POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2 PROJECT WORK

———————

PowerEnJoy Project:
Requirements Analysis and Specifications Document

Version: 1.3

Autohrs:

Jacopo FANTIN (mat. 878723)

Francesco LARGHI (mat. 876928)

Professors:

Elisabetta DI NITTO

Luca MOTTOLA

November 13, 2016

Contents

1. Introduction

1.1 Description of the problem

We will project PowerEnjoy, which is a digital management system for a car-sharing service that exclusively employs electric cars. It is based on a webapp, a car interface and a mobile application that together allow to reserve and use an electric car. Clients can use GPS mobile position to find the closest car or alternatively they can insert it manually both in the mobile application or in the Desktop interface. The cars are supposed to be recharged at specific Power Grid Stations when in need, placed all around the town where the service is offered. The clients must register to the system to reserve a car and use it, providing their credentials and payment information. They receive back a password that can be used to access the system and switch on the reserved car.

1.2 Goals

1.2.1 Generic Goals

The project's main purposes can be listed as follows:

- Provide a link between users and the car-sharing service

- Offer an easy and effective tool to the clients in order to manage the car employment

- Allow users to find the nearest car available and even reserve it for a limited time

- Charge users for a given amount of money per minute and notify them of the current charges through a screen on the car

- Allow users to leave the car where they want within a specified safe area

- Apply discounts to incentivize the virtous behaviors of the users

1.2.2   Specific Goals

We have specific goals for each actor. The system must:

G1 Allow the guest to register to the system

G2 Send a e-mail to a guest submitting a registration containing a password to access the system

G3 Allow the guest to log into the system

G4 Load a map indicating all the available cars near the detected position or specified adress

G5 Allow the user to reserve a car up to one hour before it is picked up

G6 Let the user know how much time left there is in his reservation. After one hour the reservation expires and the system applies a fee of 1 euro to the user

G7 Allow the user open a reserved car near him

G8 Allow the user starts the car with his personal password

G9 Charge the user for a given amount of money as soon as the engine ignites and show all the details about car, ride and locations through a screen in the car

G10 Lock the car and stop the time count as soon as the user leaves it

G11 Wait 10 minutes to detect possible power grid connection

G12 Calculate final discounts or fees and detuct money from the account and make it available again

1.3   Glossary

2. Preliminary considerations and assumptions

2.1 Domain properties

We suppose that all of these properties hold in the analyzed world:

- Both the mobile and car GPS give always the right position

- The users always have enough money in their credit card to pay the ride

- Vehicle plates are unique

- Users always insert correct credentials, actually corresponding to the real world

- Users have always updated documents and driving licence allowing them to drive Power Enjoy's cars

- Each e-mail used must exist and correspond to only one user

- Users normally never violate the Terms of the Contract accepting it in the registration form (even if there is a Support System to manage any kind of problem, even legal ones)

2.2 Text assumptions

- User's password is unique and used both to log into the system and switch on the car

- Users also have to provide their driving licence informations during registration

- The system is able to detect how many passengers are on board

- The system waits some minutes (we can assume 10) before detucting money from the user's account after last ride in order to detect if the car was plugged into a power grid station or not (and eventually apply the proper discount)

- We assume that there is a Support System, but it is not directly linked to our system (only to the database)

- There are operators that have to recharge cars on-site in case of empty battery (or less than 20%), but they are not handled by the Support System

- We assume that we need only few principle informations about the vehicle, like identification code (vehicle plate), position and battery (other kind of attributes used for the maintenence are hear omitted)

- We assume that there is a car interface to our application, connected to our Application Server, but we don't need to implement it.

