Politecnico di Milano

Software Engineering II project:
PowerEnjoy

Requirements Analysis and Specification Document

Gregori Giacomo and Ruaro Nicola

December 11, 2016
Version 1.1

Contents

Abstract

The main purpose of this document is to give a specification of the requirements that our system has to fulfill adopting the IEEE-830 standard for RASD documentation. It also introduces functional and non-functional requirements via high level specification of the system. In the last part a formal model is presented using Alloy.
The information contained in this document is intended for the stakeholders and developers: for the stakeholders this document presents an useful description to understand the project development, meanwhile for the developers it's quite a comfortable way to match the stakeholders' requests and the proposed solutions.

Part I

Requirements analysis

Introduction

1.1 Purpose of the system

The purpose of this project is to develop a digital management system called PowerEnJoy.
PowerEnJoy is a car-sharing service which uses only eletric-cars and allow the users to easily find a car, thanks to the location services, and to use it.

1.2 Scope of the system

Users have to be registered to the system and provide credentials (as well as payment informations), then they can access the car-sharing service. They can easily find eletric-cars thanks to the location services and reserve them for up to an hour.
When an user is near to a vehicle that she want to drive, she contacts the system telling that she's nearby that specific car, then the system unlock the car letting her to get into it. Then the system automatically calculates the charge during the ride, notifying the user 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 lock the car that become available again.
The system should encourage virtuous behaviour of the users and to do that some discounts can be applied on their last ride. For example a discount of the 10% is applied if the user took at least two other passengers onto the car.

1.3 Objectives and success criteria of the project

After a first analysis on our average users, the main features that should be provided by our application are:

1. [G1] Registration
2. [G2] Log-in and session management
3. [G3] Account managment

4. [G4] Car localization
5. [G5] Reservation and reservation managment
6. [G6] Car un-locking and end ride
7. [G7] Payments and charges application

1.4 Actors Identification

- **Guest**: is a person that hasn't already registered to the system or an user that hasn't already logged in. She can only proceed with a new registration or log in.
- **User**: we believe that PowerEnJoy can be used by a wide range of people that need only to access the system for benefit. A person became an user after her registration to the system, when she provides her credentials and payment information.
  A tipical user is a person who want to easily move around in a social end eco-friendly way, usually only for short travels near the charging stations. Our scope is to create an easy-to-use and efficent system that make the users satisfied and willing to use PowerEnJoy.

1.5 Stakeholders Identification

- **Customers**: the purpose of the customer is to maximise the performances of the system and reach the biggest profit possible.
  In our project the main stakeholder is the professor. He expects us to develop a digital menagment system, PowerEnJoy, that provide the functionality 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. We want to apply in practice what we learn during lectures and became able to manage software engineering problems, solving them in a rigorous way and being as accurate as possible. Our project includes a Requirement Analysis and Specification Document (RASD), a Design Document (DD), a testing-related activity, an assessment of the effort and cost required for the development of the project, a code inspection and bug identification activity on an existing open souce project.
  We will develop our project as close as possible to a real application that is ready to be launched in the market.

Overview of the purposed system

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.

2.1 Product perspective

2.1.1 User interfaces

We want to build an immediate and minimal graphic interface, which can be understood at first sight also by inexperienced users. Pages will be user-friendly and the link between them will appear as a natural relationship. The user should be able to interact with the system in three ways:

- Web application: is strongly cross-platform and then accessible from any device that can browse the web
- Mobile application: accessible from smartphones and mobile devices in order to guarantee portability and ease of use
- On-Board computer: accessible from the inside of any PowerEnjoy car, it must be extremely straightforward and let the user focus on the actual interaction

A common and friendly UI should be provided to create a sort of logical connection between the three different applications and make the user feel comfortable.

2.1.2 Hardware interfaces

The web application has no other hardware constraints despite the ones specified in subsection 2.4.2, it should run on any device meeting such minimum requirements. Mobile application has to communicate with GPS, Antenna and WiFi modules in order to retrieve location and query the server, requirements easily met by any modern smartphone.
On-Board computers require a car with self-diagnostic and reporting capability.

2.1.3 Software interfaces

The web application should support devices running any modern browser while the mobile application will be developed and supported on iOS, Android and eventually WP.
On-Board computer's application is device-specific and will then be designed based on the ad-hoc hardware, embedded OS and APIs.

2.2 Product functions

The system that we are to develop must let users register and then login in order to manage their account and actively access the service. The provided functionalities are clearly enumerated in section 1.3 and here graphically described to ease the complete comprehension of the system.

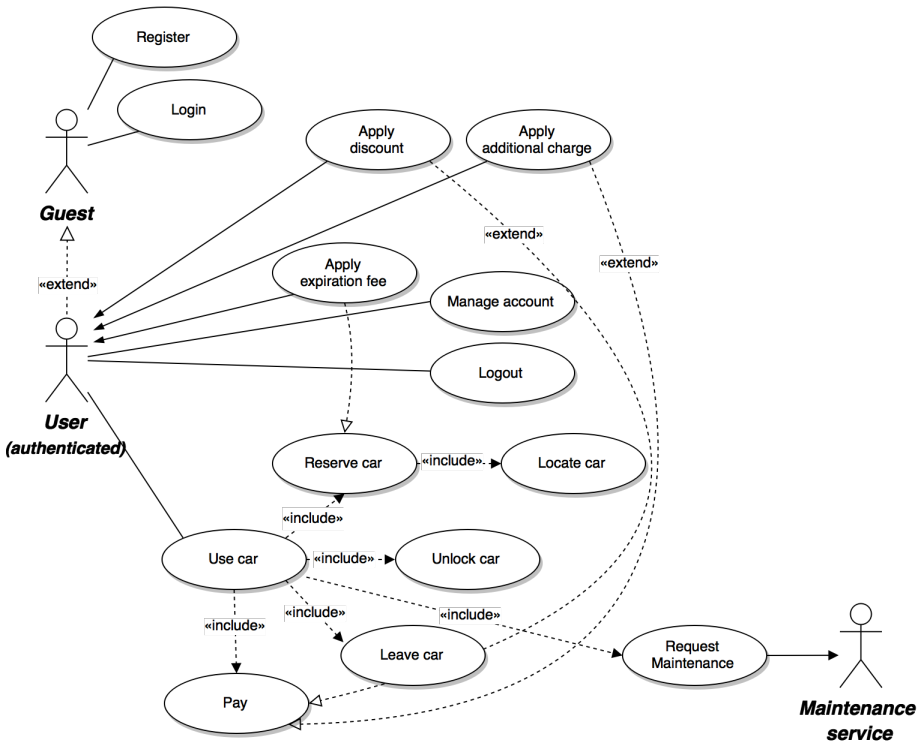


Figure 2.1: Main use-case representing all the functionalities of the system

2.3 User characteristics

The application should target users with a valid driving license, being at least 18 and providing a secure payment method, which must be checked before any reservation.
Additional legal requirements might be necessary, especially if the service is intended to spread into foreign countries

2.4 Constraints

2.4.1 Regulatory policies

Users must allow the system to collect and use personal data according to a privacy policy: data such as personal details, payment methods, transactions and locations are stored and processed in order to provide a higher service quality.

2.4.2 Hardware limitations

Hardware requirements are extremely different depending on the application that is being considered; in this document, though, the on-board computer's hardware won't be discussed. The web application development demand two main hardware constraints:

- Stable internet connection
- Modern(supported) web browser

The mobile application will be deployed for mobile devices, so even the memory constraints are consistents:

- ARM architecture
- At least 1GB of RAM
- At least 50MB of available storage
- Up-to-date operating system (minimum API level or OS version)
- Stable internet connection
- GPS module may be useful but not necessary to access the service

2.4.3 Parallel operation

The system must support heavy parallel processing because of the high number of users that are to access the service potentially at the same time.

2.4.4 Reliability requirements

The system has to be supported by an Internet connection that reliably depends by our provider.
It should be permitted to have little stops for maintenance. If occasionally the system has an unexpected stop however it can be accepted as the system doesn't cover a critical function.

