



POLITECNICO DI MILANO

SOFTWARE ENGINEERING 2 PROJECT WORK

PowerEnJoy Project: Requirements Analysis and Specifications Document

Version: 1.1

Authors:

Jacopo FANTIN (mat. 878723)

Francesco LARGHI (mat. 876928)

Professors:

Elisabetta DI NITTO

Luca MOTTOLA

November 12, 2016

Contents

1	Introduction	1
1.1	Description of the problem	1
1.2	Goals	2
1.2.1	Generic Goals	2
1.2.2	Specific Goals	3
1.3	Glossary	3
2	Preliminary considerations and assumptions	5
2.1	Domain properties	5
2.2	Text assumptions	5
3	Proposed system	7
4	Actors identification	9
5	Requirements	11
5.1	Functional requirements	11
5.2	Non-functional requirements	14
5.2.1	Mobile Interface	14
5.2.2	Desktop Interface	15
5.2.3	Car Interface	17
5.2.4	Documentation	18
5.2.5	Considerations	18
6	Scenarios identification	19
6.1	Scenario 1	19
6.2	Scenario 2	19
6.3	Scenario 3	19
6.4	Scenario 4	19
6.5	Scenario 5	19
6.6	Scenario 6	19
7	UML modeling	21
7.1	Use case diagram	21
7.2	Class diagram	22
7.3	Sequence diagram	23
7.4	Statechart	24

8 Alloy modeling	25
8.1 Model	25
8.2 Worlds generated	25
9 Used tools	27
10 Hours of work	29

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 web and a mobile application that allows to reserve an electric car using GPS position to find the closest car to the client or alternatively the client can insert it manually. The cars are supposed to be recharged at specific electricity stations when in need, placed all around the cities where the service is offered. The clients must register to the system to reserve a car, providing their credentials and payment information. They receive back a password that can be used to access the system.

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 virtuous 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:

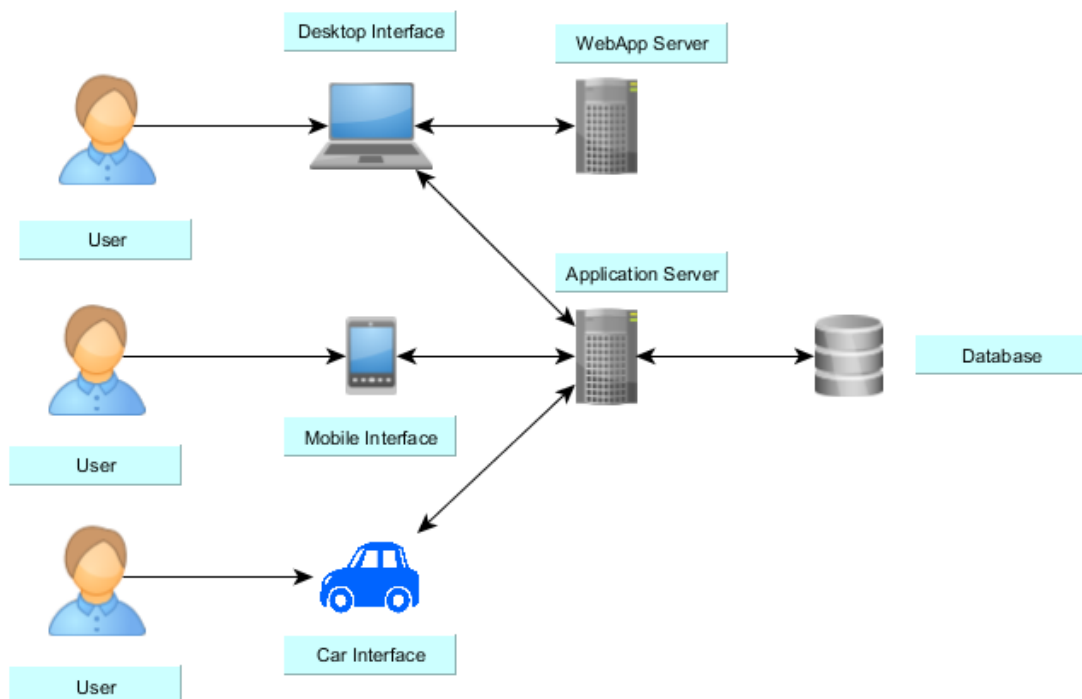
- Both the mobile and car GPS give always the right position
- The clients always have enough money in their bank account to pay the ride
- Car codes and vehicle plates are unique
- Clients registered into the system have always updated documents and driving licence allowing them to drive Power Enjoy's cars
- Each e-mail used must exist and correspond to one user

