]

The required operating systems for the mobile application are:

- iOS 8 or better
- Android 6.0 or better

A set of APIs for the several functionalities must be provided by the system

[1] http://dev.mysql.com
[2] http://www.oracle.com/technetwork/java/javase/overview/index.html
[3] http://www.oracle.com/technetwork/java/embedded/overview/index.html

3.1.4 Communications interfaces

The clients communicate with the server via HTTPS requests (port 443).

3.2 System Features

3.2.1 Driver registration

3.2.1.1 Purpose

Any guest can subscribe through web or mobile applications.

In both cases the guest has to fill a registration form and must agree to the personal data policy according to his/her country privacy laws, otherwise the registration request will be aborted.

As soon as the guest has submitted all the data, the system verifies the consistency of the information and a confirmation mail with a password is sent to the email address indicated in the registration form. The guest must confirm his/her email address to end the registration.

Once registered a guest can be recognized as a driver by logging in.

3.2.1.2 Scenario 1

Bob, a normal citizen without a car, has just discovered the existence of the PowerEnJoy service and wants to use it.

He opens the homepage of PowerEnJoy website and proceeds to register by clicking on the button "Sign up".

He fills in all the information required and authorises the personal data treatment.

The system verifies all the information that bob submitted in the form and sends him a confirmation mail.

Bob checks his mailbox, opens the email and clicks on "Confirm email". He is also notified about his current password.

The system then informs Bob that he is successfully registered.

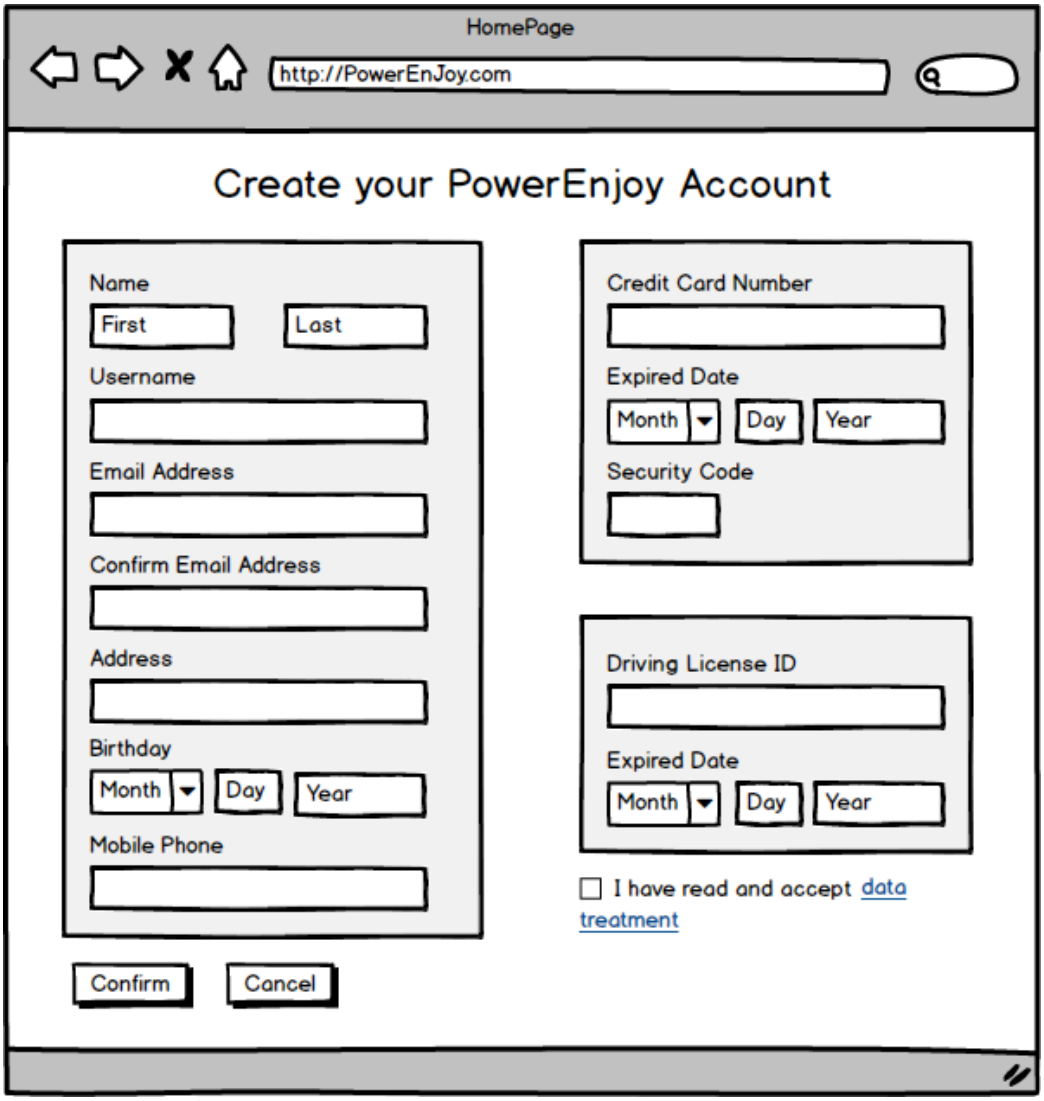

Figure 3.1: Concept for the registration webpage.