2.5 Assumptions and dependencies

In order for the system to work properly some domain assumptions are needed:

- Sensors and devices needed to support the functionalities of the system are already installed on vehicles
- The system knows the battery level, location and number of passengers in each car
- Cars are equipped with a standard diagnostic connector and are using OBD-II communication protocols
- The GPS signal is sufficiently accurate ($\pm 5m$ accuracy with A-GPS)
- All the vehicles are insured
- PowerEnJoy runs only one type of vehicle
- The battery level always refers to the full-charge
- All other functions are locked while reserving a vehicle
- Users driving licenses are checked
- The vehicle is driven by the user that reserved it
- If a vehicle is damaged during the ride because of the user, the user is responsible for it
- If a vehicle is charging it's in a charging station
- There are no time or distance limits for a single ride
- During a ride the number of passengers in a vehicle doesn't change
- The system stores all users' information
- The charging stations are into safe areas
- The data center can't be improperly modified or hacked
- If a vehicle is not usable the system suggest another reservation and charges the previous user for the reparation costs

- There are PowerEnJoy employees that move and re-charge out-of-service vehicles
- If an unlocked car is not ignited within three minutes the system starts to charge the user
- The system uses a reliable external payment system
- Bills are payed once
- There are precedence rules regarding the fee/discount application

Developer Overview

Here a complete class diagram of our system is presented before splitting up the functionalities and provide an in-depth overview:

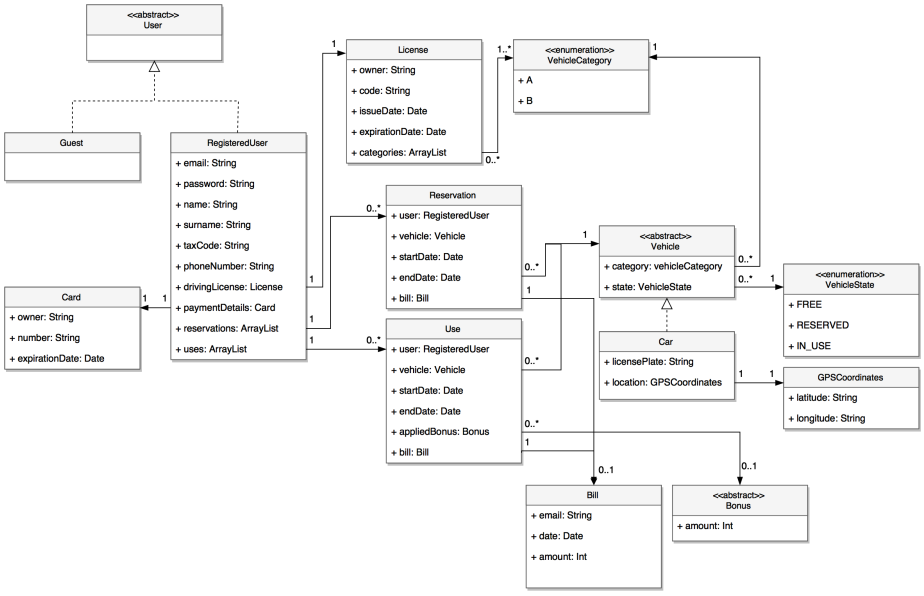


Figure 2.2: Complete class diagram of the system

Part II

Requirements specification

Specific requirements

3.1 External Interfaces

3.1.1 User interfaces

As stated in section 3.5.1 and subsection 2.1.1 the application's UI must be extremely user-friendly and functionally equivalent across devices.
Regarding the mobile UI a 3-page mockup is presented (fig. 3.1) to highlight the most important functions:

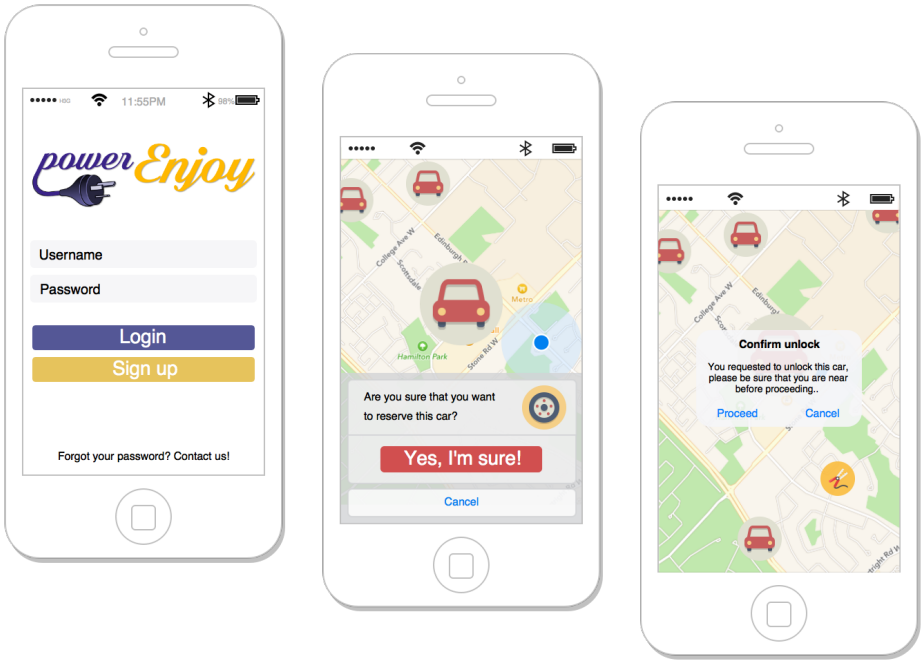


Figure 3.1: 3-page mockup representing the main pages of the mobile application

- Login: the first page presented to the end-user is a simple login/signup page where she can login using her credentials or register to the system
- Car reservation and localization: this is probably the most important use-case, the user will be able to browse and find nearby cars, eventually requesting a reservation
- Unlock: the user will request to unlock the car through a procedure similar to the one described in point 2 above; she must submit the car PIN code displayed on the vehicle's windscreen, then a confirmation will be asked to enforce security and advise the user to really locate the car before unlocking it

Additional in-detail mockups are presented in section 3.3 along with the use-cases specification.

The web application will be really similar in features to the mobile application, the main use-cases are in fact the ones presented above in figure 3.1; a basic mockup is presented (fig. 3.2) to provide completeness:

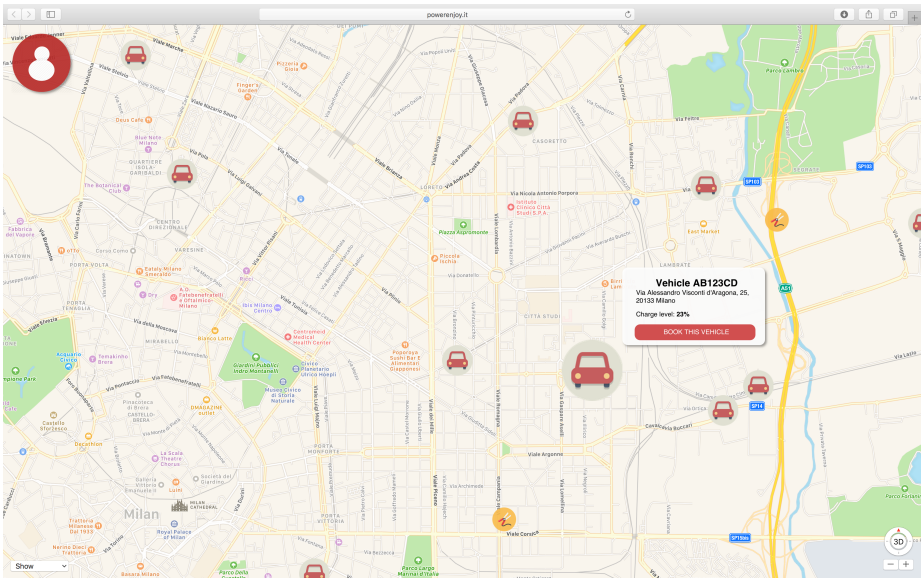


Figure 3.2: 1-page mockup representing the main page of the web application

3.1.2 Hardware interfaces

The on-board computer is equpped with a GPS. The navigation system and the map data are supplied by the Navigation Data Standard (NDS). The road database is stored in an hard disk, other media may be added by the system via internet access. It can receive and display information on traffic congestion using either TMC and RDS. The internet access is permitted by a USB modem with the Long-Term Evolution (LTE) standard that guarantee an high-speed wireless communication with the system. The on-board computer is connected with a standard diagnostic connector and are using the car's OBD-II communication protocols for self-diagnostic and reporting capability.

