PowerEnJoy

Requirement Analysis and Specification Document

Politecnico di Milano

A.Y. 2016/2017

Bolshakova Liubov, matr. 876911

Gao Xiao, matr. 876265

Kang Shuwen, matr. 876245

November 13, 2016

CONTENT

1. Introduction

1.1 Purpose

Our team is focus on specifying the accomplishment of the service PowerEnJoy, this is a system which supply the electric cars to facilitate the transportation of citizens. In particular, the PowerEnJoy will offer the accessible electric cars in the reachable areas in the city and admeasure them rationally to users; the system also set rider clauses to standardize the driving behaviors of users by provide discounts and add compensations.

This document aims to describe all the functional and non-functional requirements of system, simplify the processes and optimize the methods of operations. The RASD is intended to:

Provide a megascopic view about the system, help the manager to make decision about function expanding and service extending.

Provide a structure of system to programmers, which clarify and elaborate the programs that asked to realize.

1.2 Scope and Problem Description

In particular, the system wants to:

1) Provide an easy and efficient approach to users

2) Guarantee an energetic and accessible car service

Anyone can register to be a user by supply the valid driving license information and credit card information.

After the login, users are able to search the available cars around the location they are or around the certain position they inputted. System response the search request by enumerates the available cars around them in an available queue as well as the basic information. The basic information of cars include: the accurate location, the distance to start position, dump energy, and passenger capacity.

Users are able to reserve at most one car every time, they can cancel the reservation in free within a certain time after the reservation request. The user can't pick the car in the one hour time span should pay one euro, and the reservation will be released as well.

The system will move away the car from the available queue as soon as the car is reserved, it will move back if user cancels the reservation or the car doesn't be picked up in one hour.

If the user accesses the car with time limit, he /she can send request to pick up the car, system will unlock the door and let user enter. The system will start charging the car as soon as user ignites the engine, and the charging information will be notified through the screen in the car.

The user is able to choose the "money saving option" after picking up the car, and the system will response the optimal terminal station to park the car.

Users are able to terminate the ride in the safe area, the charging will stop and the car will be locked when all passengers leave the car, no matter how engine works.

If the user parks the car in the unsafe area and leave, it is necessary to lock the car for the reason of security; the system will consider that as a complete driving process with unsafe parking.

Users can recharge the car in the power gird station by plugging the car into power grid.

System estimates the driving behaviors in this ride according to the rider clauses as follow to reset payment:

If the user took more than 2 passengers, apply 10% discount.

1. If the user parked the car in safe area with more than 50% dump energy, apply 20% discount
2. If the user recharged the car after parking, apply 30% discount.
3. If the place user parked car is in a longer distance than 3KM to the nearest power grid station, apply 30% compensation.
4. If the user parked the car with less than 20% dump energy, apply 30% discount.
5. If the user parks the car in an unsafe area, apply 30% compensation.

1.3 Goals

The objectives that system has to satisfy:

[G1] to allow users approach the car-sharing service easily

[G2] to allow users to make a reservation

[G3] guarantee the integrity of a ride.

[G4] guarantee the availability of cars

[G5] guarantee the accessibility of cars

[G6] distribute the cars in a reasonable distribution in the city

1.4 Domain Assumptions

We suppose that the following conditions are true in the analyzed world:

- The parking slots and power grid stations of PowerEnJoy are abundance for the cars.
- Power gird stations are distributed reasonably in the city.
- All passengers will close the door while leaving.
- When cars are parking, they don't make any power consumption.
- The cars have no hitch and are maintained on time.
- The network communication between users and the system in a good condition; hence all data are accurate delivery.
- The users who can access the PowerEnJoy are all have valid driving license and credit card.
- All valid credit cards can be charged a fee.
- Cars are all equipped with a GPS positioning system and connected to the network, the PowerEnJoy app on the phone is qualified to use the GPS equipped on the phone. The positon information supplied by GPS is correct and accurate.

- The GPS on cars can't be switched off; it has its own battery which is independent of the car.
- Cars can be opened from inside even the door is locked.
- There is no accident while driving.
- The car will not be overloaded.
- The staff of the PowerEnJoy already logged in.

1.5 Test Assumption

The following assumptions are specification of information not clear in the document:

1. Only the user can execute the driving action.
2. User is a special passenger.
3. The user name is the ID of driving license correspondingly.
4. The system will keep charging the car even it is flameout by some incorrect driving action.

1.6 Glossary

1. Safe area: the admissive parking area that be pre-defined by the PowerEnJoy system, the power grid stations are the subset of safe area.
2. Car: the cars that supplied for the car-sharing service in the PowerEnJoy system.
3. Basic car information: the basic information that helps guests and users to make decisions, include the dump energy, location information, distance to the setting location, the passenger capacity.
4. Starting position: the current positons of user or the positions user inputted to start a ride.
5. Car state: every car has four states:

i. Available: the car has dump energy more than 50% and be parked in the safe area.

ii. Reserved: the car is reserved by the users but still parks in the safe area.

iii. In use: the car has been picked up by users.

iv. Out of service: the car has battery more than 50% empty or it is not been parked in the safe area

6. Available queue: a queue that maintains available cars
7. Electric devices: the GPS and power plug senor, weight sensor, display screen, battery sensor, door state sensor, locks of door in the car, and the sensor on the power grid.
8. Unsafe area: the area that is not include by pre-defined safe area
9. Incorrect parking: the parking in the unsafe area, the parking which doesn't park the low battery car in power gird station and the parking that park the low battery car in power grid station but doesn't charge the car with power grid.
10. System: the whole system which include the electric devices and the PowerEnJoy system background.
11. Complete driving process: in this system, the complete driving process is start with ignite the engine and end with all passengers leave the car.

1.7 Further Developments

This software has some limitations that could be improved in the future:

1. The emergency alarm front feed. While driving, it is better to notify the user real-timely about the traffic accident just happened in this city for a safer and faster ride.
2. The system can analyze the possible starting positions/destinations from the history record of each user and show them as starting position/destination options, to comfily the input process.
3. Allow users to leave in a short time interval in a ride.

1.8 Used Tools

The tools in used to create this RASD are:

StarUML
To create UML diagrams

Balsamiq mock-up
To create the mock-ups of the interfaces

Dropbox and Google Docs
To share the files and synchronize our work

2. Specific Requirement

According to the “scope and description”, we specified the requirement as follows:

2.1 Functional Requirements

R1: allows a guest executes the registration to become a user.

R2: allows users to login into the system.

R3: allows users to search the available cars near the starting position.

R4: allow users to make reservation.

R5: Allow users to cancel the reservation.

R6: allows the users to pick up the car.

R7: allow users to get discount or compensation by the rider clauses as follows:

7.1 Apply 10% discount when user takes more than two passengers

7.2 Apply 20% discount when user leaves more than 50% dump energy

7.3 Apply 30% discount when user recharges the car after a complete driving process

7.4 Apply 30% compensation when user parked the car 3KM away the nearest power grid station

7.5 Apply 30% compensation when user leaves less than 20% dump energy

R9: Allow users to access the “saving money option”

R10: send the submission after the complete driving process.

R11: correct the unsafe parking situations

2.2 Nonfunctional Requirements

The integrity of a ride

The life span of a ride is the time interval that a specific car relates to a specific user. It includes the complete driving process and 20 minutes extra time for users to recharge the car.

- Although the car should be parked in the safe area in a ride, but if user park the car outside the safe area and just leave for some emergency, this should be consider as a complete driving process. Because it is a waste of car resource to spend time on waiting users to continue the driving process.
- The engine should be turn off at the end of every ride.

Accessibility and security of cars

- The system should monitor the car state real-timely, the available cars which is released from reservation should be move back to the available queue immediately.
- The system should move away the car from the available queue when it was reserved at the first time, so a car can only belongs to one specific ride every time.
- After a ride, the cars which are parked inside the safe area and with higher than 50% dump energy should be move back to the available queue immediately.
- The cars which are recharged to more than 50% dump energy should be put into available queue immediately.
- When users do the search operation, the car in available queue should be listed in an increase order of distance and in a decrease order of battery.

The reasonable distribution of cars

- As soon as user stops the car, system should locate the car position and mark the car into “out of service”. Those cars are inaccessible to uses; too much inaccessible cars will lower the service quality. The system should dispatch staffs to handle this.
- after the user confirming the “nearby”, system will unlocks the door, if the user doesn’t enter the car, or user entered the car and then leaved without ignite the engine, the car will be at “in use” state forever, and can’t be released as a complete driving process. For this reason, the system detects the

passenger amount of user at the “in use “state, so long as there are no passengers in the car, the system locks the car, and move the car back into available queue.

The reasonable distribution of cars

- The system should manipulate the parking action by “saving money option” to distribute the car more reasonably. The safe area contains few available cars should be considered firstly.

3. Scenarios Identifying

We report the below list of probability scenarios, in order to clarify how PowerEnJoy system works.

3.1 Scenario 1

Elena is the user of PowerEnJoy, she really enjoy the convenient and economic service it supplies. Elena is in the shopping mall in Sunday afternoon with friends Vanessa, Jessica and Lina. At 4:00 pm she reserves one car near the mall by her mobile phone. They pick up the car successfully, but they don't live in the same place, so Elena sends them home one by one and finally parks the car in a safe area near her home. Since she picks up more than 2 passengers, she is applied a 10% discount on this ride.

3.2 Scenario 2

Jimmy, a user already reserved a car in PowerEnJoy, is walking in the parking slot with a gift in his hand, he is happy because he is going to pick up his wife in the airport, he hasn't see his wife for about 6 month, so he is exciting. He finds the correct car he reserved takes out the mobile phone and clicks "nearby", he pulls the handle of door and it is opened consequently. At this time, his gift drops from his arm, he tries to pick up his gift but he close the door accidently. Then he can't open the door anymore! This is so sad, but it is life, you can't open the door of an empty, parked car. He should wait 20minutes to release from this ride.

