POLITECNICO
MILANO 1863

"PowerEnjoy"

Requirement Analysis and Specifications Document

Version 1.1 *(11/12/2016)*

Giorgio Marzorati (876546)
Aniel Rossi (877018)
Andrea Vaghi (877710)

INDEX OF CONTENTS

INTRODUCTION

Description of the given problem

This is the Requirements Analysis and Specifications Document of the system we are going to implement, named PowerEnjoy. It is a car-sharing service for the city of Milan based on a mobile application with a single category of end users.
The system allows clients to reserve or directly identify and use available electric-powered cars in the area around the client's GPS position or an address inserted manually. If the client does not identify a car within 1 hour from the reservation, this is cancelled, the car becomes available again to other clients and a fee of 1€ is charged to the one who made the reservation.
In order to use it, clients have to register to the system, in particular they have to provide an e-mail address, biographical data and a valid driving license. On the other side, the system provides to clients a personal PIN, with which they access to the system and are allowed to reserve or identify a car.
Clients are charged at the end of every ride and can pay with one of the supported payment methods, that must be specified during the registration process and can be modified in every moment.
In addition, the system defines different discounts and overcharges for every ride, as a result of particular client's behaviours (more informations are included in the Glossary session).
Also, clients will be supported by an assistance team that will take care of recharging parked low battery cars, fix car malfunctions and assist in general clients with any kind of issue and support request. An assistance request has to be performed with a direct call to a provided phone number.
Cars are equipped with different sensors that are able to detect battery level, problems with cars components, number of passengers etc. and other data that have to be sent to the central server (for example to contact the assistance team in case of any issue concerning the car).

The system has the purpose of providing an efficient and environment-friendly alternative to public transportation to people who don't have to cover long-distance travels and don't want to (or cannot) use personal vehicles.

Actors of the system

- Client:

A potential Client of our application is a person who wants to use the car sharing option instead of, e.g, the public trasportation service.

- Assistance:

Is the team which take care of general clients help request, cars maintenance and recharging when they are parked with a <= 20% level of battery.

In the definition of the actors of our system and their use cases, we decided not to include standard external partecipants (such as electronic payment system), and to focus just on what acts actively from inside the environment.

Goals

Client

- [G1] - Register to the system
- [G2] - Log into the system
- [G3] - Find available cars from the current GPS position
- [G4] - Find available cars from a specific address
- [G5] - Reserve a car
- [G7] - Drive a car
- [G8] - Monitor current charging during a ride
- [G9] - Enable "Money Saving Mode" for a ride
- [G10] - Host Passengers for a ride
- [G11] - Park the car in a Safe Area
- [G12] - Plug the car in a Power Grid
- [G13] - Pay a ride
- [G14] - Pay a fee
- [G15] - Get a discount on a ride
- [G16] - Get an overcharge on a ride
- [G17] - Ask for assistance

Assistance

- [G18] - Get notified about car's low battery level
- [G19] - Get notified about car malfunctions
- [G20] - Get notified about client assistance requests

Glossary

1. **Client**: is a potential user of the service (a person);
2. **Username**: the *client* identifies itself in our application providing its email, which is used as *username* for the login procedure;
3. **PIN**: is a personal sequence of 6 digits provided by the system to the *client* in order to access the service and let him perform a *car identification*. The PIN is unique for every *client* and it's provided once at the end of the registration flow;
4. **Payment Information**: are the informations provided by the *client* during the registration in order to let the system charge him for the *rides* or *fees*;
5. **Position**: it is the position rappresented by the couple latitude - longitude acquired by the GPS.
6. **Car**: a vehicle with 5 total seats that belongs to our system and can be used by the *clients* for moving through the city. All the *cars* are identified by a unique licence plate;
7. **Research Area**: area in which the system researches available *cars*; it has a 1 km radius from the specified GPS position (or address) for the research;
8. **Reservation**: is a way to book a parked available *car* for a limited amount of time (one hour) in order to use it. After that amount of time the reservation is no more valid and the *client* needs to pay a *fee* of 1€ if the the *car identification* is not performed;
9. **Car Identification**: it is the procedure that allows the *client* to access an available *car*. To perform the *car identification* the *client* needs to identify the vehicle with the licence plate and itself with the personal *PIN*.
10. **Car Availability**: a *car* is available if it is not reserved, it's not in use or has a battery level > 20%;
11. **Terminate Ride**: it is the procedure with which a *client* terminates the use of the *car* after he has parked it in a *Safe Area*;
12. **Charge**: is a variable amount of money that the *client* pays proportionally to the *ride*; The normal charge (without *discounts* or *overcharges*) is based on the *ride's* time with this rate: 0.25 €/min;
13. **Fee**: is a 1€ price applied by the system to the *client* if he does not identify a *car* before 1h from its reservation;
14. **Discount**: is a percentage to subtract from the *charge* of a *ride* in case of virtuous behavior of the *client* (not cumulables):
    - 10% on the *charge* of a *ride* with 2 or more additional *passengers*
    - 20% on the *ride charge* if the *car* is parked with > 50% of battery level
    - 30% on the *ride charge* if a *car* is plugged into the *Power Grid Station* of a *Special Parking Area* when the *client* terminate its *ride*

    Overcharge: to cover costs due to uncorrect *client's* behaviours the system applies the following overcharges:
    - 30% on a *ride charge* if the *client* parks the *car* at > 3 km from the nearest *Power Grid Station;*
    - 30% on a *ride* if the *client* parks the *car* with the battery level <= 20%;
15. **Safe Area**: is a single place parking in which the *client* has to leave the *car* at the end of a *ride*;
16. **Special Parking Area**: is a *Safe Area* in which is located a single *Power Grid Station*;
17. **Battery level**: refers to the residual battery level of the *car*;

18. **Ride**: it begins when the *client* starts the engine after a *car identification* and ends when the *client* turns off the engine in a *Safe Area* and confirms to the system (on the proposed prompt) the intention of terminating it;
19. **Power Grid Station**: is a totem placed in a *Special Parking Area* that allows the distribution of electric power in order to recharge the battery of a single parked *car*. It contains a unique plug;
20. **Money Saving Mode**: is a special *ride* options in which the *client* has to specify the ending point so that the system suggests where to park the *car,* considering the availability of Special Parking Area (to get a discount) and the distribution of vehicles available in all the city area.
21. **Passenger**: is a person who does not have to be registered on our system, but participates to a *ride*. The system can not identify people partecipating to a ride except from the *client*, but can detect the number of *passengers*;
22. **Assistance**: is the team that takes care of issue concerning:
    a. Recharging parked *cars* with the level of battery <= 20%;
    b. Generic *car* malfunctions;
    c. generic help request asked by the client;

Domain Assumptions

- The GPS of the cars gives always the right position
- The GPS of the cars cannot be switched off
- Cars cannot be stolen or damaged while they are parked
- A client will not carry more passengers than the car's capacity
- An assistance request will always be evaluated
- Client will always have enough money on their balance to pay the last ride
- A client who damages a car or makes an accident will take care to contact the assistance
- When the client finishes his ride he leaves the car
- The driving license provided by the client is valid
- Number of passengers in a single ride is constant
- A client will not delete a car reservation

Text Assumptions

- Our system is not derived from an older one
- All the clients must be registered and logged in the application to let the system be aware of "who has made what"
- A client that wants to register must have a valid driving license
- We need informations on a payment method in order to charge the client
- We are always able to find all available cars in a specified geographical area
- As soon as the engine ignites the system begins to charge the client for a given amount of money per minute
- A Client which unlocks a car may or may not start a ride
- The system locks the car automatically if the client doesn't enter the car after the car's identification or if the client exits the car after finishing a ride
- The client can always enable the money saving option following the specified procedure
- We assume that that our service works only with a single category of electric cars
- We assume that discounts and overcharges are not cumulable
- We assume that the system has to feature an assistance team for issues concerning cars (low battery level, malfunctions)

Constraints

Regulatory Policies

The system must require the client's permission to acquire his position in order to make an accurate research of the available cars.
The system must also require the permission for managing sensible data that the client must provide to the system, for example through the insertion of his personal driving licence, in respect to the privacy law. The client must accept the policy document imposed by the system before being registered to it; in this policy document the system specifies in which cases the client has to take his own responsibilities, for example in case of an accident or if another person, not registered in the system, drives the car.