2.2 Text assumptions

- The user's password is unique and used both to access 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 deducting 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
- We assume that there is a maintenance car system, but it is not linked to our system
- There are employers that have to re-charge cars on-site in case of empty battery (or less than 20%), but they are not directly handled by our system, they have another system connected to our database
- We assume that we need only few principle informations about the vehicle, like identification code (vehicle plate), position and battery
- We assume that there is a car interface to our application, connected to our Application Server, but we don't need to model or manage it deeply

3. Proposed system

We will implement both a web and a mobile application with a client-server architecture. Users can use whether browser or mobile app with the same account to communicate with the unique application server. We will have also a database server communicating with our application server.



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 and insert his data and payment informations in order to have credentials to access and use the application
- USER: he can log into the system with his credentials. Then he can reserve a car, use it with his 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 address inserted/detected is valid
3. There is a proper algorithm 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

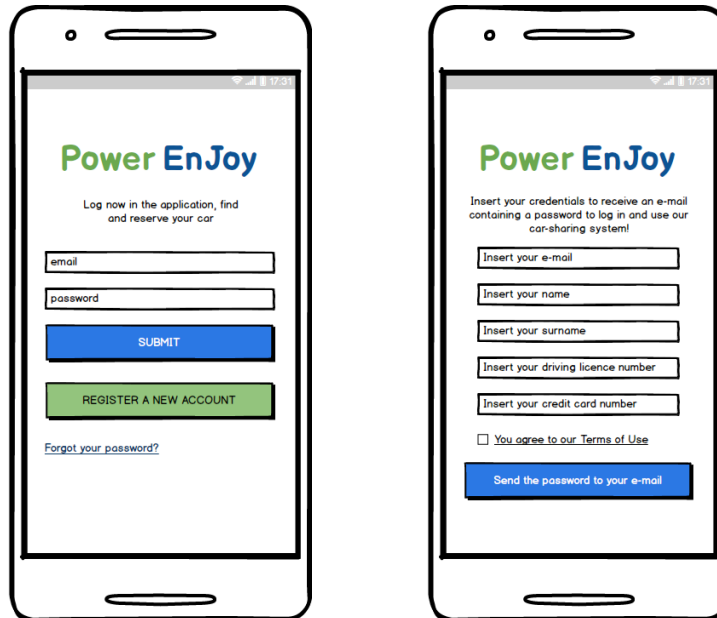


Figure 5.2.1: Login and Registration forms on mobile

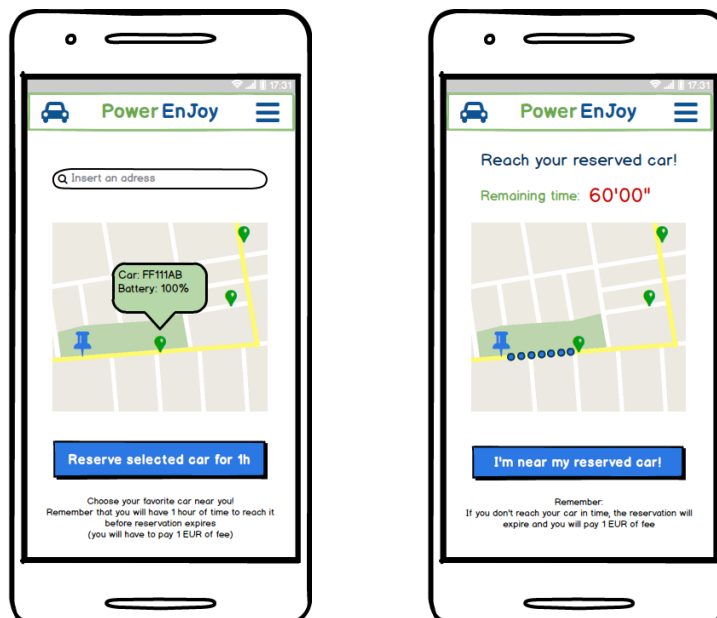


Figure 5.2.2: Reserve a car and open it on mobile

5.2.2 Desktop Interface

This is how the desktop site should be

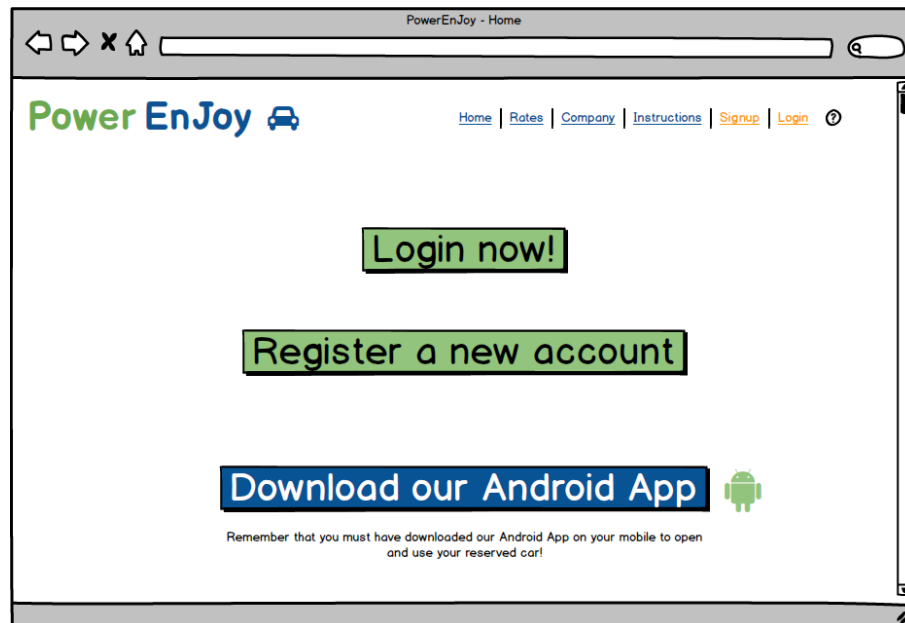


Figure 5.2.3: A sample of the Desktop Home Page

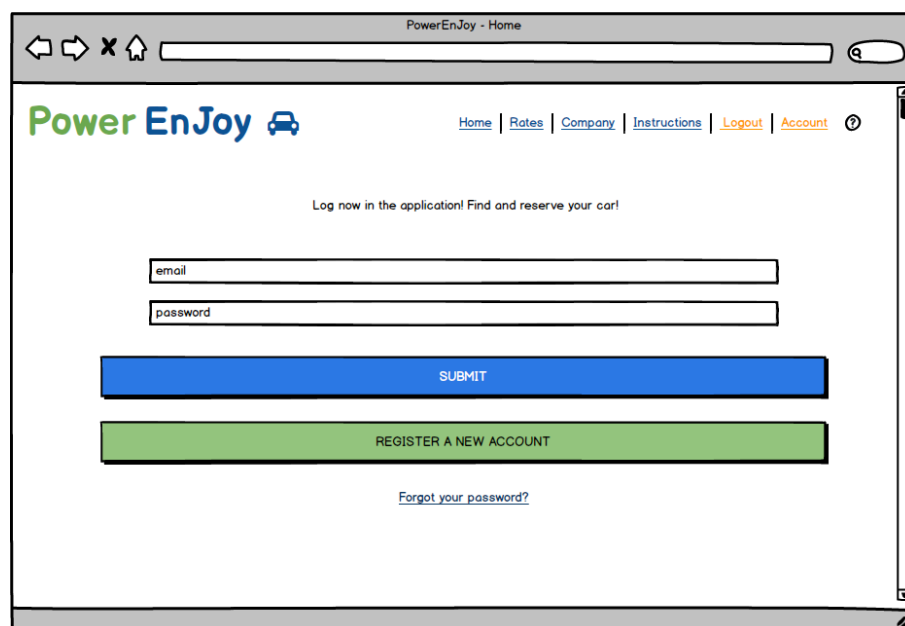
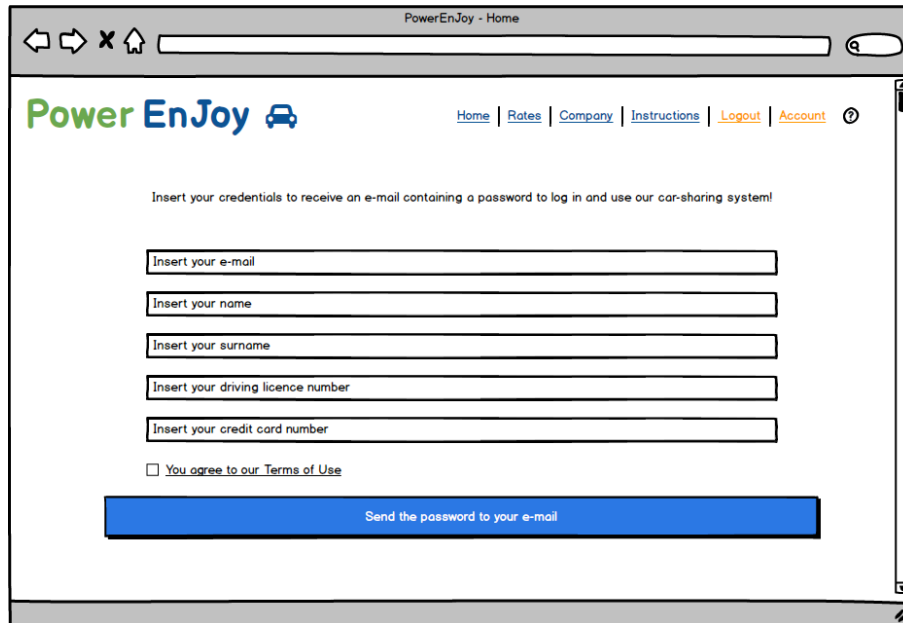


