POLITECNICO
MILANO 1863

Computer Science and Engineering

A.A. 2016/2017

Software Engineering 2 Project:

"PowerEnJoy"

Assignment 1 Requirement and

Specification Document

13/11/2016

Version 1.0

Prof.Luca Mottola

ZHOU YINAN 872686

ZHAO KAIXIN 875464

ZHAN YUAN 806508

Content

1 Introduction

1.1 The description of the system

We are required to deveplop a digital management system for a car-sharing service. More precisely, this system can be accessed by the clients from a mobile deivce and processed by a server. The system

provides electric cars renting service. The registered users can search available cars, make reservations, acquire current charging fees. Moreover, the system has certain mechanics to incentivize the virtuous behaviours of the users, which includes creating discount of the charging fees or increasing the charging fees.

1.2 Goals

- [G1]Users can register to the system and get the password by providing credentials and payment information
- [G2]Registerd users can log in by providing the correct credentials
- [G3]Registered users can find the locations of available cars within a certain distance from their current location or from a specified address.
- [G4]Among the available cars in a certain geographical region, users must be able to reserve a single car for up to one hour before they pick it up.
- [G5]If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; the user pays a fee of 1 EUR.
- [G6]A user that reaches a reserved car must be able to tell the system he’s nearby, so the system unlocks the car and the user may enter.
- [G7]As soon as the engine ignites, the system starts charging the user for a given amount of money per minute
- [G8]The user is notified of the current charges through a screen on the car
- [G9]The system stops charging the user as soon as the car is parked in a safe area and the user exits the car; at this point, the system locks the car automatically.
- [G10]The users can know the locations of the safe areas
- [G11]If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.
- [G12]If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.
- [G13]If a car is left at stations where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.
- [G14]If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.
- [G15]If the user enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

1.3 Domain properties

We suppose the following statements hold in the analyzed world.

- The credentials and payment information provided by the users are correct.
- Assume that the credits in the users credit cards or bank accounts are always enough.
- Each car and safe area and users' mobile devices contain GPS and the GPS works correctly.
- Each car has a way to determine the number of people into the car.
- Each car knows when to lock the car after user leaves it.
- Each car knows its own battery's state(percent of battery empty,charging).
- Users can not leave the car while driving.
- Cars are always connected to the network.
- The system always sends a message to user when it is necessary.
- There is a timer to take care of the management of some operations.
- There are maintainers to take care of charging the battery of cars.
- There are maintainers in the stations.
- There will be no car accident which interrupts the users while driving.

1.4 Glossary

Here presents the detail defination of some common used terms.

- Guest
    - Guests are people who haven't performed the log in operation.They may be registered clients or unregistered clients. As long as they have not performed log in, they are guests.
- User
    - Refer to the clients who have performed log in,
- Credentials
    - Some personal information about each user, including:
    - Name
    - Email address
    - Number of drive license
- Payment information
    - Number of the bank acount or credit card
    - Expired time
    - Security code
    - Holder name
    - Phone number
- Driver
    - He is the driver of the renting car. According to the regulations, only the user who rent the car can

drive.

- Passenger
    - People on the car, other than the driver.
- Safe area
    - a set of locations predefined by the system
    - It may has the power plug or may not.
- Station
    - stations are always safe areas with power plug.

1.5 Assumption

There are some information that is not clear in the specification document, so here lists the assumptions which make the specification more clear.

- The available cars are the cars with full battery
- The available cars are the cars in stations
- Since only the user uploads the information of the driving licence, only the user can act as a driver while passengers can not.

1.6 Stakeholders

Our main stakeholder is Pro.Mottola who requires a managing system for PowerJoy car sharing servive.

2 Actors Identifying

The actors involved in our system is described below

- Guest: A potential user who has not registered yet
- User: User logs into the system and performs the payment

3 Requirements

3.1 Funcitonal requirements

Assuming all the domain properties hold and we derive the correspondong requirements from the goals.

- [G1]Users can register to the system and get the password by providing credentials and payment information

    - The system must let only one registeration per email
    - The system must create and mantain only one password per email
    - The system must send the password to the corresonding email adress when the registeration is completed

- [G2]Registerd users can log in by providing the correct credentials

    - The system must check whether the password and email address are matched
    - The system must allow the user enter the system when the credentials are correct

- [G3]Registered users can find the locations of available cars within a certain distance from their current location or from a specified address

    - The system must be able to get all the locations of the cars
    - The system has a default range which identifies the term "nearby"
    - The system must be able to get all the information of the nearby cars corresponding to a given location

- [G4]Among the available cars in a certain geographical region, users must be able to reserve a single car for up to one hour before they pick it up.

    - The system must allow users to reserve a single car
    - The system maintains a clock for each car. Once the car is resvered by a user, a one-hours time clock starts
    - The system must notify the user of the remaining reservation time

- [G5]If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; the user pays a fee of 1 EUR.

    - The system must allow the user who has made a reservation for the car to stop the clock if he/she is nearby
    - The system must charge the user 1 euro for not being able to stop the clock within one hour
    - The system will notify the user if he/she is charged by the fee
    - The system will make the car state changed to available if the clock is not stopped within in one hour

- [G6]A user that reaches a reserved car must be able to tell the system she’s nearby, so the system unlocks the car and the user may enter.

    - The system must be able to check the position of the user according to the user's GPS
    - The system must be able to check the position of the car according to the car's GPS
    - The system should be able to check whether the position of the user is the same as the corresponding car
    - If the user and the corresponding car are in the same position, the system must transfer the control information to the appropriate car for unlocking the car

    - If the user and the corresponding car are not in the same position, the system must be able to keep the car locked

- [G7]As soon as the engine ignites, the system starts charging the user for a given amount of money per minute; the user is notified of the current charges through a screen on the car.
    - The system must be able to detect whether the engine of the car is ignited or not
    - The system must be able to record the total using time

- [G8]The user is notified of the current charges through the screen on the car
    - The system must be able to estimate the current fee during the renting servece
    - The system must be able to transfer the charging informatin to the appropriate car

- [G9]The system stops charging the user as soon as the car is parked in a safe area and the user exits the car; at this point, the system locks the car automatically.
    - The system must be able to check the position of the car according to the car's GPS
    - The system must be able to detect whether the car is parked or not
    - The system must be able to check whether the car is parked in the safe area
    - The system must be able to detect whether the user has exited the car
    - If the car is not parked in the safe area, the system must be able to keep the car in the unlocked state and keep charging
    - If the car is parked in the safe area and the user has exited the car, the system must be able to transfer the control information for locking the car

- [G10]The users can know the locations of the safe areas
    - The system must be able to show users the locations of the safe areas

- [G11] If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.
    - The system must be able to detect the number of passengers in the car
    - If the number is equal or more than 2, the system should register a 10% for the corresponding ride

- [G12]If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.
    - After the lock of car, the system will examine the level of battery, if over half the battery is remained, then the system will register a discount of 20% for the ride

- [G13]If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.

    - The system will register a discount of 30% on the last ride if it recognize the car is parked in stations and be plugged to the gird

- [G14]If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges 30% more on the last ride to compensate for the cost required to re-charge the car on-site.

    - system must be able to get the positions of the cars and all positions of power grid stations.
    - system will record an increase of 30% on the fee of last ride, if there are not any power grid stations within 3KM from the position of car or the car is left with no more than 20% of battery.
    - system must inform the maintainers to retrive the cars with low battery to the stations.

- [G15]If the user enables the money saving option, he/she can input his/her final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

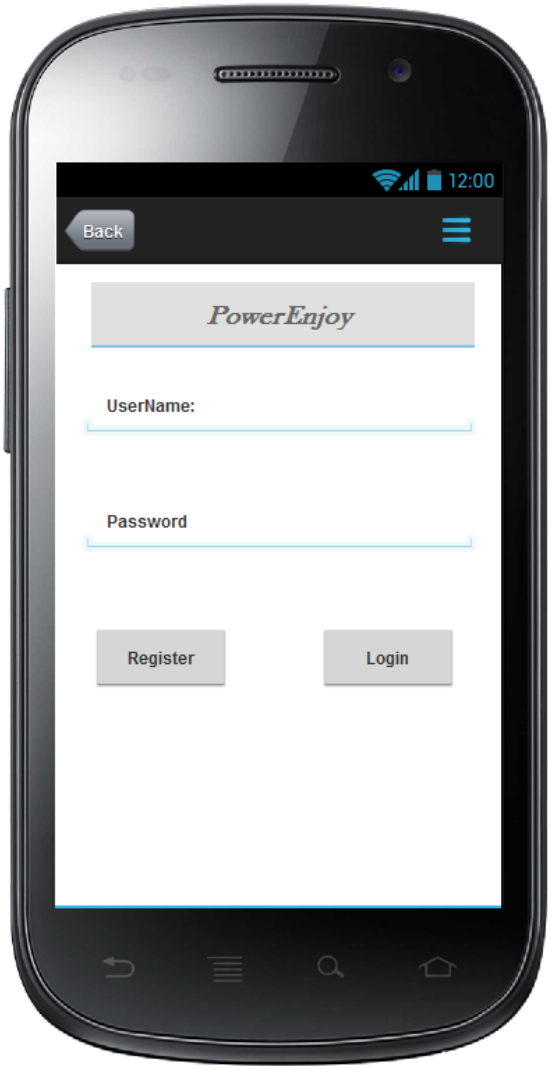
    - The system must be able to get the distribution of the cars
    - The system must be able to detect whether the user enables the money saving option or not
    - The system must be able to select the station according to the user's destination for ensuring a uniform distribution of the cars
    - The system must be able to detect the availability of power plugs at the selected station