Hardware Limitation

- in order to use our mobile application the client must have a mobile device on which there has to be:
  - 3G or WiFi connection present and persistent;
  - GPS enabled (optional);
  - Enough free space for application file installation;
- every PowerEnjoy car has:
  - GPS always enabled (to get precise position of the car)

Interfaces to other applications

- Interface with e-mail services in order to communicate important informations on the client account, like the personal PIN and with the assistance.
- Interface with Paymenth Method providers (Credit Card companies & Paypal)

Stakeholders Identification

Our main stakeholders are the owners of company interested in developing a innovative system in the transportation market, which have to be environment-friendly and mainly self maintained by the clients who use it.

Reference Documents

- Assignments+AA+2016-2017.pdf (Section 5)
- IEEE+standard+on+requirement+engineering.pdf
- Examples documents:
  - RASD sample from Oct. 20 lecture.pdf

REQUIREMENTS ANALYSIS

Functional Requirements

- *Client*
  - [G1] - Register to the system
    - The system must receive valid informations of the client (biographical data, driving license ID, email, payment method)
    - The system must check if the client's e-mail address is not registered in the system yet
    - The system must check if the client's driving license is not registered in the system yet
    - The system must send a PIN to the client
  - [G2] - Log into the system
    - The system must check if e-mail address is correct, otherwise an error is returned
    - The system must check if PIN is correct, otherwise an error is returned
  - [G3] - Find available cars from the current position:
    - The system must be able to determine the client's position from the GPS of his mobile phone
    - The system must be able to determine cars position
    - The system must be able to determine which cars are reserved
    - The system must be able to determine which cars have enough battery level left
  - [G4] - Find available cars from a specific address:
    - The system must determine the correctness of the address provided by the client
    - The system must be able to determine cars' position
    - The system must be able to determine which cars are reserved
    - The system must be able to determine which cars have enough battery level left
  - [G5] - Reserve a car:
    - The system must show the available cars on the map
    - The system must set the selected car (reserved) as unavailable
    - The system must check if the inserted client PIN is correct
  - [G6] - Identify a car:
    - The system must check if the inserted license plate is correct
    - The system must check if the inserted client PIN is correct
    - The system must check if a car is available
  - [G7] - Drive a car:
    - If G6 succeeds the system must unlocks the identified car
    - The system must begin to charge the client as soon as he/she turns on the engine
  - [G8] - Monitor current charging during a ride

        - The system must show in real time the current charging based on the elapsed time
    - [G9] - Enable “Money Saving Mode” for a ride
        - The system must check the correctness of ending position
        - The system must check the distribution of the car parked on the city area
        - The system must check the availability of Special Parking Zones near the destination
    - [G10] - Host Passengers for a ride
        - The system must be able to detect if and how many passengers are into the for a ride
    - [G11] - Park the car in a Safe Area
        - The system must ask the client to terminate the ride
        - The system must be able to detect if client has stopped the car in a Safe Area
    - [G12] - Plug the car in a Power Grid
        - The system must ask the client to plug the car into the Power Grid Station
        - The system must be able to detect if the client has succesfully plugged the car into the Power Grid Station
    - [G13] - Pay a ride
        - The system must interact with a payment system
    - [G14] - Pay a fee
        - The system must be able to determine when a reservation has expired
    - [G15] - Get a discount on a ride
        - The system must be able to determine car’s battery level at the end of a ride
        - The system must be able to determine car’s distance from the nearest Special Parking Zone
        - Reference to [G10] & [G12]
    - [G16] - Get an overcharge on a ride
        - The system must be able to determine car’s battery level at the end of a ride
        - The system must be able to determine car’s distance from the nearest Special Parking Zone
    - [G17] - Ask for assistance
        - The system must provide to the user the phone number of the assistance

- *Assistance*

    - [G18] - Get notified about car’s low battery level
        - The system must be able to determine car’s battery level at the end of a ride
        - The system must be able to forward the information above to the assistance
    - [G19] - Get notified about car malfunctions

    - The system must be able to automatically detect car’s malfunctions when they occur
    - The system must be able to forward the information above to the assistance
- [G20] - Receive client’s assistance request
    - The system must provide to the user the phone number of the assistance
    - The system must allow the assistance team to store client’s request information

Non-Funtional Requirements

Client interface sample

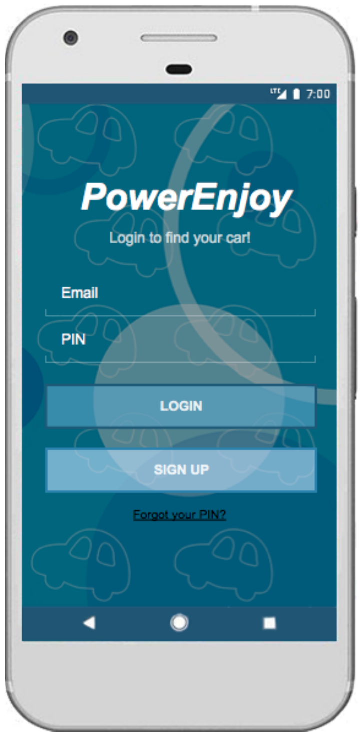


Figure 1 shows the login form of the application, with the option of signing up for a new client and a link to recover a forgotten PIN

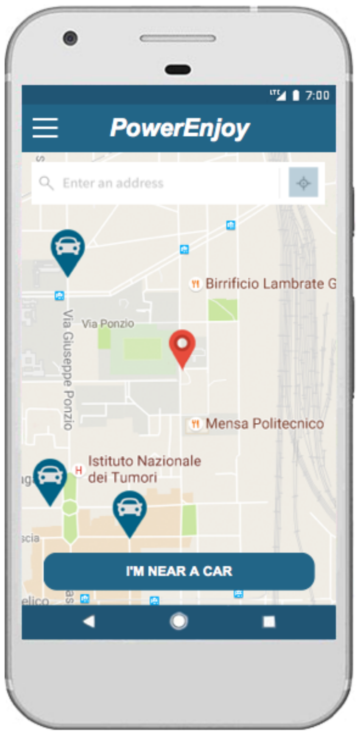
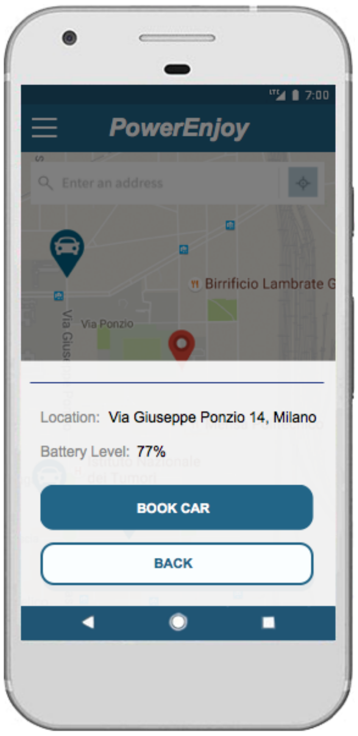


Figure 2 show the home of the application, in which the user can serach for available cars near a specified address or its GPS position



Location: Via Giuseppe Ponzio 14, Milano

Battery Level: 77%

Figure 3 shows the section in which the client focus on an available car selected on the map, with the possibility of reserving it. The display show the location and the battery level of the car.

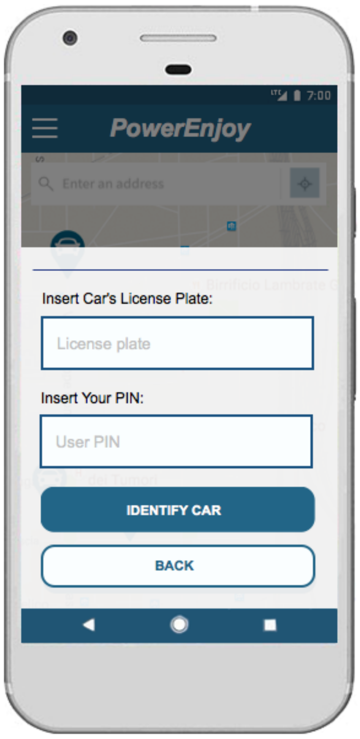


Figure 4 shows the section for a car identification, accessibile by touching the button in figure 2 “I'm near a car”. The client have to insert the car's License Plate and the personal PIN.