3.1.3 Software interfaces

- Database Management System
  - **Name**: MySQL
  - **Version**: 5.7.16
  - **Source**: http://www.mysql.com
- Java Virtual Machine
  - **Name**: JEE
  - **Version**: 7
  - **Source**: http://www.oracle.com/technetwork/java/javaee
- Application Server
  - **Name**: GlassFish
  - **Version**: 4.1.1
  - **Source**: https://glassfish.java.net
- Operating System
  - Application must be able to run on any SO which supports JVM and DBMS specified before

3.1.4 API interfaces

To provide location services we use the W3C Geolocation API combined with the GeoNames API for reverse geocoding.
As well described on the producer's website, this combination gives us an accurate way to retrieve the user position for any location on Earth.
Regarding the payment method we use the Stripe API, that gives a secure and fast way to tranfer money.
More information on dev.w3.org, geonames.org and stripe.com.

3.1.5 Communication interfaces

Protocol	Application	Port
TCP	HTTP	443
TCP	HTTPS	80
TCP	DBMS	3306(default)

3.2 The world and the machine

For a first domain analysis of the PowerEnJoy application we use "The World & The Machine" model by M. Jackson and 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 direclty by the application.

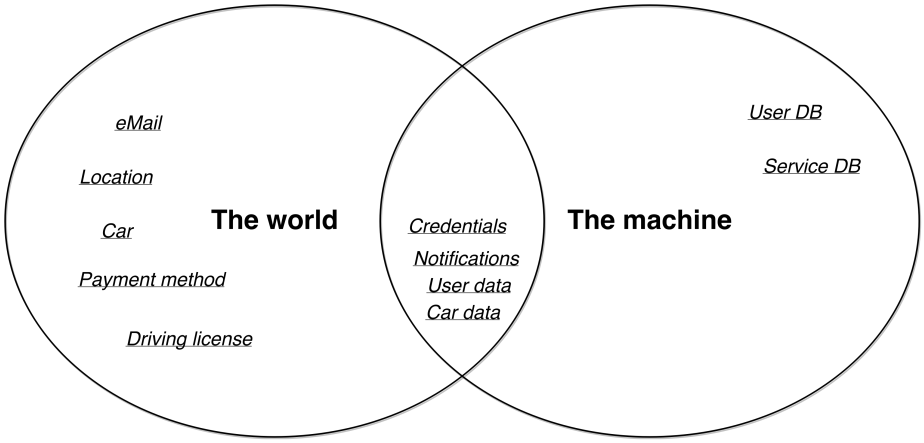


Figure 3.3: The world and the machine system model

3.3 System Functions

Looking at the objectives and success criteria of the project described in section 1.3, we can derive the functional requirements that PowerEnJoy have to implement. We choosed to group them by the actor involved in the function.

- Guest
  1. Registration
  2. Log-in
- User
  1. Manage account
  2. Create reservation
  3. Delete reservation
  4. Reservation Expiration
  5. Use car
  6. Charge ride
  7. Discounts & fees

Use-case	Actor	Brief description	Goal
1	Guest	Registration	G1
2	Guest	Log-in	G2
3	User	Manage account	G3
4	User	Create reservation	G4,G5
5	User	Delete reservation	G5
6	User	Reservation expiration	G5
7	User	Use car	G4,G6
8	User	Charge ride	G7
9	User	Discounts & fees	G7

3.3.1 Registration

Functional Requirements

- The user must insert valid informations.
- The user must provide valid payment information.
- The system must check all the form fields and create a new user.
- The system must give back a password that can be used to log-in.

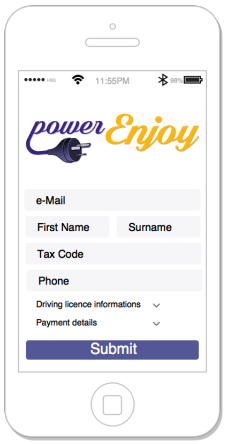
Scenario 1

Mark has been told that a new awesome car-sharing service just launched, he then decide to download the mobile application and register as a new user. After filling the sign-up module a confirmation e-mail is sent to Mark containing his new credentials.

Scenario 2

Susan was browsing the net when an advertisement caught her attention, it says: "PowerEnJoy is the new planet-friendly car sharing service! Register now and get a discount on your first ride!". She then clicks the advertisement and opens the web application: the first page is a Sign-up page and Susan fills up her personal informations and confirms. Unfortunately the informations provided are not complete and the system signals an error, she notices that the driving licence's informations are missing and compiles the remaining part of the module before submitting it and receiving a confirmation e-mail.

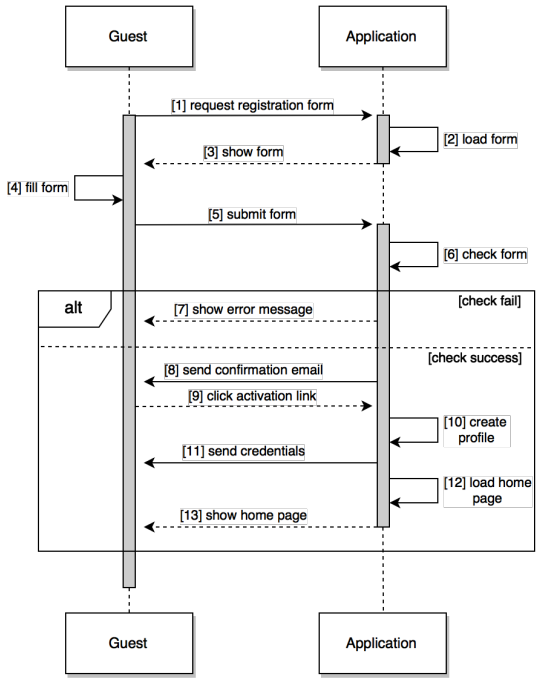
Mockups



Use-case table

Actors	Guest
Goal	G1
Entry conditions	The Guest enters the Sign-up page in the web/mobile application.
Flow of events	• The Guest is shown a form to fill with her personal informations (e-mail, name, surname, tax code, phone number, driving license, payment information) • The Guest fills the form and confirm the provided information. • The system sends a confirmation e-mail to the Guest's personal e-mail with an activation link. • The Guest clicks the link in the system's e-mail. • The system allocates the User in the database. • The system sends an e-mail to the User's e-mail with her personal pwd. • The system loads the User's homepage.
Exit conditions	The Guest is registered to PowerEnJoy and became an User. Now the User is in her homepage.
Exceptions	• The Guest does not fill one or more fields in the form (the system does not allow to proceed and signals an InformationLack). • The Guest provides in-use informations (the system signals an error). • The Guest provides wrong payment information (the system signals an error). • The Guest provides wrong driving license information (the system signals an error). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.2 Log-in

Functional Requirements

- The user must provide a valid ID.
- The user must provide a valid password.

Scenario 1

Susan has just registered to PowerEnJoy and loads the home page in order to login, she then enters her credentials and submit the form. The credential are checked and Susan is redirected to the web application as a logged user.

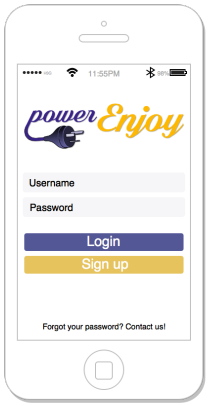
Scenario 2

Mark is a typical PowerEnJoy user, he got used to the credentials form submission and types his password quickly. The password is not recognized by the system and Mark is asked to check his credentials. He then types more accurately and finally gets redirected to the web application.

Scenario 3

A guest confused the login form with the registration one and after submiting the form is told that such user is not registered to the PowerEnJoy service. She then clicks on "Register to PowerEnJoy" and follows the registration procedure.