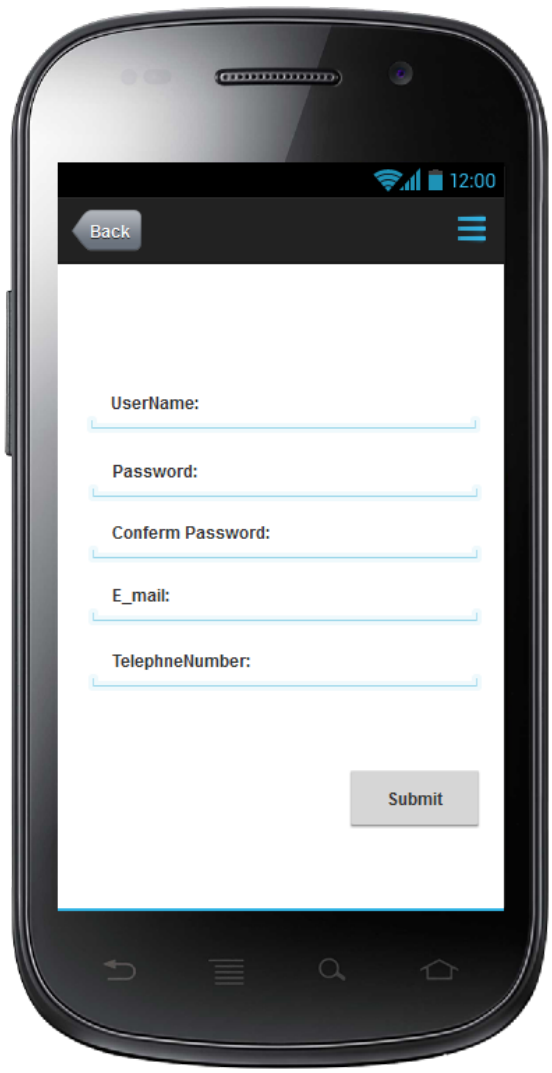
3.2 Non functional requirements

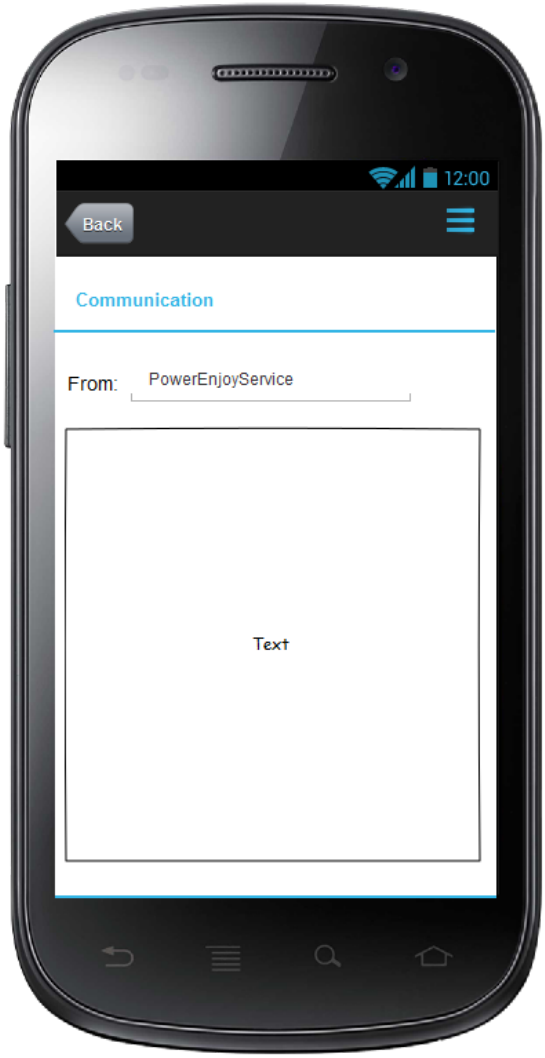
The system is constrained by these as much as possible:

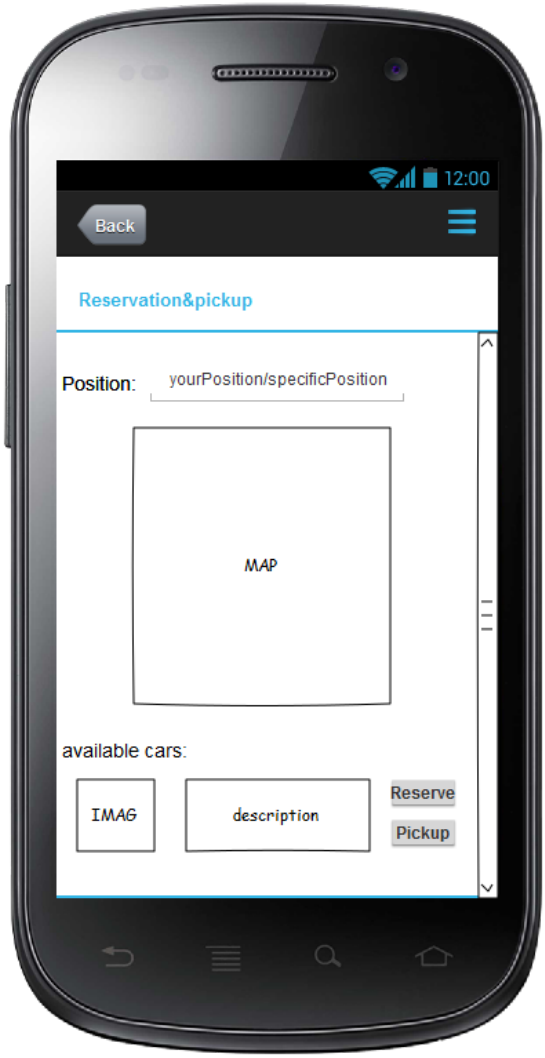
- The system must be available 24 hour a day, 7 day a week.
- The reliablity must be more than 99%
- The response time must be less than 2 second.
- The system must guarantees 99.999& of security and integrity of user's data. data security
- The system must be able to cope the input error within 99.9% of case.
- Determination of car's position in real time.
- Registration data must be stored within 100ms.
- The system must be able to be updated within 1 day.

