REQUIREMENTS ANALYSIS AND SPECIFICATION DOCUMENT

Figure 1: Politecnico di Milano

version 1.1

Artemiy Frolov, mat. 876373

autumn 2016

Contents

1 Introduction

1.1 Description of the system

The project that we are implementing is called Car-Sharing Service, with the help of which users can reserve the electrical car and use it. The service is based on web application with one target of people:

- users

Users must provide information about themselves, including their credentials and payment information in order to register an account in this system, and access it. Users can locate available electrical cars nearby or in the certain area. Also users can see current battery fulness of each car. After selecting the car, user can reserve it for up to one hour. When a user reaches the reserved car, system allows the user to unlock the car. As soon as the engine ignites, users can see current charges through the screen in the car.
User can leave the car for a short period of time without missing the car reservation. When the user tells the system that he doesn't need the car no more, it stops charging the user. At this point the car can't be controlled by the user no more and it be becomes available for users again.
System, in order to restrain the behaviour of users, and to encourage virtuous behaviours of users, will carry out some reward and punishment features.

1.1.1 Actual system

Sometime ago the car sharing companies used the system according to which the the car identifies users only by their membership cards. Such system forces users to have membership cards, that must be ordered and collected.
The system we suggest reduces the necessity of having such cards, but lets the users to use the car via web application, where only registration is needed.
During registration user provides all the necessary information, including payment credentials.

1.2 Goals

To specify the task goals were defined:

- [G1] Registered users(users) can access the system.
- [G2] Users can locate all unoccupied electric cars parked nearby or within a specific area.
- [G3] Users can see the information about battery fulness of each unoccupied electrical car.
- [G4] Users can reserve available electric car.
- [G5] Users can access the reserved car.
- [G6] Users can park the car for later usage without missing the "occupied" status.
- [G7] Users are notified about current driving charges.
- [G8] Users are encouraged to use the service properly.
- [G8.1] Users have a 10% discount when he picks at least 2 more passenger onto the car.
- [G8.2] Users have a 20% discount on the ride if he left the car with no more than 50% battery empty.
- ~~[G8.3] Users have a 30% discount on the ride if he left the car on the special parking area with the power grid station and plugged car to it.~~

1.3 Domain assumptions

We suppose that these properties hold in the analysed world:

- All electric cars have GPS navigators.
- GPS navigators state the right positions of users and cars
- Car has a set of sensors to define the current state for the car (cameras, engine state sensors, battery fulness sensors, ~~battery charge sensors~~

- Car has an inner operating system, that acquire and process information from sensors(cameras, engine state sensors, battery fulness sensors, ~~battery charge sensors~~), starts/kills engine, locks/unlocks doors, displays information on the screen.
- Users drive accurately, without car damage.
- ~~Special charging parking areas are defined in advance~~
- CarSharing Company has workers that charges cars that are left uncharged and far from charging station and park them in a convenient place

1.4 Application domain

CarSharing System can be applied in the area where

- Air pollution is serious.
- Public transportation infrastructures are imperfect.

The applicable group, people who:

- are environmentalists,
- use car occasionally,
- who don't have own cars.

1.5 Glossary

- Registered user: He/She is a client of car-sharing service who sends requests to the system. He/She is able to register and access the system, reserve and use reserved cars. While registering he/she should provide the following information:
  - Name and Surname
  - Payment credentials
  - Phone number
  - Username

  - Password
  - In order to search and reserve the car, user must provide his/her position or coordinates of specific area. Position can be acquire by the system automatically via GPS.

- User: the same as "Registered user"
- Passengers: User is allowed to pick up more people to join the trip. If the system detects the user took at least two other passengers onto the car(e.g. via camera face capture system), the system applies a discount of 10% on the last ride.
- Electric car: Car, that is provided by the CarSharing Company. Uses electrical power to ride.
- Available car: Electric Car, that is currently available, thus can be reserved by any registered user.
- Reserved car: Electric Car, that is reserved for using by the user. Reservation status lasts for 1 hour and afterwards car becomes available again and user is charged with 1 Euro, unless he occupies the car(see further)
- Occupied car: Car, that is currently occupied by some registered user. Car from the moment user starts the engine to the time he ends the trip.
- Unoccupied car: the same as "Available car"
- Trip: The period from the moment when user starts the engine to use the car until the end of the trip.
- Specific area: are that is specified by the user within which the system searches for unoccupied cars. This area is 1km$\hat{2}$.
- ~~Special parking areas: These are the areas where car can be parked and plugged into the power grid station for recharging.~~
- Position/Location: GPS coordinates
- System: Platform(Software) that is being implemented.

- Background System: the same as the ”System”, to distinguish the implemented System from the one specified in the ”Car’s inner OS” paragraph.
- Car’s inner OS: It is the car’s embedded operating system that:
  - responds to requests from background system
  - can interact with a screen that notifies the user about the current charges
  - acquires information from sensors of the car
  - provides information to background system
- ”Unlock”/”Lock” button: button in the web application that can unlock the door of car. Can be used only if the system states that the user is nearby the reserved car. It is also physically implemented inside the car.
- ”Start/kill Engine” button: This button can ignite or kill the engine. Implemented physically inside the car.
- ”End the Trip” button: Button in the web application that must be pressed to state that the user has finishes the trip. After that System stops charging the user.
- ~~Power grid station: place, where user is able to recharge the car.~~

1.6 Constrains

1.6.1 Regulatory policies

The System must:

- keep the payment information of users confidentially;
- ask users permission to use their GPS coordinates.

1.6.2 Hardware limitations

User's devices:

- 3G/4G connection
- GPS
- Space for app package

Car's inner OS:

- Internet connection with background system
- Screen (show current charges)
- GPS

1.6.3 Parallel operation

The server supports parallel operations from different users.

2 Proposed System

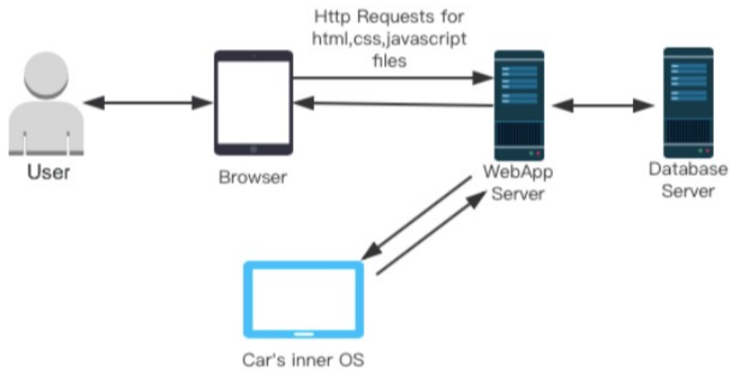


Figure 2: System architecture

The architecture in the figure 2 of Car-Sharing based on common API and MVC pattern.

2.1 Identifying stakeholders

The main purpose of this System is to provide environmental and social benefits to the communities in which it operates.
The main possible stakeholders are:

- environmentalists, that might support the idea of encouraging people to use electric cars with the simplified procedure of reservation and usage.
- goverment, that want to imrove car-sharing services, inrease the usability, efficiency and reduce the cost.

2.2 Reference documents

- Specification Document: Assignments 2016-2017.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- Examples documents: RASD sample from Oct. 20 lecture.pdf

3 Actors identifying

One actor is identified in the system:

- User: He/She is a client of car-sharing service who sends requests to the system. He/She is able to register by providing payment and other information to access the system, reserve and use electric cars. Besides, he/she should pay the rent to the car-sharing company. According to the behaviors of users, system provides users discounts or charge more for users.

4 Requirements

4.1 Functional requirements

In the "Domain assumptions" paragraph we defined expectations from the analyzed world. According to this information and stated goals we can derive functional requirements for the System.

- [G1] Registered users(users) can access the system:
  - System must provide a user with registration or sign in procedure.
- [G2] Users can locate all unoccupied electric cars parked nearby or within a specific area:
  - The system must acquire and provide to the user the information about location of all unoccupied cars within the certain area
  - The system must notify user to turn on the GPS on the smartphone while locating nearby cars.
  - The system must search available cars within the coordinates provided by user.
- [G3] Users can see the information about battery fulness of each unoccupied electrical car.
  - The system must acquire and provide to the user information about car's battery fulness
- [G4] Users can reserve available electric car.
  - System must allow the registered user to reserve 1 chosen unoccupied electric car for up to 1 hour.
  - When 1 hour after reservation is up, the system must state that the car is available and charge the user for 1 euro.
- [G5] Users can access the reserved car
  - System must provide the "lock/unlock" feature to the user when he/she reaches the reserved car

  - "Lock/unlock" feature can only be used if the user is close to the reserved car.
  - System must stop counting the reservation time and state that the car occupied after the user starts the engine for the first time

- [G6] Users can park the car for later usage without missing the "occupied" status.
  - System must provide the "end of the trip" button when the user decides to quit using the car.
  - System must charge user less, when the engine is turned off.
  - System must state that the car is available if the engine is suspended for more than 1 hour.
  - System must warn the user, when the time of the car left suspended is going to expire.