3. Proposed system

We will implement both a web and a mobile application with a client-server-database architecture. Users can use whether browser or mobile app with the same account to communicate with the unique application server and make their registration in order to login and reserve a car. To open and use a car, users must use the mobile application. There is a car interface showing all the useful informations during the ride. We will have also a database connected to our application server. There is another system connected only to our database that can read the informations about all the cars. This is the Support System with its operators who have to charge cars on-site if they are left with less than 20% of battery, call the tow truck if necessary or any other kind of maintenence and support to the users, even legal if someone disobey to the Terms of the contract (for example leaving the car outside the Safe Area or damaging the car). This system is not directly linked to our main system. We will not model this system in our documents.
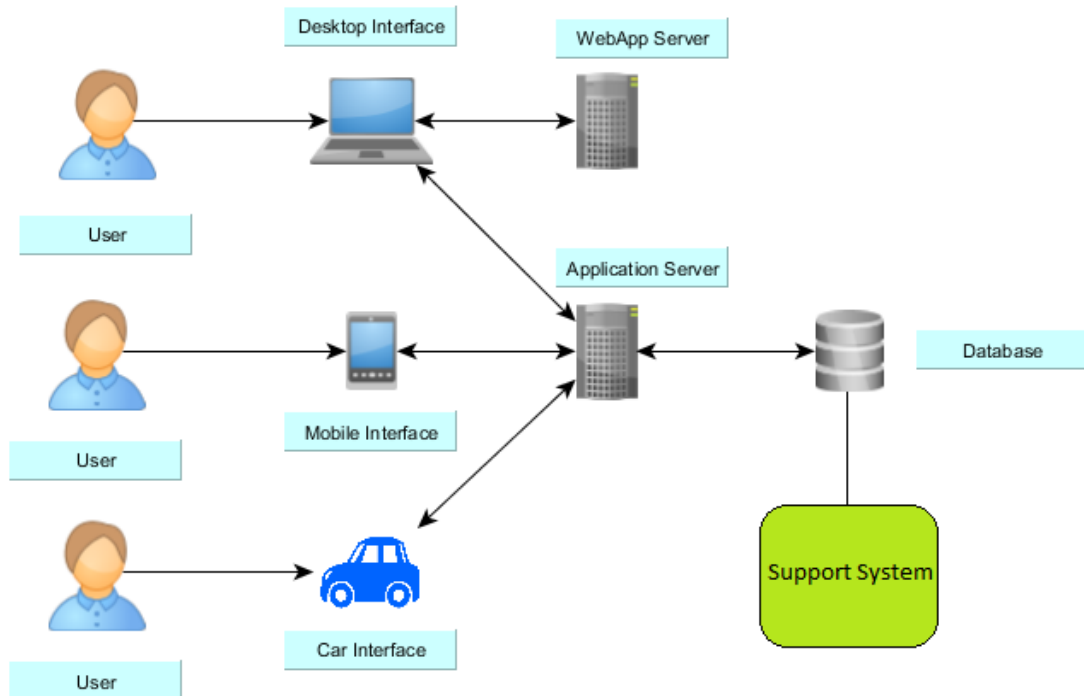
4. Actors identification

The actors of the system represents all the entities that interact with our platform. We basically have two actors:

- GUEST: he can register inserting his credentials and payment informations in order to have a password to log into the system as an User

- USER: he can view the map,find and reserve a car, use it with his personal password and use all the other included features of the application

5. Requirements

5.1 Functional requirements

We want the actors to interact properly with the system and satisfy all our goals. In order to do this all of these aspects for each actor must be true:

- **GUEST**:

  G1 :

    1. Valid e-mail and credentials inserted
    2. Valid payment informations inserted
    3. Valid driving licence inserted
    4. Guest agrees with privacy conditions of the system

  G2 :

    1. The registrations steps are successfully completed

  G3 :

    1. The guest must be actually registered to the system
    2. Credentials and password inserted must be valid
    3. He can reset the password through a new e-mail if he doesn't remember it

- **USER**:

    G4 :

    1. All available cars and their position are known
    2. The adress inserted/detected is valid
    3. There is a proper algorythm that decides which are the nearest cars to the given position

    G5 :

    1. An available car was selected by the user
    2. As soon as the car is selected it is not more available for other reservations, now it is reserved
    3. A timer starts waiting for 1 hour

    G6 :

    1. An available car was successfully reserved
    2. The user is notified every 15 minutes of the remaining time before he have to pay the fee

    G7 :

    1. The user position is close to the reserved car
    2. In the moment he tries to open it the timer is not still expired
    3. The car is actually the same car he reserved

    G8 :

    1. The car is still the same car he has reserved and opened
    2. The password inserted is valid and corresponds to the reservation

    G9 :

    1. The car's engine actually ignites
    2. An amount of money per minute is fixed
    3. The car display shows the currently charged money
    4. The amount is updated every minute
    5. The display shows remaining battery, covered distance and passed time
    6. The display shows the safe area map and the power grid stations

G10 :

1. The car must be left in the safe area
2. Every passenger on board gets out of the car

G11 :

1. The car is successfully locked and empty
2. The user has to plug the car in to the power grid station by 10 minutes

G12 :

1. The system applies a discount of 10% when the car detects at least two other passengers onto the car
2. The system applies a discount of 20% when the car is left with more than 50% of battery
3. The system applies a discount of 30% when the car detects that it is connected to a power grid station
4. The system charges 30% more on the ride when the car is left at more than 3 Km from the nearest power grid station or with less than 20% of battery