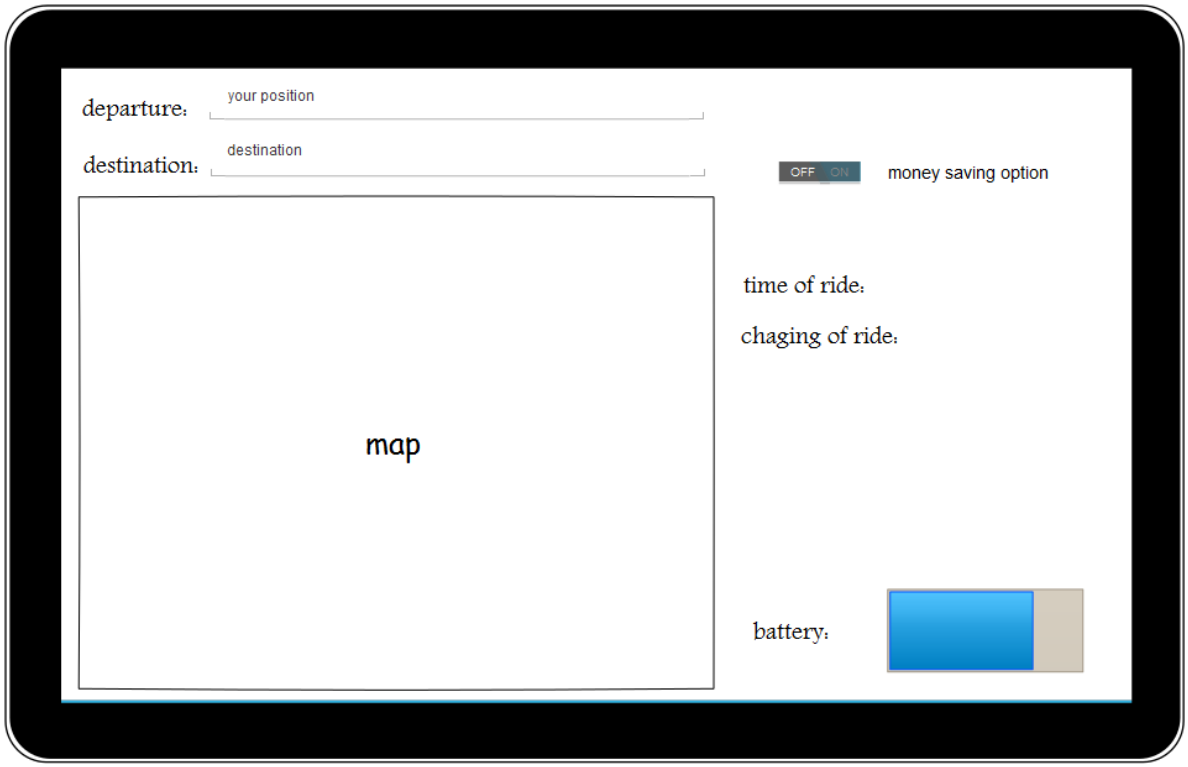
Client interface


12:00
Back
PowerEnjoy
UserName:
Password
Register
Login

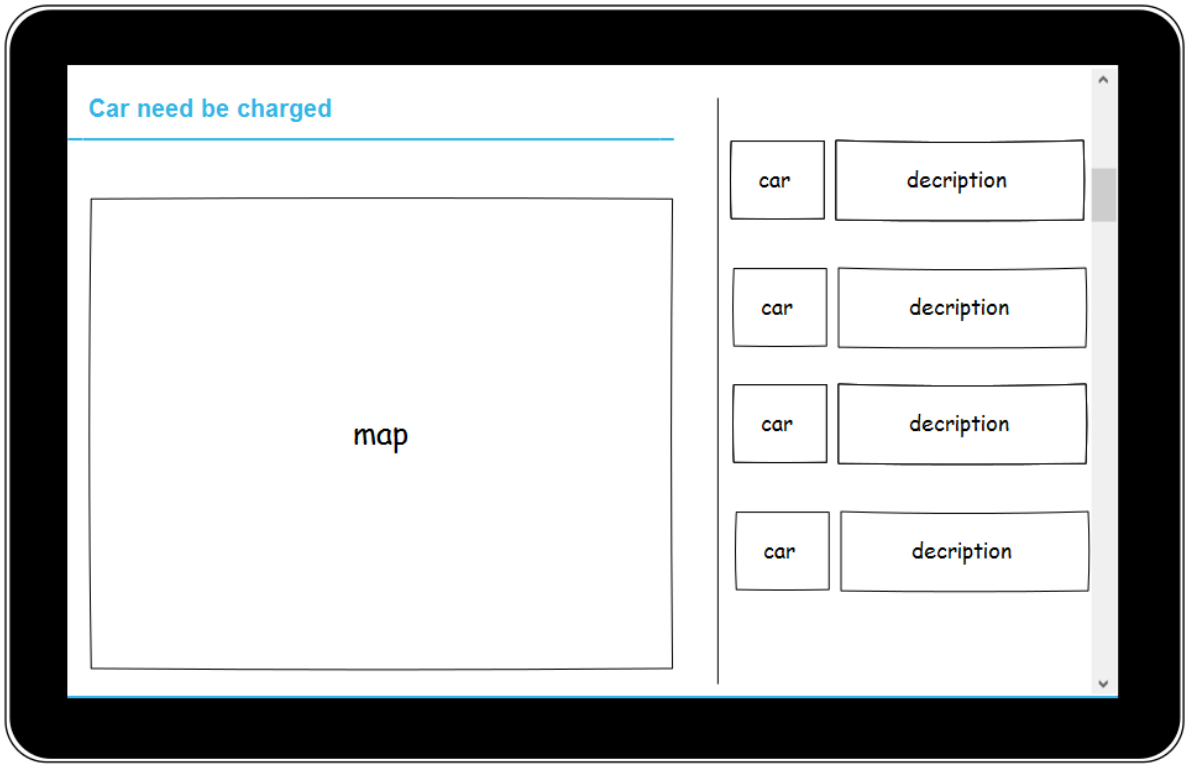

12:00
Back
UserName:
Password:
Conferm Password:
E_mail:
TelephneNumber:
Submit


12:00
Back
Communication
From:
PowerEnjoyService
Text


12:00
Back
Reservation&pickup
Position:
yourPosition/specificPosition
MAP
available cars:
IMAG
description
Reserve
Pickup


departure:
your position
destination:
destination
OFF ON
money saving option
map
time of ride:
chaging of ride:
battery:

Admin : however not implemented in the following assignments



4 Scenario identifying

Scenario 1

perfect user

Mario is a college student who possesses a driving licence. He saw the ads about PowerEnJoy and decided to try this service. He uses his laptop to access the website of PowerEnjoy and proceeds the registering procedure. He provides his credentials and payment information. After the system analyzes his data, Mario receives a password. He logs into the system and finds some available cars near his place. He selects one car and makes a reservation. 30 minutes later, Mario arrives at the car position and he uses his laptop to notify the system that he is near the car. The system checks his position and unlocks the car. Mario drives home and on the car screen he can see his current driving route,charges and safe areas nearby. He stops the car in one of the safe areas near home and finishes his journey.

Scenario 2

punished by not being able to get to the car within one hour

Tom is a previous user of the PowerEnJoy. After work, he decides to reserve a car to go home. He uses his smart phone to reserve a car and head to it. However, he encounters his colleague on his way and they chat for a while. Before he manages to get to the car, one hour has passed. He receives a message on his phone that tells him his reservation is cancelled and he is charged one euro for punishment. Tom has to log

in the system and reserve a car again.

Scenario 3 more persons in the car which enables discount

Nino is a college student. Meanwhile, he is also a user of the PowerEnjoy. He studies in the university together with two of his roommates. Finishing his lectures in the afternoon, Nino comes back home together with his roommates. First of all, Nino logs into the PowerEnJoy system using his smart phone. Then, he reserve a car and arrives at the position of the car. Nino picks it up and drives the car home. 40 minutes later, Nino stops the car at one of the safe locations near his home. Since there are 2 passengers in the car in this renting service, Nino gets 10% discount for the total renting fee.

Scenario 4 taking care of charging the car and get 30% discount from it

Gino is an employee of the technology company. Everyday, Gino rents the electric car. He is an old user of the PowerEnJoy. In the morning, He rents a car from the station nearby his home. Driving the rented car to the company, he stops the car in the station in front of the company. After exiting the car, Gino takes care of plugging the car into the power grid. After work, Gino rents an electric car in PowerEnJoy system by his smartphone again. When he arrives at home, he takes care of plugging the car into the power grid as the same as what he did in the morning. Therefore, he get 30% of discount for these two renting services in the whole day.

Scenario 5 park in unsafe area

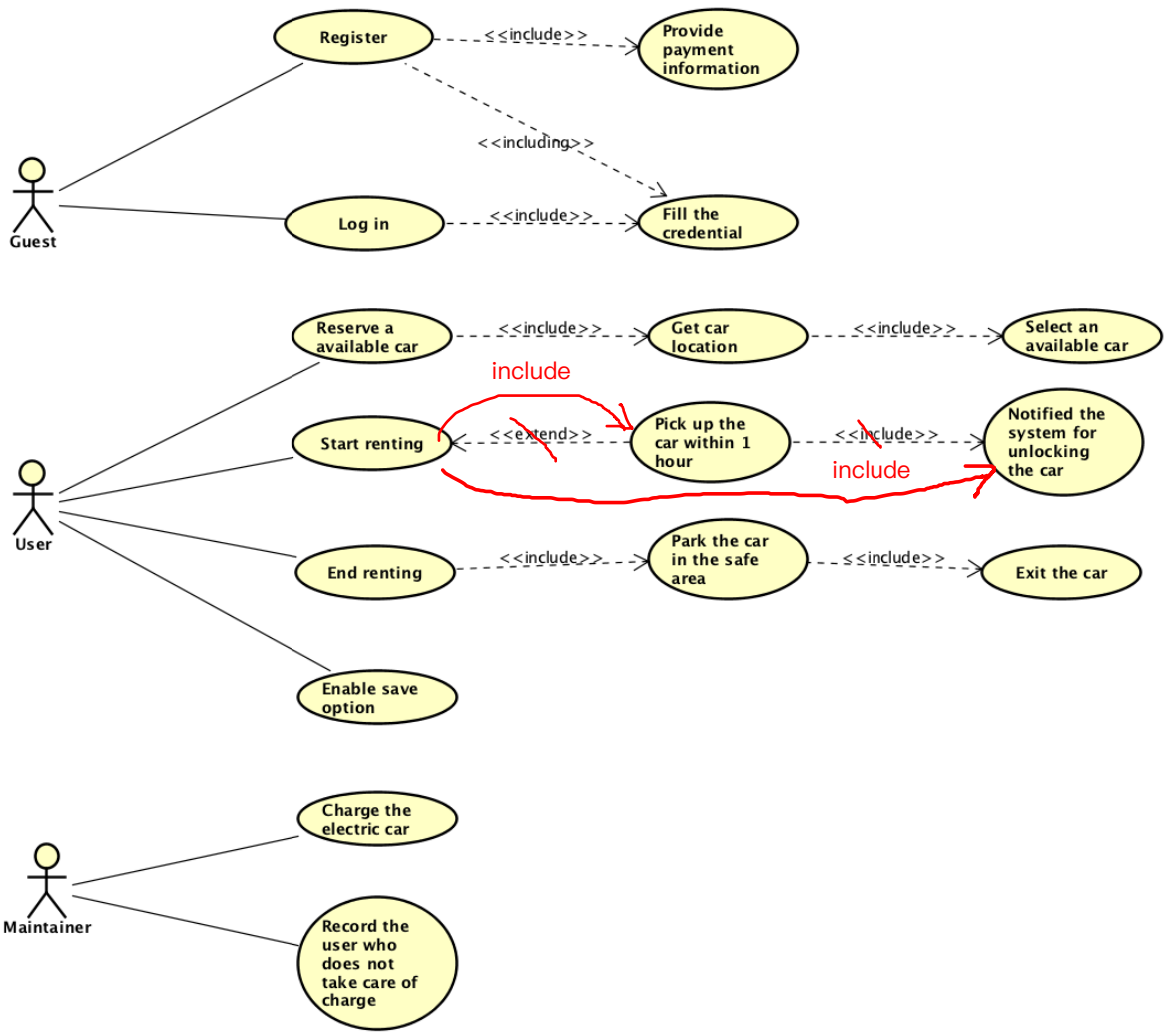
Matteo is a college student who lives in a university residence. In Christmas hoiday, he decides to go back to his parents' home using the PowerEnjoy car renting service. After registering and logging in the system, he reserves a car close to the residence. He picks up the car within an hour. He arrives at his parents' house at almost 22:00. Unfortunately, there are no stations nearby, and the battery is less than 15%. So Matteo parks the car in a safe area without charging the car. Due to a scarce battery remained and large distance from power plugs, Matteo pay 30% more on the trip.