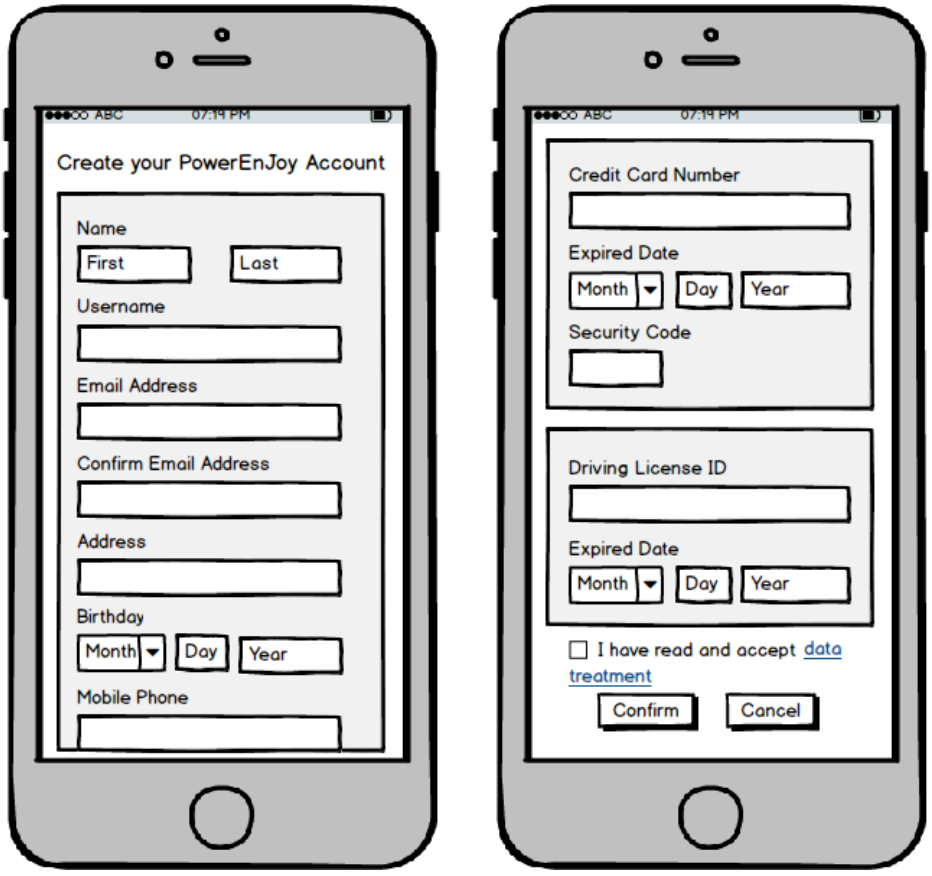

Figure 3.2: Concept for the mobile registration page.

3.2.1.3 Use case description and sequence diagram

Actor	Guest
Goal	G3
Input condition	The guest chooses to create a new driver account.
Event Flow	1. The registration form is loaded and the guest compiles it. 2. The guest authorizes the personal data treatment. 3. The guest clicks on "Confirm". 4. The system sends a confirmation email. 5. The guest reads the e-mail containing his/her password, received by PowerEnJoy and clicks on the link to confirm the registration.
Output condition	The system tells the guest that he/she has been successfully registered.
Exception	• Some exceptions are handled by notifying the guest of the problem through a dynamic message box or reloading the registration form. The requirements that generate these kind of exceptions are: 2, 3, 4, 5, 6, 7. • Some exceptions are handled aborting the registration (all guest's data are deleted). The requirements that generate these kind of exceptions are: 8, 10b.

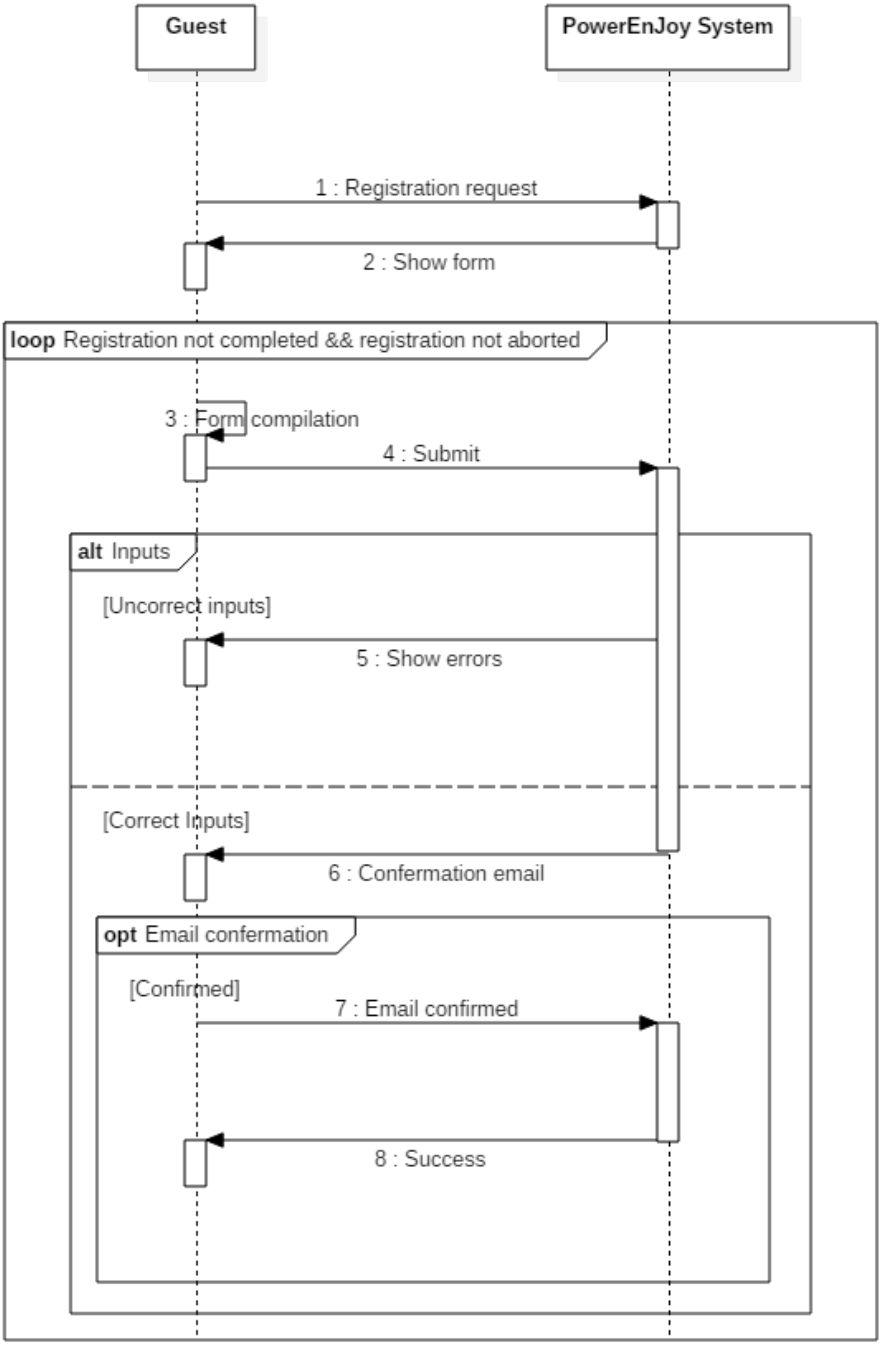

Figure 3.3: Sequence diagram of the registration process.