Quality of services attributes

- The system must work real time with the client interaction, so it has to be responsive (e.g. for car identification & unlocking, < 1 second of delay between client request and server operation)
- A car search around a indicated point must last < 5 seconds
- Since the service is mainly self mantained by the clients, it has to control and store every client activity on the cars
- The service must be active 24 hours per day, so we have to think about replication of the system
- The service must store securely the personal informations provided by the clients
- In order to achieve service's presence in other cities (in future developements), we will need more dedicated servers and DBs.

SCENARIO IDENTIFYING

Here some possible scenarios of usage of this application.

Scenario 1

Andrea downloads PowerEnjoy application. Since he does not have an account he has to subscribe to the system. He pushes the "Sign up" button and he fills the registration form. Andrea has also to provide valid driving license and payment method. The system checks the validity of all the data. The registration process is successful, so the system sends an e-mail to Andrea with a personal and unique PIN with which he can access the system and identify an available or reserved car.

Scenario 2

Aniel downloads PowerEnjoy application. Since he does not have an account he has to subscribe to the system. He pushes the "Sign up" button and he fills the registration. Aniel has also to provide valid driving license and payment method. The system checks the validity of all the data. Unfortunately the system detected an account with the same driving license ID so the system reject the registration request and it notify Aniel to enter a valid driving license ID.

Scenario 3

Aniel needs to use PowerEnjoy to get home after university courses. He sees a car and decides to take it. He logs in the application and declares to be near the car. He has to identify it with its license plate and himself with the PIN. Once the system has checked the car availability and the correctness of thePIN, it unlock the car and Aniel can access it.

Scenario 4

Giorgio wants to reserve a car after the gym, that it is in a different place from his current position. He opens the application and he searches for available cars in the gym area. Once he chooses one of the available one he reserve through the reservation method. The system marks the car as unavailable. Unfortunately he exits the gym ten minutes later than he expected and the reservation expires, so the system marks the car as available again and charges him with a 1€ fee.

Scenario 5

Aniel wants to go home by car but does not want to pay a lot, so he decides to use a PowerEnjoy car in a “Money Saving Mode”. He identifies the car and he enters it. Before starting engine he enables “Money Saving Mode” specifying the end position of his ride filling the form on the car’s screen. After the system has calculated where to park the car Aniel can start the ride. When he finishes the ride he parks the car in the Special Parking Zone suggested from the system and plugs the car in the corrispective Power Grid Station, getting a discount on the last ride.

Scenario 6

Giorgio is driving one of the PowerEnjoy cars and he has to park the car. He finds a park but unfortunately it is not a Safe Area according to the system. In the moment when he stops the car and turn off the engine, the system notifies Giorgio from the car’s monitor that it is not a place in which he is allowed to park the car, so the system continue to charge him untill he finds a Safe Area where to park the car.

Scenario 7

Aniel is driving home and near it he finds a Special Parking Area. He decides to stop the car and park there. When he stops the engine the system stop charging him and shows the charge on the car’s screen since it detected the car is in a Special Parking Area. The system asks Aniel if the ride is ended and to plug the car in Power Grid Station, to get a discount. Aniel has got 3 minutes of time for plugging the car and get a discount, otherwise the system will charge the client with the normal amount without any kind of discount.

Scenario 8

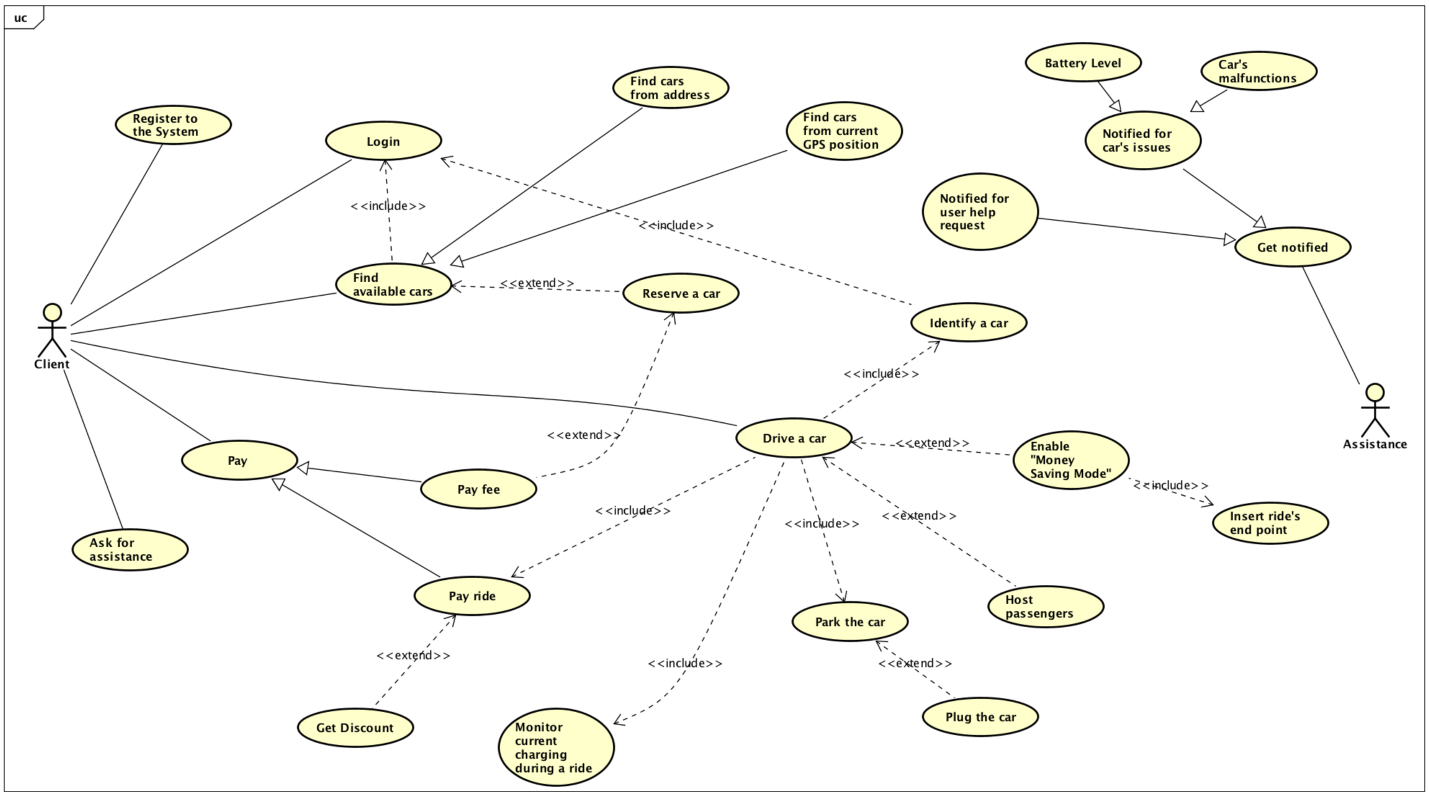
Andrea has a problem with the car while he is driving it. Andrea contacts the assistance calling the phone number provided and specifying the type of problem occured. The assistance operator picks up the request and forwards it to the team sending the position of the car and the type of issue.

Scenario 9

Giorgio is driving a car to get to the university with Aniel and Andrea. He decides to stop the car in a Safe Area near the building. When he stops the engine, the system asks him if the ride has ended and after client confirmation, it calculates the final charge decting 2 other passengers on the car, applying a 20% discount on the normal ride charge.

UML MODELS

Use Case Diagram



Use Cases Description

Identify a car

Name: Identify car
Actors: Client
Entry conditions:

- The client must be logged
- The client must be able to see the "Identify car" button

Flow of events:

- The client must push the "I'm near a car" button
- The client must enter the driving plate of the chosen car and his personal PIN
- The system checks if the car is available and if the driving plate and the PIN are correct
- The system unlocks the car pushing the "Identify Car"

Exit conditions: The car is unlocked and the client is ready to use it

Exceptions: if the client do not enter the correct informations in the car identification form or the car is unavaible, the system will reject the request from the client showing an error message.