- [G7] Users are notified about current driving charges.
  - System must display charging information on the screen in the car.

- [G8] Users are encouraged to use the service properly:

- [G8.1] Users have a 10% discount when he picks at least 2 more passenger onto the car:
  - system must provide 10% discount for the time while it gets information that there are 2 more passengers in the car.

- [G8.2] Users have a 20% discount on the ride if he left the car with no more than 50% battery empty:
  - System must provide 20% discount on the whole ride iff user states that the car is no more needed + battery's fulness is not less that 50% full.

- ~~[G8.3] Users have a 30% discount on the ride if he left the car on the special parking area with the power grid station and plugged car to it:~~

– ~~System must provide 30% discount on the whole ride iff user states that the car is no more needed + the car is on the special parking area + car's battery is charging.~~

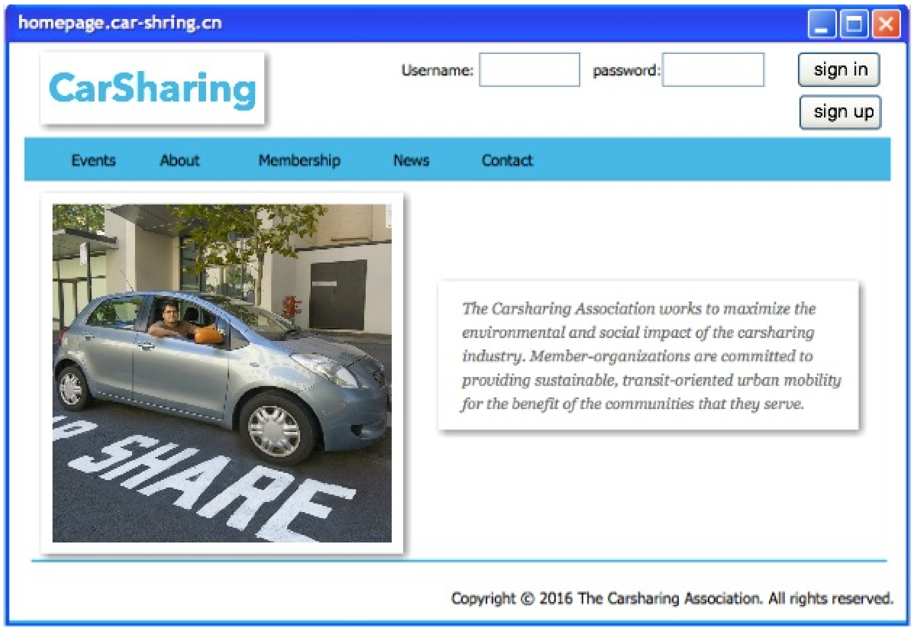
Global requirement:

- System must be able to communicate with car's inner OS to acquire processed information from car's sensors, send signals to lock/unlock door, to display information on the screen

4.2 Non-functional requirements

4.2.1 User interface

System to be implemented is web-based. Thus, the possible design of the webpages of the application in this paragraph are presented.



The Carsharing Association works to maximize the environmental and social impact of the carsharing industry. Member-organizations are committed to providing sustainable, transit-oriented urban mobility for the benefit of the communities that they serve.

Copyright © 2016 The Carsharing Association. All rights reserved.

Figure 3: Home page (plus sign in window)

In the figure 3 the possible interface of the main page and the sign in window of the System is shown. It contain useful information about the service and has 2 buttons to interract with user: sign in and sign up.

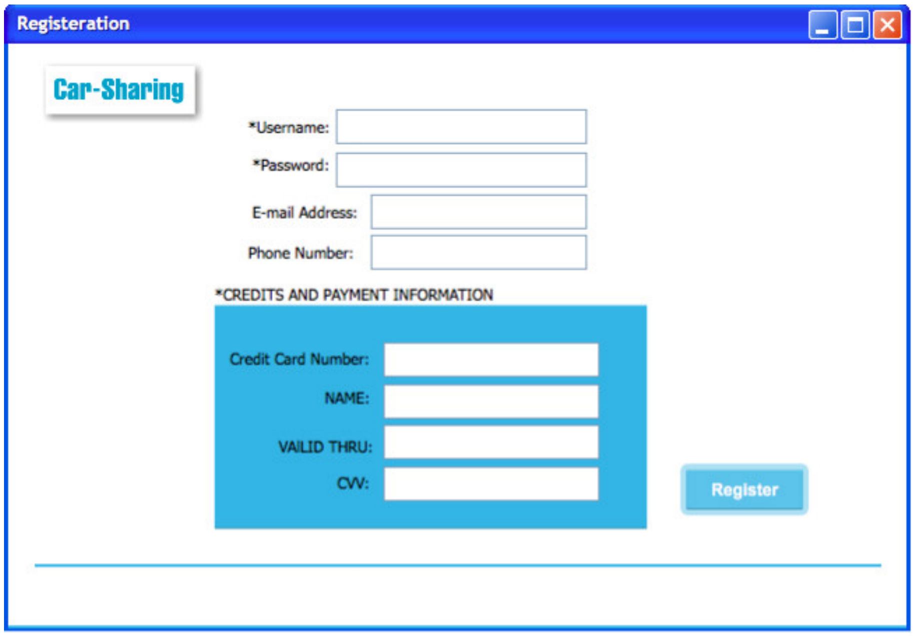
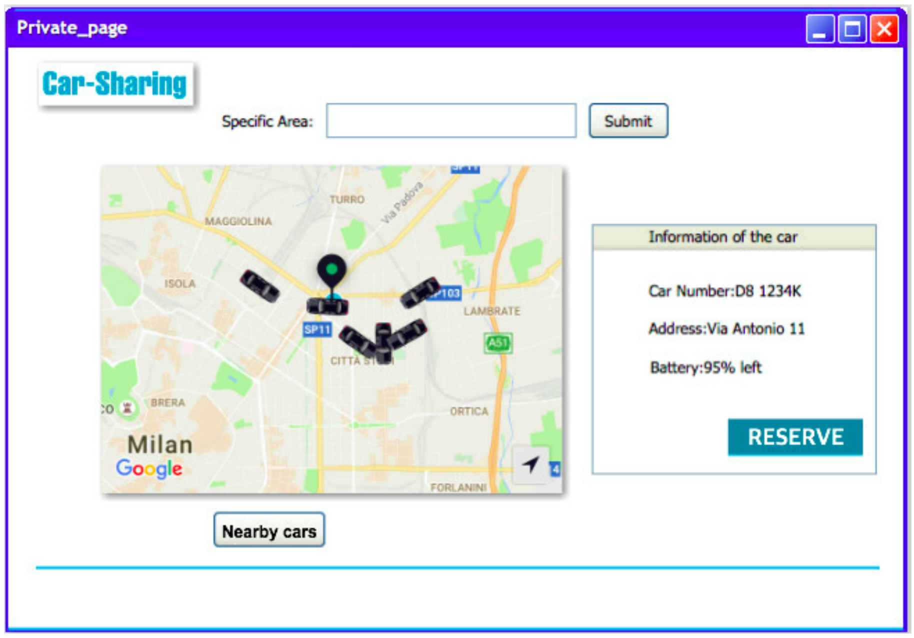


Figure 4: Registation form

In the figure 4 the possible interface of registration webpage is presented.



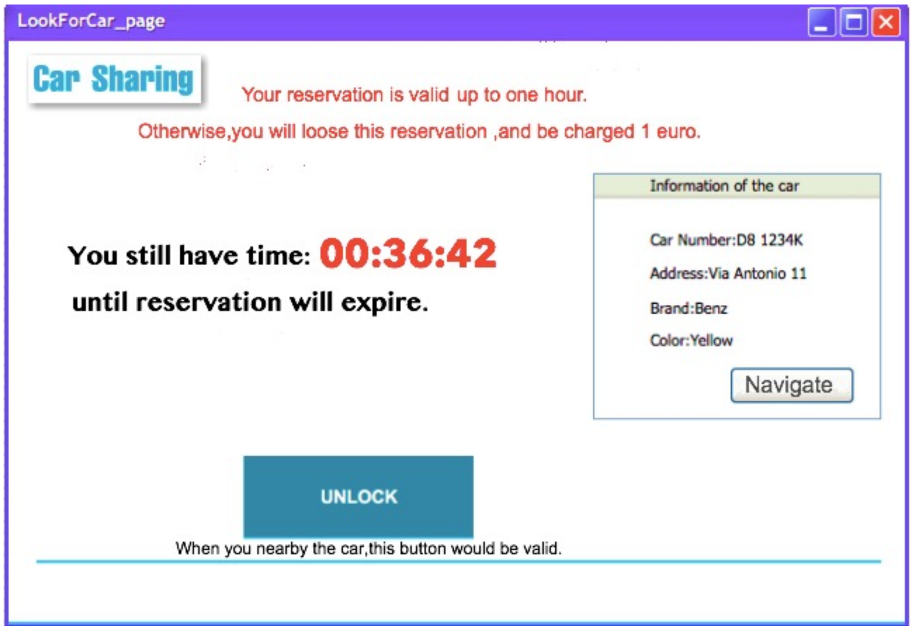
Car Number:D8 1234K

Address:Via Antonio 11

Battery:95% left

Figure 5: Choose car page

In the figure 5 the possible interface of the car choose is presented. To provide information about locations system can use external web-services.