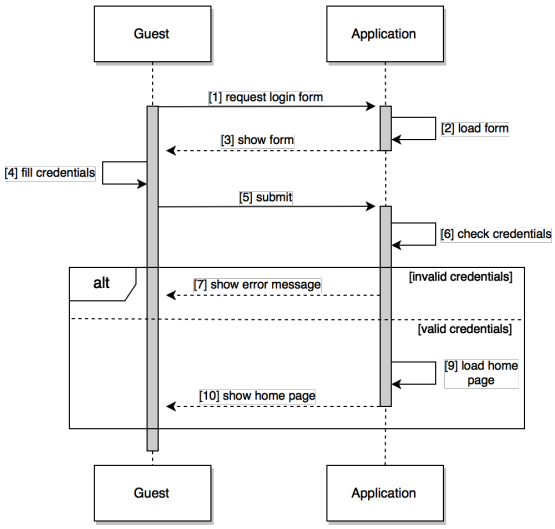
Mockups



Use-case table

Actors	Guest
Goal	G2
Entry conditions	The Guest is registered to PowerEnJoy and browse to the Log-in page in the web/mobile application.
Flow of events	• The Guest insert her ID and pwd. • The Guest clicks the Log-in button. • The system loads the User's homepage.
Exit conditions	The User is in the PowerEnJoy homepage.
Exceptions	• The Guest provides wrong username-password pair (the system signals a LoginError). • The Guest doesn't fill one of the fields (the systems signals an InformationLack). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.3 Manage account

Functional Requirements

- The system must store all the account information.
- The user must be able to modify the password.
- The system must update any changed information.
- he system shall check that all the informations are valid.

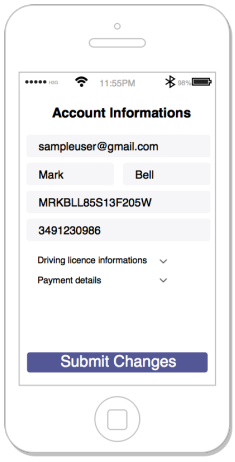
Scenario 1

After five months that Susan is using PowerEnJoyshe she has decided that is better to change her pwd. So she open her manage profile page and change it. Then she save the changes and the system upload Susan's information. Then for the future acesses Susan will use her new pwd.

Scenario 2

Mark is a typical PowerEnJoy user. Unfortunatly his credic card is expired and so he has to change it. He was using that credit card also for pay his PowerEnJoy's charges. So as soon as he has his new credit card he decides to upload his payment information on PowerEnJoy. He accesses his manage profile page and changed the payment information. Since that moment PowerEnJoy is going to charge Mark on his new credit card.

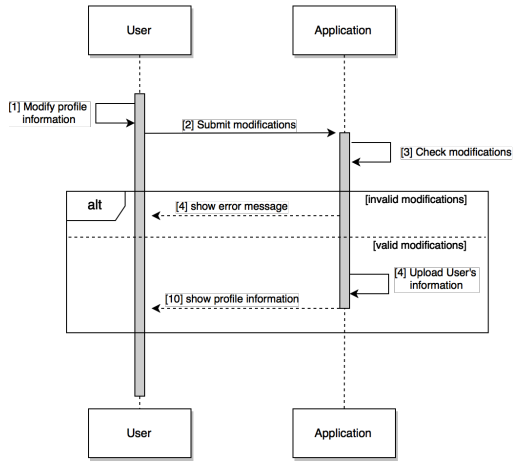
Mockups



Use-case table

Actors	User
Goal	G3
Entry conditions	The User is logged into the system and she is viewing her profile page.
Flow of events	• The User select the "Modify profile" option. • The User modifies one or more information. • The User submit the modifications. • The system checks the modifications and upload them.
Exit conditions	The User's profile is modified.
Exceptions	• The User provides empty or invalid values as modifications (the system signals an error). • The User doesn't submit the modifications (the system aborts the request). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.4 Create Reservation

Functional Requirements

- The system must correctly handle the availability of vehicles.
- The system must prevent an user from reserving more vehicles.
- The system must correctly switch the car state to FREE or RESERVED.
- The system must correctly register the reservation's time.

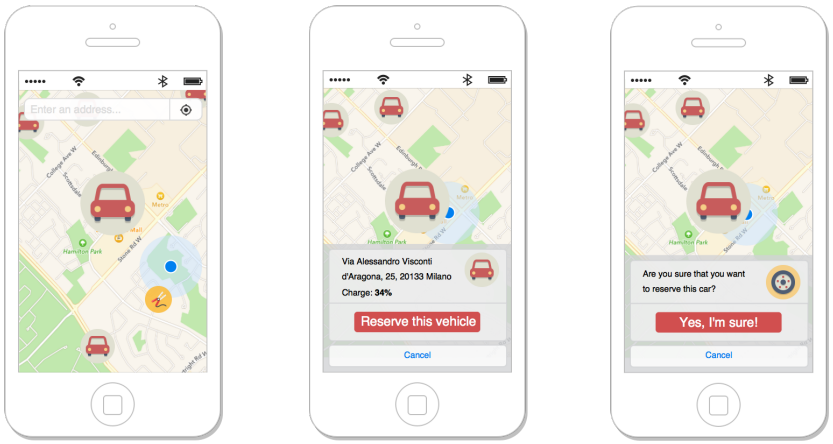
Scenario 1

After dinner Mark decides to go to the cinema and decides to use a PowerEnJoy's car. He then browses the application and reserves car near him. Unexpectedly Morgan, a friend of Mark, is having dinner in the same restaurant and when they see each other they start chatting. When they stop talking Mark notices on his PowerEnjoy app that in five minutes his reservation expires. So quickly greets Morgan and goes to unlock his car.

Scenario 2

Susan is a typical PowerEnJoy user, she was walking through a street when a friend texts her to hang out. Fortunately she saw a PowerEnJoy's car on the other side of the street. She reached the car and checked online: that's not available. Someone had probably already reserved it. But on the app she noticed that there were two avable car 5 minutes walk from her position. Then she reserved one of that two cars and easily reached it, thanks to the accurate location service provided by the system.

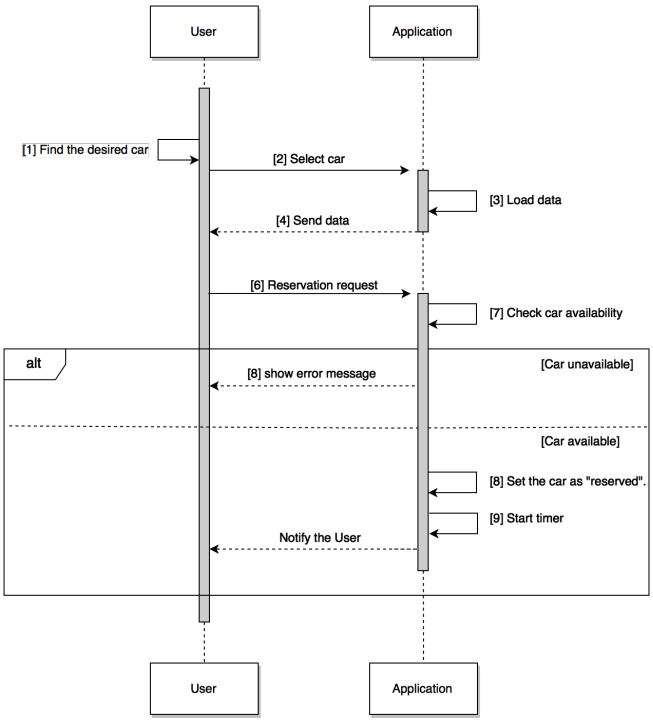
Mockups



Use-case table

Actors	User
Goal	G5
Entry conditions	• The User is in his homepage and want to reserve a car. • PowerEnJoy has available cars. • The User is not already reserving a car.
Flow of events	• The user has three ways for finding a car: – browse the map. – use her location. – enters an address. • The user selects the car that want to reserve. • the system displays the car's information. • The system turns the car "reserved". • The system notifies the user.
Exit conditions	• A car is set as "reserved". • The car is reserved for up one hour. After that the reservation expires and the user has to pay a fee of 1eur.
Exceptions	• The vehicle has already been reserved (the system signals an error). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.5 Expiration of the reservation

Functional Requirements

- The system must tag a car avaible again if it is not picked-up within an hour from the reservation.
- The system must apply a fee of 1 EUR if a car is not picked-up within an hour from the reservation.