3.3 Scenario 3

Bob, who is driving the car on the street, then catches a familiar back in the crowd—Angela. Angela is the one who borrow 10000 euro from Bob and just disappeared, if Bob doesn't take this chance, he may have no chance anymore, so Bob

just stops the car in a random place and runs to Angela. The system stops charging the car and lock the door in the same time.

3.4 Scenario 4

Maria picks up the car successfully, and she enters the car and closes the door. She wants to go to the stadium for a baseball match, but she is a new bee of driving, so she never been there by car. She chooses the “saving money option” to get the correct information about optimal parking position around destination. She inputs the name of stadium, when she gets the response from system; she accurately knows how to go there. She ignites the engine, and the display screen on the car is lighted meanwhile. While the driving, the current charging value which is displayed on the screen is updated every second.

She stops the car in power gird station, but she forgets to tail-off the engine. As soon as she closes the door, the current charging value stops. She leaves less than 30% dump energy, but she also recharge the car, so finally system synthesized the discount and the compensation together. She gets either discount or compensation in this ride.

3.5 Scenario 5

Francesca, the PowerEnJoy staff, gets a mission of moving a car. She is supplied the specific location information about that car. She finds that car in front of a restaurant which is not a safe area, and the car is powered off. She moves hat car with crane. After she placing the car in the nearest power girds station, she takes out her mobile phone for work, and clicks the “mission accomplished “button.

3.6 Scenario 6

Livia is from another city and plans to visit Mr. Wang, a friend of her father who lives here. Livia is suggested to access the PowerEnJoy service for transportation. Livia downloads the PowerEnJoy app, as a guest, she is asked to register first. Livia registers successfully and she logs in. Livia makes a reservation and picks up the car

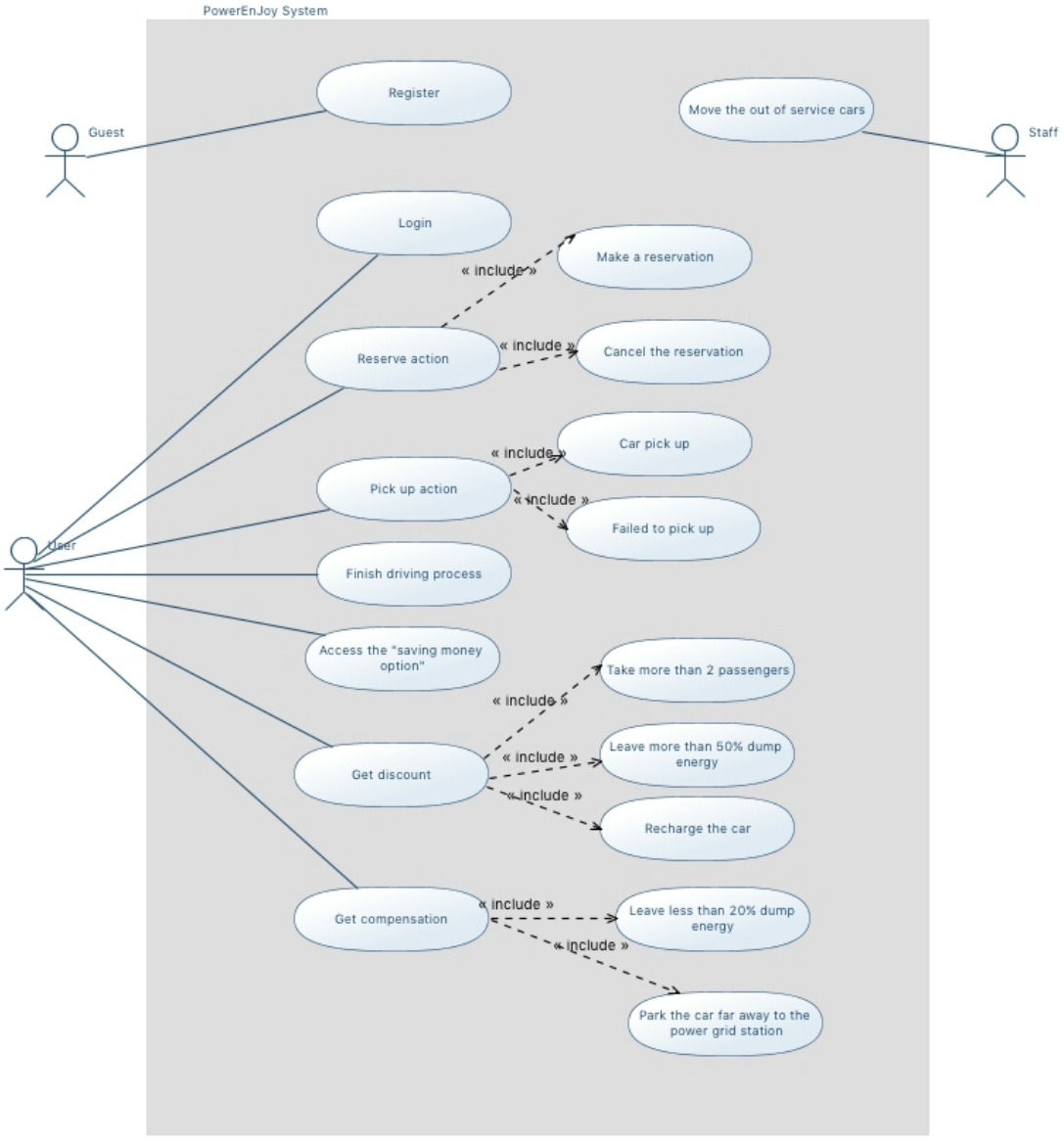
successfully. While she is driving, Mr. Wang calls her and asks if she can do him a favorer to buy a cake in “Cristina”—a cake shop. Livia changes the direction to “Cristina”, consider that she should leave the car while buying cake; it is more convenient to make another reservation near the “Cristina”. She inputs the position of “Cristina” as the starting position, and reserves a car near it.

Livia takes the cake inside one hour; she picks up the new-reserved car and continues her driving to Mr. Wang’s home.

4. UML Models and Use Cases

4.1 Use Cases Diagram

The following is the general use case for this system：



4.2 Actors Identifying

Guest:

The people who are interested to PowerEnJoy service and are willing to become the user by registration.

User:

The people who already done the registration process and has the authority to make reservation and driving the car.

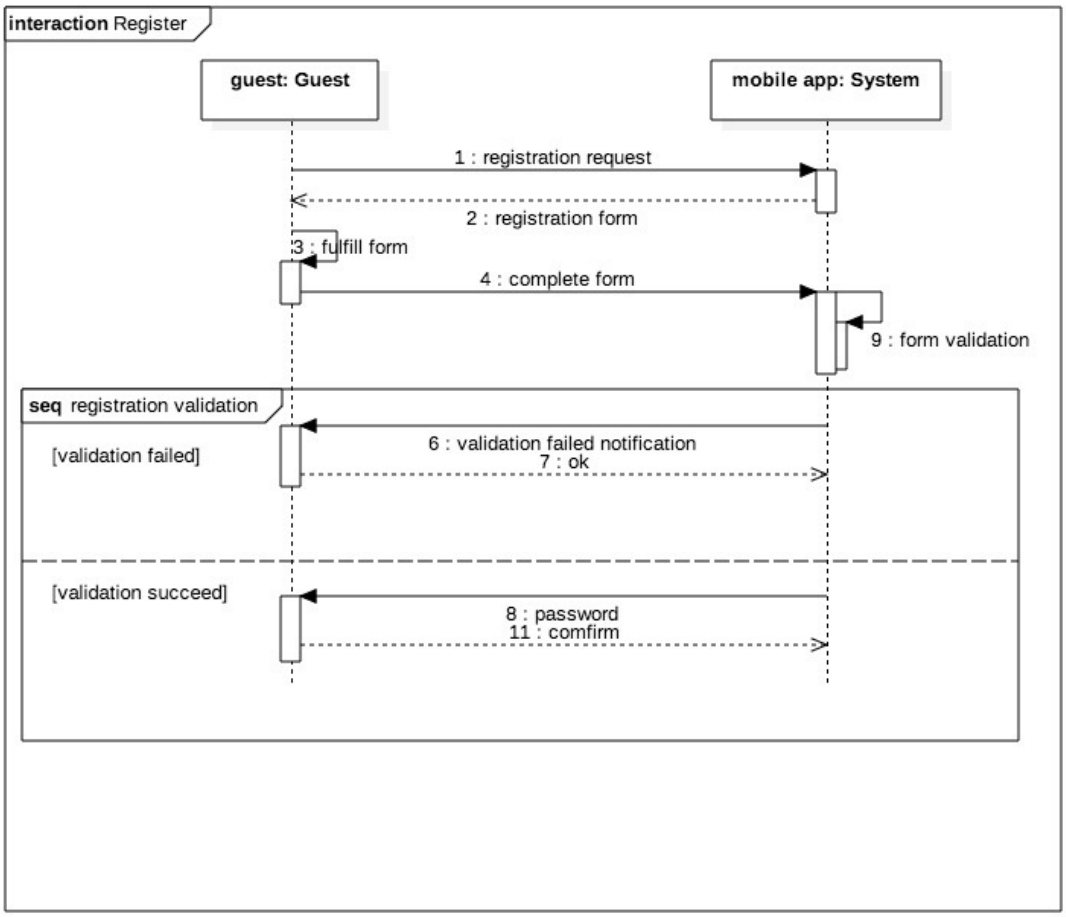
Staff:

The person, who is employed and authorized, is to modify the car which in incorrect parking.