3.2.1.4 Associated functional requirements

1. Guests must provide the following information:
    - name
    - surname
    - username
    - email address
    - address
    - day of birth
    - phone number
    - credit card number
    - credit card date of expire
    - credit card security code
    - driving license ID
    - driving license date of expire
2. There mustn't be another user already subscribed with the same username and/or email.
3. There mustn't be another driver already subscribed with the same driving license.
4. Username must match the regular expression "`[a-zA-Z][a-zA-Z0-9]{2,20}`"
5. The system accepts the email only if it is entered identically both times.
6. The driving license must be valid and not expired.
7. The payment method must be valid and not expired.
8. If the personal data treatment is not authorised, the subscription is canceled.
9. The system must allow the guest to abort the registration process at any time.
10. Email confirmation process:

(a) The subscription ends successfully when the guest clicks on the link in the confirmation email. It must contain a password for the new driver.

(b) After one day without an answer, the guest's registration info are deleted and the guest may re-try the registration process.

3.2.2 Search and reservation

3.2.2.1 Purpose

Any logged driver, in possession of a valid driving license and a regular status of payments, can search and reserve a car through the web application or the mobile app.

The system shows a map, of the area nearby him/her, with the available cars represented as colorful markers. There are 3 diffent colors for the markers:

- red: 25-50% of battery charge
- yellow: 50-75% of battery charge
- green: 75-100% of battery charge

The driver can also search nearby cars from a given address through a search box on the page.

Once the driver chooses the car he can click on its marker and a new dynamic frame will show more informations about the car and the possibility to reserving it by clicking on the button "Reserve". The system then notifies the driver that he/she succesfully reserved a car.

Cars with less charge than 25% of battery charge require assistance to be recharged and the system won't show them as available to the drivers.

3.2.2.2 Scenario 1

Bob logs in the PowerEnJoy application through his web browser. He wants to reserve a car because an important meeting is waiting for him. Once logged, the system redirects him to the homepage that now shows the car search page. Bob realizes there's only a car with a red marker within 1km from him. Bob has to hurry up so he clicks on the red marker and then on the button "Reserve". Bob has succesfully reserved the car and he get redirect to the reservation page.

3.2.2.3 Scenario 2

Alice logs in the PowerEnJoy application through his mobile phone. She wants to reserve a car but everytime she clicks on the button "Reserve" of an available car the system shows her a dynamic message box that remember her to solve her pending payments. Once any pending payments is solved she will be able to reserve an available car.

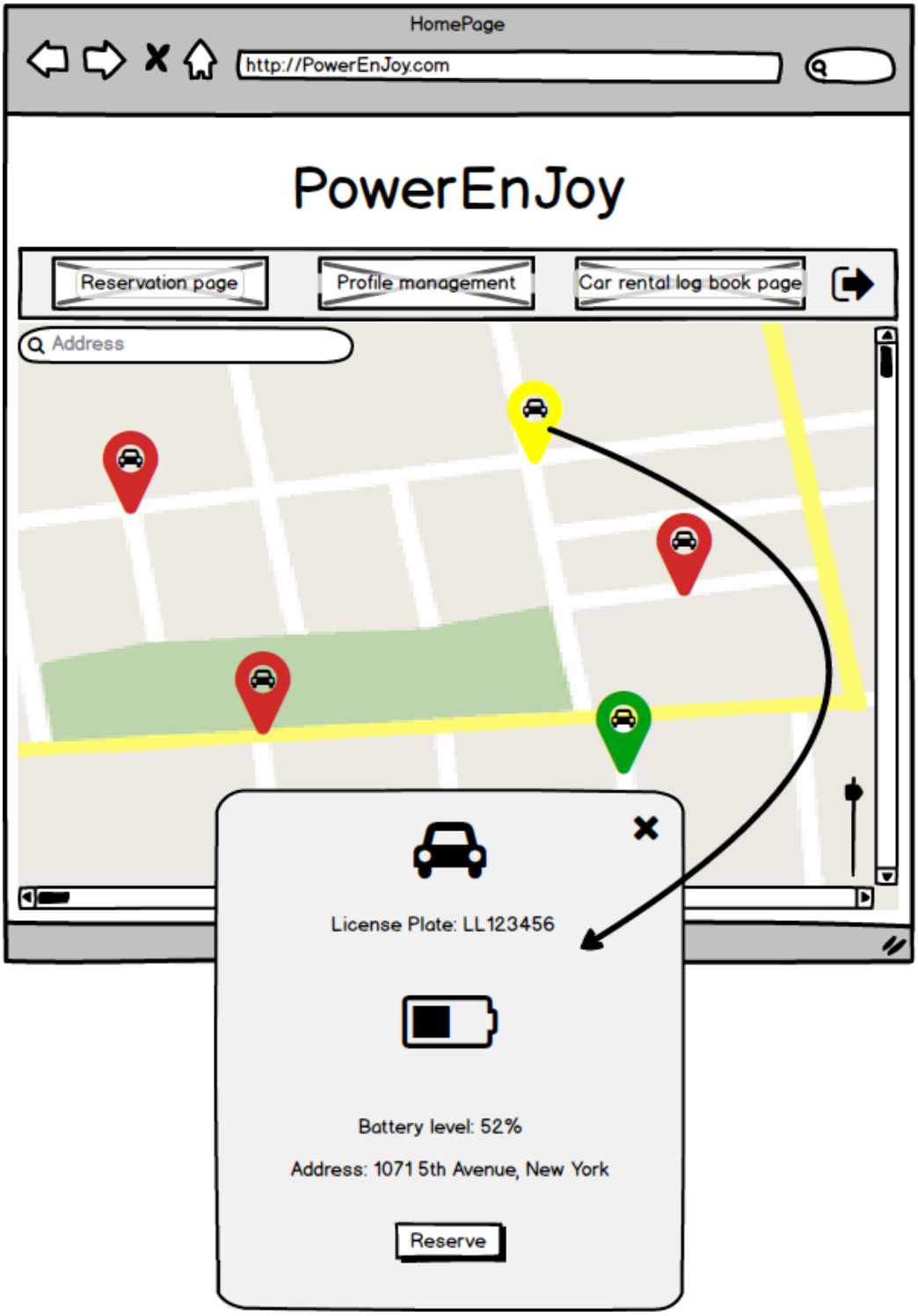

Figure 3.4: Concept for the search webpage.