5.2   Non-functional requirements

5.2.1   Mobile Interface

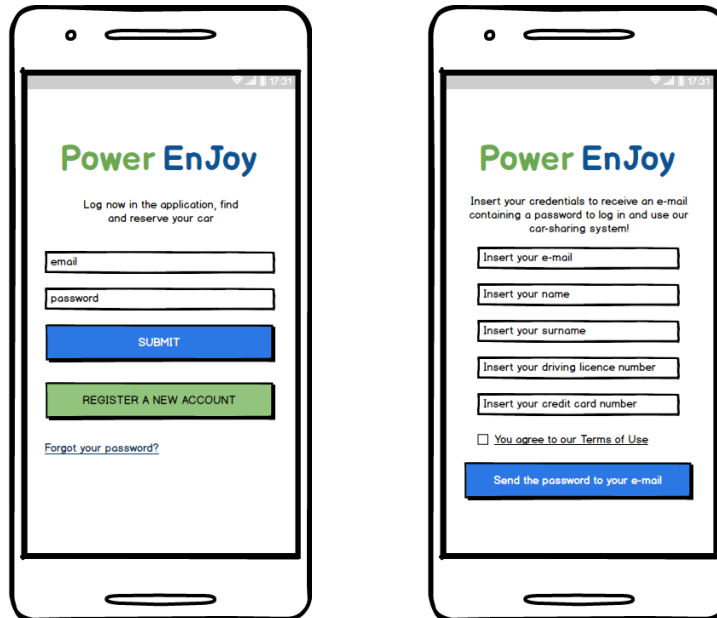These are some mockups to have an idea of how the mobile app should look like

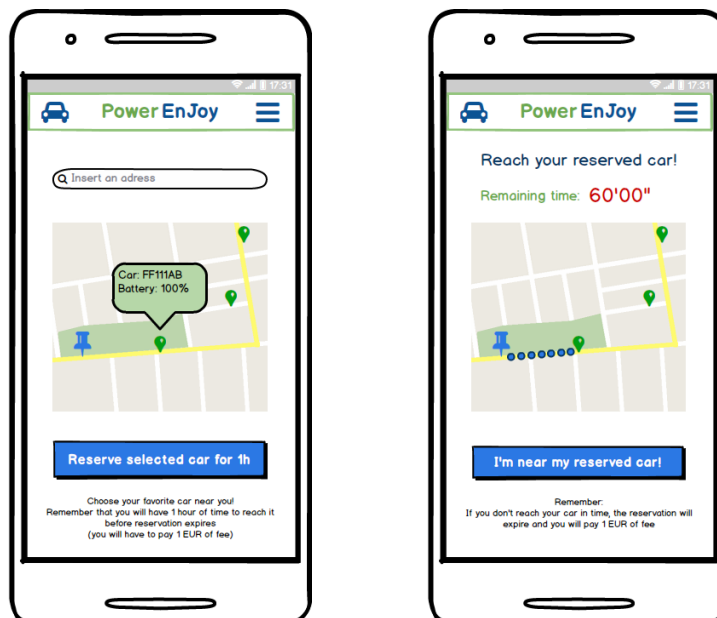Figure 5.2.1: Login and Registration forms on mobile

Figure 5.2.2: Reserve a car and open it on mobile

5.2.2   Desktop Interface
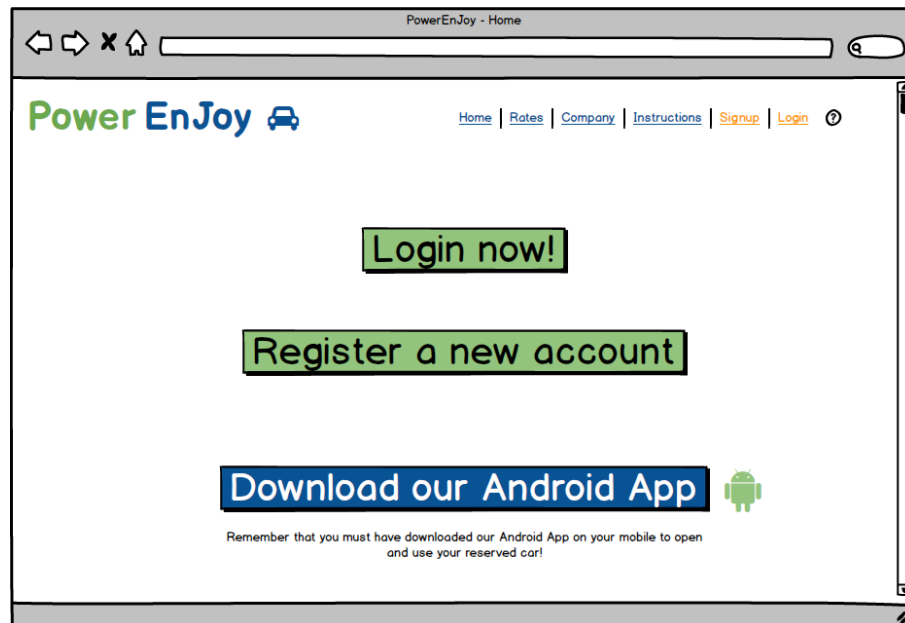
This is how the desktop site should be
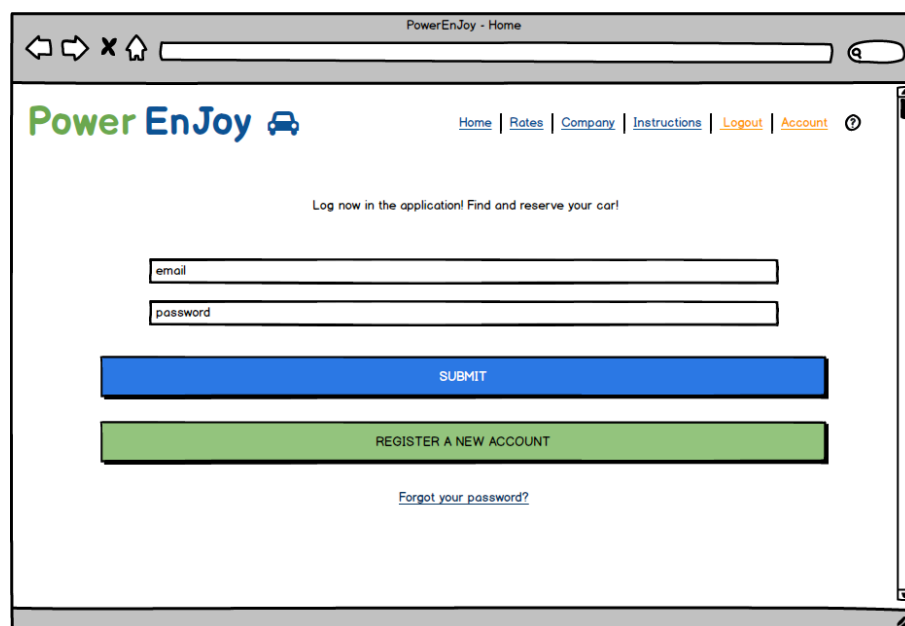