Drive a car

Name: Drive a car
Actors: Client
Entry conditions:
- The client must have identified a car
- The car must be unlocked
- The client enters in the car

Flow of events:
- The client starts the engine
- The system start calculating the charge of the ride

Exit conditions: the client parks the car in a safe area, terminating the ride
Exceptions: there are no exceptions

Reserve a car

Name: Reserve a car
Actors: Client
Entry conditions:
- The client is able to see the available cars on the map

Flow of events:
- The client can choose one of the available cars from the interactive map
- The client insert its personal PIN in the relative box
- The system will tag that car as unavailable for all the other clients except the client that now sees it as "Reserved"

Exit conditions: the client has reserved the chosen car
Exceptions:
- The client insert a wrong PIN

Enable "Money Saving Mode"

Name: Enable "Money Saving Mode"
Actors: Client
Entry conditions:
- The client has identified the car that he is going to use

Flow of events:
- The client enables the "Money Saving Mode" from the monitor in the car
- The system shows a "Final Destination Form"
- The client must enter the final destination in "Final Destination Form
- The system suggests where to park the car

Exit conditions: the client parks the car in a safe area, terminating the ride

Exceptions: if the client do not enter the correct informations in "Final Destination Form" the system is not able to suggest any area where the client can park the car. In this case the system notifies the client that the informations provided are wrong. If the client does not park the car in the suggested point the system will not apply the discount

---

Register to the system

Name: Register to the system
Actors: Client
Entry Conditions: There are no entry conditions except the intention of use the application
Flow of events:
- The client opens the application
- The client presses the button "Register"
- The client fills the registration form
- The system checks the validity of the data
- The system sends back a PIN via the provided email

Exit Conditions: The client is successfully registered to the system
Exceptions:
- The data are not valid. In this case the client is invited to fill the form again with correct data
- The client is already registered in the system (control on email and driving license). In this case the client is notified that he is already registered

---

Find available cars

Name: Find available cars
Actors: Client
Entry Conditions: The client must be logged in the system
Flow of events:
- The client select the type of the research
- The system gets the current position of the client or the provided address depending from the choice of the client
- The system gets all the GPS positions of the available cars in the limited geographic area depending from previous choices
- The system shows on the map the location of found cars

Exit Conditions: The client can see the available cars in the desired area
Exceptions:
- Client address is incorrect

Park the car

Name: Park the car in a Safe Area
Actors: Client
Entry Conditions: The Client must be driving a car and being charged by the system

Flow of events:

- The Client must switch of the engine and exit from the car
- The system stops the charging of the client
- The system calculates if there are discounts on the ride
- The systems redirects the charge to the payment system
- The system checks the battery level of the car
- If the battery level is <20% the system contacts the Assistance Team

Exit Conditions: The client has parked the car and the charge is ready to be paid
Exceptions: There are no exceptions here

Ask for assistance

Name: Ask for assistance
Actors: Client
Entry Conditions: The Client must be logged in or he is driving a car
Flow of events:

- The Client makes a phone call to the provided phone number
- The assistance team asks for details about the issue, in particular car's license plate and client's driving license are mandatory
- The assistance locks the car and sets it unavailable

Exit Conditions: The assistance takes care of resolving the issue
Exceptions: There are no exceptions here

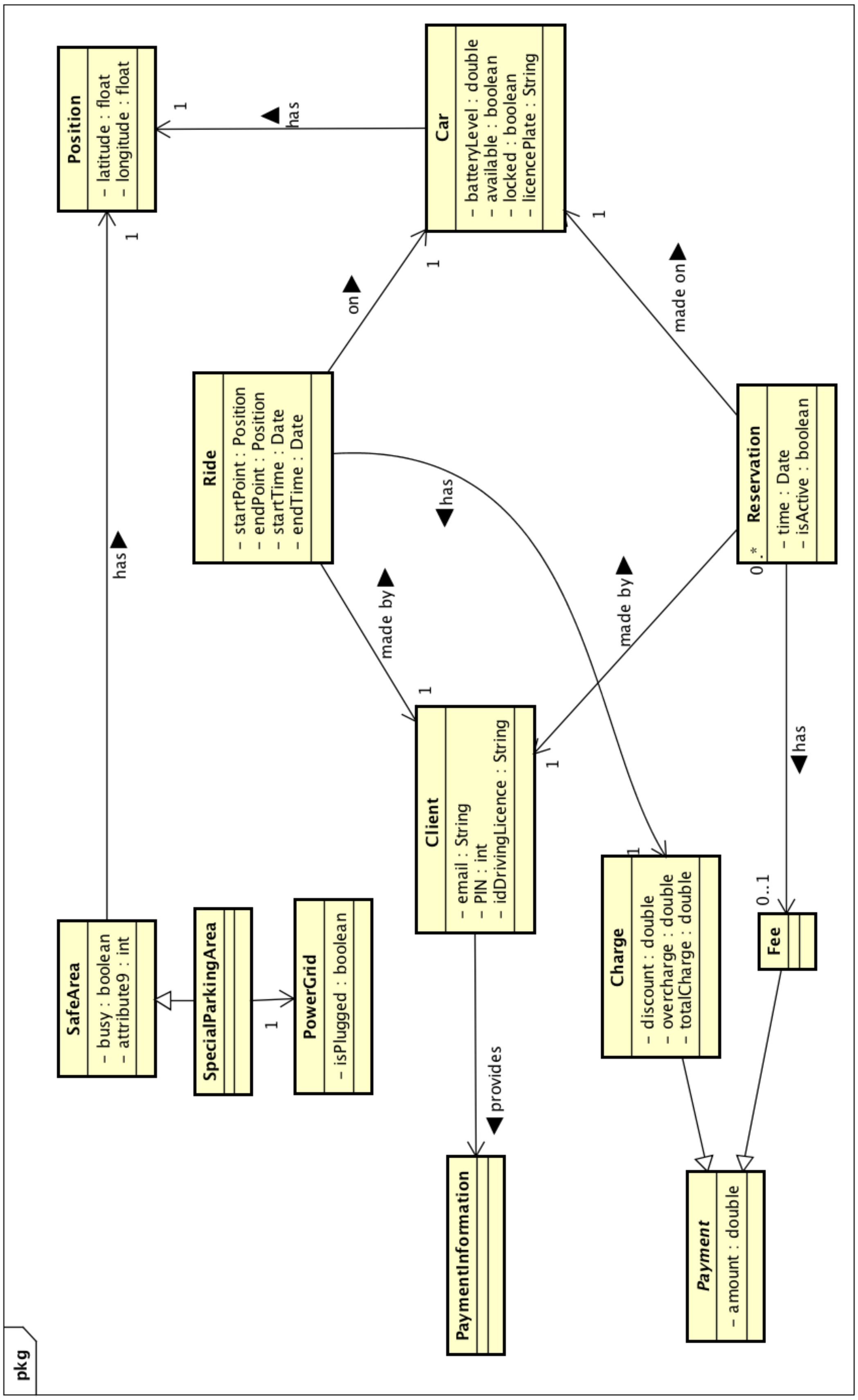
Host Passengers

Name: Host Passengers during a ride
Actors: Client
Entry Conditions: Client must be in the car and he is going to drive
Flow of events:

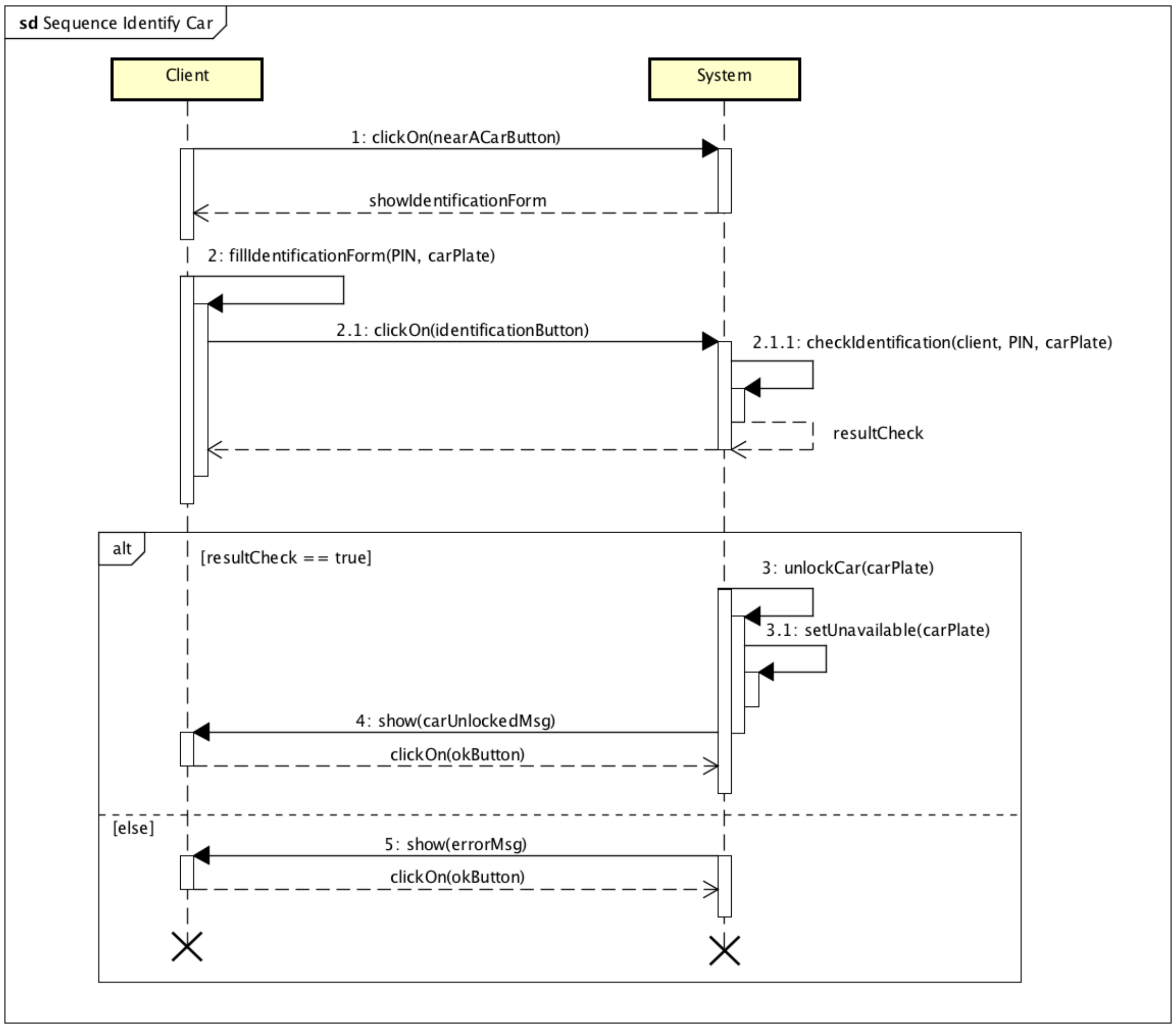
- The Client is driving hosting two or more passengers in the car (respecting the limit capacity of the car)
- The system detects there are more people in the car
- The system will apply a discount of 10% on the ride

Exit Conditions: The system calculates a discount for the last Client's ride
Exceptions: there are no exception