LookForCar_page

Car Sharing

Your reservation is valid up to one hour.

Otherwise,you will loose this reservation ,and be charged 1 euro.

You still have time: 00:36:42

until reservation will expire.

Information of the car

Car Number:D8 1234K

Address:Via Antonio 11

Brand:Benz

Color:Yellow

When you nearby the car,this button would be valid.

Figure 6: Reservation page

In the figure 6 the possible interface of the reservation page is presented. The counter is used to notify user about the reservation expiring.

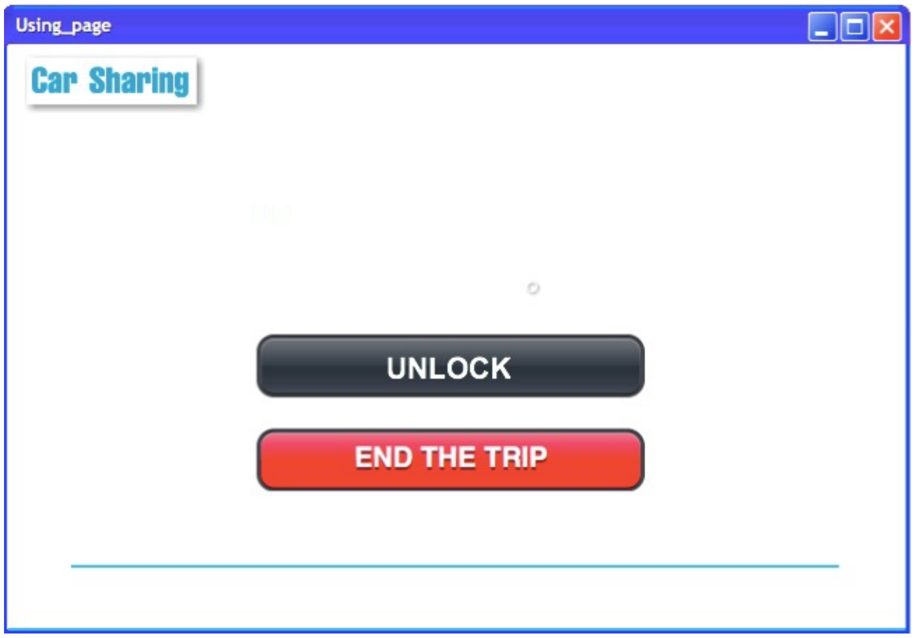


Figure 7: Occupation page

In the figure 7 the possible interface of the car usage page is presented. The button "Lock/Unlock" is activated when GPS coordinates of the user are close to the GPS coordinates of the car.

5 Scenarios identifying

Here some possible scenarios of usage of application.

Scenario 1

Melissa needs to get to the job, which is on the other side of Milano. But due to the protests on Friday, metro is closed. Melissa new about it in advance, so she decided to use the Car-Sharing system. She registered and reserved the nearest electric car in the morning and reached the job.

Scenario 2

Jack is the tourist and he decides to watch all the main places of Milano. He decides to use the Car-Sharing system. He finishes the registration via smartphone, reserves the nearest car and drives it to the different destinations. When he reaches the destination he leaves the car suspended and locked to watch the attractions. When he returns he unlocks the car and drive it to another places.

Scenario 3

Nicola lives far from the city and needs to get his son from school, but his car is broken. He already has an account in the Car-Sharing system and reserves the electric car on the specific area that is close to the train station. He reaches this station by train and drives to the son's school be the reserved car. When he picks his son he drives back home and leaves it nearby his house with the 85% battery empty. Nicola is charged 30% more.

Scenario 4

Kate decides to take a ride home by the electric car after meeting her friend in the cafe. She reserved the car nearby and continued to talk to her friend. Suddenly she realizes that the time of reservation is going to expire. She misses the reservation time and is charged by 1 Euro.

Scenario 5

Luci's car is broken. In the morning she always drive her children Peggy and Steve to the school and then she drives to her job. She reserved the electric car not far from her house with 100% battery fulness. When family gets to the car it captures that there are 3 people in the car and system starts to charge Luci 10% less. After she drove Peggy and Steve to the school, system stops the discount charging. After Luci drove to her job, the battery was more than 50% full. The total price for the ride is the sum of the discount time charging, when she was driving with her kids and the time charging without discount, when she was driving alone. Plus she gets 10% discount on the total price, because car is left with more than 50% full battery.

~~Scenario 6~~

~~Caroline has reserved the car electric and want to go to the bank. Luckily bank is situated near the special parking area. When she gets there, she parks on that area and plug the car to the power station. Caroline gets 30% discount on the ride.~~

6 UML models

6.1 Use case diagram

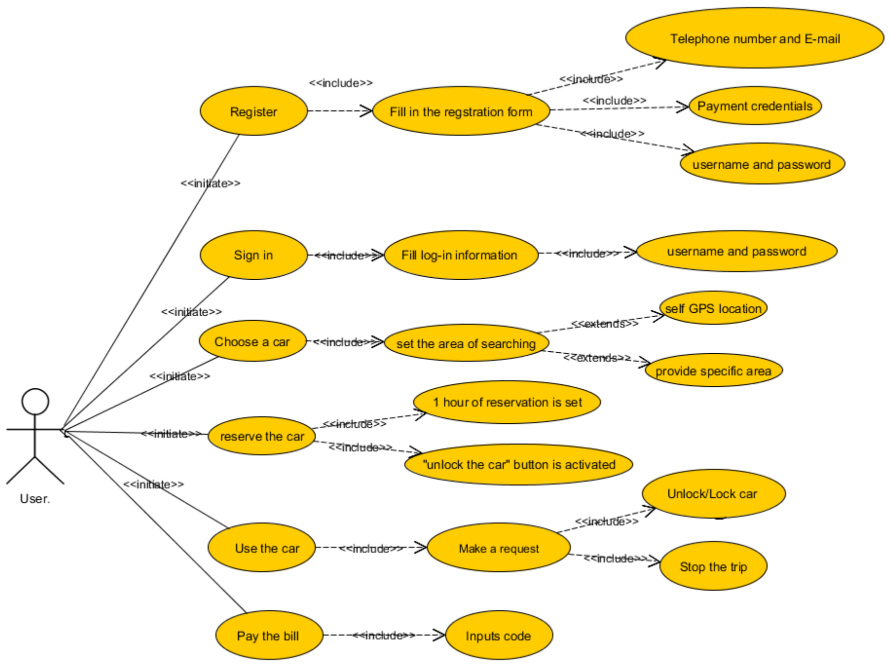


Figure 8: Use case diagram

6.1.1 Use case description

User registers
Name: User registers
Actor: User
Entry condition: there is no entry condition
Flow of events:

- User go to the System website.

- User clicks on the registration button.

- The system redirects the user to a form where he has to furnish the following information:

- Username
- Password
- Drive license
- E-mail address
- Telephone number
- Payment and credentials
- User press ”register” button.System sends the e-mail confirmation letter to the stated e-mail address.
- User goes to the e-mail and press on confirmation link in the confirmation letter.

Exit condition: The system notifies the user registered successfully, and redirects the user to login page. Exceptions:The user inputs an invalid payment information(eg.wrong cvv of credit card), system will notifies the user that he should to check his payment information and input again. Until user provides information all valid, system redirects user to login page.

User log-in
Name: User log-in
Actors: User
Entry condition: The user has already registered successfully.
Flow of events:

- The user inputs his username and password.
- The user click on log-in button.

Exit condition: The user is successfully access into home page,and he can see both his location and all available cars form the e-map of app.
Exceptions: The user inputs incorrect password,system prevent user from accessing into reservation page,and notifies user that the password or username is incorrect.Besides,system allows user to input username and password again.

User choose a car
Name: User choose a car
Actors: User
Entry condition: The user must already logged in successfully.
Flow of events:

- The user location himself postion or inputs an address of a specific area.
- The user search all unoccupied electric cars parked nearby or inputs a specific area.
- The user click the icon of one car.

Exit condition:

- The location of all unoccupied cars nearby or within a specific area can be shown on user's devices.
- The user is able to see the detail information of any available car when user click the icon of one available car.