4.3 Use Cases

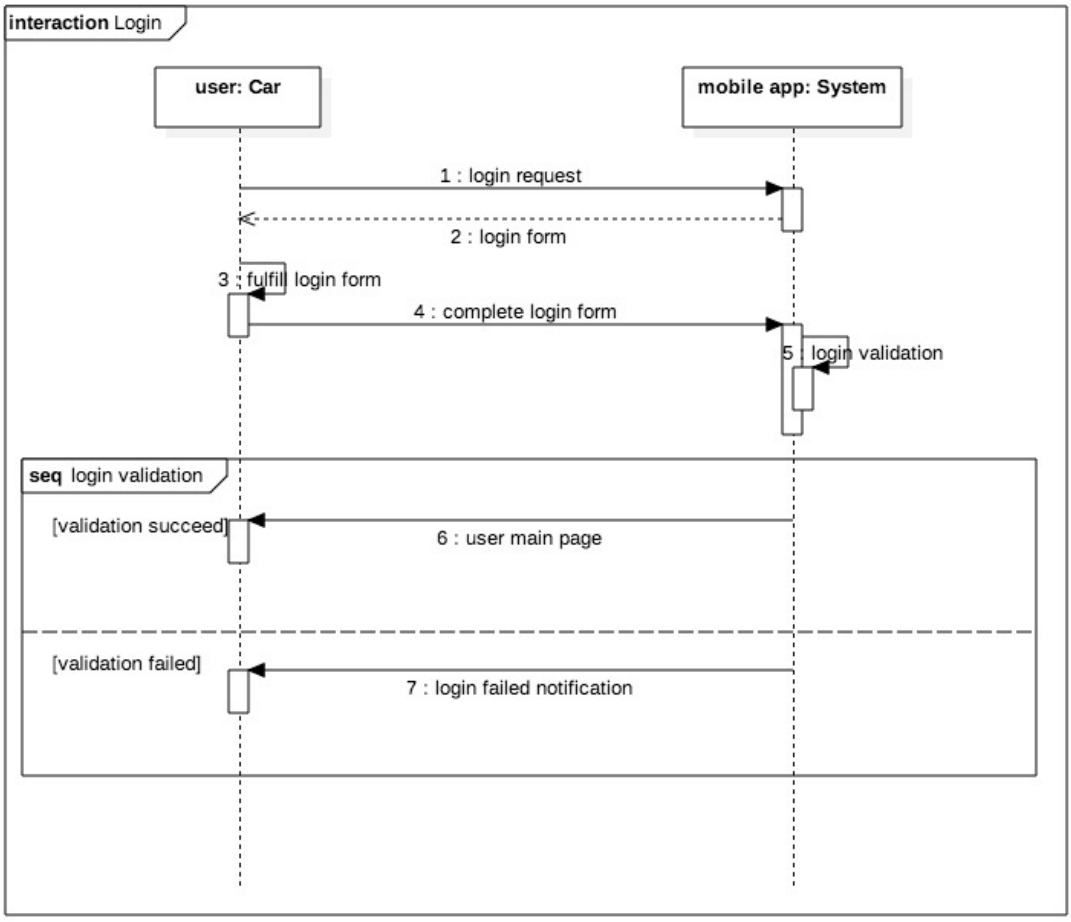
4.3.1 Register

Name	**Register**
Actors	Guests
Assumptions	The guests have PowerEnJoy app in the mobile phone
Flow of events	open the app click the “register” button fill the required information press “confirm” button the system verify the information the system response a unique password system stores the user information and according password in the system database
Exit conditions	Guest registered successfully
Exceptions	The information guest supplied has some spelling errors At least one document is invalid The driving license is already be used for another registration



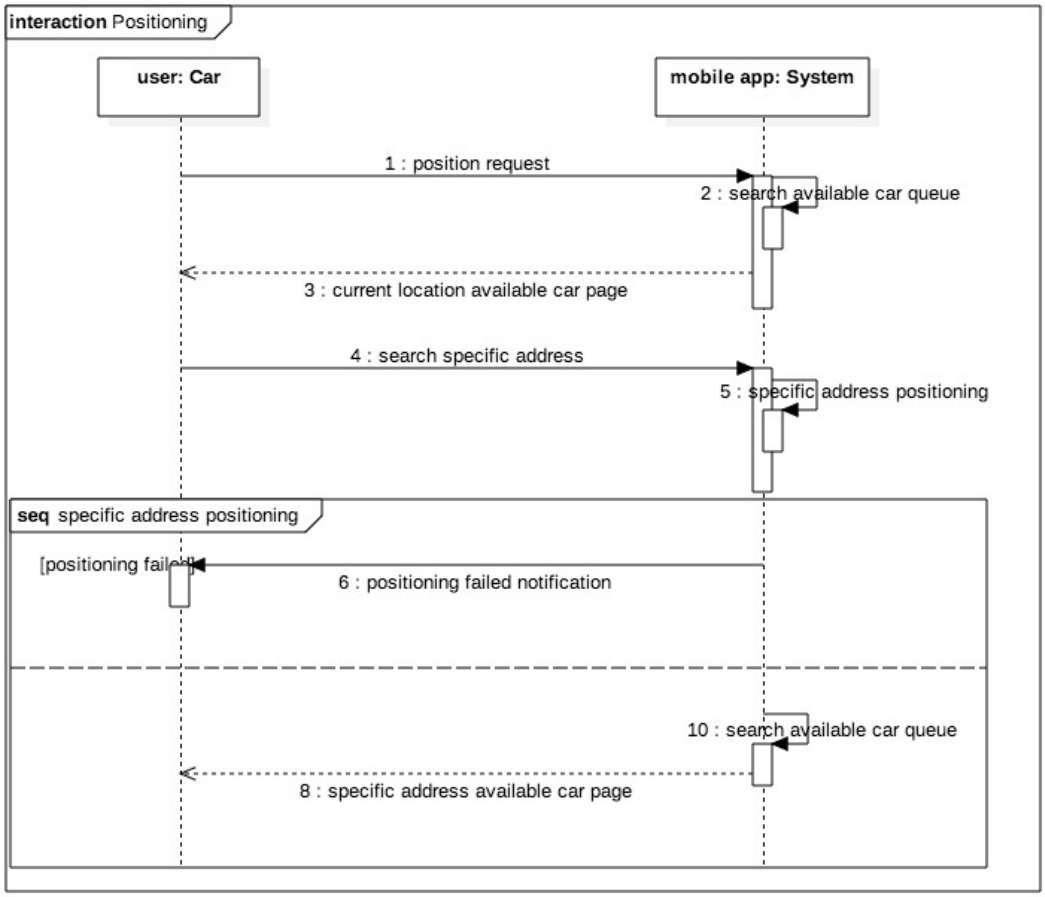
4.3.2 Login

Name	**Login**
Actors	Users
Assumptions	Users are registered successfully
Flow of events	open the app on the mobile phone input the user ID and password click the “login” button system check the login data
Exit conditions	Users login successfully
Exceptions	There are no inputted user ID in the system database The password is not correct



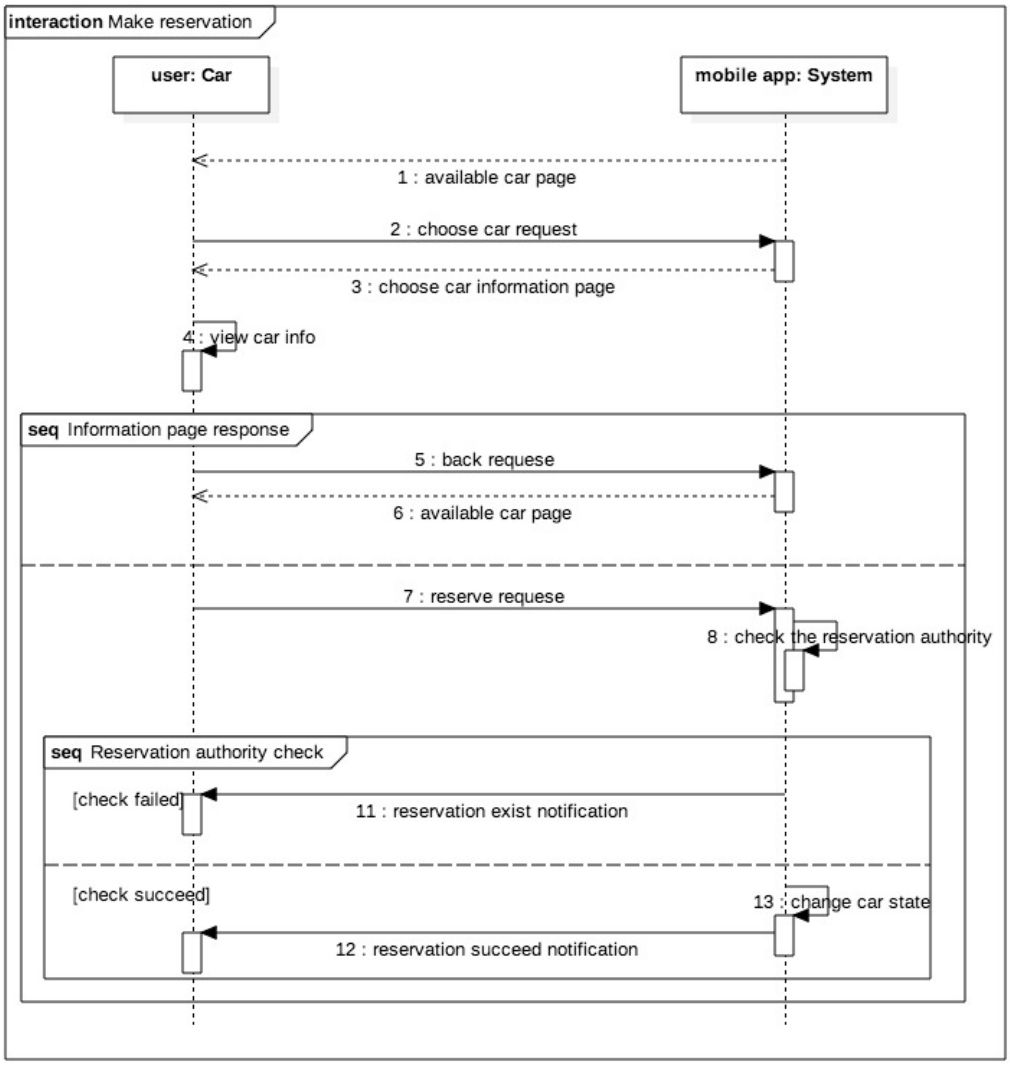
4.3.3 Positioning

Name	**Positioning**
Actors	Users
Assumptions	The users already logged in The GPS equipped in the phone works well
Flow of events	click the “position” button choose the “set other location” by the specific needs input an accurate position if the choice is “set other location” click the “search” button
Exit conditions	System return an available car page
Exceptions	Incorrect specific address.