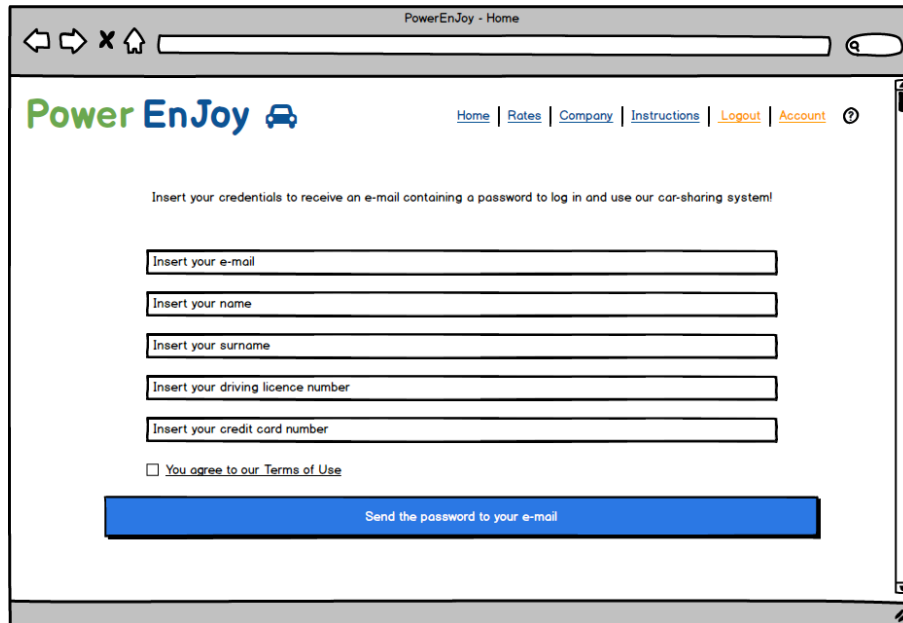Figure 5.2.3: A sample of the Desktop Home Page

Figure 5.2.4: A sample of Desktop Login Page

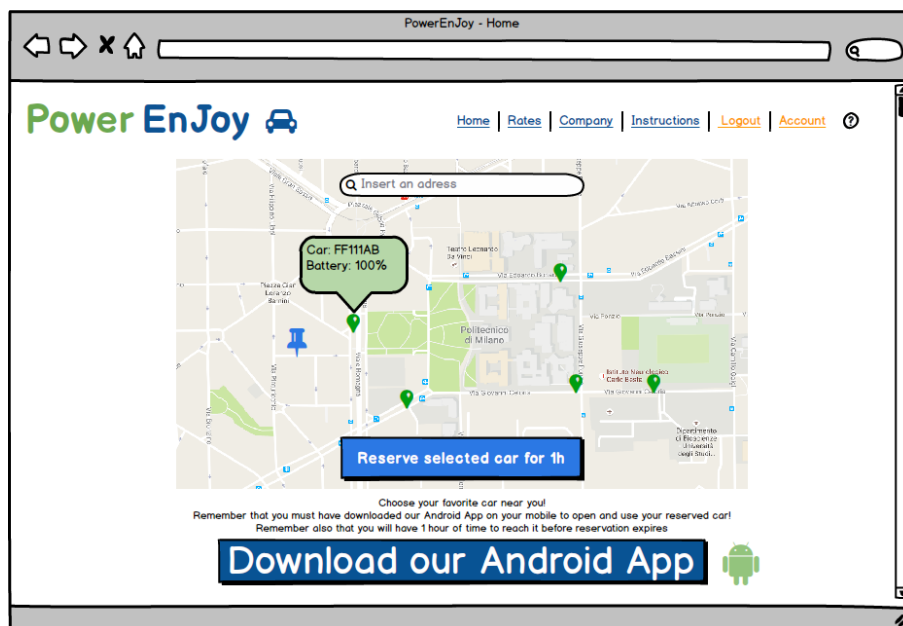Figure 5.2.5: A sample of the Desktop Registration Page