Exceptions: The user didn't turn on his GPS function in his devices, system can not acquire the!18 location of user,which leads to system unsuccessfully provides information for user.System would notifies him to turn on it and try again.

User reserve a car
Name: User reserve a car
Actors: User
Entry condition: The use has already click the icon of one of the car he choosed.
Flow of events:

- The user click "Reserve" button.
- The user click "Yes" button,when system ask him whether he is sure to reserve this car.

Exit condition:

- System directs the user to the page where system timing the time of one hour for user's reservation.
- When user nearby the reserved car,the “unlock the car”button is activated

Exceptions:

- The user just clicks into the page about details of the car,but he didn't click the ”Reserve” button.So his reservation is invalid.
- After one hour the user still doesn't nearby the car,”unlock the car” button can not be activated. His reservation turns to invalid.Here is one euros also be charged by system.

User use the car Name: User use the car
Actors: User
Entry condition: The ”unlock the car” button has been activated.
Flow of events:

- The user click ”Unlock” button via app then get on the car.
- When user get off the car,click ”Lock car” button.
- The user click ”End the trip” to stop current trip,system stop charging.

Exit condition:

- System unlock the car when user click ”Unlock car” button,and the screen of the car be turn on where user can check current charge.
- System lock the car when user click ”Lock car” button.
- System automatically lock the car and stop charging when user click ”End the trip” button.
- User can see details about the bill for this trip from his devices.

Exceptions: There are no exceptions for this use case.

User pay the bill
Name: User pay the bill
Actors: User
Entry condition: System has already provides bill information to user.
Flow of events:

- The user choose a way to pay the bill.
- The user inputs his code of the payment he choose.

Exit condition:The system redirects the user to a success-paid page,where the system notifies user has already successfully paid this trip.
Exceptions: The user input an incorrect code, in that case, system redirects the user to a unsuccessful-paid page, and notifies user should to input code again. Until user input correct code,system redirects the user to a success-paid page.

6.2 Class diagram

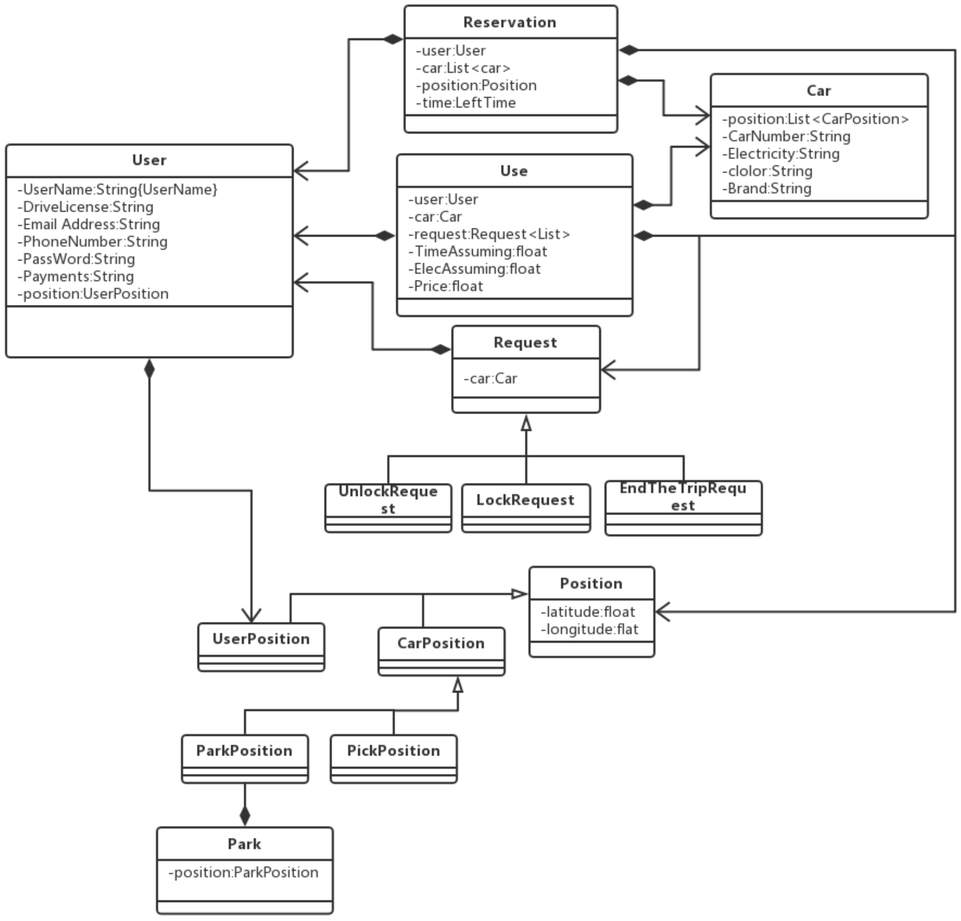


Figure 9: Class diagram

6.3 Sequence diagrams

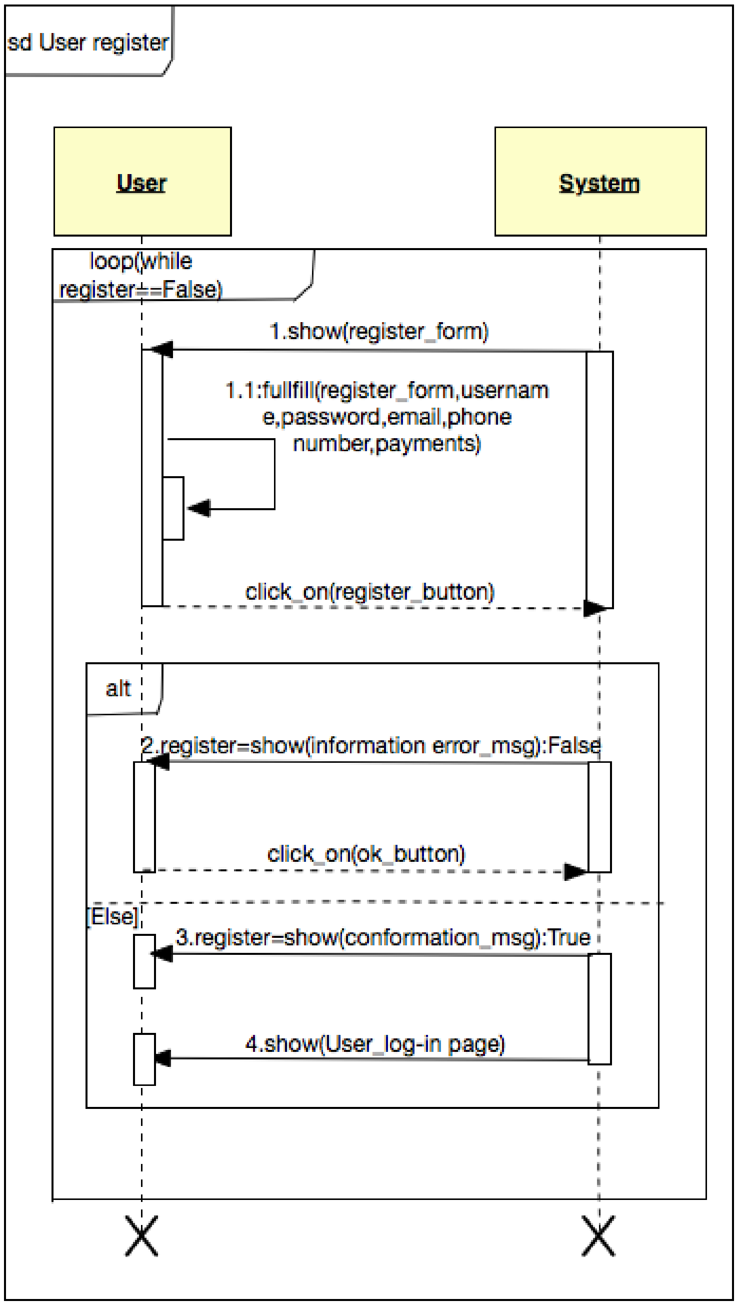

sd User register
User
System
loop(while register==False)
1.show(register_form)
1.1:fullfill(register_form,username,password,email,phone number,payments)
click_on(register_button)
alt
2.register=show(information error_msg):False
click_on(ok_button)
[Else]
3.register=show(conformation_msg):True
4.show(User_log-in page)