Scenario 6 discounts added up

Andrea is an emploee of a company and he is an expert user. During a holiday, he organizes a picnic outside the city with 4 other friends. They decide to meet at railroad station and then go to other places together. He inserts the position of station into the system, and he reserves one of the cars offered by the system. They successfully pick up the car within one hour. Before beginning their journey, Andrea enables the money saving option. After they arrive at the destination, Andrea parks the car into the safe area given by the system and pluggs the car into power grid,even though the car has still more than 70% of battery. Therefore, he gets the accounts, 10% for taking more than two passegers, 20% for using little battery, and 30 % for plugging the care, all bonus into once ride.

5 Use case

5.1 Use case diagram



5.2 Use case description

(1)Guest registers in the system
Name: Guest registers in the system
Actor: Guest
Entry condition: No
Flow of events:

- The user enters the web home page or opens the mobile app and clicks "Sign up"
- The user fills in the credentials
- The user fills in the payment information
- The system gives the user a password for loging in

Exit condition: The user receives password from his/her email box.
Exceptions:

- The user misses information during filling blanks.
- The user has already registered with the same credentials.

Exceptions handling: The user can either continue filling all the correct information or terminates the registering process.

(2)Guest logs in to the system
Name: Guest logs into the system
Actor: Guest
Entry condition: The guest provides correct email address and password
Flow of events:

- The user enters his/her email address and password
- The user pushes the "log in " button
- The system redirects the user to his personal home page

Exit condition: The user is redirected to the personal home page.
Exceptions:

- The system database can not find registering information of this current email address.
- The user provides wrong match between email address and password.

Exception handlings: In either case, the user can choose to continue filling the blanks until correct match occurs or the user can exit the process.

(3)User reserves an available car
Name: User reserves an available car
Actor: User
Entry condition: The user successfully login with his/her email address and password
Flow of events:

- The user clicks on the "From current address" button or sets the "From specified address" box to the desired value.
- The user sets the "Certain distance" to the desired value.
- The system notifies the user with the available cars within the desired distance of desired address.
- The user selects one of the available cars
- The user clicks on the "Reserve this car" button.
- The system notifies the user for the reservation has been done.

Exit condition: The user receives the notification about the reservation is completed and clicks the buttom "Confirm".
Exception:

- There is no available car within desired distance of desired address.
- The user does not want to reserve any of the available car

- The desired car has been reserved before the user completes the reservation

Exception handing: The user can reset the desired distance and desired address, or select another available car. The user can also cancel this reservation.

(4)User starts renting
Name:User starts renting
Actor:User
Entry condition:The user has reserved a car and pick it up within 1 hour
Flow of events:

- The user notifies the system with his/her nearby.
- The system checks his/her location and unlocks the corresponding car.
- The user enters the car and ignites the engine.
- The system starts charging.

Exit condition:The user is notified of the current charges through a screen on the car. **Exception**:The position of the user is not as the same as the position of the corresponding car.
Exception handing:The system should notify the user that he/she is not in the correct position

(5)User ends renting User can not park the car in the “unsafe” areas
Name:User ends renting
Actor:User
Entry condition:The user has started the renting successfully
Flow of events:

- The user parks the renting car in the safe area.
- The user exits the car.
- The system locks the car automatically.
- The system detects the passenger in this renting service, if the passenger is no less then 2, applies 10% discount for this renting service.
- The system detects the battery remaining amount, if the battery is no more than 50% empty, applies 20% discount for this renting service.
- The system detects whether the car is plugged into the power gird or not, if yes, applies 30% discount for this renting service.
- The system detects the position of the car, if the car is 3KM away from the power station and the battery is more than 80% empty, charges 30% more for this renting service.

Exit condition:The system calculates the total fee for this renting servcie
Exception:The user does not park the car in the safe area.
Exception handing:The system notifies the user that he/she does not park the car in the correct position.

(6)User enables save option
Name:User enables save option

Actor:User
Entry condition:User has started the renting successfully
Flow of events:

- The user clicks the "Money saving option" button.
- The user sets the "Destination" to the desired value.
- The system tells the user about the station where the user should park the car in.

Exit condition: The user receives the message about where to park the car. **Exception**:There is no station for user to park the car.
Exception handing:The user can either reset the destination or cancel this application.

(7)Maintainer charges the electric car
Name:Maintainer charges the electric car
Actor:Maintainer
Entry condition:The user does not park the car in the power grid station or does not plug the car into the power grid. **Flow of events**:

- The user does not park the car in the station or leaves the car without charging.
- The maintainer sent the message to the system for asking authority.
- The system unlocks the corresponding car.
- The maintainer restarts the car.
- The maintainer plugs the power grid into the car.
- The maintainer exits the car.
- The system locks the car.

Exit condition:The system receives the confirmation about the car is plugged into the power grid.
Exception:

- There is not available power grid in the corresponding station
- The car is out of power in the way to the power grid station

Exception handing:

- The maintainer changes others power grid station for charging the car.
- The maintainer asks for special assistance.

(8)Maintainer records the user who does not charge the car **Name**:Maintainer records the user who does not charge the car
Actor:Maintainer
Entry condition:The user does not park the car in the power grid station or does not plug the car into the power grid.
Flow of events:

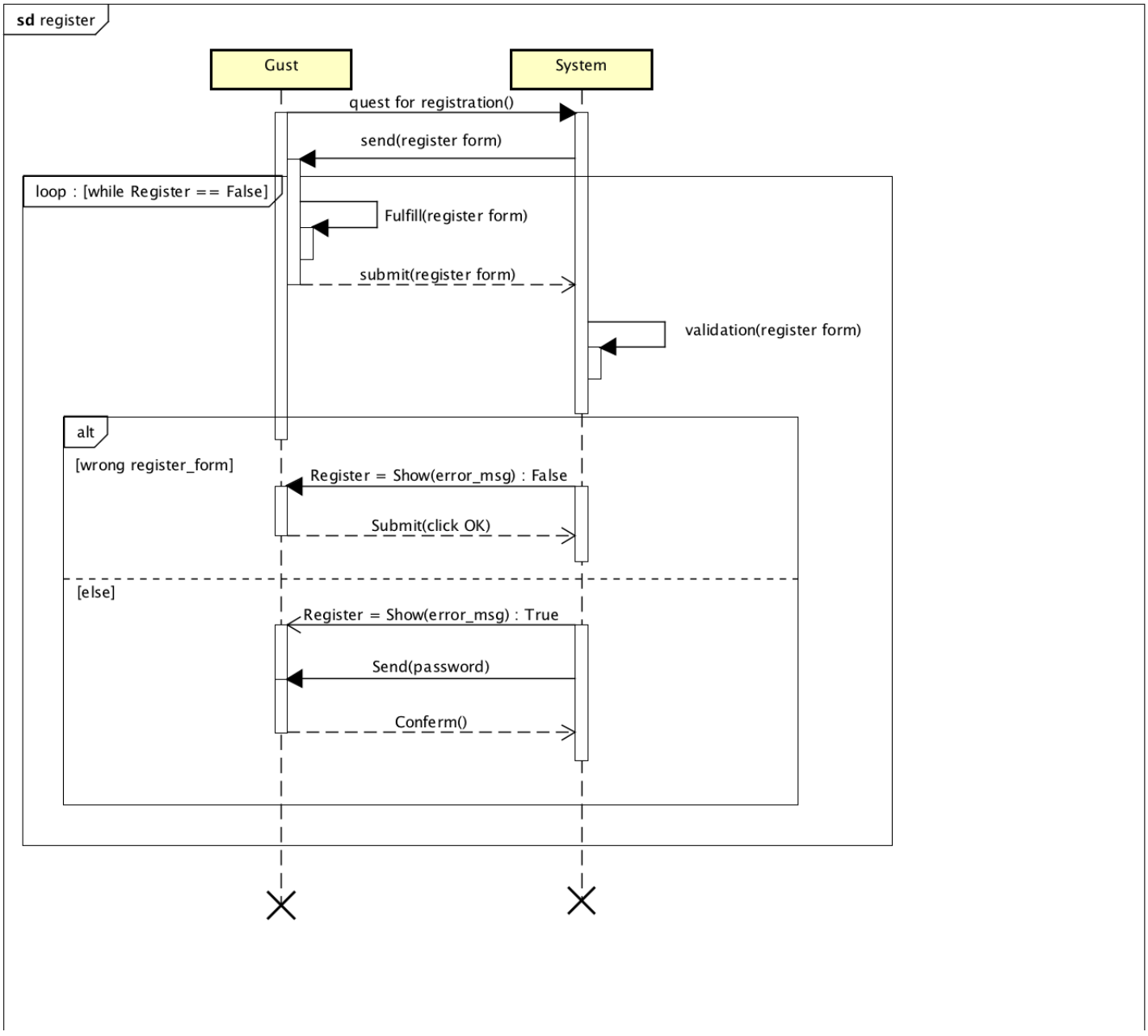
- The user does not park the car in the power grid station or does not plug the car into the power grid.