Figure 5.2.6: A sample of th Desktop Reservation Page

5.2.3 Car Interface

Figure 5.2.7: This is a sample of how the screen on the car should look like during the confirmation of the password
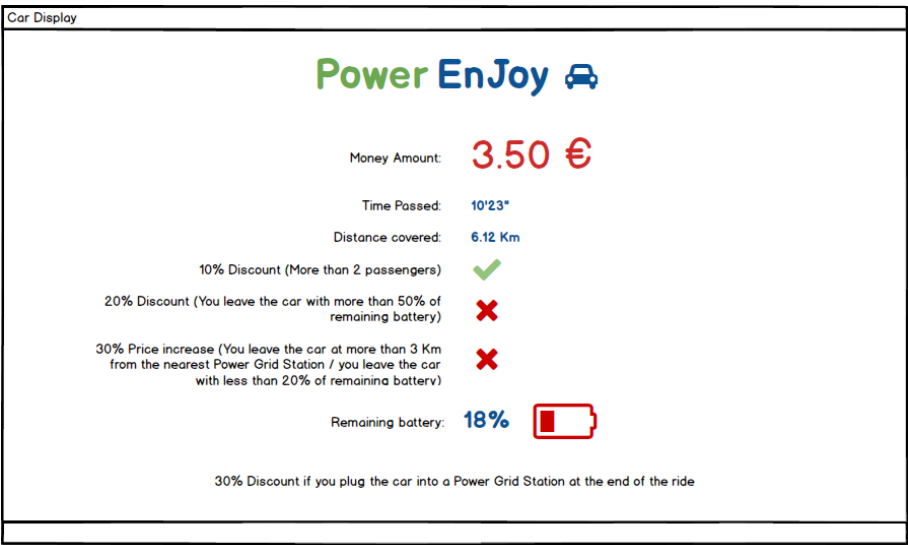

Figure 5.2.8: A sample of the car display during the ride, showing Money Amount rising and all the other info and discounts

5.2.4   Documentation

We will draft these documets in order to have a complete description of our system:

- Requirements Analysis and Specifications Document (RASD): contains the description of the scenarios, Uml and Alloy Models, goals and requirements of the system.

- Design Document (DD): contains the Architecture, the Components and design decisions of the system.

- Project Testing Document: contains all the specifications and details about the testing of the system.

- Project Management Documentation: contains assigment of tasks, goals to reach in order to develop the real system correctly.

5.2.5   Hardware and Software Considerations

We will use the following technologies:

- Apache server

- Mysql as sql server to store data persistently in the Database

- REST and JSON for API communication over HTTP

- Javascript (with AngularJs framework), CSS and HTML to create responsive site that communicate to server using REST API and HTTP

- Modern browser with javascript

- Java for our android appplication

- Internet connection for communication of data

6. Scenarios identification

6.1 Scenario 1: Registration and reservation

Luca is a guy who wants to visit his parents but he is too far from the nearest underground station. He discovered on socials that there is a car-sharing service called PowerEnJoy. He decides to try it and so he download the Android application. Luca registers himself in the system giving his data, driving licence and credit card numbers. An e-mail is sent to him containing the password to access the system. As soon as he logs in the system using his new password the map is shown, giving all the informations about nearest available cars to his position (detected by GPS on his mobile phone). So he decides to click a car which is only 100 meters far from his position and reserve it. Now he has 1 hour to reach it, but it takes only few minutes to him.

6.2 Scenario 2: Reservation through Desktop

Lorenzo is at home. He is still registered to the system. He decides to reserve a car throungh the web application on his personal computer. He browse to the homepage https://powerenjoy.com and logs in the system. He has already downloaded the application, he only wants to reserve a car near him. He looks to the map and finds an available car with full battery, because he needs to use it a lot. Then he reserves it and now he has one hour to reach it. Luckily it takes 59 minutes to get ready, reach the car and open it with his application!

6.3   Scenario 3: Reservation expired

Giovanni has reserved a car and now he has only 60 minutes to reach it. Luckily he can use the map integrated in the application, but he meets an old friend of him on the way. Then they find a nice bar where to take a coffee. He forgets his reservation and the time passes. He remembers now of his reservation, but it's too late, it's expired and 1 EUR was detucted from his account as a fee. Giovanni now needs to make another reservation.