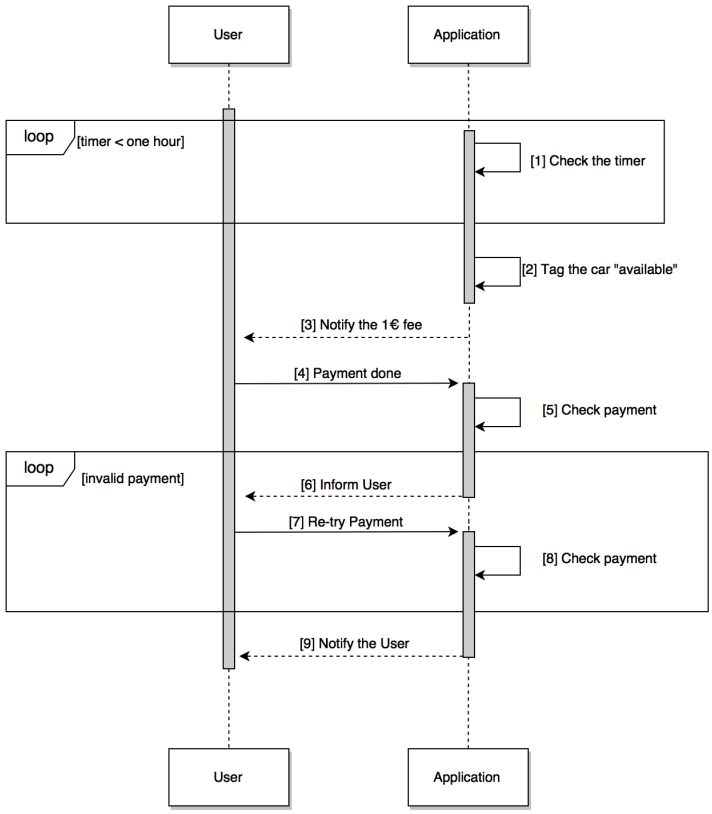
Scenario 1

Mark needs to go to the city center in the late evening and decides to use a PowerEnJoy's car. Just before having a quick lunch in a near restaurant he browses the application and reserves car near him to make sure that it will still be available when he'll be back. Unexpectedly Morgan, a friend of Mark, is lunching in the same restaurant and when they see each other they start chatting about everything and anything. When they stop talking Mark notices an e-mail from PowerEnJoy: it's about his reservation that has expired. His bank has also notified him that a fee of 1eur was applied by PowerEnJoy.

Use-case table

Actors	User
Goal	G5
Entry conditions	• The User has reserved a car. • The system has activeted the timer.
Flow of events	• The system checks the timer. If it's up an hour the reservetion is expired. • The system applies a fee of 1eur to the User. • The system notified the User. • The User pays the fee. • The system notifies the User of the success of the payment. • The system turn the car "available".
Exit conditions	• A car is set as "available". • The User's reservation is delated.
Exceptions	• The timer has some issues (the system signals an error). • The payment fails (the system signals an error). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.6 Delete Reservation

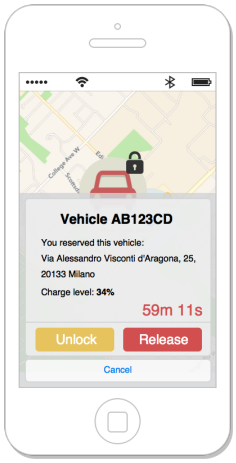
Functional Requirements

- The system must tag a car avaible again if a related reservation is deleted.
- The system must not apply any charge if the reservation is deleted before expiration.

Scenario 1

Michael is in a hurry because he has a meeting downtown in 20 minutes, he is really late and after locating a PowerEnJoy car he submits a reservation. He is about to unlock the car when he hears someone calling his name, a collegue recognized him and stopped to offer him a ride. Michael accepts but he immediately cancels his reservation in order to avoid the expiration fine and allow other users to take advantage of the service. The system processed his request and now the car is available again.

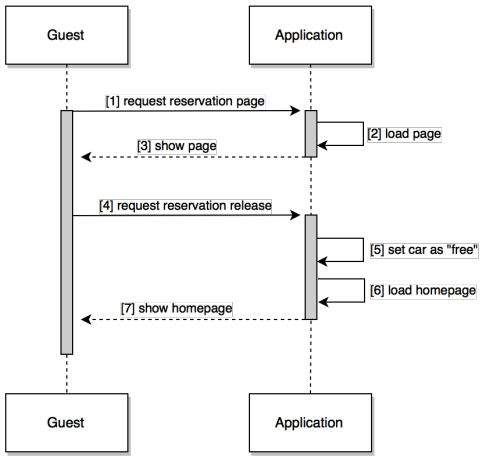
Mockups



Use-case table

Actors	User
Goal	G5
Entry conditions	The User has previously reserved a vehicle and wants to delete the reservation
Flow of events	• The User opens the application • The Users visualizes in his home page the active reservation • The User clicks "Delete" in order to release his reservation • The system processes his request and updates the car status • The system loads the User's homepage.
Exit conditions	The User is in his homepage but his last reservation has been deleted and the vehicle is free again
Exceptions	• The reservation is already expired (the system signals an error). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.7 Use Car

Functional Requirements

- The system must unlock the right car when an user requests to.
- The system must not unlock any car that does not match the user's reservation and the submitted vehicle-PIN.
- The system must switch the car state to "IN USE" only when the user unlocks it.
- The system must set the car state to "FREE" only when the user leaves it.
- The system must close the reservation as soon as the car state is set to "IN USE".
- The onboard computer must unlock the car commands if and only if the right PIN code has been inserted.
- The system must calculate the current charges and display it through the onboard computer.
- The system must correctly detect accidents and set the car state to "OUT OF SERVICE".
- The system must correctly detect when a car is outside the safe-area and prevent the user from ending the ride.
- If a car is left at more than 3Km from the nearest power grid station or with more than 80% of the battery empty, its state must be set to "OUT OF SERVICE" and an operator must charge it on-site and notify the system to set its state to "FREE" again.

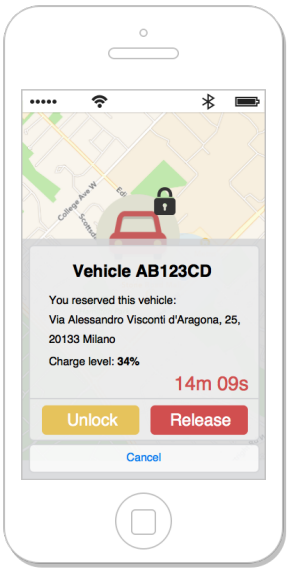
Scenario 1

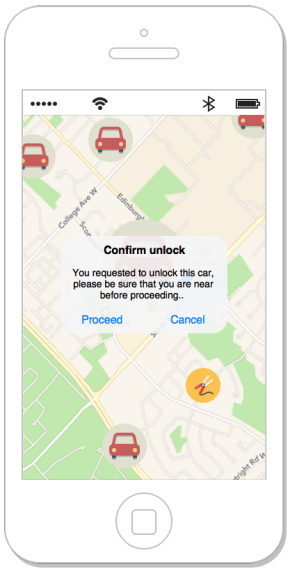
Anna has a pending reservation and is just in front of her PowerEnJoy car, she requests to unlock the car through the application end enters. She is then asked for her PIN code and after a correctness check the engine is unlocked and Anna starts driving. When at destination, she notices that the car is not parked inside the safe-area so she moves it until inside and leaves the car after parking it, immediately receiving a payment notification.

Scenario 2

Michael and his two brothers are far from home and its really late, the subway is already closed and since they have seen a PowerEnJoy vehicle near them, they decided to reserve it and proceed with the unlock procedure. After entering his PIN, Michael starts driving and once at home they park and leave the car. But the system notices that the battery level is below 20% and (even if two additional passengers were taken) Michael is charged 30% more.