- The maintainer send the message to the system for reporting the code of the car.
- The system retrieves the car and the corresponding user.
- The system send the confirmation to the maintainer.

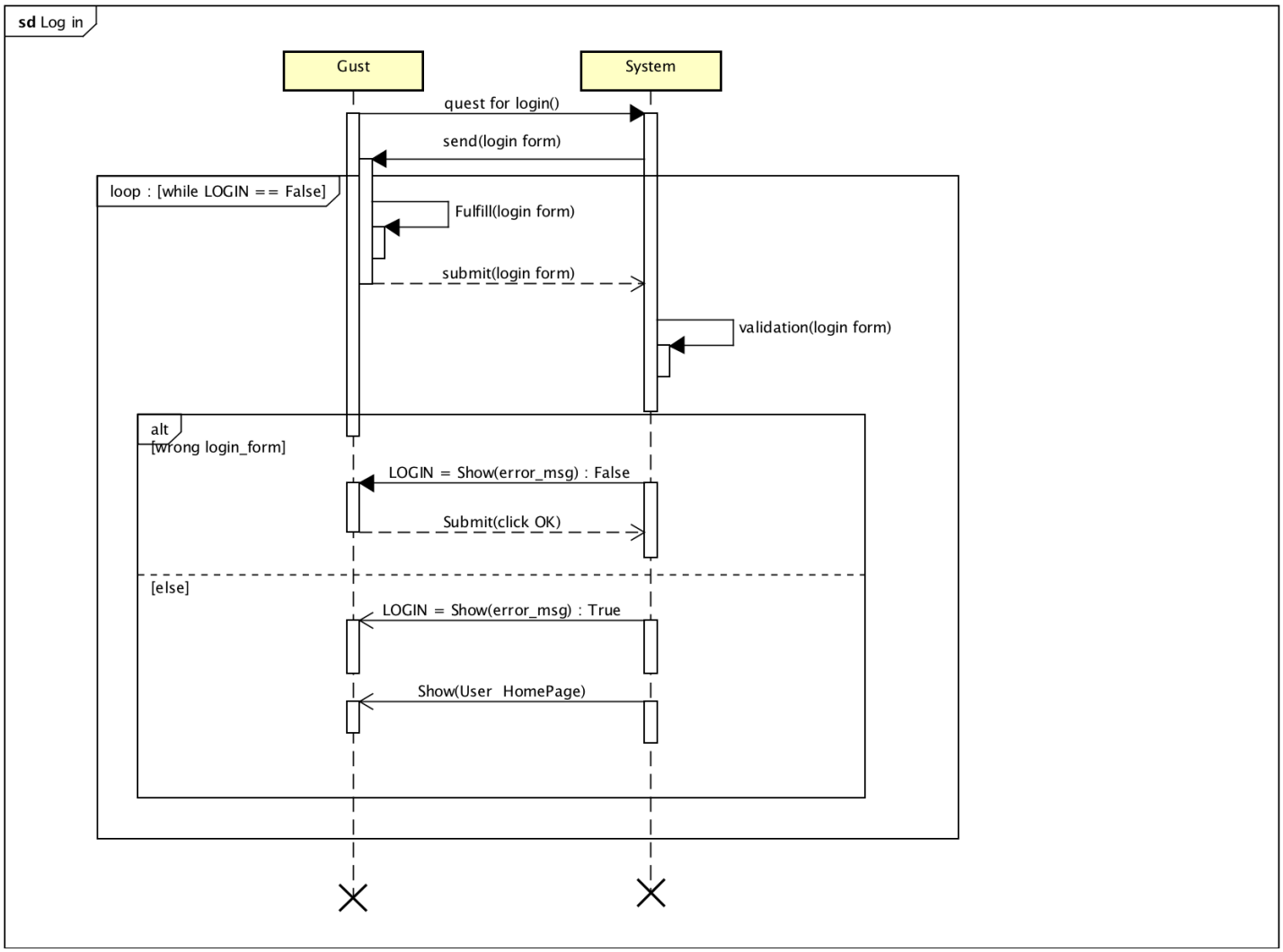
Exit condition:The maintainer receives the confirmation
Exception:There is no available power grid in the corresponding station.
Exception handing:The maintainer send the message to the system for reporing there is no power grid available.