## 6.4   Scenario 4: Travel, park and lock the car

Giulia has reserved a car to go to a party with her friends. There are 4 people including Giulia. They find the reserved car in few minutes and open it through the application. She uses her personal password to switch on the car and the timer starts. The travel to reach the party lasts 20 minutes. Giulia and her friends can check at every time the informations about the travel and costs on the car display. She has to pay only 4 EUR, that are automatically detucted from her credit card. The 20% discount was activated because there more than 2 passengers. Then they decide to split the bill. They get out from the car and it automatically locks.

## 6.5   Scenario 5: Many discounts

Andrea and his friends have reserved a car after a party in the night. They only need to make few Kms to reach their home. Andrea reserves a car with 100% of battery. Luckily their destination is very close to a Power Grid Station located in the map so they decide to connect it for an extra discount. So they use it for only 15 minutes, leaving it with more than 50% of battery, gaining a 20% discount in addition to the 10% discount (beacause there are 3 passengers). Finally Andrea plugs the car in the Power Grid Station so they gain an extra 30% discount for their virtuous behavior.

7. UML modeling

7.1 Use case diagram
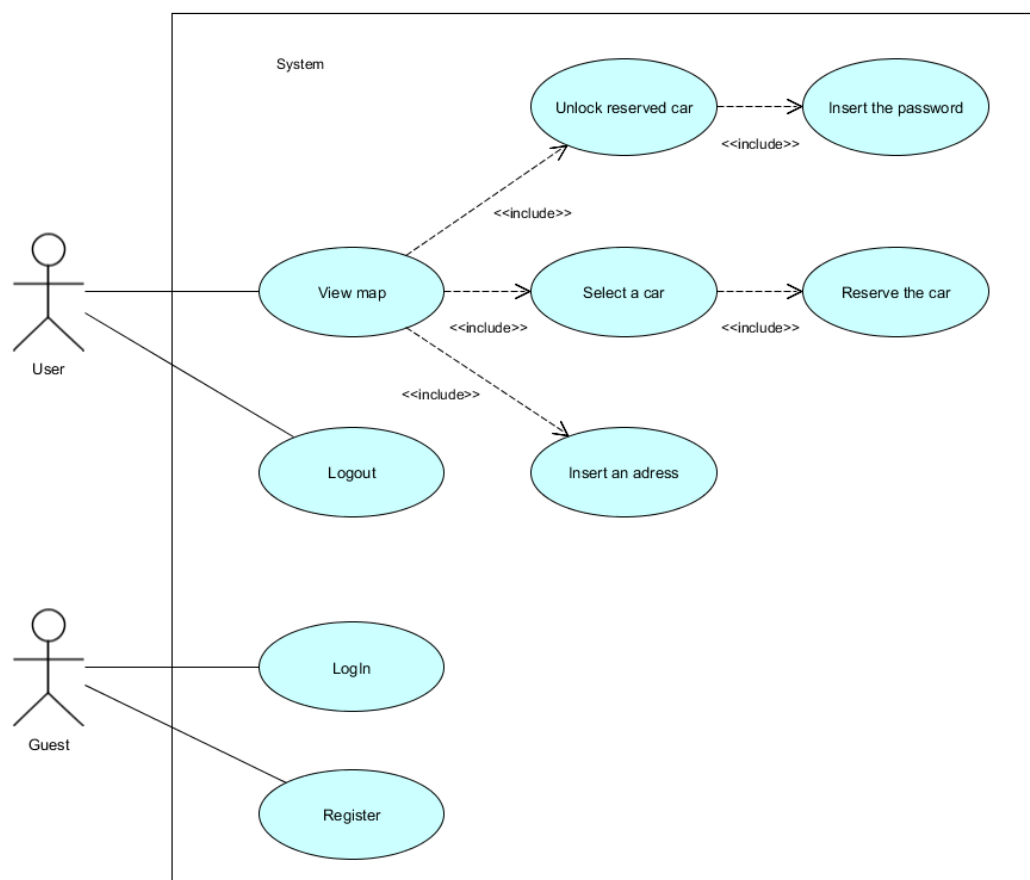
7.2   Class diagram
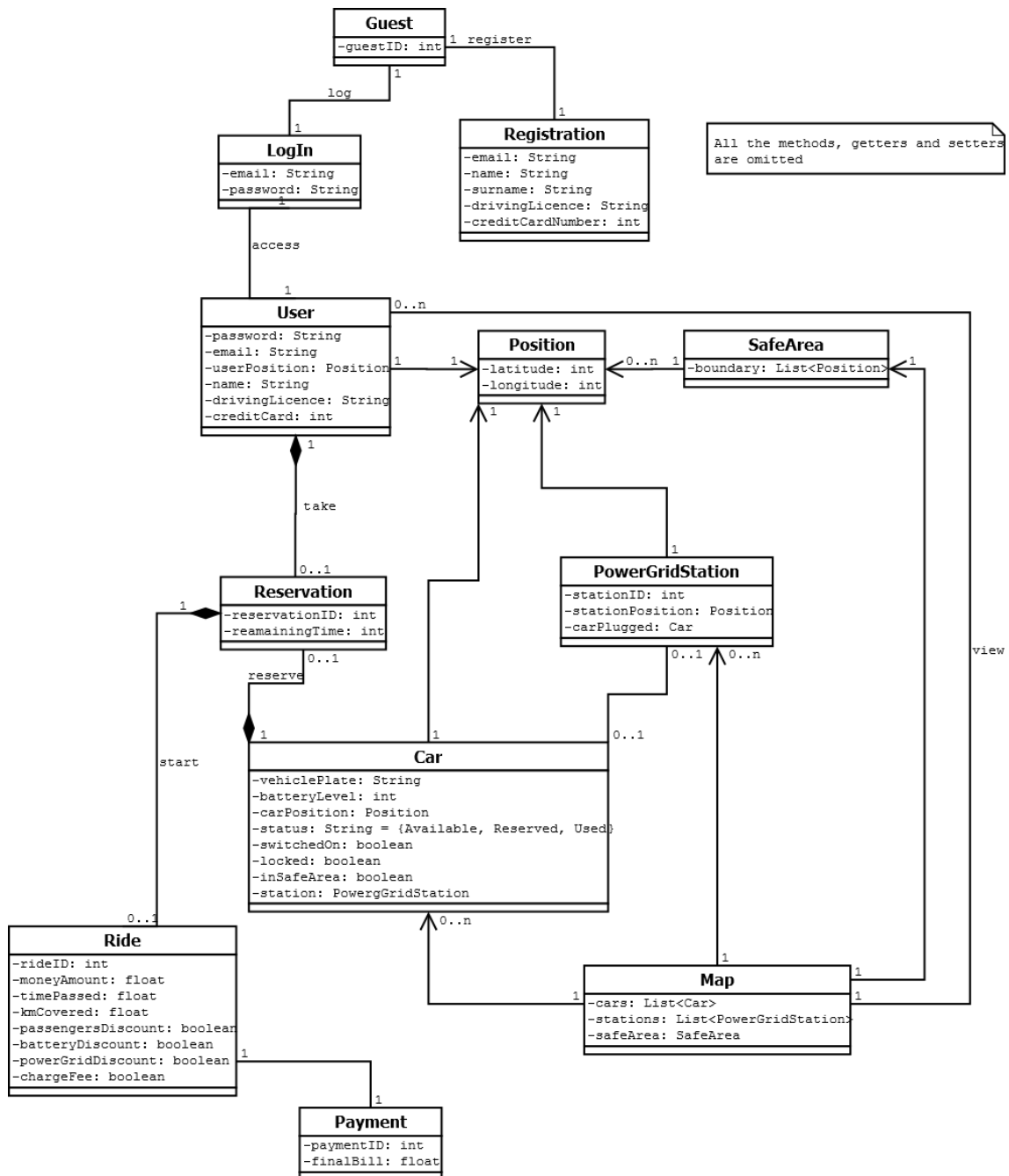
7.3 Sequence diagram