Figure 10: Sequence diagram - user registers

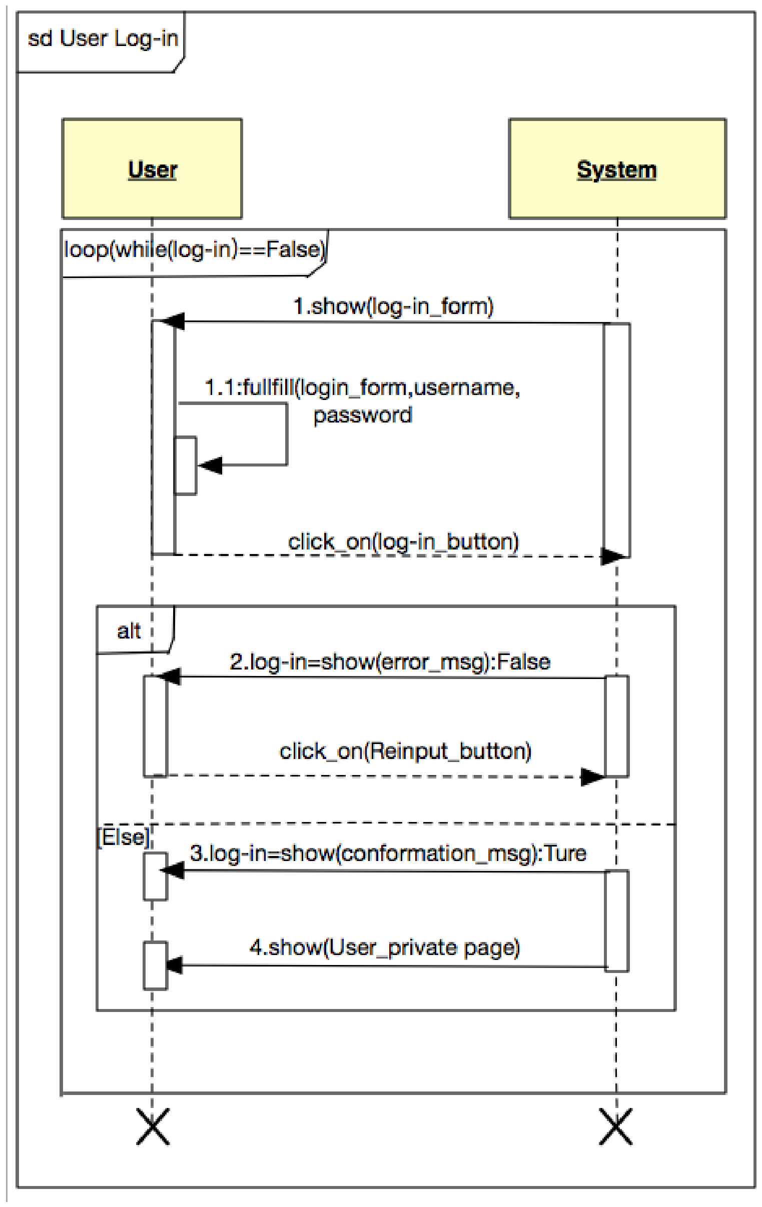

sd User Log-in
User
System
loop(while(log-in)==False)
1.show(log-in_form)
1.1:fullfill(login_form,username, password
click_on(log-in_button)
alt
2.log-in=show(error_msg):False
click_on(Reinput_button)
[Else]
3.log-in=show(conformation_msg):Ture
4.show(User_private page)