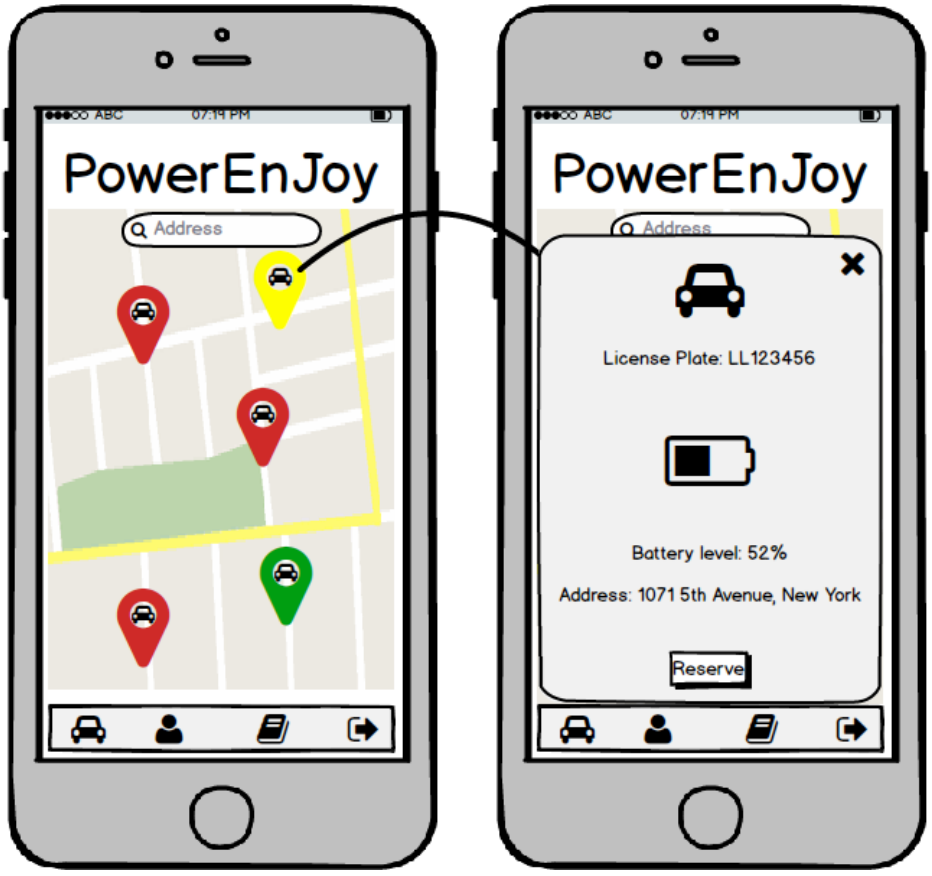

Figure 3.5: Concept for the mobile search page.

3.2.2.4 Use case description and sequence diagram

Actor	Driver
Goal	G1.1
Input condition	The driver is already logged and chooses to reserve a car.
Event Flow	1. The driver opens the home page and search an available car. 2. Once chosen, the driver clicks on "Reserve".
Output condition	The system notifies the driver that he/she succesfully reserved the chosen car
Exception	• Some exceptions are handled by notifying the driver of the problem through a dynamic message box. The requirements that generate these kind of exceptions are: 4, 5. • No car is available in the driver's area.