Mockups

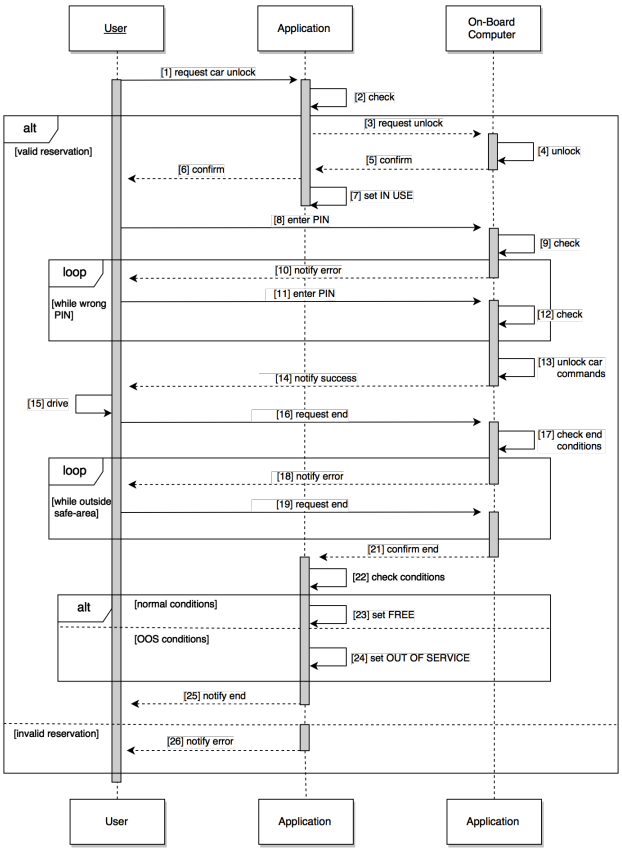

Vehicle AB123CD
You reserved this vehicle:
Via Alessandro Visconti d'Aragona, 25,
20133 Milano
Charge level: 34%
14m 09s
Unlock
Release
Cancel


Confirm unlock
You requested to unlock this car,
please be sure that you are near
before proceeding..
Proceed
Cancel

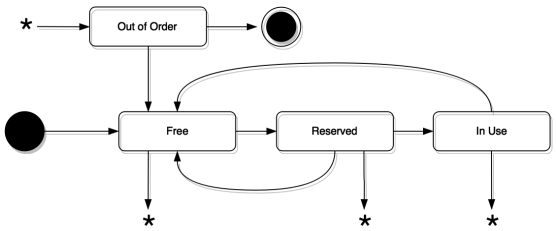
Use-case table

Actors	User
Goal	G1
Entry conditions	The Guest is near the car with a matching valid reservation active.
Flow of events	• The User requests to unlock the car through the application, the unlock request must include the vehicle-PIN. • The system checks the validity of the request and unlocks the car, setting its state to "IN USE". • The User enters the car and digits the PIN on the onboard computer. • The PIN is checked and the engine is unlocked. • The User uses the car and is showed the charges in real-time. • The User parks in a safe area and leaves the car in order to end the rental. • The system charges the user and tags the car as "FREE".
Exit conditions	The User has left the car and has no active reservations, he has been charged for the rental. The car state is "FREE".
Exceptions	• The reservation expires just before the unlock request (the system notifies an InvalidReservation exception). • An accident/engine fault occurs (the system tags the car as "OUT OF SERVICE" and requests maintenance). • The user doesn't leave the car in a safe area (the system tags the car as "OUT OF SERVICE").

Sequence diagram



State machine diagram



3.3.8 Charge Ride

Functional Requirements

- The system must charge the user using her payment information after each utilisation.
- The system must start to calculate the charges as soon as the engine ignites.
- The system must stop to calculate the final charges as soon as the car is parked in a safe area and the user exits the car.
- The system must start to calculate the charge as soon as the engine ignites.
- The system must calculate the current charges. The amount of money per minute is given.
- The system must notify the current charges to the user through a screen inside the car.

Scenario 1

Mark has just used PowerEnJoy for going from his house to his office. During his travel he frequently check the car's monitor in order to know the amount of the current charges. When he arrives in a park near his office, which is in a safe area, he turns off the engine and exits the car. At this point the system automatically lock the car and stops charging Mark. After a while Mark receives a notification by the system about the last ride, with travel information and the final charge. The bank inform Mark that the charge was applied on his bank account.

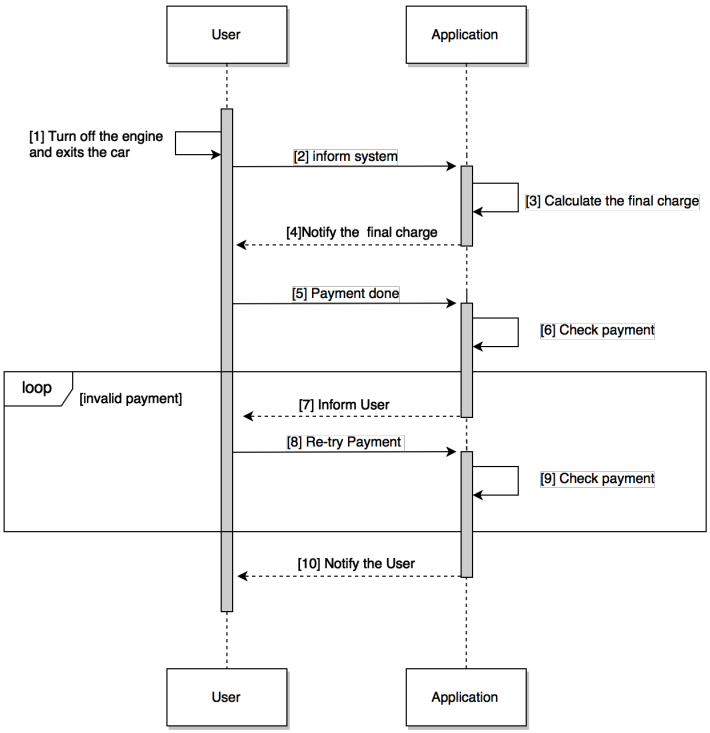
Scenario 2

During a sunny day Susan decides to use PowerEnJoy to go to the countryside. She takes a car and drives for an hour. When she arrives she notices that the system doesn't stop to charge her. After a quick check she discovers that it isn't a safe area. Unfortunatly for stopping the charge she has to drive back for a while and park the car in a safe area. She finds a bus and uses it for reach the countryside again. The PowerEnJoy service should be used mostly for travels around the city, the cars should be available alongside the city ready for being shared between the users.

Use-case table

Actors	User
Goal	G7
Entry conditions	• The user has turns off the engine and exits the car. • The system stops charging the user and locks the car.
Flow of events	• The system check the car location. • The system notify the user about her final charge. • The user does the payment. • The system notify the success of the payment
Exit conditions	The user has finished his ride and has payed PowerEnJoy for the service.
Exceptions	• The payment fails (the system signals an error). • The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.3.9 Discounts & Fees

Functional Requirements

- The system must apply a discount of 10% on the last ride if it detects that the user took at least two other passengers in the car.
- The system must apply a discount of 20% on the last ride if the car is left with no more than 50% of the battery empty.
- The system must apply a discount of 30% on the last ride If the car is left in a special parking area and the user takes care of plugging the car into the power grid.
- The system must charge 30% more on the last ride if a car is left at more then 3Km from the nearest power grid station or with more than 80% of the battery empty.

Scenario 1

Mark is a PowerEnJoy user. One day Jenny hang out with Tom. They decided to take a PowerEnJoy car for going in a club in the other side of the city. Mark is the only one with an account, so he reserves a car, unlocks it and drives. Near the club there is a charging station, so he decides to go there to park the car. When he turns off the engine and they exit the car he also plugs the car into the power grid. After a while Mark receivs an e-mail from PowrEnJoy. He opens it and happily finds out that a discount of 30% was applied on his last ride due to his virtuous behaviour.