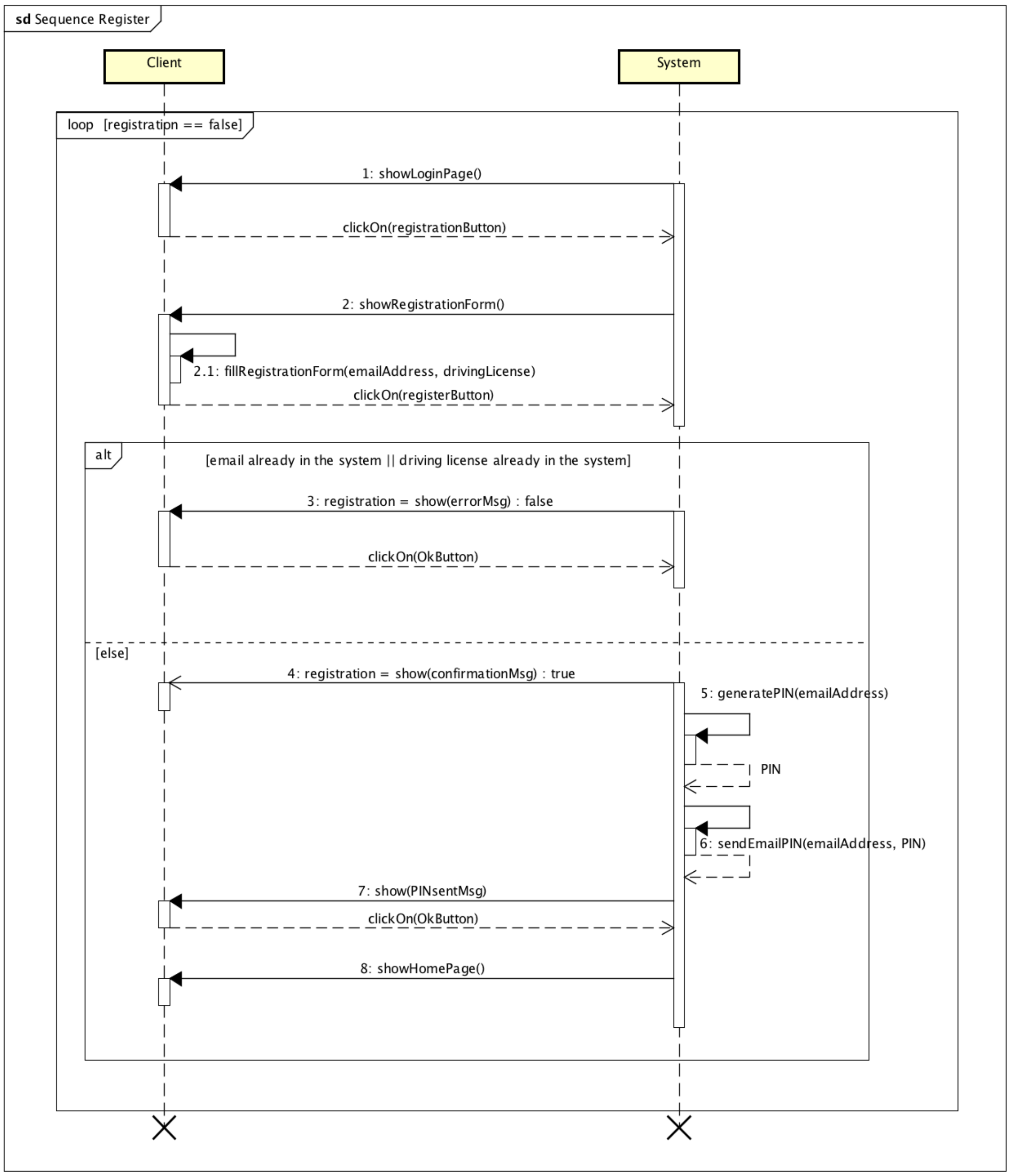
Class Diagram



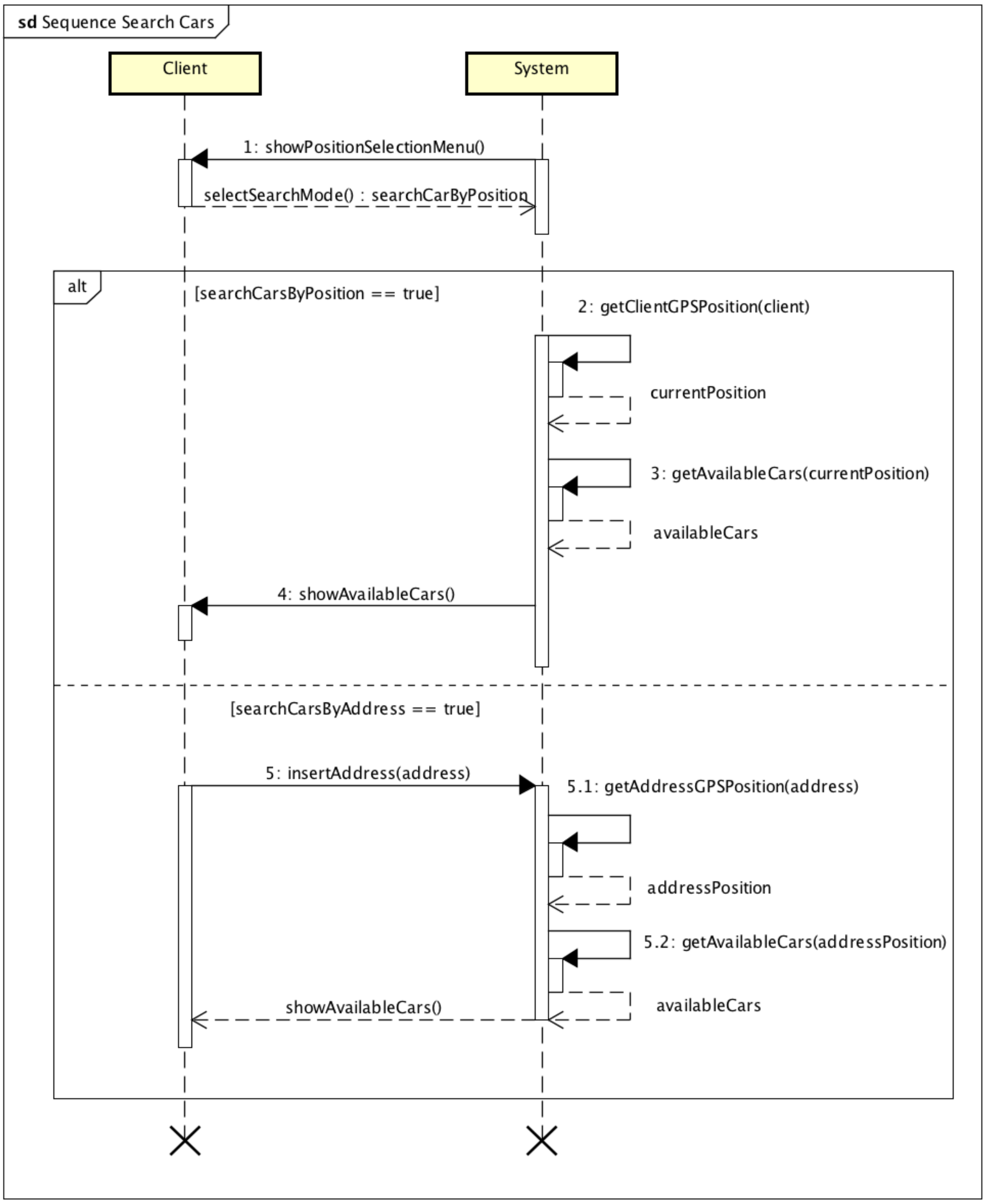
Sequence Diagrams



Car identification

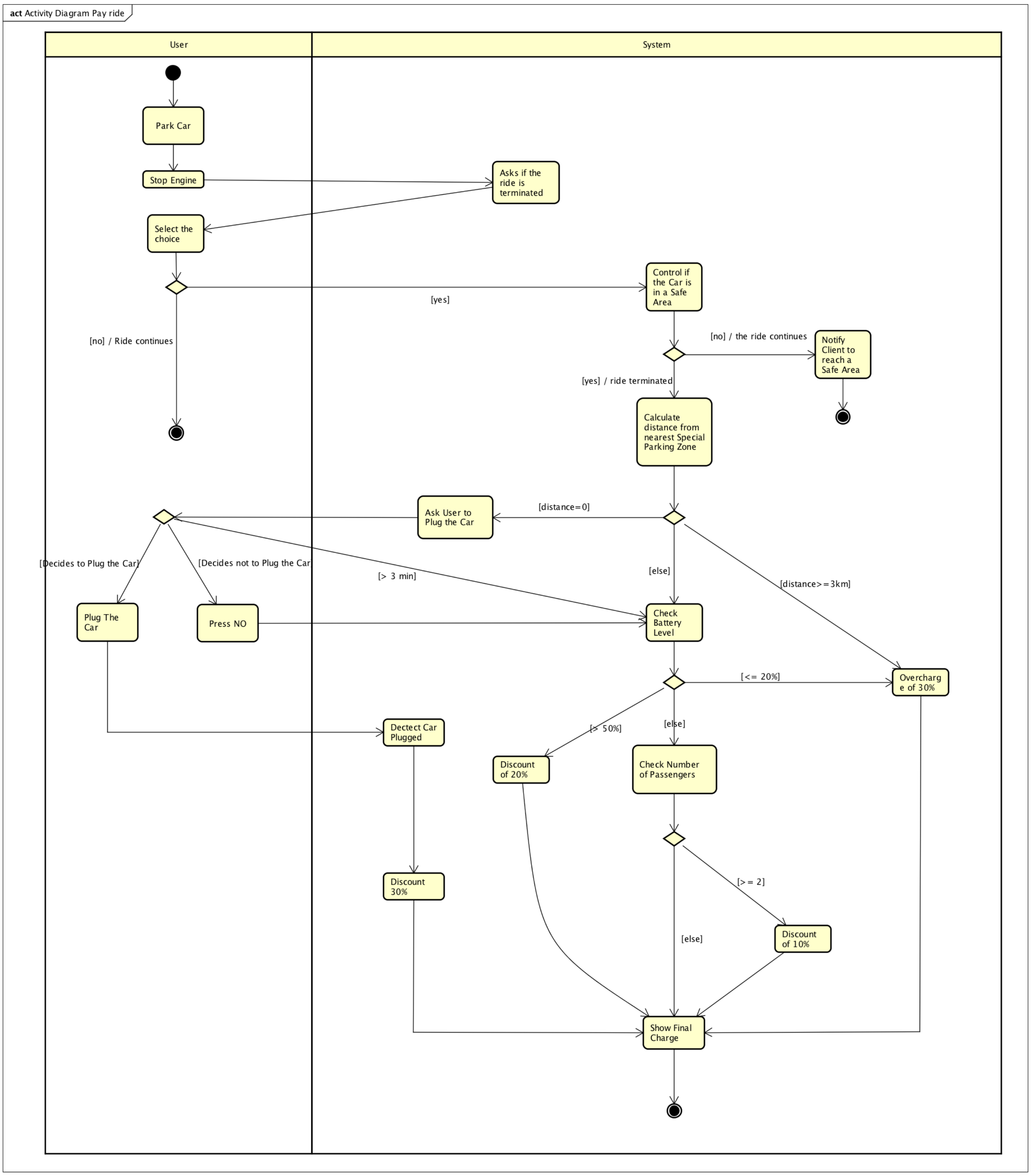

sd Sequence Register
Client
System
loop [registration == false]
1: showLoginPage()
clickOn(registrationButton)
2: showRegistrationForm()
2.1: fillRegistrationForm(emailAddress, drivingLicense)
clickOn(registerButton)
alt
[email already in the system || driving license already in the system]
3: registration = show(errorMsg) : false
clickOn(OkButton)
[else]
4: registration = show(confirmationMsg) : true
5: generatePIN(emailAddress)
PIN
6: sendEmailPIN(emailAddress, PIN)
7: show(PINsentMsg)
clickOn(OkButton)
8: showHomePage()

User registration


sd Sequence Search Cars
Client
System
1: showPositionSelectionMenu()
selectSearchMode() : searchCarByPosition
alt
[searchCarsByPosition == true]
2: getClientGPSPosition(client)
currentPosition
3: getAvailableCars(currentPosition)
availableCars
4: showAvailableCars()
[searchCarsByAddress == true]
5: insertAddress(address)
5.1: getAddressGPSPosition(address)
addressPosition
5.2: getAvailableCars(addressPosition)
availableCars
showAvailableCars()