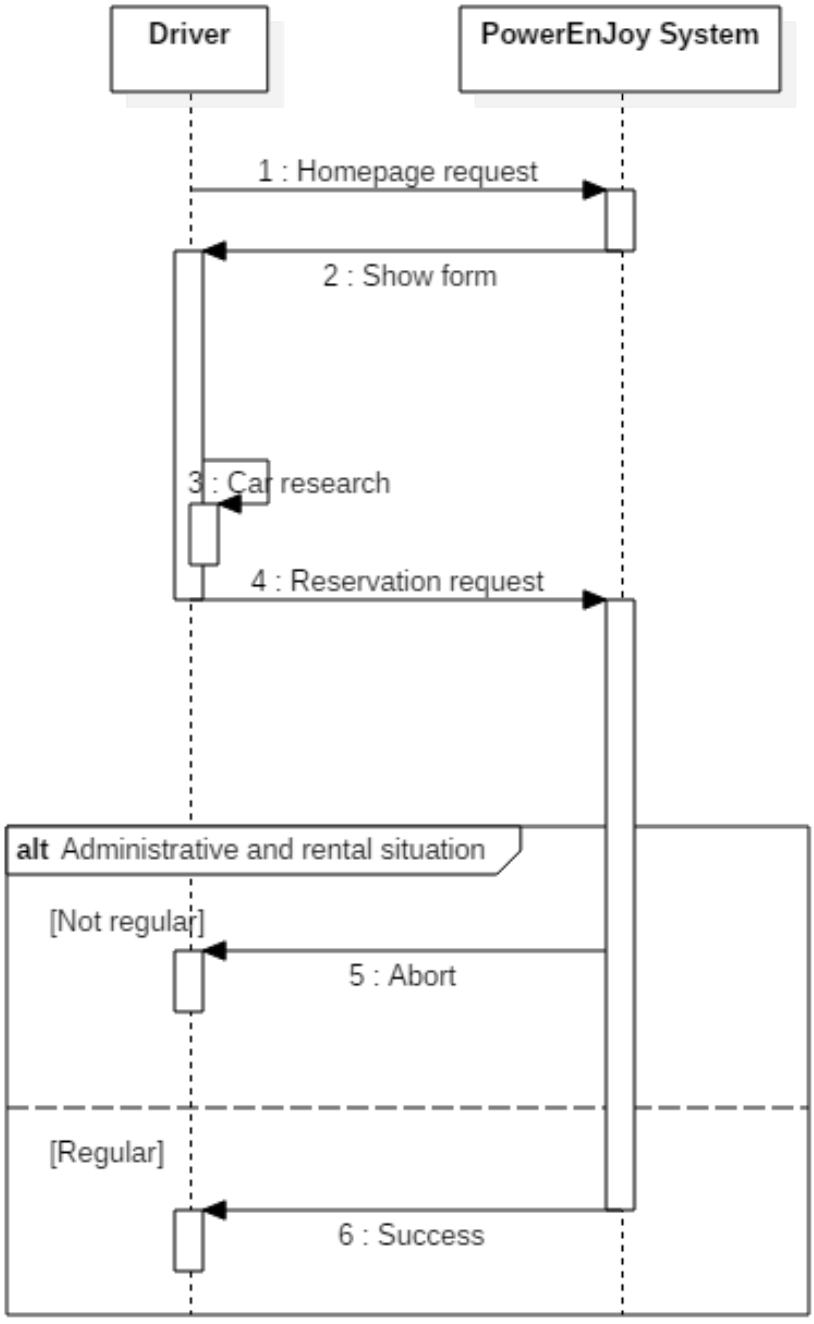

Figure 3.6: Sequence diagram of search and reservation process.

3.2.2.5 Associated functional requirements

Interactions between system and driver occur through car or mobile applications.

1. The driver must be logged into the system in order to search and reserve a car.
2. The system must show the driver a map of the area, within a 1km range from him/her or from a given address, with the available cars.
3. The system must provide more informations about an available car. These informations are:
   - license plate
   - address
   - battery level
4. In order to reserve a car the driver's driving license mustn't be expired.
5. In order to reserve a car the driver's status of payments must be regular.
6. The system must prevent the driver from reserving more than one car at time.
7. The system must prevent the driver from reserving a car already reserved, or in use by another driver.
8. The system must prevent the driver from reserving an "Out_of_service" car.
9. After a reservation is completed, the system automatically tags the reserved car as "Reserved".
10. After a reservation is completed, the system automatically redirects the driver to the reservation page completed by with the reservation informations.

3.2.3 Car usage

3.2.3.1 Purpose

The driver can use the car after he/she reserved it.

When the driver is nearby the reserved car he/she can click the button "I'm nearby" on the mobile reservation page and the system will check the GPS signal of both the car and the driver's mobile phone, to assess if they are within a 25 meters range. If they are within that range, the system will unlock the car and the driver can enter it. This operation is available only through the mobile application.

As soon as the engine ignites, the system starts charging the driver for a given amount of money per minute; the driver is notified of the current charges through a screen on the car.

3.2.3.2 Scenario 1

Bob has already reserved a car and he is going to pick it up. He is about 50 meters from the car and he clicks the button "I'm nearby" on the mobile application. The system checks the GPS signal of both the car and Bob's mobile phone, and doesn't unlock the car notifying Bob that he's too far from the car.

Once Bob is in front of the car he clicks again on "I'm nearby", the system unlocks the car and he can finally enter.

Bob starts the car and the system notifies him about his current charges through the car application. He can start his ride.