6 Sequence diagram

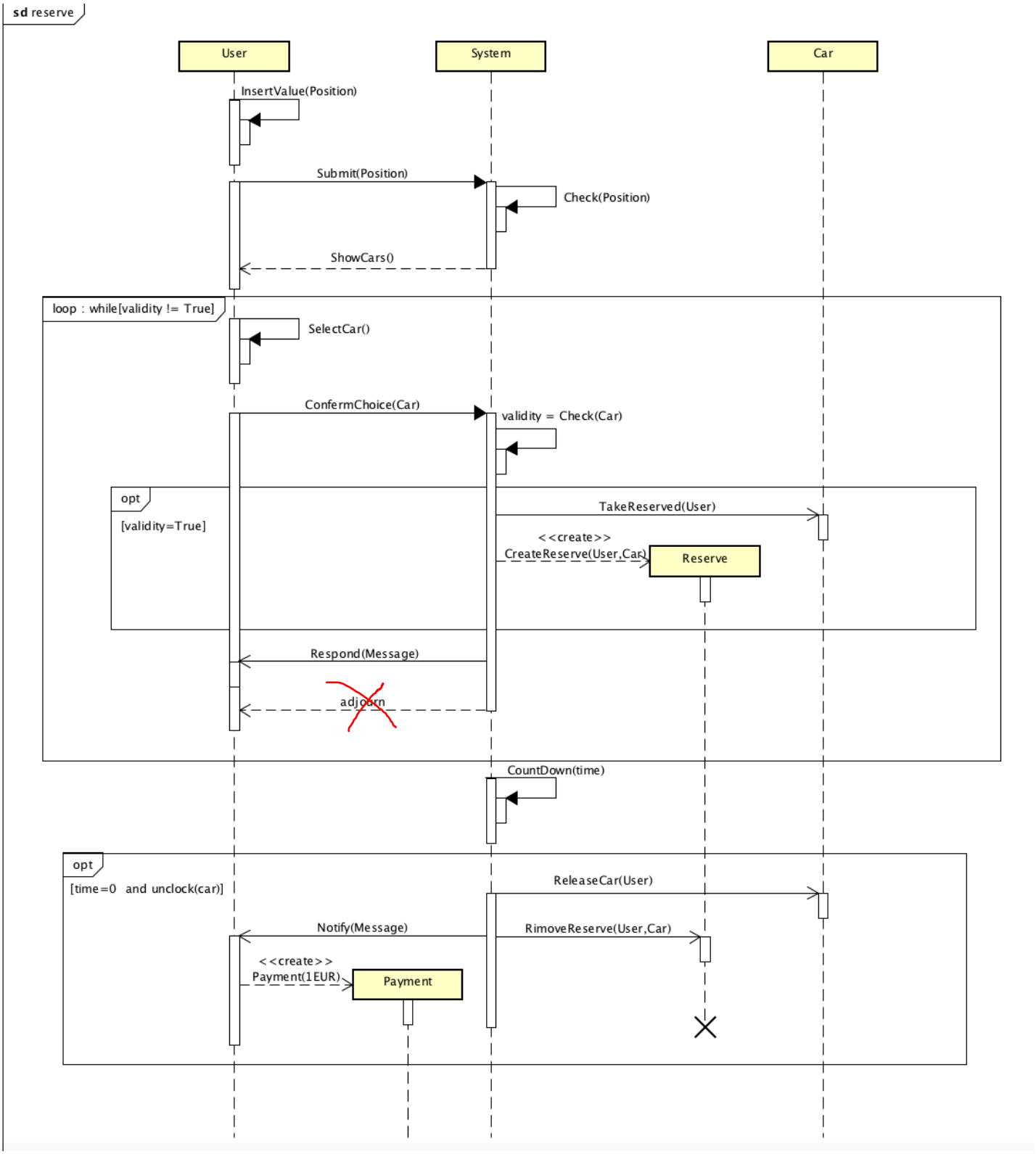
6.1 Register



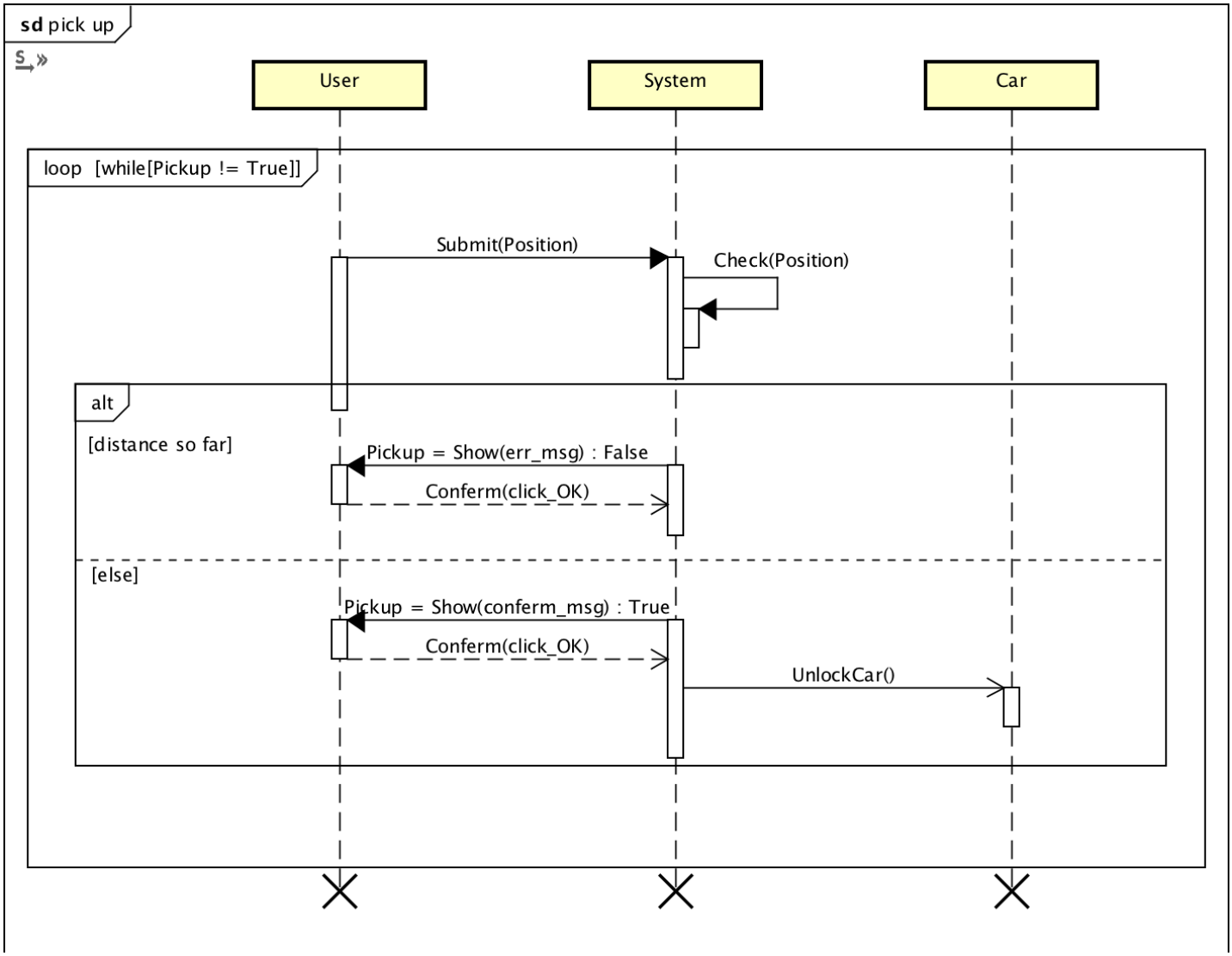
6.2 Log in



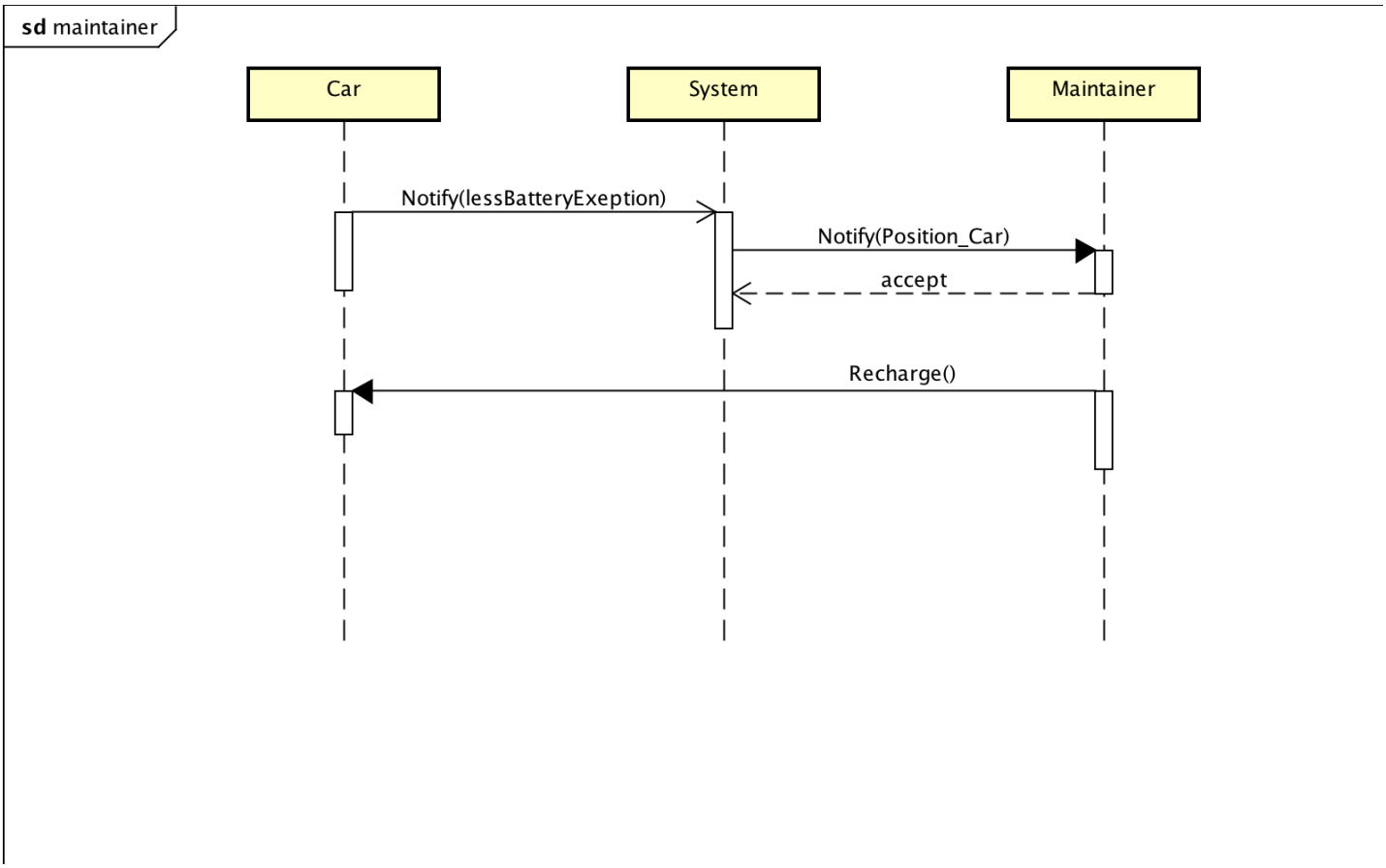
6.3 Reserve



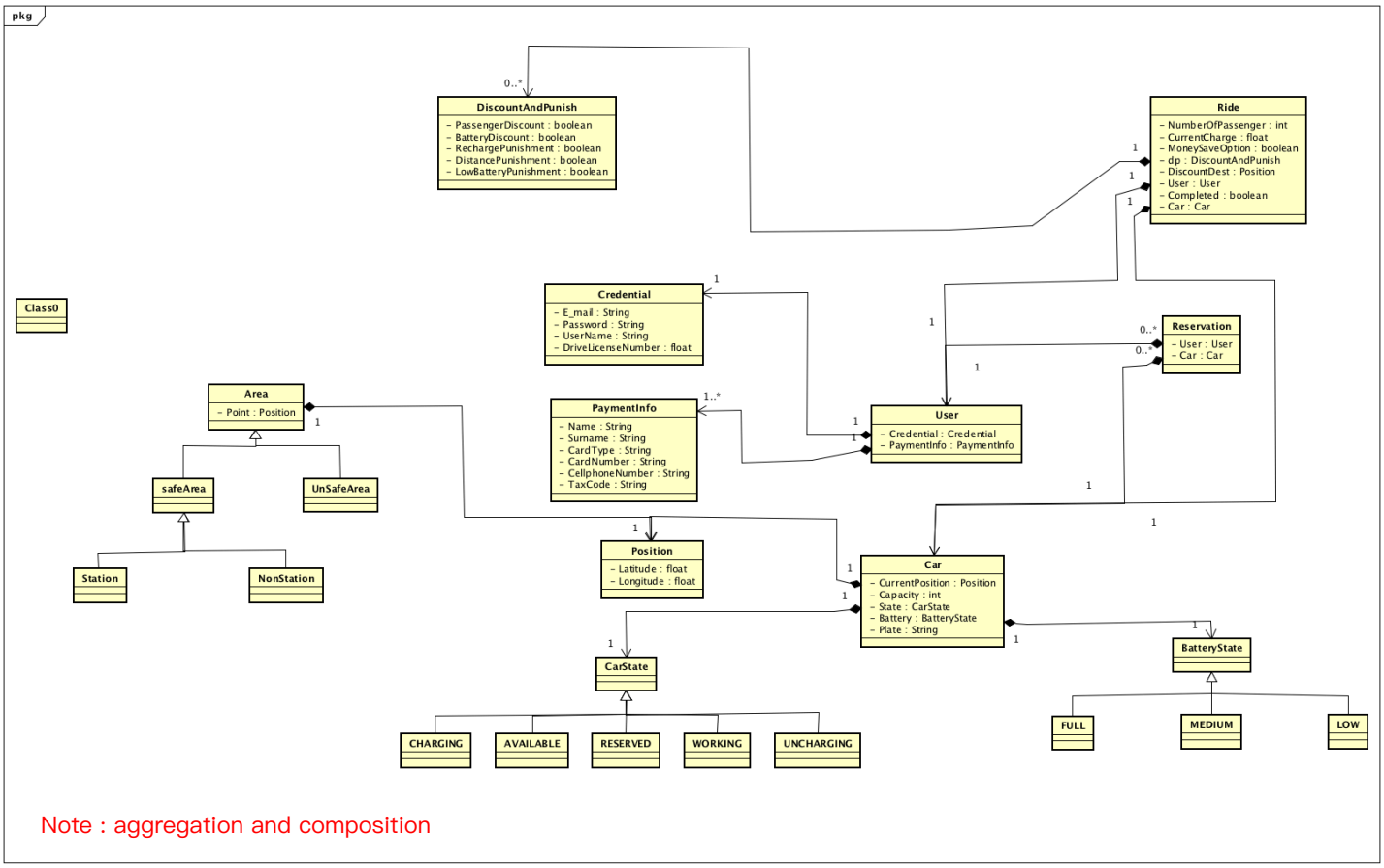
6.4 Pick up the car


sd pick up
User
System
Car
loop [while[Pickup != True]]
Submit(Position)
Check(Position)
alt
[distance so far]
Pickup = Show(err_msg) : False
Conferm(click_OK)
[else]
Pickup = Show(conferm_msg) : True
Conferm(click_OK)
UnlockCar()

6.5 Maintain



7 Class digram



8 Alloy model

8.1 Alloy code

```
open util/boolean

sig Float { }
sig Astring { }

sig Credential {
    email : one Astring,
    password : one Astring,
    username : one Astring,
    drivingLicenseNo : one Astring,
}
sig PaymentInfo {
    name : one Astring,
    surname : one Astring,
    cardType : one Astring,
    cardNo : one Astring,
    cellphoneNo : one Astring,
    taxCode : one Astring
} {
    all p : PaymentInfo | p in User.paymentInfo
}

sig User {
    credential : one Credential,
    paymentInfo : set PaymentInfo,

} {
    paymentInfo != none
}

sig Position {
    latitude : one Float,
    longitude : one Float
} {
    all p : Position | p in Area.points
}

//Area should be a range, but for simplicity we consider only a set of points

abstract sig Area {
    points : set Position
}
sig SafeArea extends Area { }
```

```
sig UnSafeArea extends Area { }
sig Station extends SafeArea { }
sig NonStation extends SafeArea { }

abstract sig CarState { }
one sig AVAILABLE extends CarState { }
one sig RESERVED extends CarState { }
one sig WORKING extends CarState { }
one sig CHARGING extends CarState { }
one sig UNCHARGING extends CarState { }

abstract sig BatteryState { }
one sig FULL extends BatteryState { }
one sig MEDIUM extends BatteryState { }
one sig LOW extends BatteryState { }

sig Car {
    plate : one Astring,
    capacity : one Int,
    state : one CarState,
    battery : one BatteryState,
    currentPosition : one Position,
} {                                          Assume that all cars in station are charging
    capacity > 0 and capacity <= 4
    state = AVAILABLE <=> battery = FULL
    state = RESERVED <=> battery = FULL
    state = AVAILABLE implies currentPosition in Station.points
    state = RESERVED implies currentPosition in Station.points
    state = CHARGING implies currentPosition in Station.points
    state = UNCHARGING implies currentPosition in NonStation.points
    state = WORKING implies this in Ride.car
    state = CHARGING implies this in Ride.car
    state = UNCHARGING implies this in Ride.car
    state = RESERVED implies this in Reservation.car
}


sig Reservation {
    user : one User,
    car : one Car,
} {
    car.state = RESERVED
}

one sig System {
    users : set User,
    cars : set Car,
    safeAreas : set SafeArea
```

```
} {
    all u : User | u in users
    all c : Car | c in cars
    all sa : SafeArea | sa in safeAreas
}

sig Ride {
    user : one User,
    car : one Car,
    numberOfPassenger : one Int,
    currentCharge : one Float,
    moneySaveOption : one Bool,
    discountDest : lone Position,
    completed : one Bool,
    dp : one DiscountAndPunish
} {                              to handle discount condition