Figure 11: Sequence diagram - user sign in

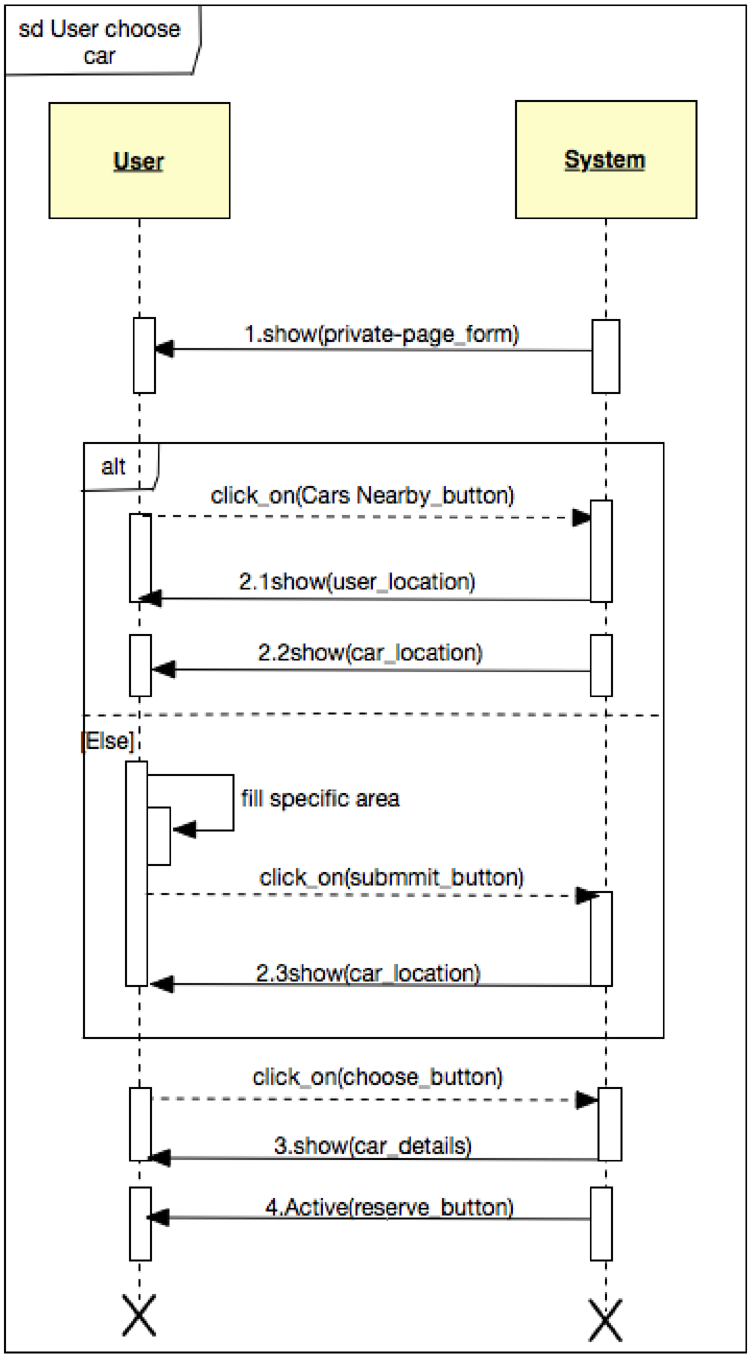


Figure 12: Sequence diagram - user choose available car

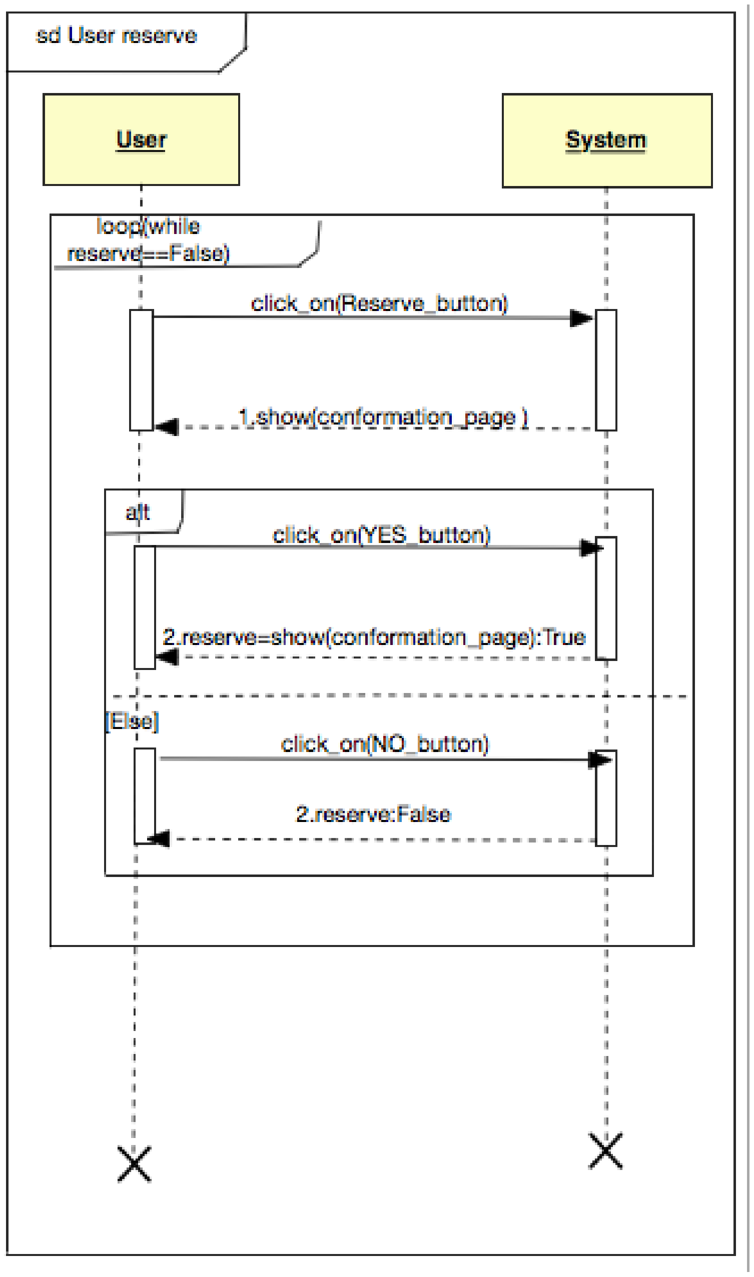


Figure 13: Sequence diagram - user reserves the car

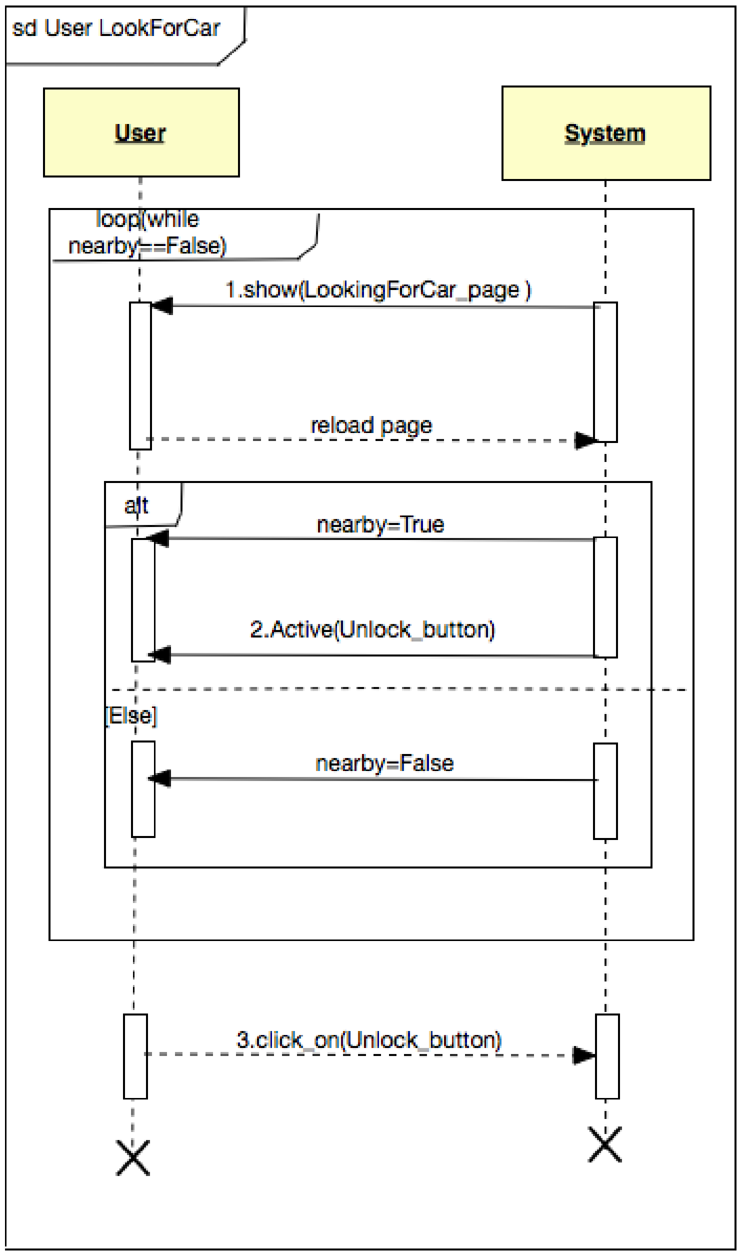


Figure 14: Sequence diagram - user approaches the reserved car

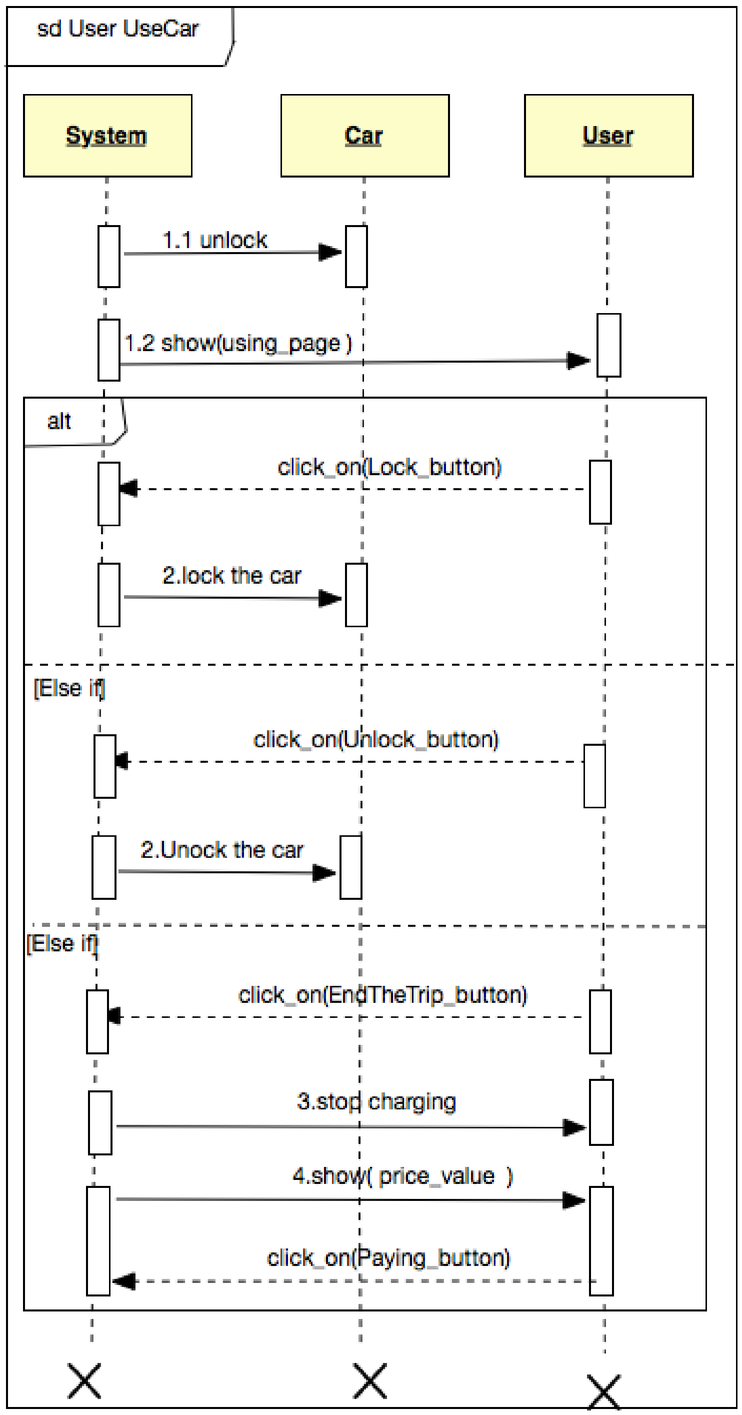


Figure 15: Sequence diagram - user uses the car

6.4 Activity diagrams

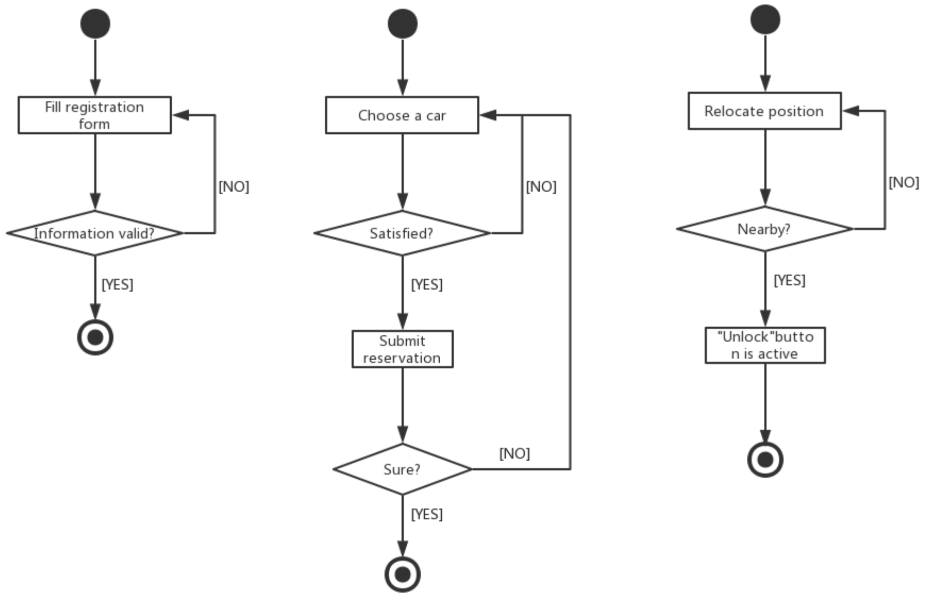


Figure 16: Activity diagrams

6.5 State diagrams

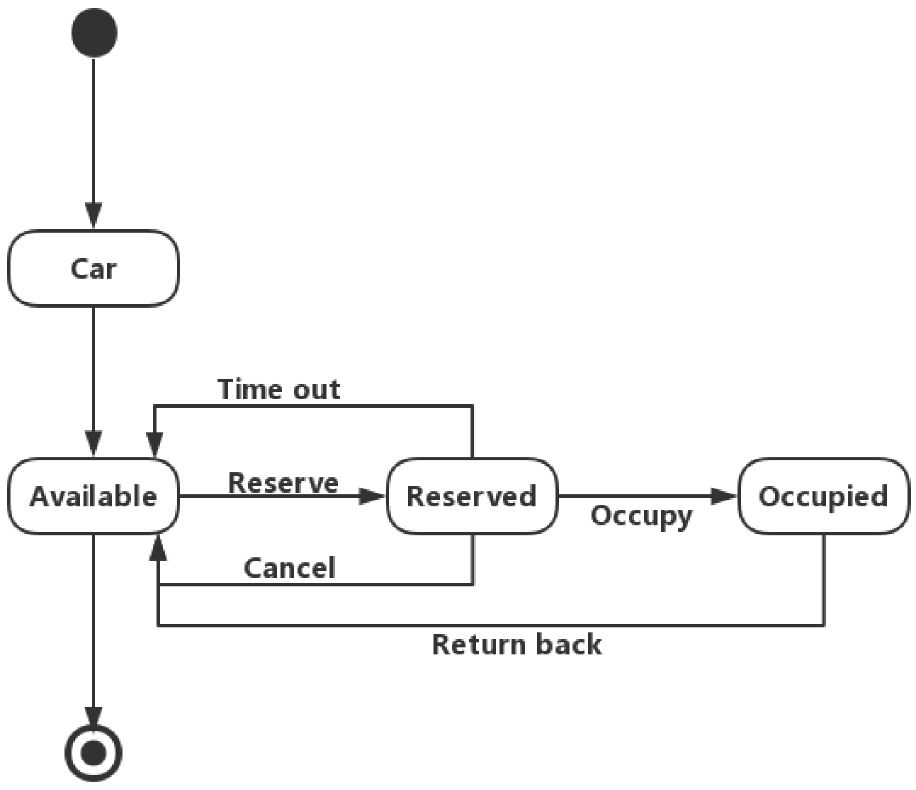


Figure 17: Activity diagrams

7 Alloy modeling

```
// ALLOY SPECIFICATION
// FOR CARSHARING SYSTEM RASD
// AUTHOR: ARTEMIY FROLOV

// PEOPLE
// ==============================

// Any person
abstract sig Person{}
// Registered User
sig RegUser extends Person {
payment: lone Payment
}
// Unregistered User
sig UnReg extends Person {}

// PAYMENT
// ==============================
// Discounts
abstract sig Discount {}
lone sig Discount10Percent extends Discount {}
lone sig Discount20Percent extends Discount {}
lone sig Discount30Percent extends Discount {}
abstract sig Charge {}
lone sig Euro1 extends Charge{}

sig Payment {
discounts: set Discount,
charges: set Charge
} {
#discounts >= 0
#charges >= 0
}

// PARKING
// ==============================
abstract sig Parked {}
sig SpecialParkingArea extends Parked{}
sig AnyParkingArea extends Parked {}
lone sig Driven extends Parked{}
// TIME
// =============================
abstract sig Time {}
lone sig Less1h extends Time{}
lone sig More1h extends Time{}
// CAR
// =============================
// Car occupation states
abstract sig CarOccupationState {}
lone sig Available extends CarOccupationState {}
lone sig Reserved extends CarOccupationState {}
lone sig Occupied extends CarOccupationState {}
lone sig Released extends CarOccupationState {}
lone sig LostReservation extends CarOccupationState {}
lone sig LostOccupation extends CarOccupationState {}
// Car charging states
abstract sig CarChargingState {}
lone sig Charging extends CarChargingState {}
lone sig NotCharging extends CarChargingState {}
// Battery fulness
abstract sig BatteryFulness {}
lone sig More50 extends BatteryFulness {}
lone sig Less20 extends BatteryFulness {}
lone sig More20Less50 extends BatteryFulness {}
// Engine State
abstract sig CarEngineState {}
lone sig EngineOn extends CarEngineState {}
lone sig EngineOff extends CarEngineState {}
// Car
sig Car {
occupationstate: one CarOccupationState,
chargingstate: one CarChargingState,
takenby: lone RegUser,
wastakenby: lone RegUser,
passengers: set Person,
hadpassengers: set Person,
battery: one BatteryFulness,
parked: one Parked,
enginestate: one CarEngineState,
reservationtime: lone Time
}{
#passengers >= 0
#passengers <= 4
#hadpassengers >= 0
#hadpassengers <= 4
}

// CONVENTIONS
// ==============================

fact TakenWasTakenConventions {
all c: Car | ((c.takenby != none) and
(c.occupationstate != Available)) implies (c.wastakenby = none)
all c: Car | ((c.wastakenby != none) and
(c.occupationstate != Available)) implies (c.takenby = none)
}

fact OwnerIsAPassenger {
all c: Car | (c.takenby != none) implies (c.takenby in c.passengers)
all c: Car | (c.wastakenby != none) implies (c.wastakenby in c.hadpassengers)
}

fact NoUserCanUseTheSameCar {
all c1, c2: Car | ((c1 != c2) and ((c1.takenby != none) and
(c2.takenby != none))) implies (c1.takenby != c2.takenby)
all c1, c2: Car | ((c1 != c2) and ((c1.wastakenby != none) and
(c2.wastakenby != none))) implies (c1.wastakenby != c2.wastakenby)
all c1, c2: Car | ((c1 != c2) and ((c1.takenby != none) and
(c2.wastakenby != none))) implies (c1.takenby != c2.wastakenby)
}

fact NoTheSamePassengers {
```

```
all c1, c2: Car | ((c1 != c2) and (c1.passengers != none) and
(c2.passengers != none)) implies (c1.passengers & c2.passengers) = none
all c1, c2: Car | ((c1 != c2) and
(c1.hadpassengers != none) and
(c2.hadpassengers != none)) implies (c1.hadpassengers & c2.hadpassengers) = none
all c1, c2: Car | ((c1 != c2) and
(c1.takenby != none) and
(c2.passengers != none)) implies (c1.takenby not in c2.passengers)
all c1, c2: Car | ((c1 != c2) and
(c1.wastakenby != none) and
(c2.hadpassengers != none)) implies (c1.wastakenby not in c2.hadpassengers)
}

fact TakenByDoesntHavePayment {
all c: Car | (c.takenby != none) implies (no c.takenby.payment)
}

fact NoTheSamePayment {
all c1, c2: Car | ((c1 != c2) and
(c1.wastakenby != none) and