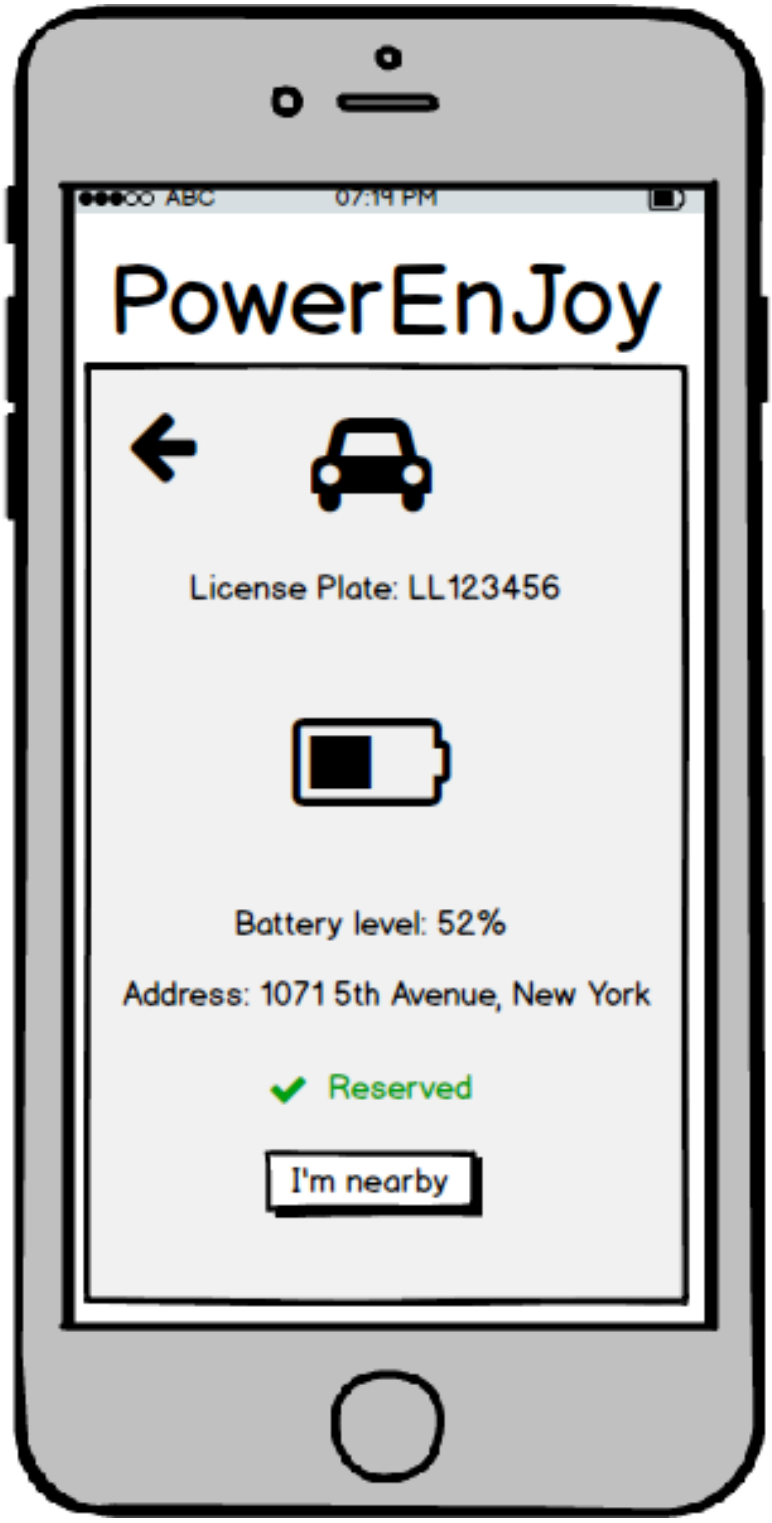

Figure 3.7: Concept for the mobile car reservation page.

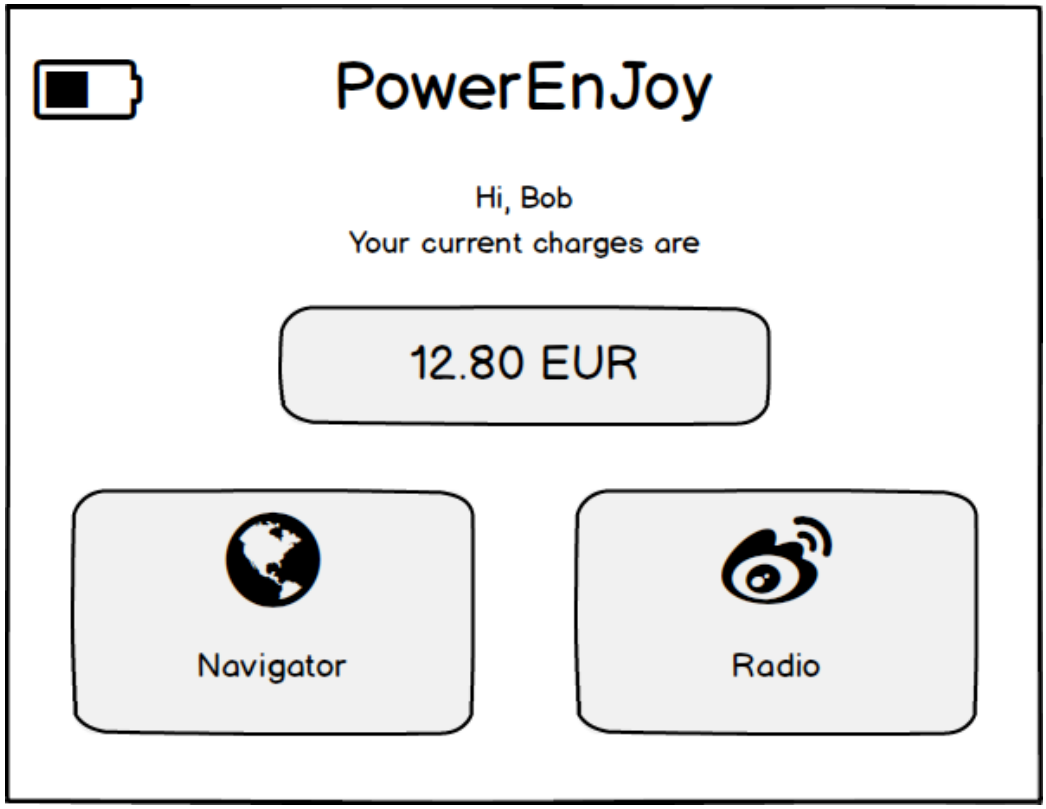

Figure 3.8: Concept for the car application.

Actor	Driver
Goal	G1.2
Input condition	The driver has already reserved a car and he/she is going to pick it up.
Event Flow	1. The driver is within 25 meters from the car. 2. The driver opens the reservation mobile page, through a button of the homepage, and clicks on "I'm nearby". 3. The system unlocks the car. 4. The driver enters and starts the car.
Output condition	The driver starts his/her ride and the system keeps notifying him/her about his/her current charges.
Exception	• Some exceptions are handled by notifying the driver of the problem through a dynamic message box. The requirements that generate these kind of exceptions are: 2, 3. • GPS signal of the driver's mobile phone is not available. • GPS signal of the car is not available.

