Requirements Analysis and Specifications Document

Figure 1: Logo Politecnico di Milano

Figure 2: Logo PowerEnjoy

- Maria Chiara Zaccardi
- Nicola Sosio
- Riccardo Redaelli

Contents

1

Introduction

1.1 Description of the given problem

We will project and implement PowerEnjoy, which is a car-sharing service that offers only electric cars. The price varies just depending on the real use of each individual rental of the car.

The system allows clients to reserve a car via mobile or web app, using GPS or an address to identify client's zone and find a car in the same zone. On the other side the mobile app allows operators to know what they must do. The system provide the functionality normally provided by car-sharing services. Indeed the system let the registered users to see cars location through the city and to reserve a car until one hour before. Moreover, PowerEnjoy wants to incentivize the virtuous behaviors of the users through discounts for those users that park in power grid station, charge the car and travel with more than two passengers, and penalties for those who leave the car with more than 80% of battery empty or far away from a power grid station.

The system offers also a money saving option that users can enable inside the car, which 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 destination of the user and on the availability of power plugs at the selected station.

1.2 Goals

1.2.1 Visitor

- [G1] Allow visitors to register in the system

1.2.2 User

- [G2] Allows users to login in the system

- [G3] Users must be able to find the locations of available cars
- [G4] Allows users to request for the reservation of a car until one hour before
- [G5] Allows users to unlock the reserved car
- [G6] Allows users to know the current battery level of each available cars
- [G7] Allows users to know which are the "safe area"
- [G8] Users should enable the money saving option in the car
- [G9] Users should know where the power grid stations are
- [G10] Users can report if the reserved car has damage
- [G11] Allows users to cancel a reservation
- [G12] Allows users to have discount
- [G13] Allows users to end the ride

1.2.3 Operator

- [G14] Allows operators to login in the system
- [G15] Allows operators to know all the information of cars
- [G16] Allows operators to unlock cars
- [G17] Operators should receive a notification for incoming request of reparation
- [G18] Operators must be able to report the operation they have done
- [G19] Operators should receive a notification for moving a car
- [G20] Operators should receive a notification for charging a car

1.3 Domain properties

We suppose that these properties hold in the analyzed world:

- [D1] All the GPS car always give the right position
- [D2] The GPS of cars cannot be switched off
- [D3] Each car has a maximum number of passenger
- [D5] Sensors detect that the user took at least other two passengers if and only if passengers are really onto the car

- [D6] Operators will take care of all the requests that they receive through notifications
- [D7] A car can be in only one position at the same time
- [D8] Operators GPS must be always on
- [D9] The company is supported by an external payments service (es. Paypal)
- [D10] If the user hasn't enough money on the credit card the external payments service takes care of it
- [D11] We assume that each ride has a fixed price per minute
- [D12] An user report a damage of a car if and only if the car really has a damage
- [D13] For every maintenance services, details about the operation are correctly encoded
- [D14] Operators already have credentials for login which are given by PowerEnjoy when they were hired
- [D15] Car codes are unique
- [D16] If a car is in charge, it will become available as soon as its battery is fully charged
- [D17] Email address used for registration must be a valid address
- [D18] Payment information must be formally correct
- [D19] The car's code must be readable
- [D20] The car's code can't be hidden
- [D21] We assume that the operators during their working hours are always available

1.4 Glossary

- User: a person already registered in the system. A user has a profile that includes all these information:
  - Name
  - Surname
  - Email
  - Phone number
  - Username

  - Password
  - Address
  - Number of credit card (with deadline and CVV)
  - Number of Driving Licence

- Operator: an employee of society. The system gave him references to access to their page where they find their assignment. Operator has a profile that includes this information:
  - Name
  - Surname
  - IDNumber
  - Password

- Visitors: a visitor is a person who hasn't signed up yet. Visitors have less power in the system then user, they don't have the user skills, and the only function they can use is to sign up and read a description of the service.

- Passengers: a passenger is a person travelling in the car but not controlling it, that not need to sign in or sign up in the system. They had no power

- Available car: a car that isn't reserved by another user, and it's charge must be more than 50% and it isn't plug in

- GPS: abbreviation for global positioning system: a system that uses satellites (devices that move around the earth) to show the position of a person or thing anywhere in the world

- Reservation: only users can use cars. One user can reserve a single car for up to one hour before they pick it up. 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. Now user pays a fee of 1€

- Safe Area: it is an area where user must park cars and they don't pay any fee. This area belongs or has a partnership with the society

- Power grid station: special areas with electricity station where people could park cars and recharge it

- Operator notification: is the notification that the system sends to the nearest operator to the car that needs the operator.