(c2.wastakenby != none)) implies (c1.wastakenby.payment != c2.wastakenby.payment)
}

fact UsersWithoutTheCarCannotHavePayment {
all c: Car, r: RegUser | ((c.takenby != none) and
(r not in c.takenby)) implies (no r.payment)
all c: Car, r: RegUser | ((c.wastakenby != none) and
(r not in c.wastakenby)) implies (no r.payment)
}

fact NoPaymentWithoutOwner {
all c: Car, p: Payment | ((c.takenby != none) and
(p not in c.takenby.payment)) implies (no p)
all c: Car, p: Payment | ((c.wastakenby != none) and
(p not in c.wastakenby.payment)) implies (no p)
}

fact CarChargingConventions {
```

```
all c: Car | (c.chargingstate = Charging) implies (c.parked != Driven)
}

fact CarEngineOffConventions {
all c: Car | c.enginestate = EngineOff implies {
c.parked != Driven
}}

// STATES CONVENTIONS
// ==============================
fact CarIsAvailableConventions {
all c: Car | (c.occupationstate = Available) implies
(

no c.takenby and
no c.wastakenby and
no c.passengers and
no c.hadpassengers and
c.enginestate = EngineOff and
no c.reservationtime
)}

fact CarIsReservedConventions {
all c: Car | (c.occupationstate = Reserved) implies (
c.takenby != none and
(#c.passengers = 1) and
no c.wastakenby and
no c.hadpassengers and
c.enginestate = EngineOff and
c.reservationtime = Less1h
)}

fact CarIsLostReservationConventions {
all c: Car | (c.occupationstate = LostReservation) implies (
(no c.takenby) and
(no c.passengers) and
(c.wastakenby != none) and
(#c.hadpassengers = 1) and
(c.enginestate = EngineOff) and
(c.reservationtime = More1h) and
(c.wastakenby.payment != none) and
(Euro1 in c.wastakenby.payment.charges)
)}

fact CarIsOccupiedConventions {
all c: Car | (c.occupationstate = Occupied) implies (
(c.takenby != none) and
(no c.wastakenby) and
(no c.hadpassengers) and
(no c.reservationtime)
)}

fact CarIsLostOccupationConventions {
all c: Car | (c.occupationstate = LostOccupation) implies (
(no c.takenby) and
(no c.passengers) and
(c.wastakenby != none) and
(c.hadpassengers != none) and
(c.enginestate = EngineOff) and
(c.reservationtime = More1h) and
(c.wastakenby.payment != none)
)}

fact CarIsReleasedConventions {
all c: Car |  (c.occupationstate = Released) implies (
(no c.takenby) and
(no c.passengers) and
(c.wastakenby != none) and
(c.hadpassengers != none) and
(c.enginestate = EngineOff) and
(c.wastakenby.payment != none) and
(no c.reservationtime)
)}

// ENCOURAGEMENT
// ==============================
fact More2PassengersDiscount {
all c: Car, oc: c.occupationstate, p: c.wastakenby.payment |
(((oc = Released) or (oc = LostOccupation)) && (#c.hadpassengers > 2))
iff (Discount10Percent in p.discounts)
}

fact More50PercentEnergyDiscount{
all c: Car, oc: c.occupationstate, p: c.wastakenby.payment |
((( oc = Released) or (oc = LostOccupation)) && (c.battery = More50))
iff (Discount20Percent in p.discounts)
}

fact ChargingOfTheCarDiscount {
all c: Car, oc: c.occupationstate, p: c.wastakenby.payment |
((( oc = Released) or (oc = LostOccupation)) &&
(c.chargingstate = Charging) &&
(c.parked = SpecialParkingArea)) iff
(Discount30Percent in p.discounts)
// We suppose that charging of the car can only
// be done on the special parking area
}
// ==============================

// TESTING
// ==============================
// Car is Available
pred available () {
some c: Car | (c.occupationstate = Available) and
(c.parked = AnyParkingArea) and
(c.enginestate = EngineOff)
}
// Car is Reserved
pred reserved () {
some c: Car | (c.occupationstate = Reserved) and
(c.parked = AnyParkingArea) and
(c.enginestate = EngineOff)
}
// User has Lost Reservation
```

```
pred lostreservation () {
some c: Car | (c.occupationstate = LostReservation) and
(c.parked = AnyParkingArea) and
(c.enginestate = EngineOff)
}
// Car is Occupied
pred occupied () {
some c: Car | (c.occupationstate = Occupied) and
(c.parked = Driven) and
(c.enginestate = EngineOn)
}
// User has Lost Occupation
pred lostoccupation () {
some c: Car | (c.occupationstate = LostOccupation) and
(c.parked = SpecialParkingArea) and
(c.enginestate = EngineOff)
}
//Car is Released
pred released() {
some c1: Car | (c1.occupationstate = Released) and
(c1.parked = SpecialParkingArea) and
(c1.chargingstate = Charging)
}
// Any
pred any() {
#Car = 1 and
some c: Car | (c.occupationstate = LostReservation)
}
// Empty
pred show {}

//run available for 6
//run reserved for 6
//run lostreservation for 6
//run occupied for 6
//run lostoccupation for 6
run released
//run any for 8
```

```
//run show
```