Figure 5.2.4: A sample of Desktop Login Page



The screenshot shows a web browser window titled "PowerEnJoy - Home". The page features the "Power EnJoy" logo with a car icon. A navigation bar includes links for Home, Rates, Company, Instructions, Logout, and Account. The main content area prompts users to "Insert your credentials to receive an e-mail containing a password to log in and use our car-sharing system!". It contains five input fields: "Insert your e-mail", "Insert your name", "Insert your surname", "Insert your driving licence number", and "Insert your credit card number". Below these fields is a checkbox labeled "You agree to our Terms of Use" and a large blue button labeled "Send the password to your e-mail".

Figure 5.2.5: A sample of the Desktop Registration Page

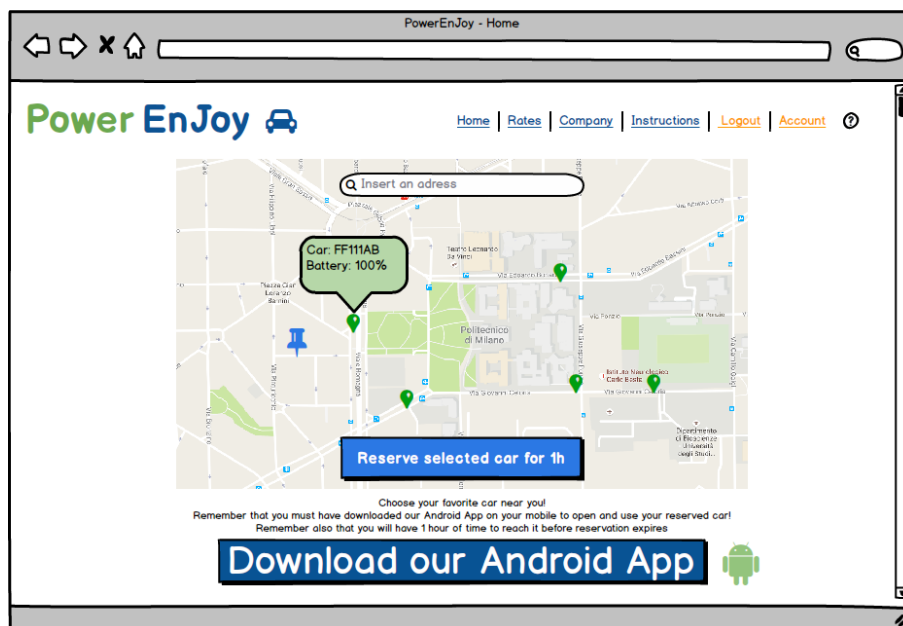


Figure 5.2.6: A sample of the Desktop Reservation Page

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

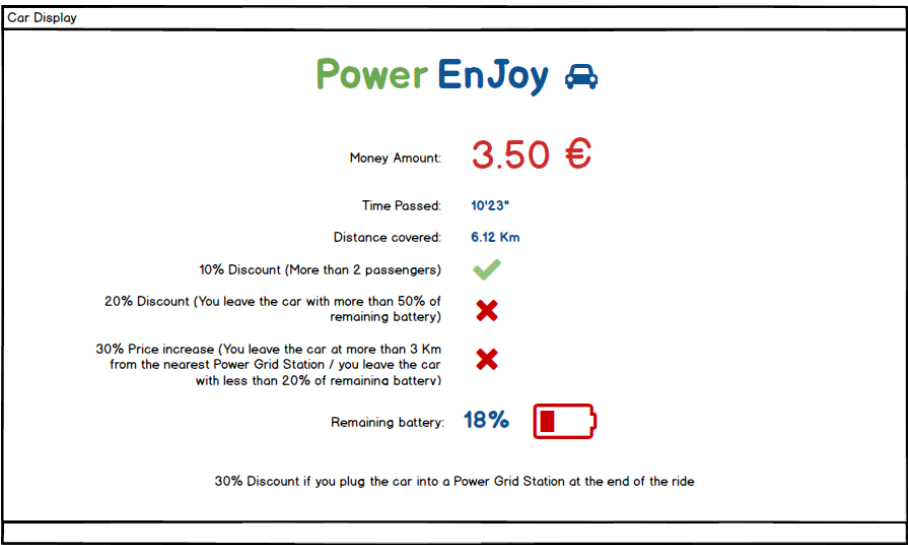


Figure 5.2.8: A sample of the car display during the ride, showing Money Amount rising and all the other info and discounts