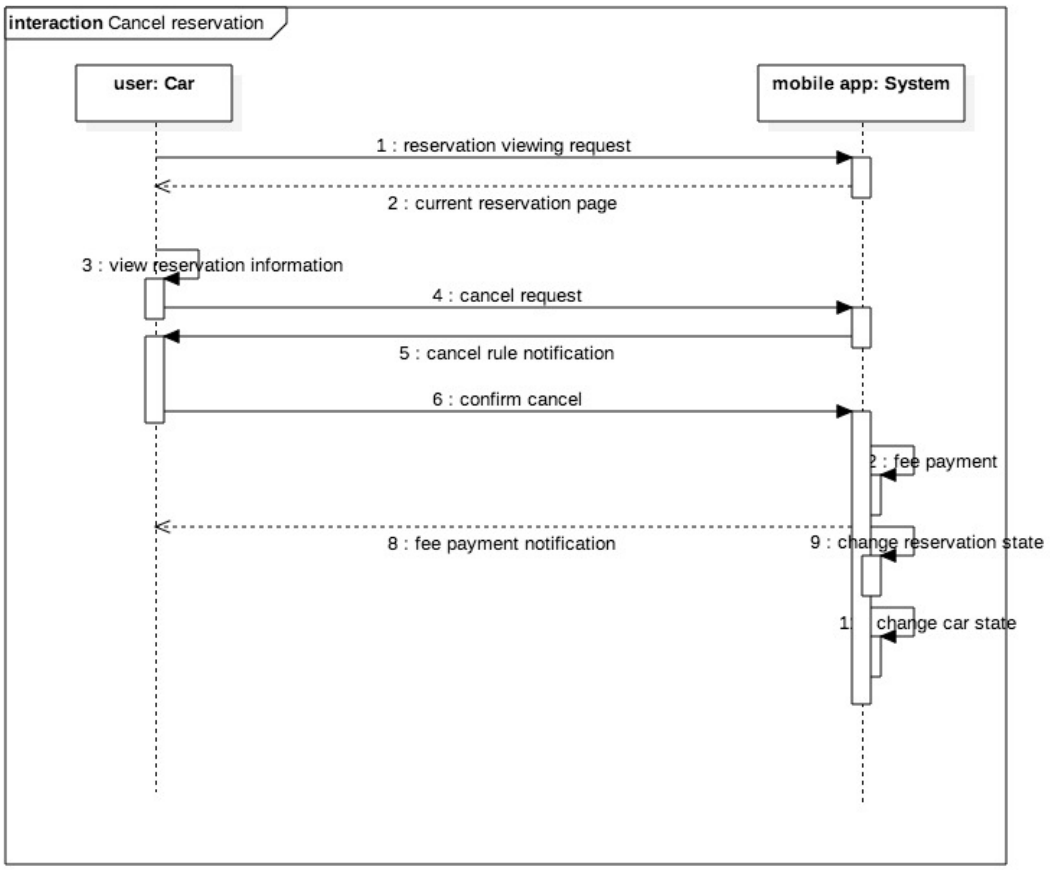
4.3.4 Make a reservation

Name	**Make a reservation**
Actors	Users
Assumptions	The users already logged in The GPS equipped in the phone works well System has returned an available car page
Flow of events	choose the favorite car in the available queue confirm the basic car information in a new page click “reserve” button system check the reservation authority of user system change the car state into reserved system notify the user that you reserved successfully
Exit conditions	User reserve a car successfully
Exceptions	User already has one reservation in this moment



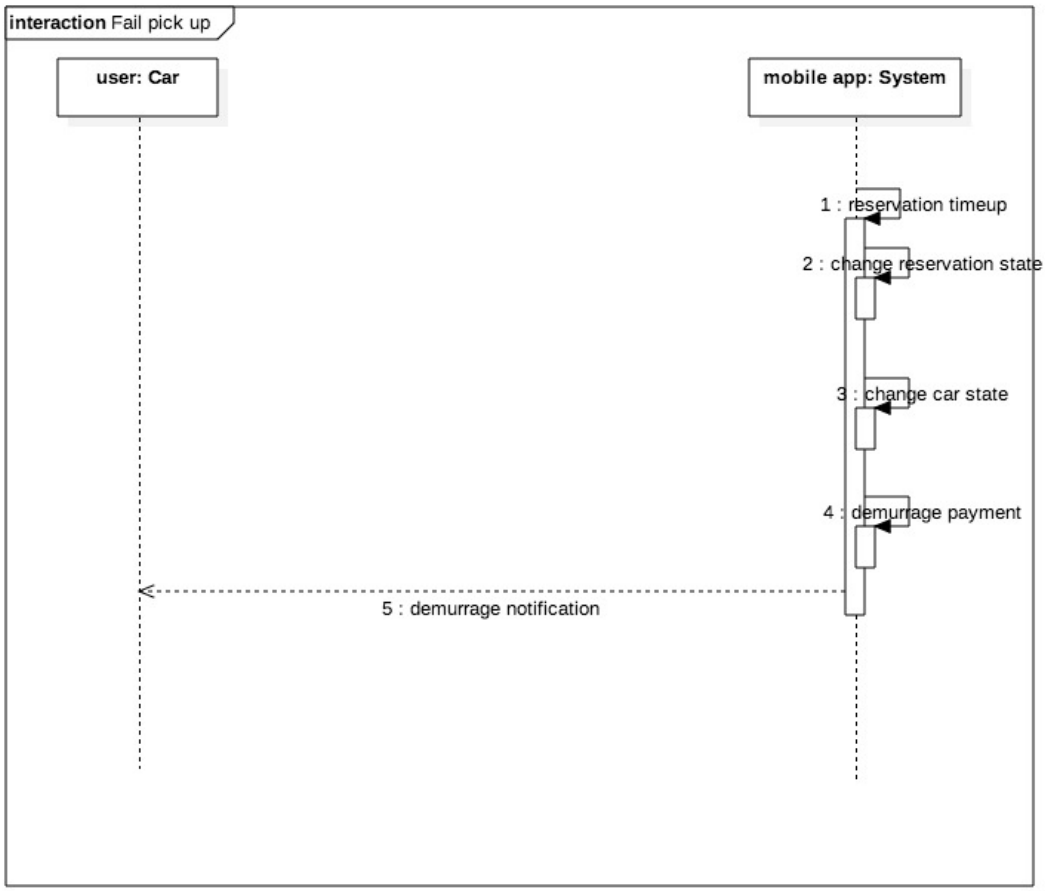
4.3.5 Cancel the reservation

Name	**Cancel the reservation**
Actors	Users
Assumptions	The user has a reservation in this moment
Flow of events	User clicks the “my reservation” button User checks the reservation information User clicks the “cancel” button System asks for confirmation User clicks the “yes” button System notify the user that you canceled this reservation successfully
Exit conditions	Cancel the reservation successfully
Exceptions	The reservation is already be released because it overstep the available time span of canceling



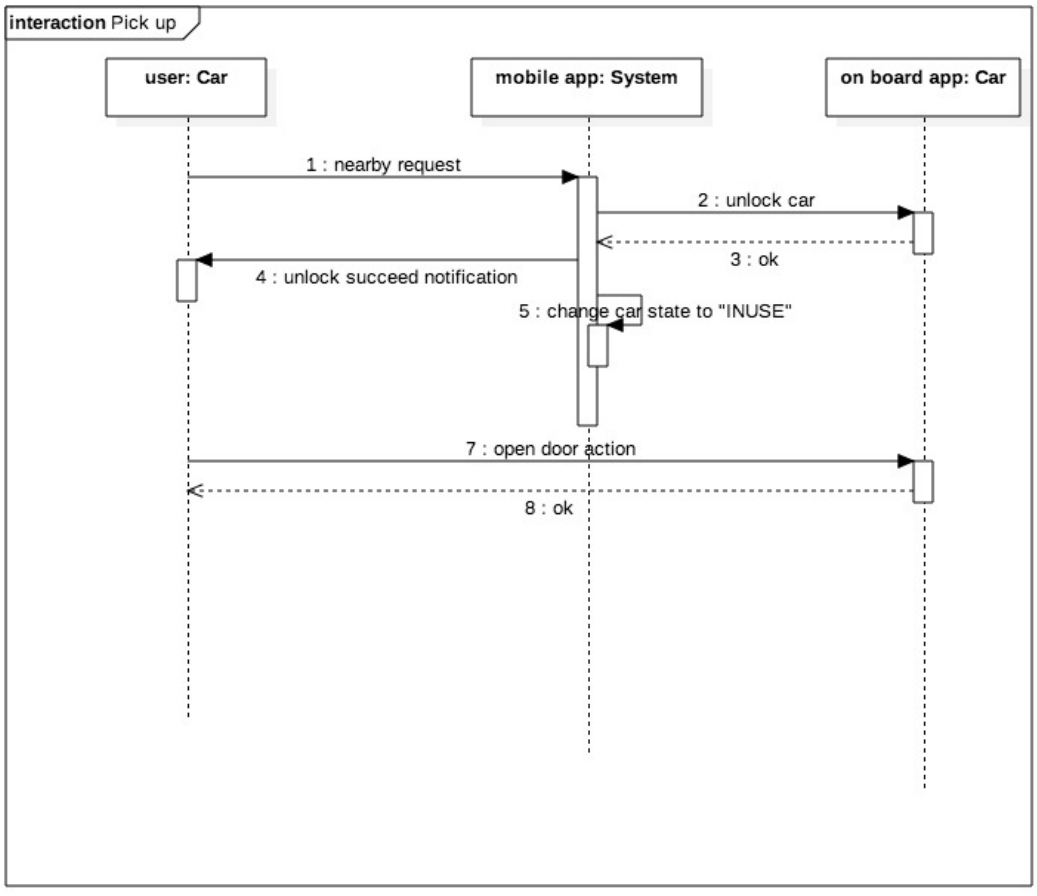
4.3.6 Failed to pick up

Name	**Failed to pick up**
Actors	Users
Assumptions	Users already made the reservation Users has not unlock the car yet
Flow of events	Users didn't pick up the car in time System check the reservation in one hour after the car is reserved System take one euro from users account as demurrage System move back the car into available queue
Exit conditions	Car is available again System take one euro as demurrage
Exceptions	Credit card charge failed