    car.state = WORKING or car.state = CHARGING or car.state = UNCHARGING
    car.state = WORKING implies completed = False
    car.state != WORKING implies completed = True
    moneySaveOption = True <=> #discountDest = 1
    discountDest in Station.points
    dp.batteryDiscount = True implies completed = True
    dp.rechargeDiscount = True implies completed = True
    dp.distancePunishment = True implies completed = True
    dp.lowBatteryPunishment = True implies completed = True
}

sig DiscountAndPunish {
    passengerDiscount : one Bool,
    batteryDiscount : one Bool,
    rechargeDiscount : one Bool,
    distancePunishment : one Bool,
    lowBatteryPunishment : one Bool
}

// -----------FACTS-------------
fact NoSameCredential {
    no disjoint u1,u2 : User | u1.credential = u2.credential
}

fact NoSamePaymentInfo {
    no disjoint u1,u2 : User | u1.paymentInfo & u2.paymentInfo != none
}

fact NoSameArea {
    no disjoint a1,a2 : Area | a1.points & a2.points != none
}
```

```
fact NoCarAtSamePlace {
    no disjoint c1,c2 : Car | c1.currentPosition = c2.currentPosition
}

fact NoSameReservationUserAndCar {
    no disjoint r1,r2 : Reservation | r1.user = r2.user
    no disjoint r1,r2 : Reservation | r1.car = r2.car
}

fact DiscountAndPunishmentCondition {
    no r : Ride | r.dp.passengerDiscount = True and r.numberOfPassenger <= 2
    no r : Ride | r.dp.batteryDiscount = True and r.car.battery != MEDIUM
    no r : Ride | r.dp.rechargeDiscount = True and r.car.currentPosition not in St
ation.points
    no r : Ride | r.dp.rechargeDiscount = True and r.dp.distancePunishment = True
    no r : Ride | r.dp.distancePunishment = True and r.car.currentPosition in Stat
ion.points
    no r : Ride | r.dp.lowBatteryPunishment = True and r.car.battery != LOW
}

//There should be no intersection between {user,car} in Reservation and {user,car}
 in Ride
fact ReservationRideDistinction {
    all r1 : Reservation, r2 : Ride | r1.user != r2.user and r1.car != r2.car
}

//No user and car can reserve and drive simuteneously
fact NoBusyUserCar {
    all r1 : Reservation, r2 : Ride | r1.user != r2.user
    all r1 : Reservation, r2 : Ride | r1.car != r2.car
    no disjoint r1,r2 : Reservation | r1.user = r2.user
    no disjoint r1,r2 : Reservation | r1.car = r2.car
    no disjoint r1,r2 : Ride | r1.user = r2.user
    no disjoint r1,r2 : Ride | r1.car = r2.car
}

//------------ASSERTIONS----------------

assert AvailableCarPosition {
    no c : Car | c.state = AVAILABLE and c.currentPosition in UnSafeArea.points
}

assert ReservedCarPosition {
    no c : Car | c.state = RESERVED and c.currentPosition in UnSafeArea.points
}

assert LowBatteryPunishmentCheck {
```

and not low

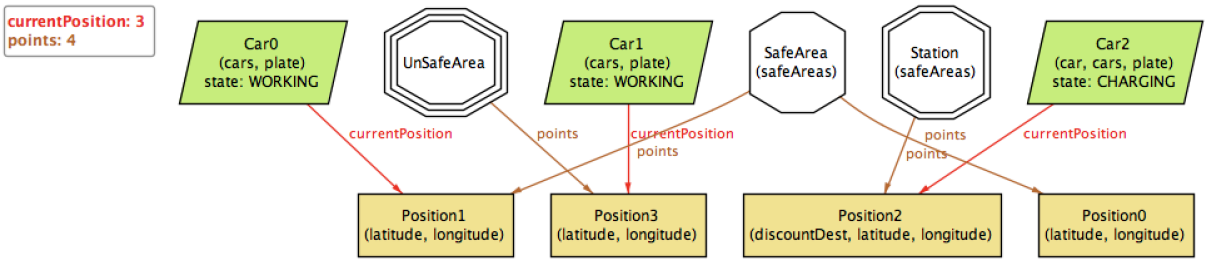
redundant

```
    no r : Ride | r.dp.lowBatteryPunishment = True and r.car.battery = MEDIUM
}

//-----------------SHOW----------------------

pred example {

/*
    some c : Car | c.state = AVAILABLE
    some c : Car | c.state = RESERVED
   some c : Car | c.state = CHARGING
    some c : Car | c.state = WORKING
    some c : Car | c.state = UNCHARGING
*/

    #Car = 3

}

run example  for 4


check LowBatteryPunishmentCheck
check AvailableCarPosition
check ReservedCarPosition
```