5.2.4 Documentation

5.2.5 Considerations

6. Scenarios identification

6.1 Scenario 1

6.2 Scenario 2

6.3 Scenario 3

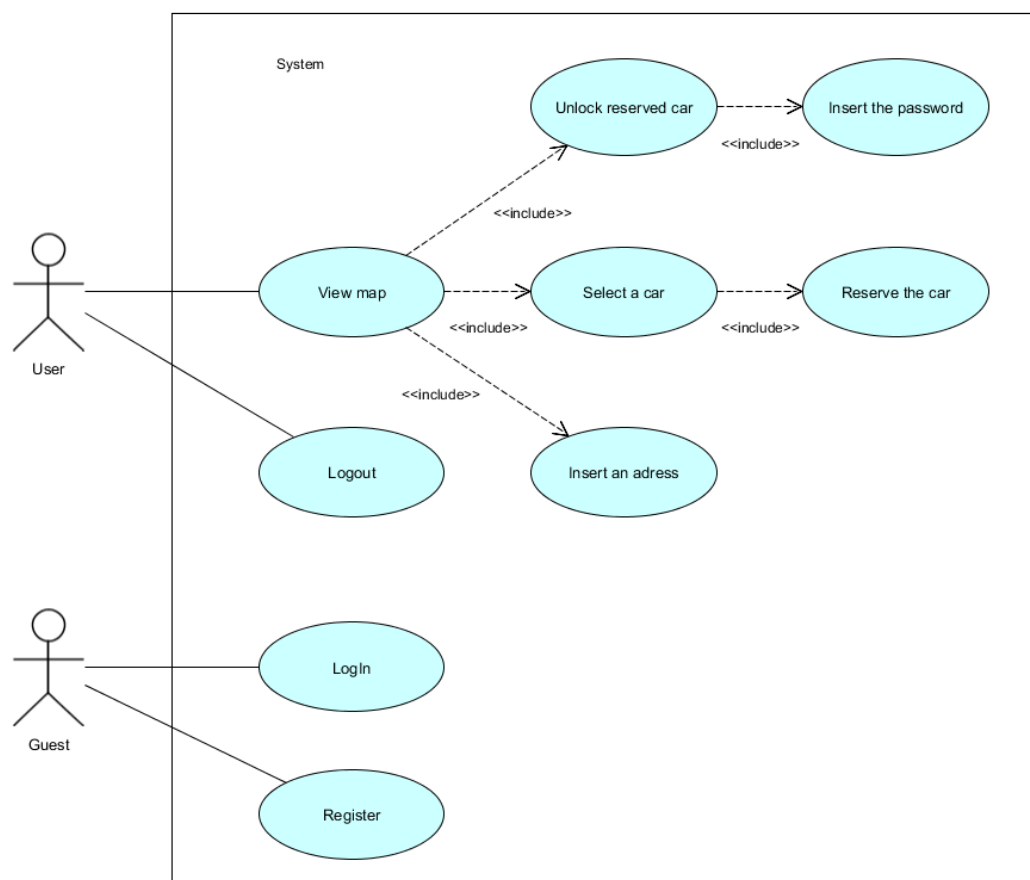
6.4 Scenario 4

6.5 Scenario 5

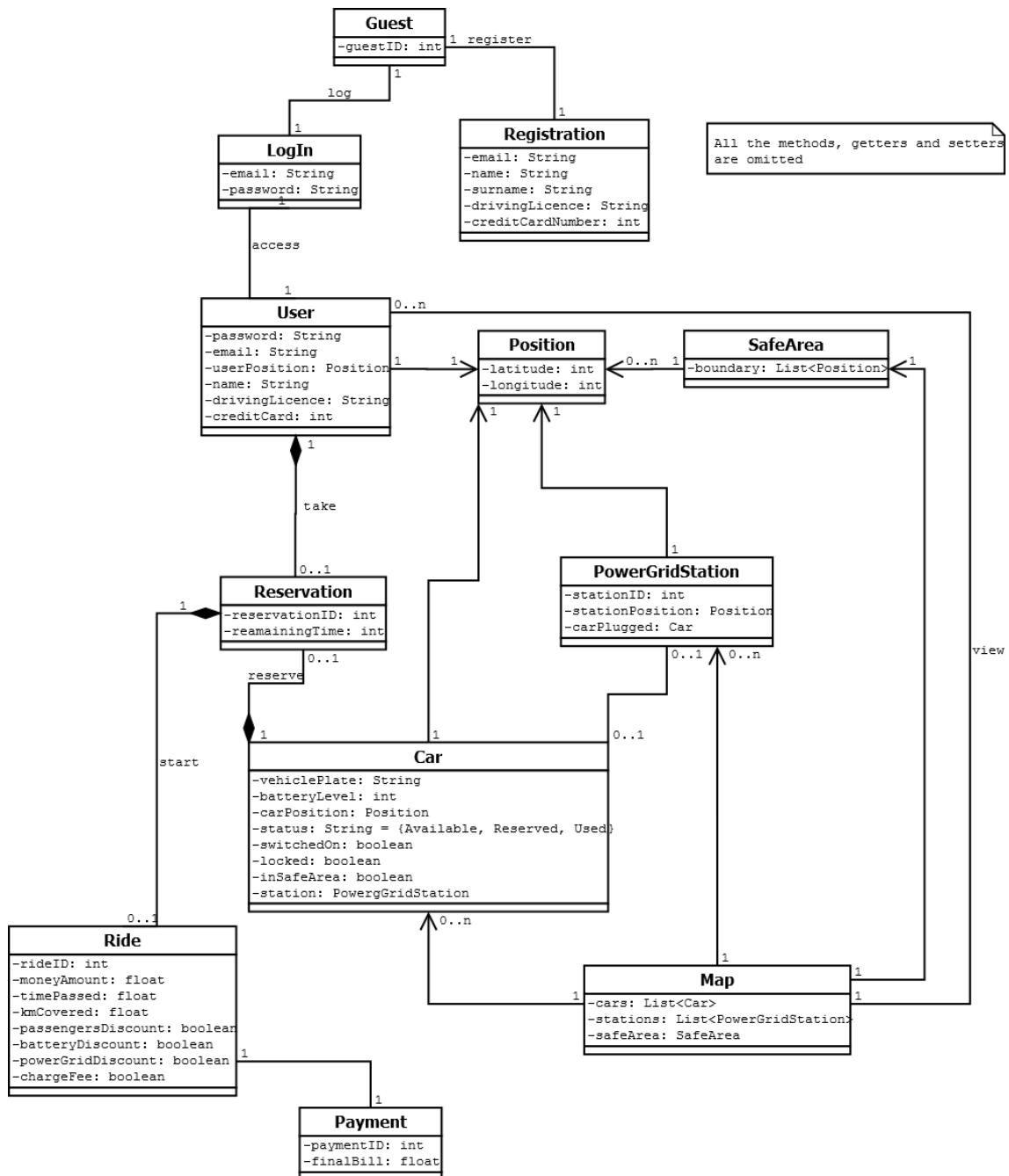
6.6 Scenario 6

7. UML modeling

7.1 Use case diagram



7.2 Class diagram



7.3 Sequence diagram

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
  stationPosition: 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 the graphs and UML models
- DIA: to create the Class Diagram
- 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