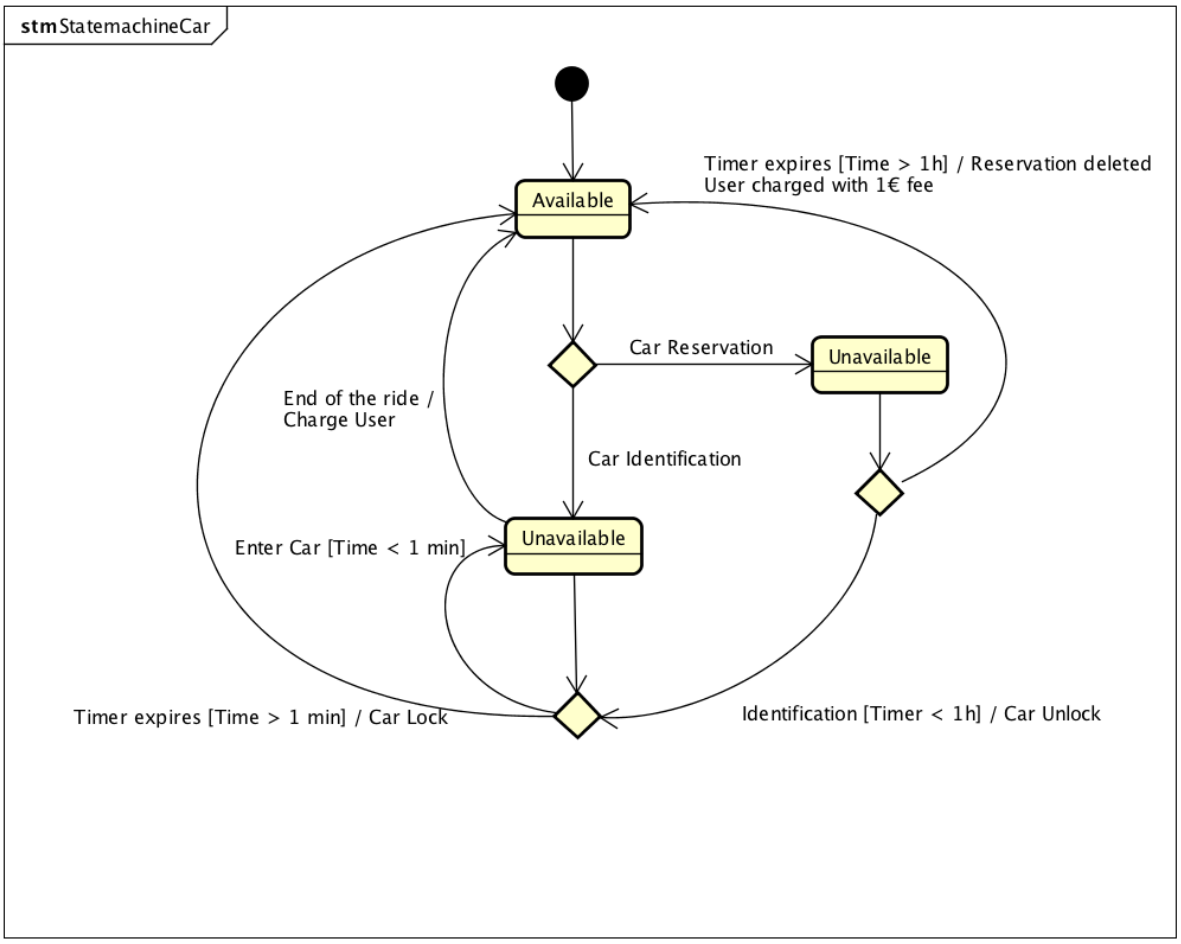
Car research

Activity Diagram



Ride Charging

State Diagram



Car states

ALLOY MODELING

Model

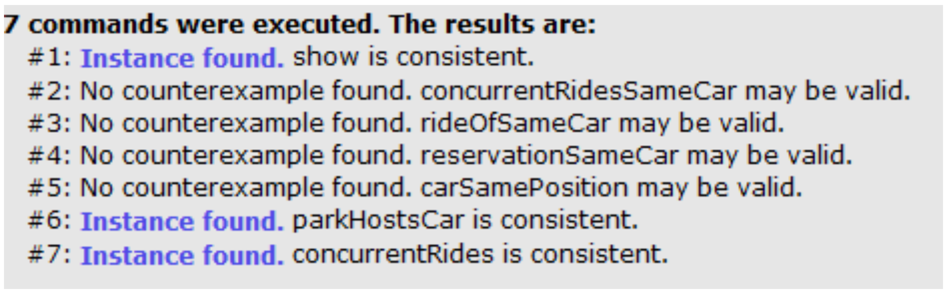
```
open util/boolean

//CENTRALMANAGER

one sig CentralManager{
        cars: some Car,
        safeAreas: some SafeArea,
        clients: some Client,
        rides: some Ride,
        reservations: some Reservation
}
{
        cars = Car
        safeAreas = SafeArea
        clients = Client
        reservations = Reservation
        rides = Ride
}

//////////////////////////////////////////// CAR

sig Car{
        position: one Position,
        availability: one Bool,
        locked: one Bool,
        licensePlate: one LicensePlate,
        batteryLevel: one Int,
        engineOn: one Bool
}
{
        batteryLevel>= 0 && batteryLevel <=100
        engineOn=True => availability=False && locked=False
        batteryLevel<= 20 => availability = False
        availability=True => engineOn=False && locked=True
        batteryLevel = 0 => engineOn = False
}

fact engineOffWhilePlugged{
        all c:Car, s:SpecialSafeArea | parkHostsCar[c,s] and s.powerGrid.used=True=>
c.engineOn = False
}

fact unicityPositionAndLicensePlate{
        no disjoint c1,c2:Car | c1.position = c2.position
        no disjoint c1,c2:Car | c1.licensePlate = c2.licensePlate
}

assert unlockedCarIsUnavailable{
        all c:Car | c.locked=False => c.availability=False
}

fact noLockedCarInUse{
        all c:Car, p:SafeArea | one r:Ride| not parkHostsCar[c,p] => r.car = c && r.charge =
none
}

sig Position{}

sig LicensePlate{}

////////////////////////////////// SAFEAREA

sig SafeArea{
        position: one Position
}

sig SpecialSafeArea extends SafeArea{
        powerGrid: one PowerGrid
}
```

```
fact unicityPositionSA{
	no disjoint s1,s2: SafeArea | s1.position = s2.position
}
fact unicityPowerGrid{
	no disjoint sp1,sp2:SpecialSafeArea | sp1.powerGrid = sp2.powerGrid
}

fact usedPowerGridImpliesCarOnSafeArea{
	all s:SpecialSafeArea| one c:Car| s.powerGrid.used=True =>  parkHostsCar[c,s]
}

sig PowerGrid{
	used: one Bool
}

///////////////////////////////////// CLIENT

sig Client{
	email : one Email,
	pin: one Int,
	drivingLicense: one DrivingLicense
}
{
	pin > 0
}

fact unicityEmailAndPINAndDL{
	no disjoint c1,c2:Client | c1.email = c2.email or c1.pin = c2.pin or
	c1.drivingLicense = c2.drivingLicense
}

sig Email{}

sig DrivingLicense{}

//////////////////////////////////////// RIDE

sig Ride{
	car: one Car,
	driver: one Client,
	startPosition: one Position,
	endPosition: lone Position,
	timeStart: one Int,
	timeEnd: lone Int,
	numberOfPassengers: one Int,
	charge: lone Charge
}
{
	numberOfPassengers >= 0 && numberOfPassengers <= 4
	timeStart >= 0
	timeEnd > 0
	timeEnd > timeStart
	charge != none => timeEnd != none
	startPosition = SafeArea.position
	endPosition = SafeArea.position
}

sig Charge{}

pred concurrentRides[r1, r2: Ride]{
	r1 != r2 && r2.timeStart >= r1.timeStart && r1.timeEnd >= r2.timeStart
}

fact noConcurrentRidesSameCar{
	no disjoint r1,r2:Ride | concurrentRides[r1,r2] => r1.car = r2.car || r1.driver =
r2.driver
}

assert noConcurrentRidesSameCar{
	all r1,r2:Ride | r1!=r2 and r1.driver = r2.driver => not concurrentRides[r1,r2]
}

pred parkHostsCar[c:Car,p:SafeArea]{
	c.position=p.position
}
```

```
fact carInUseUnclocked{
	all r:Ride | r.timeEnd = none => r.car.locked = False && r.car.availability = False
}

//////////////////////////////////////// RESERVATION

sig Reservation{
	car: one Car,
	client: one Client,
	timeStart: one Int,
	timeEnd: lone Int,
	active: one Bool
}
{
	timeStart >= 0
	timeEnd != none => timeEnd > timeStart
	active = True => car.availability = False && car.engineOn=False && car.locked=True
	active = True <=> timeEnd = none
	timeEnd != none <=> active=False
}

fact noMultipleActiveReservationsOnSameCar{
	no disjoint r1,r2:Reservation | r1.active=True and r2.active = True => r1.car =r2.car
}

fact noPostRideOnActiveReservation{
	all reserv:Reservation | no ride:Ride | reserv.active=True => ride.car = reserv.car &&
ride.timeStart > reserv.timeStart
}

pred sameCarRideReserv[ride:Ride, reserv:Reservation]{
	ride.car = reserv.car
}

fact followingReservation{
	all r1,r2:Reservation | r1 != r2 && r1.car = r2.car && r2.timeStart > r1.timeStart =>
r1.active = False
}

fact rideFollowsReserv{
	all r1:Reservation, ride:Ride | r1.car=ride.car && ride.timeStart>r1.timeStart =>
r1.active= False
}

pred concurrentRideAndReserv[ride:Ride, res:Reservation]{
	(res.timeStart>=ride.timeStart && res.timeStart <=ride.timeEnd) or
(ride.timeStart>=res.timeStart && ride.timeStart <=res.timeEnd)
}

fact noConcurrencyOfReservationAndRide{
	all res:Reservation, ride:Ride | res.car=ride.car => not concurrentRideAndReserv[ride,
res]
}

/////////////////////////////////////////////////////////////////////////////////////

fact setsRelations{
	LicensePlate = Car.licensePlate
	Charge = Ride.charge
	Email = Client.email
	PowerGrid = SpecialSafeArea.powerGrid
	DrivingLicense = Client.drivingLicense
	Position = (Car.position + SafeArea.position)
}

pred show{}

run show for 3
check unlockedCarIsUnavailable
check noConcurrentRidesSameCar
run concurrentRides
run parkHostsCar
run sameCarRideReserv
```