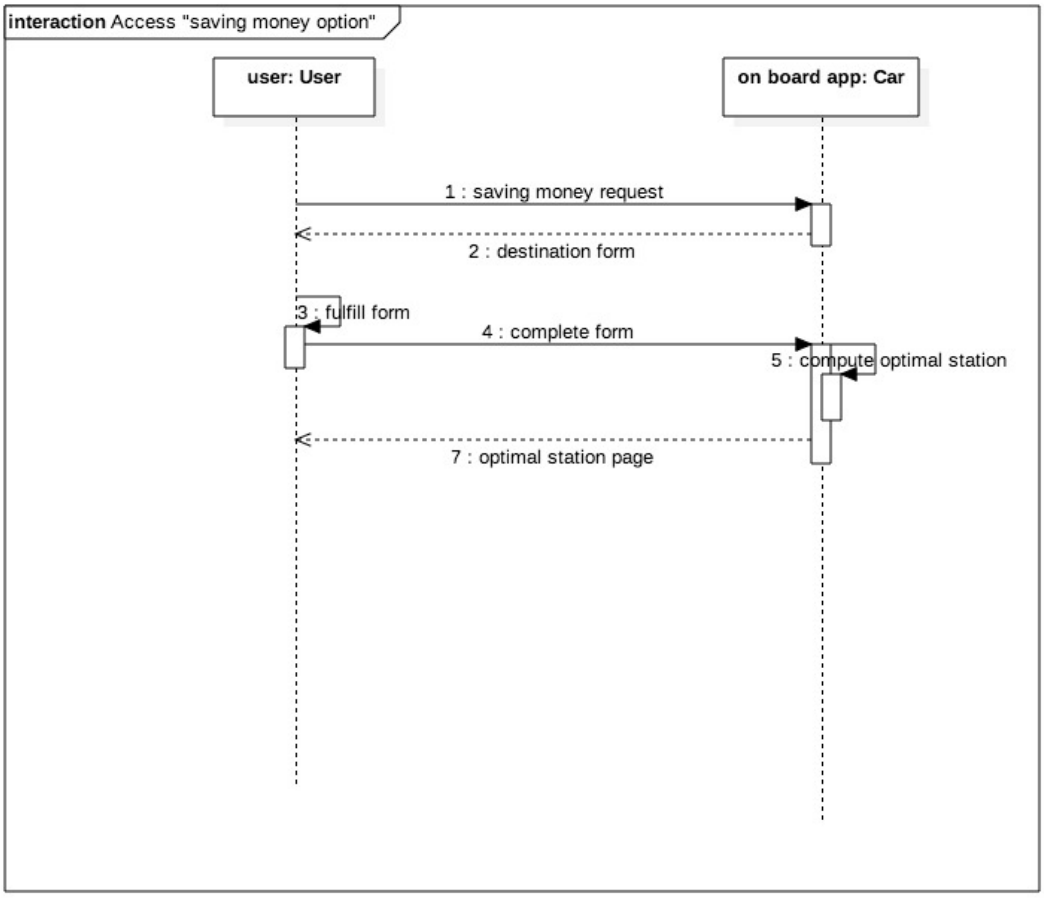
4.3.7 Car pick up

Name	**Car pick up**
Actors	User
Assumptions	The user already made a reservation The user gets to the car within one hour after reserving the car
Flow of events	User clicks the “nearby” button System unlock the car System gets the confirmation and change the car state into “in use” All passengers enter the car Close all the door
Exit conditions	All passengers enter the car and close the door
Exceptions	There is some mechanic breakdown; the door can’t be unlocked by system. The door can’t be closed in a safe condition



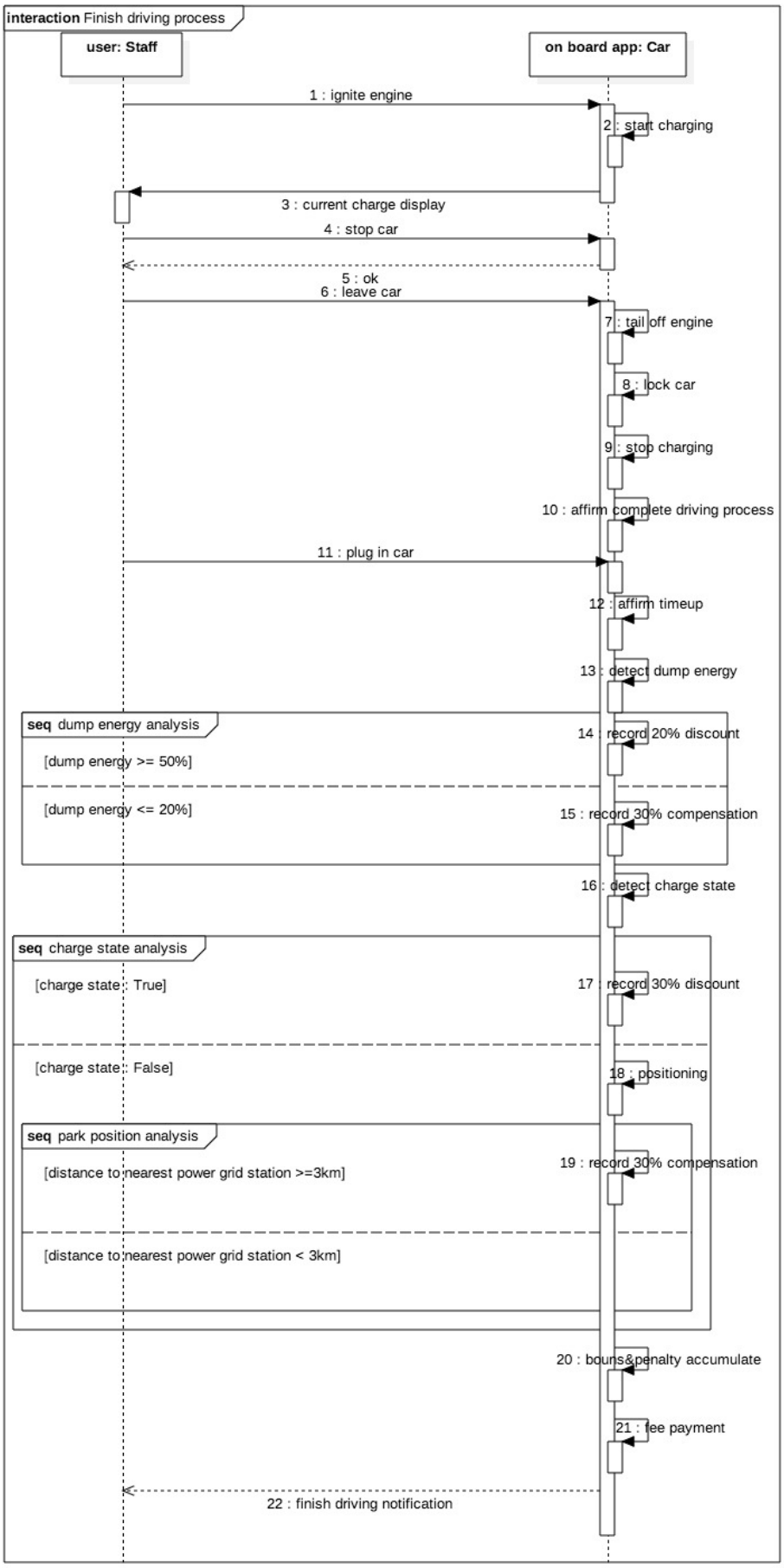
4.3.8 Access the "saving money option"

Name	**Access the "saving money option"**
Actors	Users
Assumptions	Users already picked up the car successfully
Flow of events	User clicks the "saving money" button in the mobile phone User inputs the destination User clicks the "go" button System computes the optimal station for the inputting destination System responses the optimal station and the corresponding information to user
Exit conditions	--
Exceptions	The destination user inputted is not exist in this city The destination spelled wrong