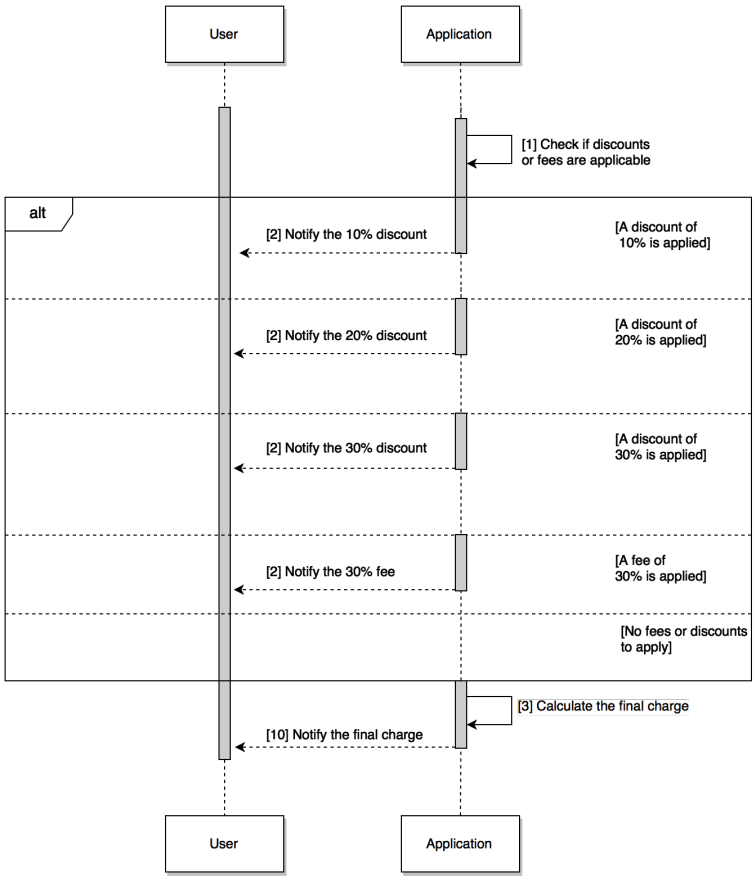
Scenario 2

Susan decides to go to the supermarket with her sister and her mother. She knows that using PowerEnJoy there'is a discount of the 10% the user took at least two other passengers onto the car. But when she arrives to the supermarket the battery of the car was 85% empty. That is a problem, because the cost for re-charge the cars on-site are very hight and Susan knows that and also know that the system charges 30% more in this kind of situations. So, when the notification of the final charge arrives she already knew that the charge is higher than expected.

Use-case table

Actors	User
Goal	G7
Entry conditions	• The User finishes his ride. • The system is calculing the final charge
Flow of events	• The system checks if one discount or one fee have to be apply to the final charge. • The system send a notification to the User.
Exit conditions	• A possible modification of the final charge. • The User has to do the payment now, as show in the "Charge ride" function.
Exceptions	• The system is not able to complete the operation due to some internal issues or connection broken (the system signals a ConnectionFailure).

Sequence diagram



3.4 Performance Requirements

The system shall support about 500000 terminals in the first implementation. At the same time the system should accept 1000 simultaneous users, and at least 90% of the transactions should be process in less then 3 seconds.
The amount of information handled by the system is on the order of few Terabytes. Most of the information can be dividen into location service info, cars status info, users info and reservations info.
The performance will also depend on the Internet connection's speed and reliability. In order to provide a service always available, the server should have a stable Internet connection with an adequate bandwidth. The interactions between the user and the system has to be reduced to a minimum, in order to not overload the net.

3.5 Software System Attributes

3.5.1 Usability

The use interface has to be user-friendly. It has to be simple, intuitive and well-organized. A typical user has not a previous knowledge of the system and we shall do it as easily to use as possible.

3.5.2 Reliability

The system must be opportunely reliable in order to support users while accessing the service, a 3-nines availability (~9hours/year) is more than enough for a non life-critical system.

3.5.3 Portability

Our system should be very portable due to the very wide range of users. It should be compatible to all the major hardware and software components. Our mobile application should work on the most used mobile OS. The web application needs to be supported by all the most used browsers. The code should be developed so that only a minor part should be adapted at the specific operation system.
The cars on-board computers OS have to be compatible with the system.

3.5.4 Maintainability

The maintainability of the system is guarantee by the administrator. The development of a 100% bug-free software is desiderable but impossible to achive, so the administrator has to fix the system every time is needed. The administrator can also access to the data of the system for manually modify or update them.

3.5.5 Consistency

In order to not lose the data in case of system fault they have always to be duplicate in a backup server.

3.5.6 Security

We need to apply security protocols at different levels for ensure a correct access to the data. First of all each user can access to her page using her personal ID and pwd, as esplain in the user interface. The pwd is gived by the system and should be an alphanumeric code of 8 digits, with at least two upper case letter, randomly selected. The pwd can be upload by the user, but it have to respect the 8-digits lenght and the two upper case letter have to be present.
Secondly every acess to the database have to filtered in order to avoid an uncorrect managment and the incosistency of the data.
Thirdly we have to guarantee a safe interaction between the cars on-board computers and the system.
Finally the connection with the system respect the https protocl, wich ensure the quality and the privacy of the connection.

Alloy modelling

4.1 Signatures

```
module PowerEnJoy


// SIGNATURES

abstract sig Bool{}
one sig True, False extends Bool {}

abstract sig Category {}
one sig A_CATEGORY, B_CATEGORY extends Category {}

abstract sig State {}
one sig FREE, RESERVED, IN_USE, OUT_OF_SERVICE extends State {}

sig Position {}
one sig SafeArea {
    coverage: set Position
}{#coverage>0}

sig ChargingStation {
    position: Position
}{position in SafeArea.coverage}

abstract sig BatteryLevel {}
one sig LOW, MEDIUM, HIGH extends BatteryLevel {}
```

```
abstract sig Vehicle {
    category: one Category,
    state: one State,
    position: one Position,
    batteryLevel: one BatteryLevel,
    plugged: one Bool
}{
    state=OUT_OF_SERVICE <=> not plugged=True and
        (batteryLevel=LOW or not (position in SafeArea.coverage))

    plugged=True implies (position in SafeArea.coverage)
}
sig Car extends Vehicle{}{category=B_CATEGORY}

sig License{
    categories: set Category
}{#categories>0}

sig User {
    license: one License,
    isLocked: one Bool
}

abstract sig BillType {}
one sig EXPIRATION_BILL extends BillType {}
one sig STD_BILL extends BillType {}
one sig DISCOUNT_BILL extends BillType {}
one sig FEE_BILL extends BillType {}

sig Bill {
    type: one BillType,
    payed: one Bool
}

abstract sig Event {
    user: one User,
    vehicle: one Vehicle,
    startTime: one Int,
    endTime: lone Int
}{
    startTime>0
    #endTime=1 implies endTime>startTime
}

sig Reservation extends Event{
    isExpired: one Bool,
    bill: lone Bill
}{
    isActive[this] implies isExpired=False
```

```
    isExpired=True implies not isActive[this]
    #bill=1 <=> isExpired=True
    #bill=1 implies bill.type=EXPIRATION_BILL
}

sig Ride extends Event{
    reservation: one Reservation,
    startPosition: one Position,
    endPosition: lone Position,
    hasAdditionalPassengers: one Bool,
    hasLeftLowBattery: one Bool,
    hasLeftHighBattery: one Bool,
    hasLeftPlugged: one Bool,
    bill: lone Bill
}{
    startPosition in SafeArea.coverage
    startPosition!=endPosition
    hasLeftHighBattery=True implies not hasLeftLowBattery=True
    hasLeftLowBattery=True implies not hasLeftHighBattery=True
    isActive[this] implies (not hasLeftLowBattery=True and
                            not hasLeftHighBattery=True)
    #bill=1 <=> #endPosition=1
    #endPosition=1 <=> not isActive[this]
    bill.type!=EXPIRATION_BILL
    bill.type=DISCOUNT_BILL <=>
        not isActive[this] and
        (endPosition in SafeArea.coverage) and
        (hasAdditionalPassengers=True or
         hasLeftHighBattery=True or
         hasLeftPlugged=True)
    bill.type=FEE_BILL <=>
        not isActive[this] and
        not bill.type=DISCOUNT_BILL and
        (not(endPosition in SafeArea.coverage) or
         hasLeftLowBattery=True)
    bill.type=STD_BILL <=>
        not isActive[this] and
        not bill.type=DISCOUNT_BILL and
        not bill.type=FEE_BILL
}
```