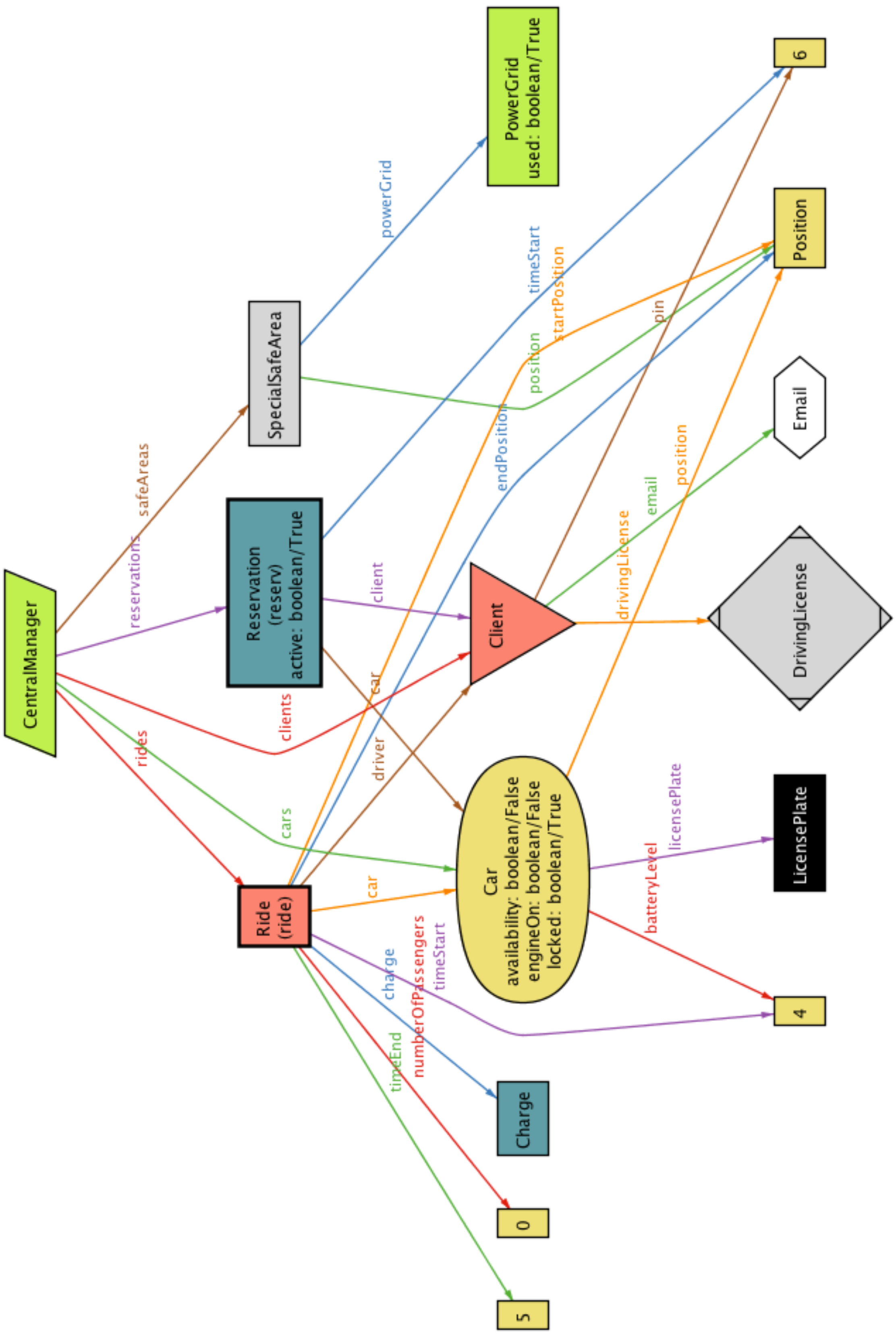
Alloy Result

Model consistent:



```
7 commands were executed. The results are:
  #1: Instance found. show is consistent.
  #2: No counterexample found. concurrentRidesSameCar may be valid.
  #3: No counterexample found. rideOfSameCar may be valid.
  #4: No counterexample found. reservationSameCar may be valid.
  #5: No counterexample found. carSamePosition may be valid.
  #6: Instance found. parkHostsCar is consistent.
  #7: Instance found. concurrentRides is consistent.
```

World Generated



FUTURE DEVELOPEMENT

In the future the service could be extended to other vehicles like bikes, scooters or bigger cars.
In the future the service could be extended to other cities.
In the future the service could feature a system of notification to the clients to keep them informed about some aspect of the system. (E.g. expiring reservations)

USED TOOLS

The tools we used to create this RASD document are:

• *Astah Professional*: for uml models
• *Github and Google Drive*: for version controller
• *Pencil / Adobe Photoshop*: for mockup
• *Microsoft Word*: to write the document
• *Alloy Analizer 4.2*: to prove the consistency of our model

HOURS OF WORKS

Giorgio Marzorati

27.10.2016 - 2.30 h
29.10.2016 - 2.30 h
30.10.2016 - 2.30 h
02.11.2016 - 1.00 h
03.11.2016 - 3.00 h
04.11.2016 - 3.00 h
06.11.2016 - 1.00 h
07.11.2016 - 1.00 h
09.11.2016 - 2.00 h
11.11.2016 - 5.00 h
12.11.2016 - 8.00 h
13.11.2016 - 10.00 h

Aniel Rossi

27.10.2016 - 2.30 h
29.10.2016 - 2.30 h
30.10.2016 - 2.30 h
02.11.2016 - 2.00 h
03.11.2016 - 3.00 h
05.11.2016 - 3.00 h
06.11.2016 - 2.00 h
07.11.2016 - 1.00 h
10.11.2016 - 5.00 h
11.11.2016 - 2.00 h
12.11.2016 - 8.00 h
13.11.2016 - 9.00 h

Andrea Vaghi

27.10.2016 - 2.30 h
29.10.2016 - 2.30 h
30.10.2016 - 2.30 h
02.11.2016 - 1.00 h
03.11.2016 - 3.00 h
04.11.2016 - 3.00 h
06.11.2016 - 1.00 h
08.11.2016 - 3.00 h
09.11.2016 - 2.00 h
10.11.2016 - 4.00 h
12.11.2016 - 9.00 h
13.11.2016 - 10.00 h

CHANGELOG

- v1.1
  - General document's look improvement
  - Alloy model improvement
  - Quality of service requirements added
  - The Assistance team is now an external resource