Here we present two example diagrams about two distinct processes: the first registration of a guest to the service within the reservation of a car freely choosen from the map, and a ride with a previously reserved car.
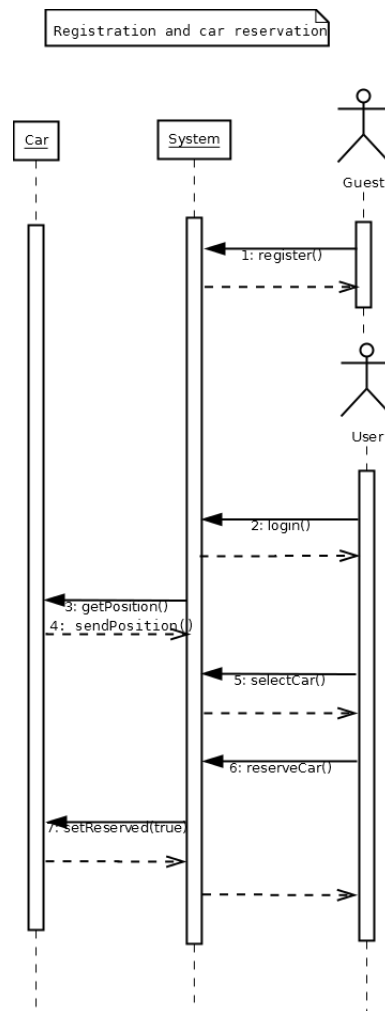