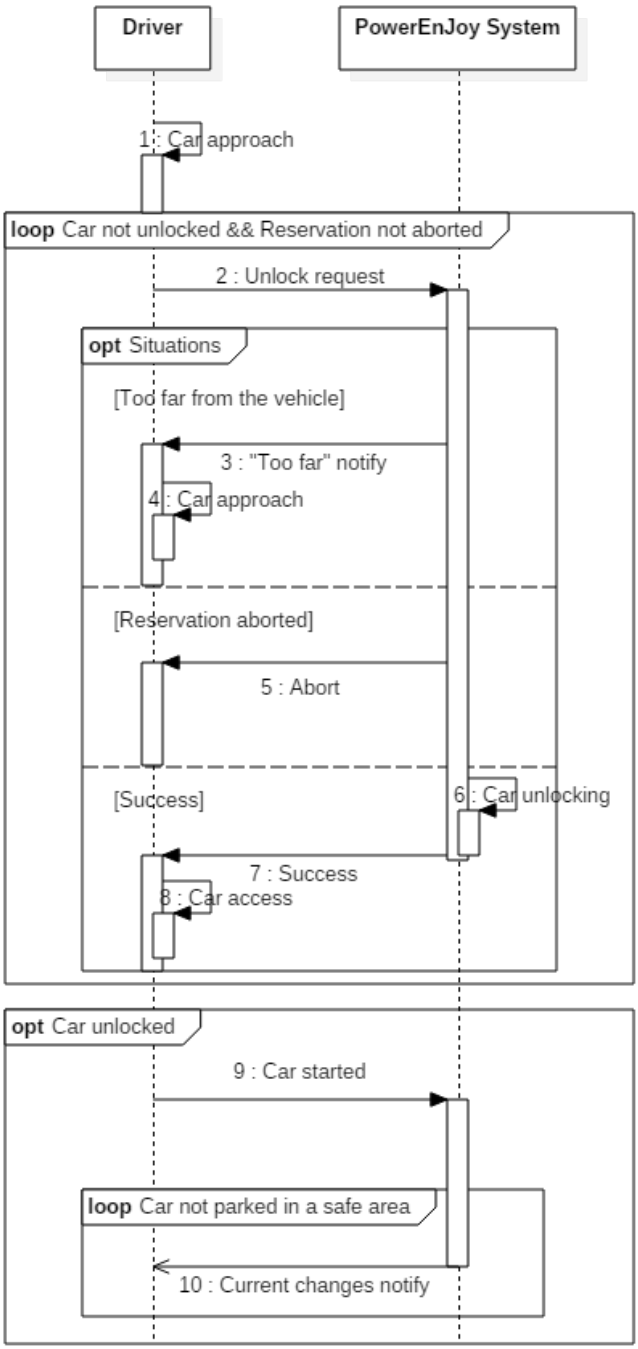

Figure 3.9: Sequence diagram of the car usage process.

3.2.3.3 Associated functional requirements

1. The driver can only use the mobile application to unlock the car.
2. From the reservation to the unlocking of the car must not pass more than one hours. Otherwise the system will tag again the car as "Available". The driver is notified about the end of the rental operation.
3. The system checks the GPS signal of both the car and driver's mobile phone, to assess if they are within a 25 meters range. If they are within that range, the system will unlock the car.
4. The driver must enter in the car to start it.
5. The system will start charging the driver once he/she starts the car.
6. Once the car is unlocked, the system tags it as "In_use".
7. The car application must notify the driver about current charging.

3.2.4 Car parking

3.2.4.1 Purpose

Once the driver has finished using the car he/she should be able to park it in safe area provided by the system. These areas are displayed by the navigator car application.

After parking the car, the driver can exit from it. The system will lock the car and automatically charges the driver on the credit card indicated during the registration.

If a driver brought other people with him/her, he/she will get a 10% discount on the final charge.

3.2.4.2 Scenario 1

Bob is almost at his important meeting. The GPS navigator application shows several parking areas and Bob chooses one of them to park his car. He checks if there's at least one available parking spot and he finds one. Once he turns off the car and gets out, the system stops charging him and he can finally enjoy his evening.

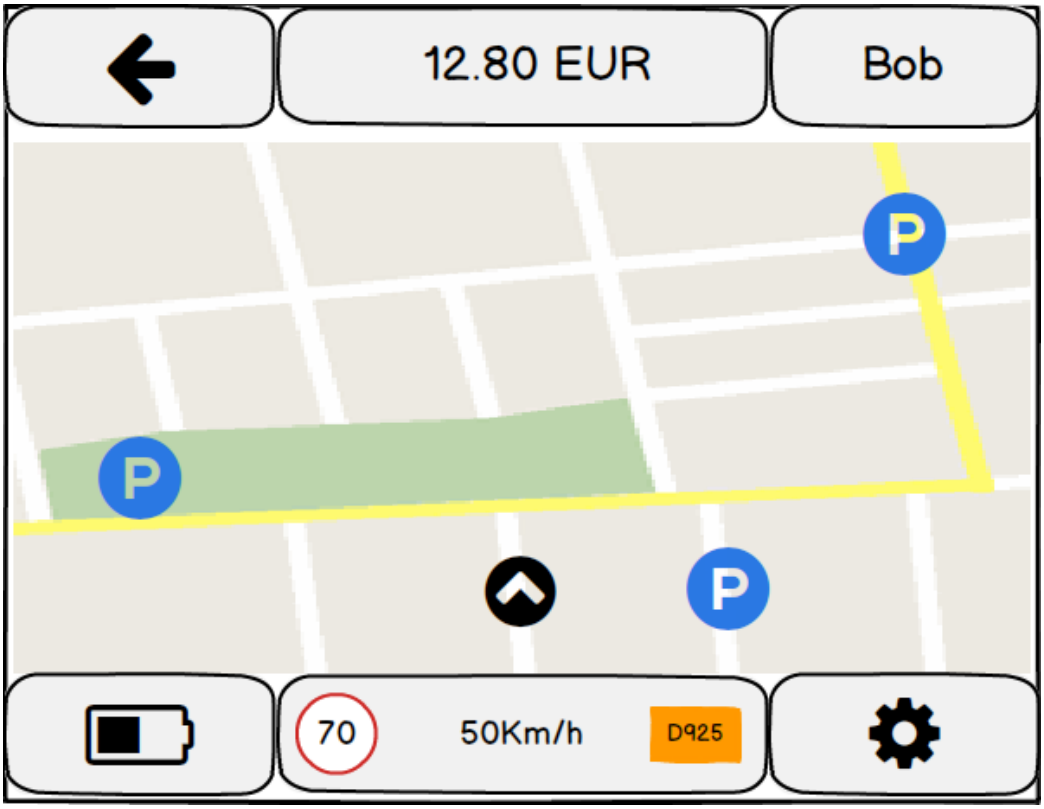

Figure 3.10: Concept for the car application.

Actor	Driver with or without guests (passengers)
Goals	G1.3, G3
Input condition	The driver is using the reserved car and wants to park it nearby his/her destination. He/she could be accompanied by some passengers.
Event Flow	1. The driver finds a parking spot via the car application. 2. The driver parks the car in an available spot of a safe area. 3. The driver turns off the car and gets out. If he/she is accompanied by someone, they get out too. 4. The system checks if the car can be locked and if discounts are applicable.
Output condition	The system locks the car, directly charges the driver via the payment method that he/she indicated during the registration process and notifies him/her that the rental has ended.
Exception	• The driver has an accident. • The car battery completly run out. • There's no parking spots available in a nearby safe areas.

Figure 3.11: Sequence diagram of the car parking process.