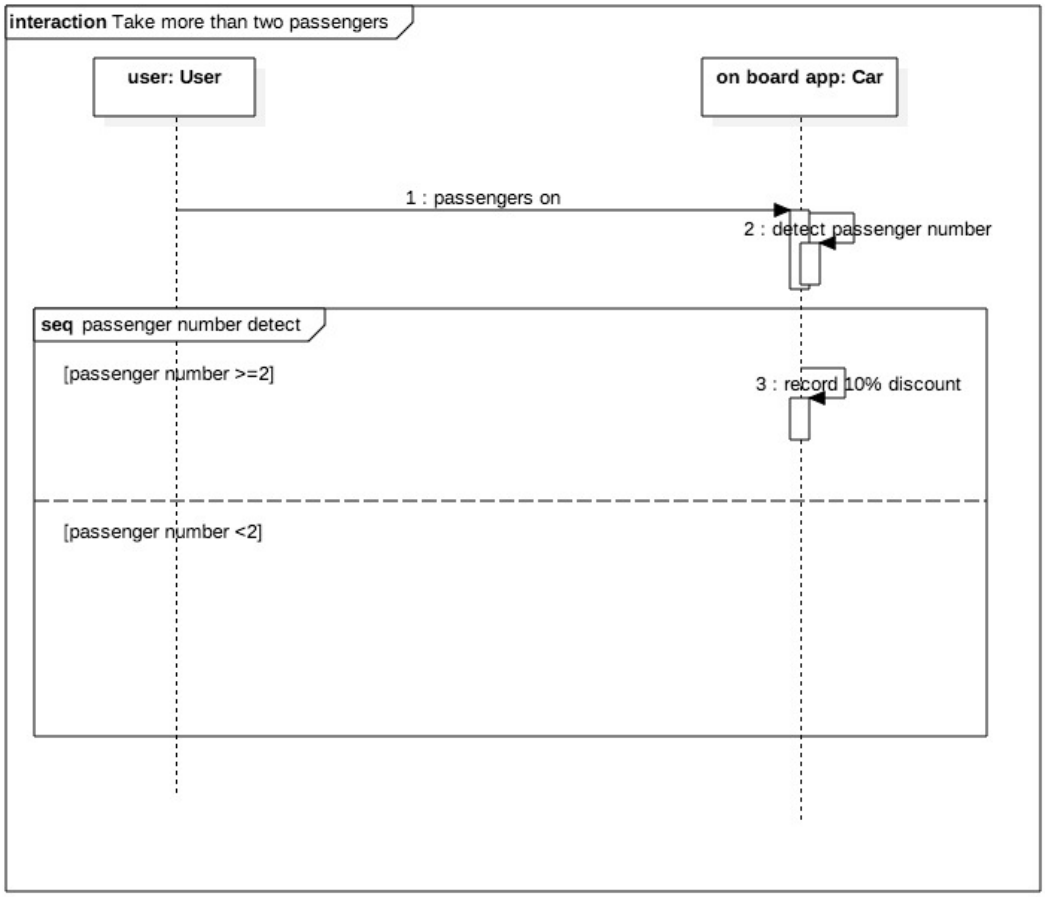
4.3.9 Finish driving process

Name	**Finish driving process**
Actors	Users
Assumptions	User picked up one car successfully
Flow of events	User ignites the engine System start charging the car System calculate the current charging System displays the current charging detail on the screen on the car User stop the car 5.1User tails-off the engine 5.2user doesn’t tail-off the engine User (passengers) leave the car System detects the car is empty and be parked System stops charging the car System locks the doors of car 9.1 system tails-off the engine if it is not turned-off by user System affirms it as a complete driving process
Exit conditions	System affirms the complete driving process
Exceptions	--


interaction Finish driving process
user: Staff
on board app: Car
1 : ignite engine
2 : start charging
3 : current charge display
4 : stop car
5 : ok
6 : leave car
7 : tail off engine
8 : lock car
9 : stop charging
10 : affirm complete driving process
11 : plug in car
12 : affirm timeup
13 : detect dump energy
seq dump energy analysis
[dump energy >= 50%]
14 : record 20% discount
[dump energy <= 20%]
15 : record 30% compensation
16 : detect charge state
seq charge state analysis
[charge state : True]
17 : record 30% discount
[charge state : False]
18 : positioning
seq park position analysis
[distance to nearest power grid station >=3km]
19 : record 30% compensation
[distance to nearest power grid station < 3km]
20 : bouns&penalty accumulate
21 : fee payment
22 : finish driving notification

4.3.10 Take more than 2 passengers

Name	**Take more than 2 passengers**
Actors	User
Assumptions	User picks up the car
Flow of events	User takes more than 2 passengers to the car System check the weight sensors User accomplishes a complete driving process System affirms the complete driving process System applies a 10% discount
Exit conditions	System applies the 10% discount
Exceptions	Passengers are to light (child) The weight sensors in the car don't work well



4.3.11 Leave more than 50% dump energy

Name	**Leave more than 50% dump energy**
Actors	Users
Assumptions	User picks up the car
Flow of events	User accomplishes a complete driving process System affirms the complete driving process System checks the battery sensor System applies a 20% discount
Exit conditions	System applies the 20% discount
Exceptions	The battery sensor doesn't work well

4.3.12Park the car far away to the power grid station

Name	**Park the car far away to the power grid station**
Actors	Users
Assumptions	User picks up the car
Flow of events	User accomplishes a complete driving process System affirms the complete driving process System locates the car System applies a 30% compensation
Exit conditions	System applies the 30% compensation
Exceptions	--

4.3.13 Leave less than 20% dump energy

Name	**Leave less than 20% dump energy**
Actors	Users
Assumptions	User picks up the car
Flow of events	User accomplishes a complete driving process System affirms the complete driving process System check the battery senor System applies a 30% compensation
Exit conditions	System applies the 30% compensation
Exceptions	The battery sensor doesn't work well

4.3.14 Recharge the car

Name	**Recharge the car**
Actors	Users
Assumptions	User picks up the car
Flow of events	User parks the car in power grid station User leave the car System affirms the complete driving process User puts the plug the car into power gird System checks the power grid sensor System applies a 30% discount
Exit conditions	System applies the 30% discount
Exceptions	Car doesn't be plugged in a right way The plug sensor doesn't work well

4.3.15 Modify the incorrect parking

Name	**Modify the incorrect parking**
Actors	Staffs
Assumptions	There are cars be parked in incorrect parking Staff is dispatched for specific car Staff uses crane to move the car
Flow of events	Staff gets the location of specific cars and the description Staff moves the specific car If the car is parked in unsafe area with more than 50% dump energy, move into the nearest safe area If the car is in low battery, move into the nearest power grid station and charge it If the car which parked in the power gird station is not be recharged, plug the car into the power grid Staff clicks the "mission accomplished" button System changes the car state, if the car is available now, move it into available queue System confirms the mission of staff
Exit conditions	System confirms the mission of staff
Exceptions	There are no empty slot in the nearest safe area or power gird station

5. External Interfaces

5.1 Hardware Interface

The mobile application must be authorized to connect into the internet.
The GPS is not mandatory if the user doesn't want to leak their position.

5.2 Software Interface

The mockups for the user interface are as follows:

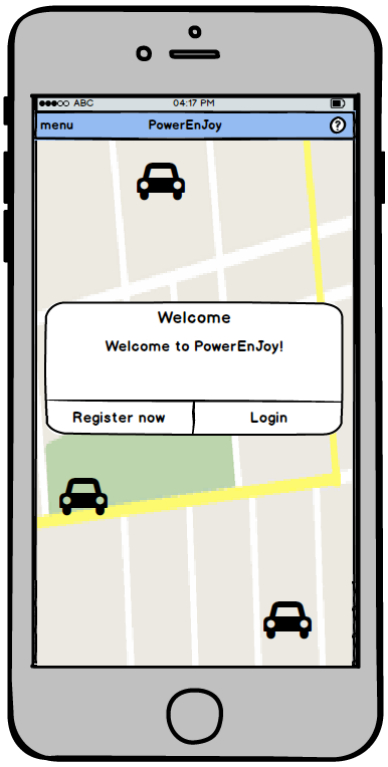


Figure 1: welcome page

Figure 2: register page

Figure 3 current position searching

Figure 4 specific position searching

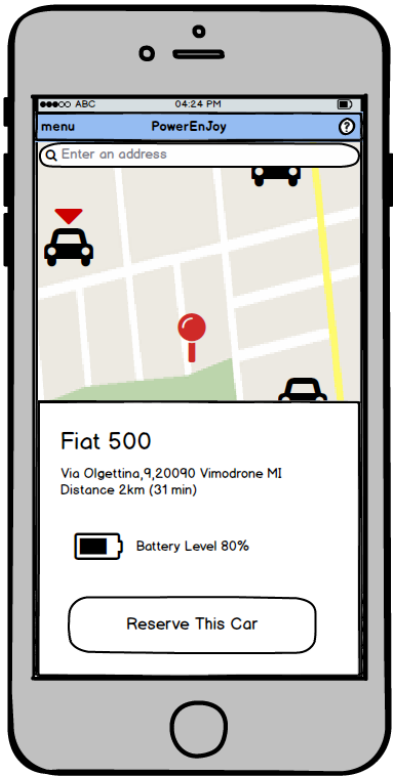


Figure 5 make reservation

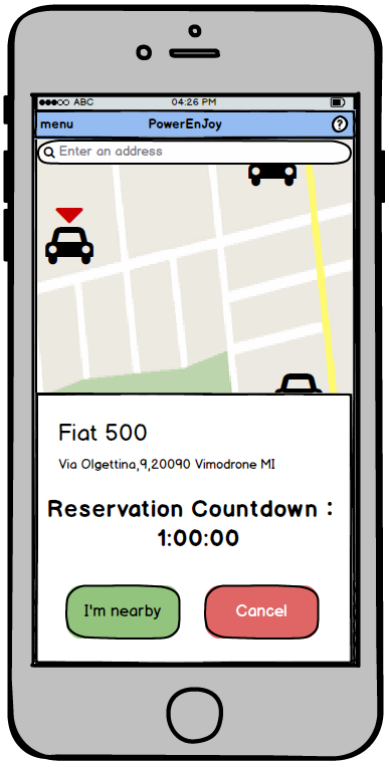


Figure 6 pick up the car

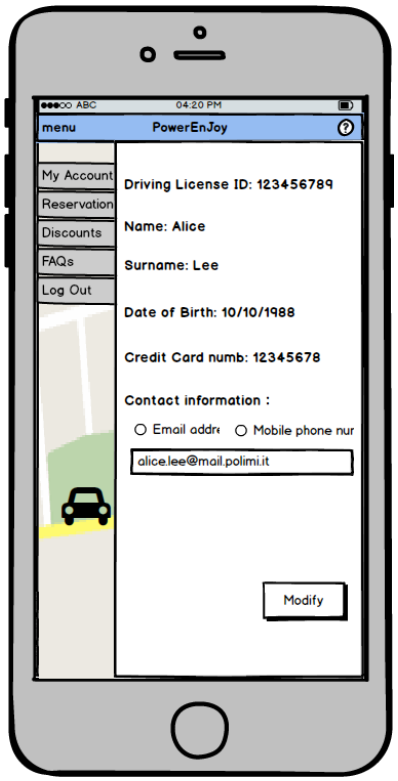


Figure 7 check the user information

6. Alloy Model

We will use the software Alloy Analyzer, in order to verify the consistency of our model and relative constraints.