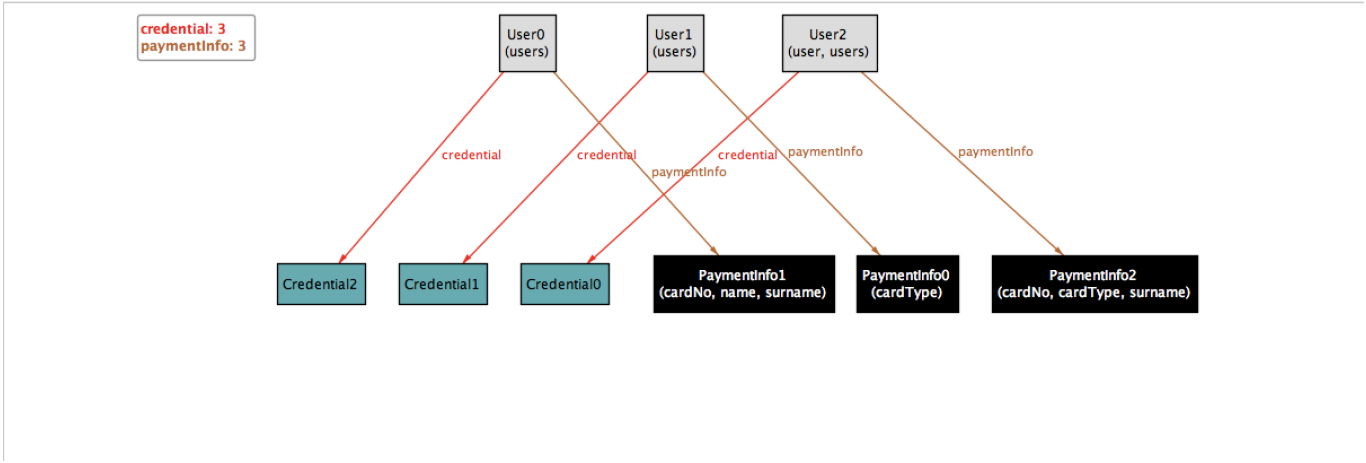
8.2 Alloy graph

Here are some alloy graphs describing the logic relations

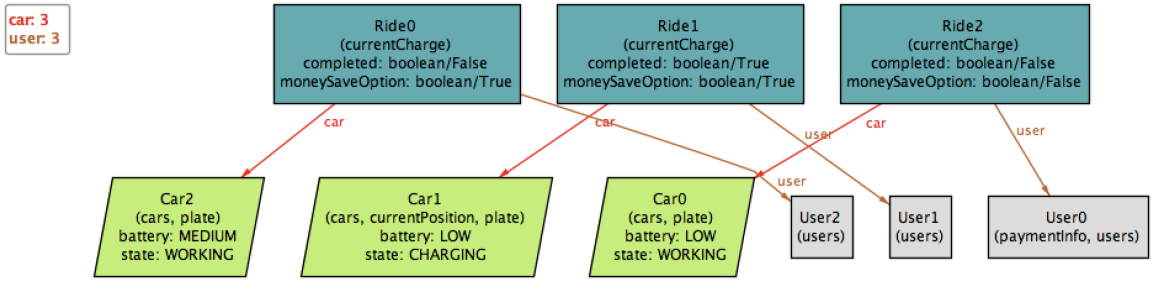
1. Area charing must be in station; working can be in whatever place



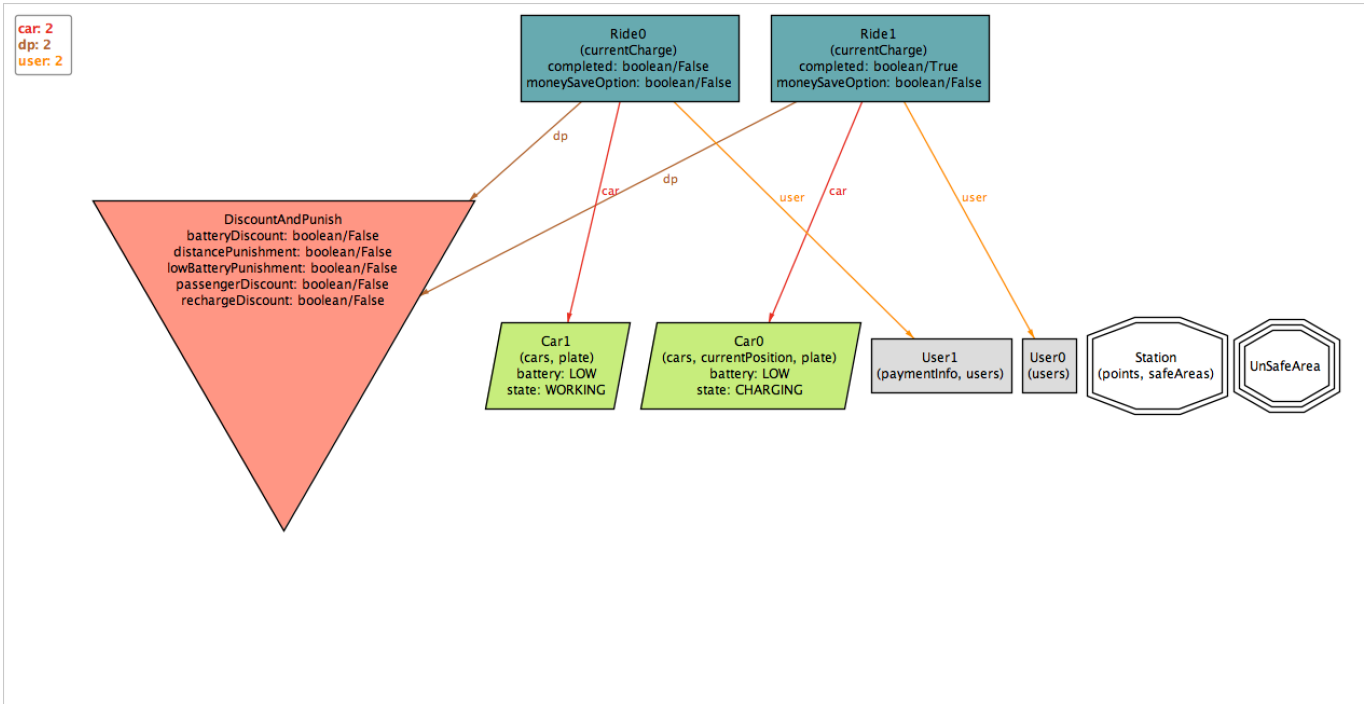
2. User credential and paymentInfo relation



3. Ride



4. Discount



9 Work Time

ZHOU YINAN

- **22/10/2016 1.5h** create RASD file, write introduction, goals,domain properties and some requirements
- **29/10/2016 2h** add content table, glossary, actor identifying, and first two scenario

- **31/10/2016 2.5h** add User cases
- **4/11/2016 4h** start writing alloy
- **5/11/2016 4h** alloy
- **8/11/2016 4h** modify RASD documents and change alloy
- **10/11/2016 2h** alloy
- **11/11/2016 1h** modify RASD
- **12/11/2016 3h** modify RASD for version 1.0

ZHAN YUAN

- **25/10/2016 40m** add some requirements
- **4/11/2016 4H** adjourn the domain property, assumption, requirement and add two scenario
- **5/11/2016 5h** class diagram and sequential diagram
- **6/11/2016 2.5h** complete sequential diagram and activity diagram
- **8/11/2016 2.5h** state diagram and interface
- **12/11/2016 2H** non functional requirement

ZHAO KAIXIN

- **29/10/2016 2h** add some requirements
- **1/11/2016 3h** adjust the glossary
- **2/11/2016 3h** adjust the use case
- **3/11/2016 4h** adjust the use case and add use case description
- **4/11/2016 3h** class diagram
- **5/11/2016 3h** sequence diagram
- **11/11/2016 2h** adjust ClassDiagram