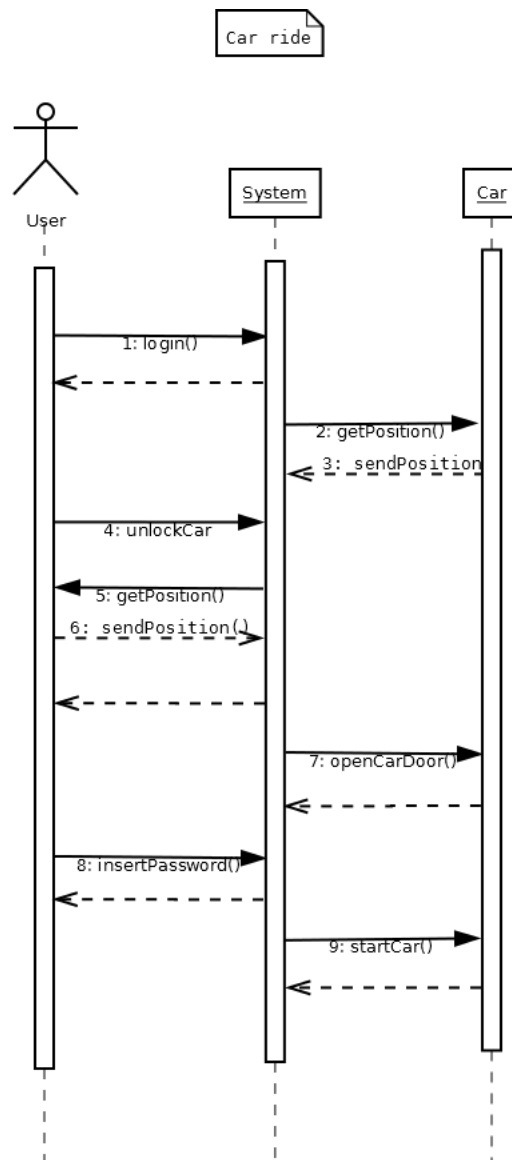Figure 7.3.1: Sequence diagram for the registration and car reservation process.

Figure 7.3.2: Sequence diagram for the car actual use.

7.4   Statechart

8. Alloy modeling

8.1 Model

In order to verify the system correctness, considering all the assumed requirements and specifications, an Alloy model has been employed.

```
// DATA TYPES

open util/boolean
sig Float {
    leftPart: one int,
    rightPart: one int }
sig float extends BaseDataType{value: int}
    {value >= 0}
sig string extends BaseDataType{}

// SIGNATURES

// Guests are still-not-registered-people
sig Guest {
    guestID: one int
}

// Users are registered people
sig User {
    name: one string,
    email: one string,
    drivingLicence: one string,
    password: one string,
    creditCard: one int,
    userPosition: one Position }
```

```
sig Car {
    vehiclePlate: one string,
    passengersNumber: one int,
    costPerMinute: one float,
    status: enum {Available, Reserved, Driven},
    carPosition: one Position,
    batteryLevel: one int,
    started: one Bool,
    locked: one Bool,
    inSafeArea: one Bool,
    station: lone PowerGridStation }
    { passengersNumber <= 4
        0 <= batteryLevel <= 100 }

sig Ride {
    rideID: one int,
    moneyAmount: one float,
    timePassed: one float,
    kmCovered: one float,
    passengersDiscount: one Bool,
    batteryDiscount: one Bool,
    powerGridDiscount: one Bool,
    chargeFee: one Bool }
    { moneyAmount >= 0
        timePassed >= 0
        kmCovered >= 0 }

sig Reservation {
    reservationID: one int,
    remainingTime: one int,
    user: one User,
    reservedCar: one Car }
    { remainingTime >= 0 }
```

```
sig Payment {
    paymentID: one int,
    finalBill: one float }
    { finalBill >= 0 }

sig Position {
    latitude: one int,
    Longitude: one int }

sig PowerGridStation {
    stationID: one int
    stiationPosition: one Position
    carPlugged: lone Car }

sig SafePlace {
    boundary: set Position }

// A map is a city area with one safe area
sig Map {
    cars: set Car,
    stations: set PowerGridStation,
    safeArea: one SafeArea }
```

```
// Returns the list of reservations one user made
fun reservationsLookup [u: User] : set Reservation {
    r: Reservation | r.user = u }

// Displays the cars one user rented more than one time
pred favouriteCars [u: user] {
    c: Car in (r: Reservation in u.reservationsLookup).reservedCar }

run favouriteCars

// Reserves a car to a user
pred reserveCar [u: User, c: Car, r,r': Reservation] {
    r'.reservationID = r.reservationID + 1
    r'.user = u
    r'.reservedCar = c }
```

8.2 Worlds generated

9. Used tools

The tools used to create this RASD are:

- Github: for version controller

- Alloy Analyzer 4.2: to create and prove our model

- yEd Graph Editor: to create graphs

- DIA Diagram: to create UML diagrams

- TexMaker: to create pdf documents with LaTex

- Balsamiq Mockups 3: to design and create mockups

10. Hours of work

Last Name	First Name	Total Hours
Fantin	Jacopo	?? h
Larghi	Francesco	?? h