6.1 Alloy Source Model

The complete model of the system in alloy code is reported as follows:

```
open util/boolean

sig Position {}

sig Plug {}

one abstract sig SafeArea{
	positions: set Position
}{	#positions >0	}

sig PowerGridStation extends SafeArea{
	plugs: set Plug
}{	#positions = #plugs	}

sig ParkingLot extends SafeArea {}

sig User {
	doReserve: lone Reservation,
	doPickUp: Bool lone -> lone Car,
	doCancel: Bool lone -> lone Reservation,
	hasRide: lone Ride,
	park: Car lone -> lone Position,
	plugIn: Car lone -> lone Plug,
	accountState: one Bool
}{
```

```
    #doPickUp =1
    Bool.doPickUp = this.(doReserve.reserve)
    (this.(doReserve.reserve).(~doPickUp) = True)
        implies doReserve.reservationStatus = COMPLETE
    Bool.doCancel = doReserve
    (doReserve.(~doCancel) = True)
        implies doReserve.reservationStatus = CANCELED
    Position.(~park) = True.doPickUp    & this.((False.(~rideStatus)).drive)
    (#plugIn =1) implies (Car.park in PowerGridStation.positions)
    Plug.(~plugIn) = (True.doPickUp & this.(False.(~rideStatus).drive))
}
sig Car {
    carStatus: one CarStatus,
    batteryLevel: one BatteryLevel,
    parkPosition: lone Position,
    chargeState: Bool,
    engineState: Bool
}{
    (carStatus = AVAILABLE)
        implies (batteryLevel = HIGH)
            and ((one parkPosition) and (parkPosition in SafeArea.positions))
            and (engineState = False)
            and ((this.(~(Reservation.reserve)) = none)
                or (no res: Reservation | this.(~(res.reserve)) != none and
res.reservationStatus = ACTIVE)
                or (one res: Reservation | this.(~(res.reserve)) != none and
res.reservationStatus = TIMEUP)
                or (one res: Reservation | this.(~(res.reserve)) != none and
res.reservationStatus = CANCELED)
                or (one res: Reservation | this.(~(res.reserve)) != none and
res.reservationStatus = COMPLETE and
                    (one rd: Ride| rd.rideStatus = False and
this.(~(Reservation.reserve)).(rd.drive)= this)))
    (carStatus = RESERVED)
        implies (batteryLevel = HIGH)
```

```
            and ((one parkPosition) and (parkPosition in SafeArea.positions))
            and (engineState = False)
            and (one res: Reservation |this.(~(res.reserve)) != none and
res.reservationStatus = ACTIVE)
            and (this.(~(User.doPickUp)) = False)
    (carStatus = INUSE)
        implies (batteryLevel != EMPTY)
            and (no parkPosition)
            and (this.~(User.doPickUp) = True) and (one rd: Ride| User.(rd.drive) =
this and rd.rideStatus = True)
            and (one res: Reservation| this.(~(res.reserve)) != none and
res.reservationStatus = COMPLETE)
    (carStatus = OUTOFSERVICE)
        <=> (batteryLevel = EMPTY) or ((one parkPosition) and not(parkPosition in
SafeArea.positions))
    (chargeState = True)
        implies (one parkPosition)
            and (parkPosition in PowerGridStation.positions)
    (engineState = True)
        implies (carStatus = INUSE)
    (carStatus = INUSE) <=> (parkPosition = none)
    (carStatus = INUSE)  implies this.(User.park) = none
    (this.(User.plugIn) != none) => (chargeState = True)
}

abstract sig CarStatus {}
one sig AVAILABLE extends CarStatus {}
one sig RESERVED extends CarStatus {}
one sig INUSE extends CarStatus {}
one sig OUTOFSERVICE extends CarStatus {}

abstract sig BatteryLevel {}
one sig LOW extends BatteryLevel {}
one sig HIGH extends BatteryLevel {}
one sig EMPTY extends BatteryLevel {}
```

```
sig Reservation {
	reserve: User lone -> lone Car,
	reservationStatus: one ReservationStatus,
	countingTimeUp: Bool,
	fee: Int
}{
	// no Reservation without User
	this.(~doReserve) != none
	#reserve = 1
	Car.(~reserve) = this.(~doReserve)
	fee >=0
	(reservationStatus = TIMEUP) <=>(countingTimeUp = True)
	(reservationStatus = TIMEUP) implies (fee = 1) and (User.reserve.carStatus = AVAILABLE)
	(reservationStatus = ACTIVE)
		implies (fee = 0)
			and (countingTimeUp = False)
			and (User.reserve.carStatus = RESERVED)
	(reservationStatus = CANCELED)
		implies (fee = 1) and (countingTimeUp = False)
			and ( True.(this.(~doReserve).doCancel) = this)
			and (User.reserve.carStatus = AVAILABLE)
	(reservationStatus = COMPLETE) implies (fee = 0)
			and (countingTimeUp = False)
			and ( True. (this.(~doReserve).doPickUp) =User.reserve)
}

abstract sig ReservationStatus {}
one sig TIMEUP extends ReservationStatus {}
one sig ACTIVE extends ReservationStatus {}
one sig CANCELED extends ReservationStatus {}
one sig COMPLETE extends ReservationStatus {}

sig Ride {
```

```
    drive: User lone -> lone Car,
    rideStatus: one Bool,
    passengerNum: one Int,
    disscount: set Discount,
    compensation: set Compensation,
    standardFee: one Int,
    paymentAmount: one Int
}{
    // no ride without user
    this.(~hasRide) != none
    #drive = 1
    Car.(~drive) = this.(~hasRide)
    (True.(this.(~hasRide).doPickUp) = User.drive)
    (rideStatus = True) implies (User.drive.carStatus = INUSE)
    (rideStatus = False) implies (User.drive.carStatus != INUSE)

}

abstract sig Discount {}
abstract sig Compensation{}

/*------------------FACTS-------------------------*/
// No isolated plugs.
fact NoIsolatedPlug {
    no p: Plug | p.(~plugs) = none
}
// No two users share one reservation.
fact NoSharedReservation {
    no disj u1, u2: User| u1.doReserve = u2.doReserve
}
// No two users share one ride.
fact NoSharedRide {
    no disj u1, u2: User| u1.hasRide = u2.hasRide
}
// No two cars share one position.
```

```
fact NoSharedPosition {
	no disj c1, c2: Car| c1.parkPosition = c2.parkPosition
}
// No two reservations share one car.
fact NoSharedCar {
	no disj res1, res2: Reservation | User.(res1.reserve) = User.(res2.reserve)
}
// No two users share plug
fact NoSharedPlug {
	no disj u1, u2: User | Car.(u1.plugIn) = Car.(u2.plugIn)
}


/*------------------ASSERTS----------------------*/

assert noEMPTYbatteryCarIsINUSE {
	no c: Car | c.batteryLevel = EMPTY and c.carStatus = INUSE
}
check noEMPTYbatteryCarIsINUSE for 3
assert noLOWbatteryCarIsAVAILABLE {
	no c: Car | c.batteryLevel = LOW and c.carStatus = AVAILABLE
}
check noLOWbatteryCarIsAVAILABLE for 3
assert noCANCELEDreservationReserveCarhasStateRESERVED {
	no res: Reservation| res.reservationStatus = CANCELED and
User.(res.reserve).carStatus = RESERVED
}
check noCANCELEDreservationReserveCarhasStateRESERVED for 3
assert noTIMEUPreservationReserveCarhasStateRESERVED {
	no res: Reservation| res.reservationStatus = TIMEUP and
User.(res.reserve).carStatus = RESERVED
}
pred example {}
run example
```

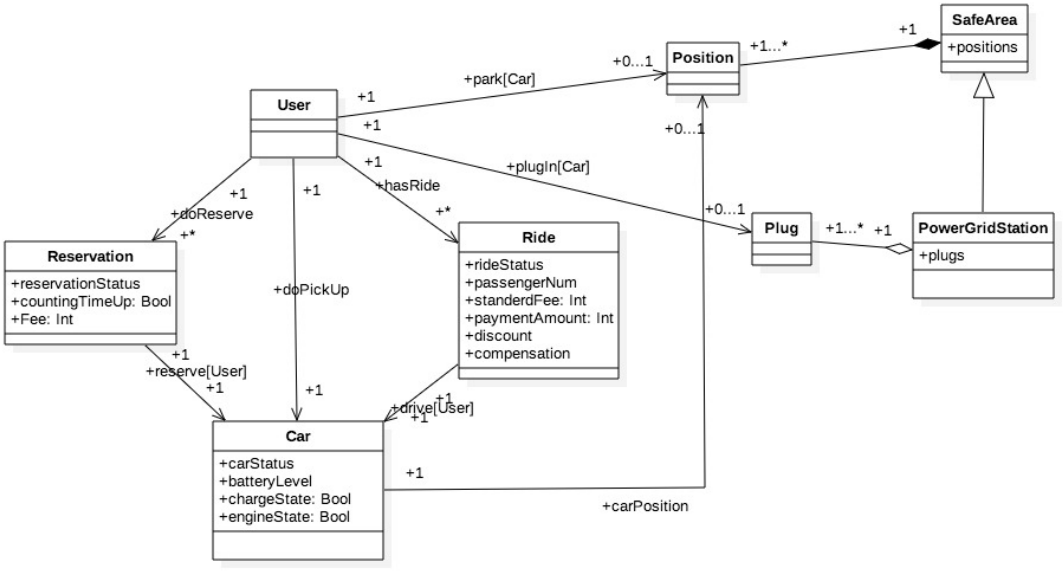
6.2 Analyzer Result

The report of Alloy Analyzer is listed as follows:

```
Executing "Check noEMPTYbatteryCarIsINUSE for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  6657 vars. 479 primary vars. 14066 clauses. 62ms.
  No counterexample found. Assertion may be valid. 5ms.

Executing "Check noLOWbatteryCarIsAVAILABLE for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  6657 vars. 479 primary vars. 14066 clauses. 54ms.
  No counterexample found. Assertion may be valid. 3ms.

Executing "Check noCANCELEDreservationReserveCarhasStateRESERVED for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  6677 vars. 479 primary vars. 14173 clauses. 50ms.
  No counterexample found. Assertion may be valid. 18ms.

Executing "Run example"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  6602 vars. 476 primary vars. 13977 clauses. 44ms.
  Instance found. Predicate is consistent. 36ms.

4 commands were executed. The results are:
  #1: No counterexample found. noEMPTYbatteryCarIsINUSE may be valid.
  #2: No counterexample found. noLOWbatteryCarIsAVAILABLE may be valid.
  #3: No counterexample found. noCANCELEDreservationReserveCarhasStateRESERVED may be valid.
  #4: Instance found. example is consistent.
```