7.0.1 Modelling results

Result #1

```
//Car is Released
pred released() {
some c1: Car | (c1.occupationstate = Released) and
(c1.parked = SpecialParkingArea) and
(c1.chargingstate = Charging)
}
run released
```

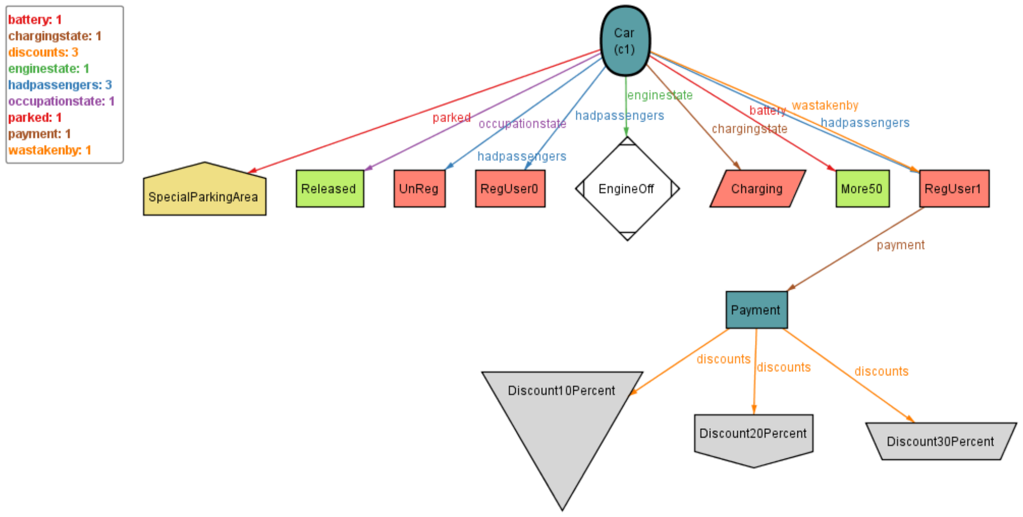


Figure 18: Alloy modeling result #1

Result #2

```
// Any
pred any() {
#Car = 3 and
some c: Car | (c.occupationstate = LostReservation)
}
run any for 8
```

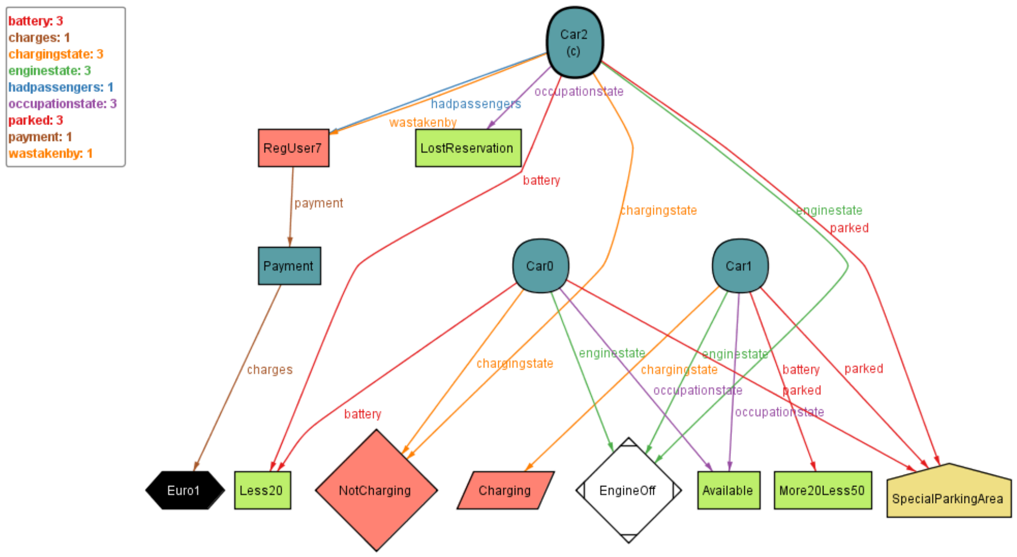


Figure 19: Alloy modeling result #2

8 Future developments

We plan to implement mobile version for IOS and Android smartphones in the future work.

9 Used tools

The tools we used to create this RASD document are:

- ProcessOn: for uml models
- Github: for version controller
- Pencil: for mockup
- OmniGaffle: for sequence diagram
- Gmail: for chenk other's modified documents
- Alloy Analizer 4.2: to prove the consistency of our model.
- yEd Graph Editor: to implement graphs and diagrams

10 Hours of work

Artemiy Frolov:

```
26/10, 1h
28/10, 2.5h
29/10, 3h
31/10, 2h
2/11, 1h
4/11, 4h
5/11, 2h
6/11, 3h
7/11, 2h
9/11, 1h
10/11, 3h
11/11, 3h
12/11, 4h
13/11, 10h
```

Lu Jia:

```
26/10, 2h
27/10, 2h
28/10, 1h
29/10, 1h
30/10, 2h
02/11, 1h
03/11, 2h
04/11,1h
05/11,2h
06/11,3h
07/11,2h
09/11,2.5h
10/11,3h
11/11,8h
12/11,7h
13/11,8h
```

11 Changelog

- v1.1:
  - [G8.3] removed (due to the group split)
  - any information somehow considering goal [G8.3] is deleted, except Alloy modelling
  - some rephrasings in the text are made
  - images of the interface pages are corrected