3.2.4.3 Associated functional requirements

1. The driver must park his/her car in a safe area.
2. The driver must get out from the car remembering to close all doors and to roll up all windows of the car, otherwise the system won't stop charging him/her, and after 5 minutes will send him/her a sms notification to remember him/her that the rental operation can't be ended correctly. After 30 minutes the system tags the car as "Out_of_service" and it will apply a fee of 20 euros to the driver.
3. Once the car is locked, the system must directly charge the driver via the previosly indicate payment method.
4. Once the car is locked, the system must send a message to notify the driver about rental costs. The rental is now concluded. The reservation page will be cleared and closed by the system.
5. Once the car is locked, the system checks its conditions. If battery power is lower than 25% or a component of it is not working correctly, the system must tag the car as "Out_of_service". Otherwise the system tags the car as "Available".
6. Other requirements for G3 only are:
   (a) car's sensors must detect correctly the number of passengers.
   (b) if the driver is accompanied by at least one passenger during the whole ride, the system will apply him/her a direct discount of 10%, otherwise there won't be any discount.

3.2.5 Service discounts and fees

3.2.5.1 Purpose

During the rental of a car, the driver can be rewarded with discounts or punished with fines. For instance, if a driver leaves a car with more than 50% of battery charge, the system applies a discount of 20% on the last ride.

3.2.5.2 Scenario 1

Bob is going too fast because he is late for his important meeting. He commits a serious infraction and gets a speeding ticket. The system pays his fine and directly charges him the same amount of the fine plus an addictional fee of 10 euros. Bob is not happy.

3.2.5.3 Associated functional requirements

Requirements for G4 are the following:

1. Discounts must be applied one at time on total rental price excluding any fee or fine. The total rental price is updated everytime a discount is applied.
2. Fines or fees are applied after any discount.
3. If a driver doesn't pick up a reserved car within 1 hour, the system applies a fine of 1 euro.
4. If a driver gets a fine, the system pays his/her fine and directly charges him/her by the same amount of the fine plus an addictional fee of 10 euros.
5. If a driver leaves a car with more than 50% of battery charge, the system applies a discount of 20% on the last ride.

3.3 Performance Requirements

1. The system must support at least 1000 connected driver at time.
2. 95% of requests must be processed in less than 5 s.
3. 100% of requests must be processed in less than 10 s.
4. There is no limit on the total number of registered drivers.

3.4 Alloy

```
// ----- SIGNATURES -----

sig Driver
{
        username: one Stringa,
        email: one Stringa,
        password: one Stringa,
        name: one Stringa,
        surname: one Stringa,
        birthday: one Stringa,
        address: one Stringa,
        phonenumber: one Stringa,
```

```
        position: lone Position,
        creditcard: one CreditCard,
        drivinglicense: one DrivingLicense,
}

sig Stringa {}

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

sig CreditCard
{
        ccnumber: one Stringa,
        money: one Int,
        expireddate: one Int,
}
{
        expireddate > 0
}

sig DrivingLicense
{
        licenseid: one Int,
        expireddate: one Int,
}
{
        expireddate > 0
}

sig Car
{
        licenseplate: one Int,
        batterylevel: one Int,
        position: one Position,
        carstatus: one CarStatus,
}
{
        batterylevel ≥ 0
        batterylevel ≤ 100
        (batterylevel ≤ 25) implies (carstatus = OutOfService)
```

```
56 }

58 abstract sig CarStatus {}
   one sig Available extends CarStatus {}
60 one sig Reserved extends CarStatus {}
   one sig InUse extends CarStatus {}
62 one sig OutOfService extends CarStatus {}

64 sig Rent
   {
66         reservationdate: one Int,
           begindate: lone Int,
68         enddate: lone Int,
           numofpassenger: one Int,
70         price: one Int,
           rentstatus: one RentStatus,
72 }
   {
74         reservationdate > 0
           reservationdate < begindate
76         begindate < enddate
           price > 0
78         numofpassenger ≥ 0
           numofpassenger ≤ 4
80 }

82 abstract sig RentStatus {}
   one sig InProgress extends RentStatus {}
84 one sig Completed extends RentStatus {}
   one sig ReservationAborted extends RentStatus {}
86
   // ----- FACTS -----
88
   //No drivers with the same username, driving license or email
90 fact UniqueDriver
   {
92         no disjoint d1, d2: Driver • (d1.username = d2.username or d1.email
   d2.email or d1.drivinglicense.licenseid = d2.drivinglicense.licenseid)
   }
94
   //No license without driver
96 fact LicenseWithoutDriver
   {
```

```
98          ∀ dl: DrivingLicense • one d: Driver • dl in d.drivinglicense
    }
100
    //No creditcard without driver
102 fact LicenseWithoutDriver
    {
104         ∀ cc: CreditCard • some d: Driver • cc in d.creditcard
    }
106
    pred show()
108 {
            #Driver ≥ 3
110         #CreditCard ≥ 2
            #Car ≥ 2
112         #Rent ≥ 5
    }
114
    run show for 10
```

4 Appendix

4.1 Used tools

To create the RASD we used the following tools:

- **MikTex**
  Distribution of the typesetting system LaTeX
  http://miktex.org/
- **TexStudio**
  OpenSource cross-platform LaTeX editor we used to write the RASD.
  http://texstudio.sourceforge.net/
- **StarUML**
  UML modelling tool we used to build the graphs
  http://staruml.io/
- **Balsamiq**
  The mockup builder we used to design the mockups.
  https://balsamiq.com/
- **Alloy analyzer 4**
  used to build strong and substantial models
  http://alloy.mit.edu/alloy/
- **GitHub desktop**
  Desktop application of the web-based Git repository hosting service. Used to collaborate in the team and to have a track of the changes.
  https://desktop.github.com/

4.2 Hours of work

This is the time spent redacting the RASD

- Lorenzo Binosi - 80 hours

References

[1] Software Engineering 2 Project, AA 2016/2017 - *Project goal, schedule and rules*

[2] Software Engineering 2 Project, AA 2016/2017 - *Assignments 1*

[3] IEEE Standard 830-1998: *IEEE Recommended Practice for Software Requirements Specifications*

[4] W3C, *HTML5 - W3C Recommendation 28 October 2014*, http://www.w3.org/TR/html5/.

[5] W3C, *Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification*, http://www.w3.org/TR/CSS21/

[6] Apple, *iOS Human Interface Guidelines*, https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/

[7] Google, *Android Developers - Design*, https://developer.android.com/design/index.html