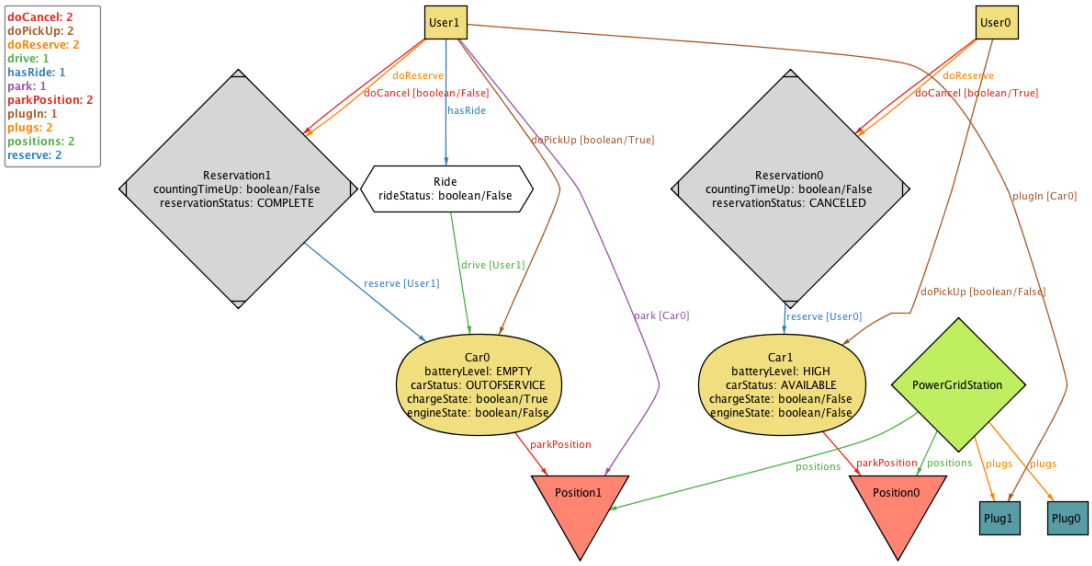
6.3 Model Description

In this subsection we will visualize some instances generated by Alloy Analyzer. The following class diagram normally represents the relevant features of our alloy model.



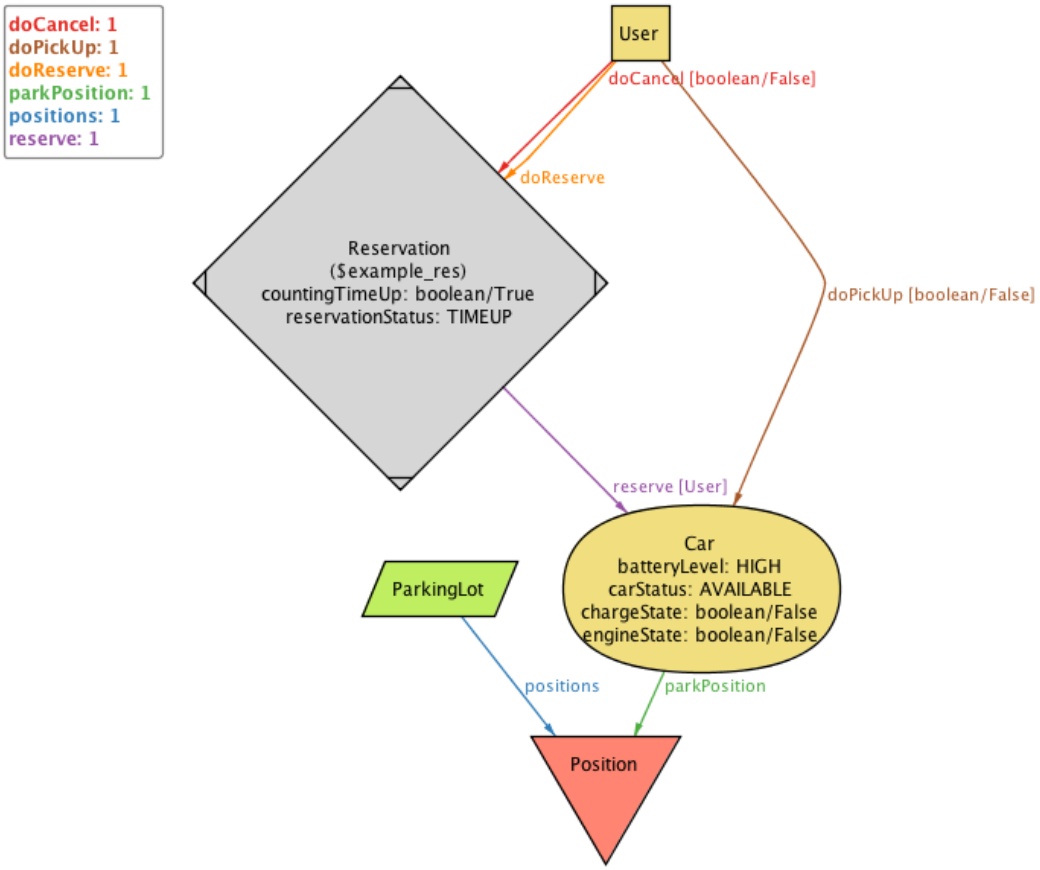
6.3.1 Instance I: Overall World's Description

The first instance is an overall world's description, which contains two users with two reservations in different status. The right one implies that User0 has reserved a car, Car1, which is parked in a power grid station. However User0 changed his idea after the reservation and cancelled the reservation, which, according to our model, makes the status of Car1 back to available again. Another user, User1, also has reserved a car, Car0. User1 picked up the car on time and made the reservation status become complete. After picking up, User1 has finished a ride with Car0. He used out the power of Car0, and finally parked it in the power grid station, as well as plugged the car into the power grid. As we can see from the model, the status of Car0 became out of service because of the empty battery level, and the charging state of Car0 stays True, thanks to the plug in action.



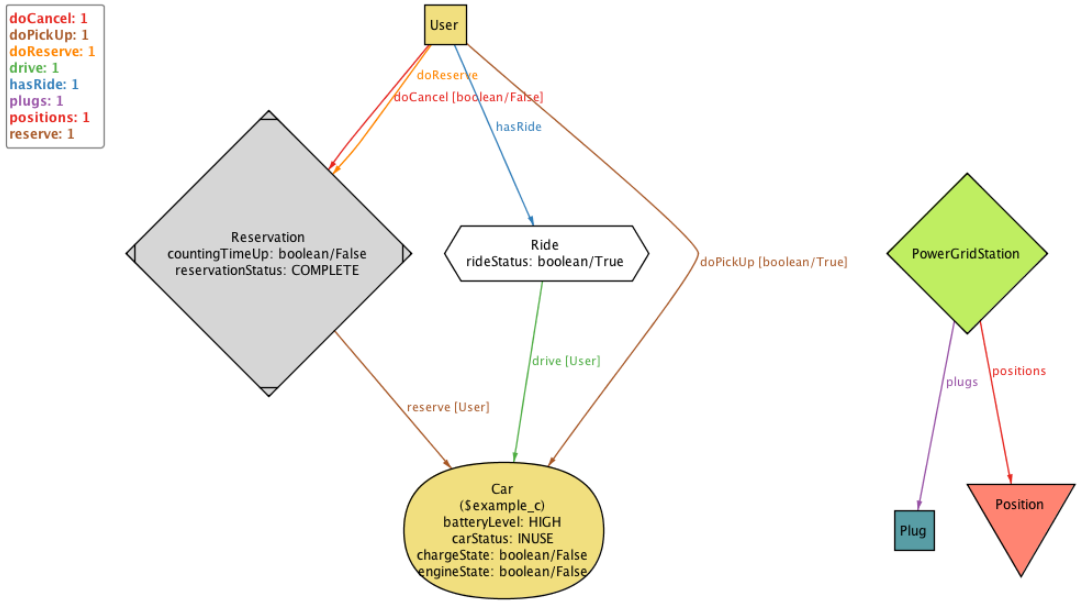
6.3.2 Instance II: One Time up Reservation

This particular instance describes a reservation with time up status. According to our model, a user can reserve a car for 1 hour, which means that the user is able to either pick up the car or cancel the reservation within an hour. If the user failed to pick up the car and didn't cancel the reservation in an hour, the reservation status should become time up and our system will release the reserved car and change it status to available again.



6.3.2 Instance III: One In Use Car

This instance contains a car, which is currently using by a user. According to our model, the user should have reserved the car and have completed the reservation by picking up the car within an hour. After that the car would be used for a ride, and the car status should stay in use as long as the user doesn't end this ride.



7. Hours of work

GAO Xiao	**40 Hours**
KANG Shuwen	**45 Hours**
Liubov Bolshakova	**30 Hours**