4.2 Facts

```
// FACTS

// all reservations are assigned to at most one ride and have coherent user/vehicle
fact ReservationMatchRide {
    no disj r1,r2:Ride | r1.reservation=r2.reservation
    all ride:Ride | ride.user=ride.reservation.user
    all ride:Ride | ride.vehicle=ride.reservation.vehicle
}

// no more active events from the same user
fact NoEventOverlap {
    no disj e1, e2:Event | overlap[e1, e2] and e1.user=e2.user
    no disj e1, e2:Event | overlap[e1, e2] and e1.vehicle=e2.vehicle
}

// all licenses belong to one user
fact LicenseMatchUser {
    all l:License | one u:User | u.license=l
}

// users cannot reserve/drive vehicles without the correct license
fact LicenseMatchVehicle {
    all e:Event | e.vehicle.category in e.user.license.categories
}

// all rides must have a valid reservation associated
fact NoRandomRide {
    no r:Ride | isActive[r.reservation]
    no r:Ride | r.reservation.isExpired=True
    all ride:Ride | ride.startTime>=ride.reservation.endTime
}

// banned users cannot reserve/drive cars
fact NoLockedUserAction {
    no e:Event | isActive[e] and e.user.isLocked=True
}

// vehicle state should be consistent
fact VehicleStateConsistency {
    all v:Vehicle | v.state=FREE or v.state=OUT_OF_SERVICE <=>
                        (no e:Event | e.vehicle=v and isActive[e])
    all v:Vehicle | v.state=RESERVED <=>
                        (one r:Reservation | r.vehicle=v and isActive[r])
    all v:Vehicle | v.state=IN_USE <=>
                        (one r:Ride | r.vehicle=v and isActive[r])
}
```

```
// all bills are assigned to a single ride or reservation
fact NoRandomBill {
    no disj r1,r2:Ride | #r1.bill=1 and #r2.bill=1 and r1.bill=r2.bill
    no disj r1,r2:Reservation | #r1.bill=1 and #r2.bill=1 and r1.bill=r2.bill
    no r1:Ride, r2:Reservation | #r1.bill=1 and #r2.bill=1 and r1.bill=r2.bill
    no b:Bill | no r1:Ride, r2:Reservation |  (#r1.bill=1 and r1.bill=b) or
                                              (#r2.bill=1 and r2.bill=b)
}

//if a vehicle is charging is in a charging station
fact ConsistentCharging {
    all v:Vehicle | v.plugged=True implies
                    (some c:ChargingStation | c.position=v.position)
}

//if vehicle was used its position should match with last ride endPosition
fact ConsistentVehiclePosition {
    all v:Vehicle | (some r:Ride | r.vehicle=v) implies
                    (some r1:Ride | isLast[r1] and (r1.endPosition=v.position))
}

//if vehicle was used its plugging state should match with last ride plugging state
fact ConsistentVehiclePlugging {
    all r:Ride | some c:ChargingStation | r.hasLeftPlugged=True implies
                                          r.endPosition=c.position
    all v:Vehicle | (some r:Ride | r.vehicle=v) implies
                    (some r1:Ride | isLast[r1] and
                                    (r1.hasLeftPlugged=True <=>
                                     v.plugged=True))
}

//if vehicle was used its batteryLevel should match with last ride batteryLevel
fact ConsistentVehicleBattery {
    all v:Vehicle | (some r:Ride | r.vehicle=v) implies
                    (some r1:Ride | isLast[r1] and
                                    (r1.hasLeftHighBattery=True <=>
                                     v.batteryLevel=HIGH))
    all v:Vehicle | (some r:Ride | r.vehicle=v) implies
                    (some r1:Ride | isLast[r1] and
                                    (r1.hasLeftLowBattery=True <=>
                                     v.batteryLevel=LOW))
    all v:Vehicle | (some r:Ride | r.vehicle=v) implies
                    (some r1:Ride | isLast[r1] and
                                    (r1.hasLeftHighBattery=False and
                                     r1.hasLeftLowBattery=False <=>
                                     v.batteryLevel=MEDIUM))
}
```

```
// no two distinct objects with the same position
fact ConsistentPosition {
    no disj v1,v2:Vehicle | v1.position = v2.position
    no disj c1,c2:ChargingStation | c1.position = c2.position
}
```

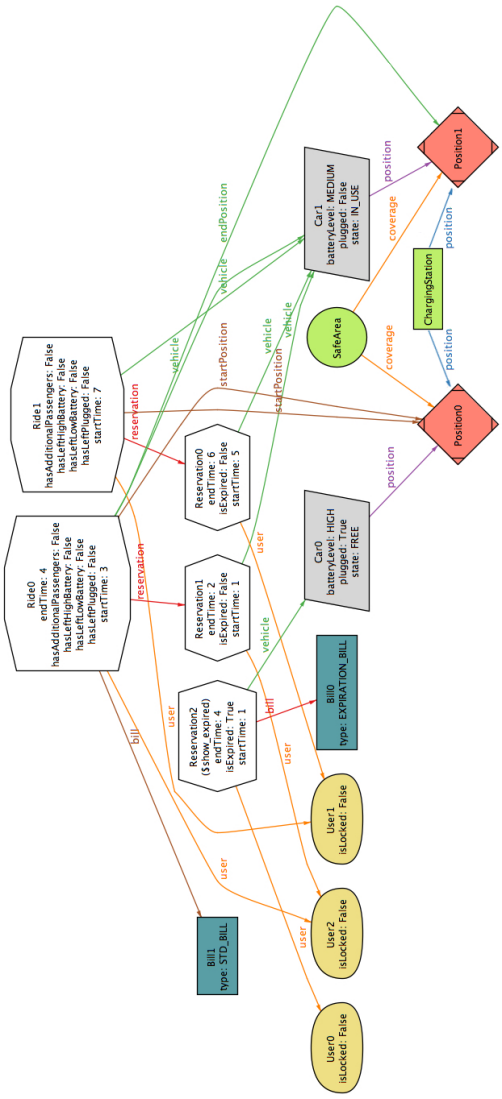
4.3 Predicates

```
// PREDICATES

pred isActive [e: Event]{
    #e.endTime=0
}

pred overlap [e1,e2: Event]{
    e1.startTime<=e2.startTime and e2.startTime<=e1.endTime or
    e2.startTime<=e1.startTime and e1.startTime<=e2.endTime or
    isActive[e1] and isActive[e2]
}

pred isLast [r:Ride] {
    all r2:Ride | r2!=r and r2.vehicle=r.vehicle implies r2.endTime<r.endTime
}

pred show {
    #Ride>1
    some expired:Reservation | expired.isExpired=True
}


run show for 3 but 5 Event
```

4.4 Results

Executing "Run show for 3"
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
5642 vars. 302 primary vars. 13223 clauses. 151ms.
Instance found. Predicate is consistent. 49ms.

4.5 Generated World



Appendix A: Used Tools

A.1 LaTeX

Used to format and redact this document

A.2 *g*it

Used as version control system in order to lead development

A.3 *d*raw.io

Used to draw mockups and diagrams

A.4 *A*lloy analyzer

Used to analyze and verify our specification

Appendix B: Hours of work

These are the hours of work spent by each group member in order to redact this document:

- Ruaro Nicola: 30 hours
- Gregori Giacomo: 30 hours
- Total worktime: 60 hours

Appendix C: Revisions

These sections will be eventually redacted during future post-release updates in order to approach the RASD modifiability providing a comfortable and higly effective way to trace changes:

C.1 Goals

- **v1.1** Goals have been rephrased and reordered: "User's account and session management" split into "Login and session management" and "Account management".

C.2 Assumptions

- **v1.1** Assumptions have been rephrased and reordered. Added "During a ride the number of passengers in a vehicle doesnt change".

C.3 Use-cases

- **v1.1** Specified that to unlock a car an user needs to input the vehicle-PIN.

C.4 Diagrams

C.5 Alloy

- **v1.1** Added fact for "If a car is charging is in a charging station".

Bibliography

[1] IEEE Std 830, *Recommended Practice for Software Requirements Specifications*, 1998

[2] Luca Mottola and Elisabetta Di Nitto, *Software Engineering 2: Project goal, schedule and rules*, 2016