- Ride: it starts when the user starts the engine and ends when the user leaves the car in a safe area.

1.5 Text assumptions

- We should develop a mobile application able to use its own position through a GPS or asking it to the client
- We assume that there is a fixed price per minute
- We assume that once the user unlock the car, he/she didn't need to stay logged
- We assume that operators must stay logged during their working hours. Only in this way they can receive notifications from the system.
- We assume that for unlock a reserved car the user has to insert through the app, the car's code that is unique and it's in the front of the car
- We assume that discounts are not cumulative and if a user is entitled to more than only one discount, the system will calculate the highest one.
- If a car is left with no more than 20% of battery we assume that an operator take care of it, so no one user can pick a car with more than 80% of empty battery
- We assume that if a car is left by some user at more than 3 km from the nearest power grid station, the nearest operator will move the car to a power grid station in order to ensure a uniform distribution of cars in the city. This wasn't specified in the assignments but we think it's more reasonable to move the car in a more popular and central area.
- As soon as the engine ignites, the system start charging the user, meaning that the system start counting the amount of money per minute that the external payment service has to charge to the user at the end of the ride. PayPal will actually charge the user when the ride is ended and the system notify to PayPal the final total cost of the rent.
- We assume that if a user couldn't pay last ride the payment society notify the system who block the user's account until he/she could pay

1.6 Constrains

1.6.1 Regulatory policies

The system must require to clients and operators the permission to get their gps positions and they have to manage sensible data (position, phone number, credit cards information) respecting the privacy law. Furthermore the systems must not use notifications to send SPAM to operators respecting the privacy law.

1.6.2 Hardware limitation

Cars have a on-board desktop where users can see safe areas and power grid stations. Cars have also a GPS always available that cannot be remove. Operators and users need to have a phone with installed the respective mobile app; the minimum technical characteristic of the phone are a GPS and an internet connection. If users use the web app they must have a modern browser with AJAX support.

1.6.3 Interfaces to other applications

- Our system needs to interface with the push service(s) via own APIs to send push notifications to operators. Often we need an interface for each platform: android, iOS and windows.
- Our system needs also to interface with the external payment service. PayPal is an online payment service that allows individuals and business to transfer funds electronically.

1.6.4 Parallel operation

The server allow and support parallel operations from different clients and different operators.

1.7 Identifying stakeholders

We have only one stakeholder: the government of the city who wants to improve pollution and traffic promoting green cars. We could adapt this system to other cities, just changing the safe areas and the power grid station because they must be modified for each different cities.

1.8 Reference documents

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

2

Actors identifying

- Visitor: all this people can only see the login form or registration form; they can also see a short description and the rules of the service.
- User : all this people, after successful login, are the only allow to see all available car, reserve a car and open it.
- Operator: after the login, all these people can see all cars position and their condition (power and maintenance). They can also open and turn on cars.
- PayPal : it's not a proactive actor. Our system collaborate with PayPal for the payment process.

3

Requirements

3.1 Functional requirements

3.1.1 Visitor

- [G1] Allow visitors to register in the system
  - [R1] Visitors must not be already registered to perform registration process
  - [R2] Visitors must provide in the registration form their credentials and payment information. They receive back a password that can be used to access the system
  - [R3] Visitors can only see login page with also the registration form and a short description of the service
  - [R4] Visitors can register themselves only with a credit card

3.1.2 User

- [G2] Allow users to log in in the system
  - [R1] User must be already registered
  - [R2] User must enter the email address used during the registration and password that they receive back, to success login
  - [R3] Email and password insert during the login process must be correct
  - [R4] Wrong credentials will not let the users enter in their account
  - [R5] Visitors can't see available cars before registration

- [G3] Users must be able to find the locations of available cars
  - [R1] User must be already logged
  - [R2] The system must be able to check the GPS position of the user or receive an address from the user and show him the nearest available car
  - [R3] User must insert a correct address
- [G4] Allows users to request for the reservation of a car at least one hour in advance
  - [R1] User must be already logged
  - [R2] The system must be able to recognize the origin of reservation
  - [R3] As soon as the user reserve the car, the system delete the car from the list of available car
  - [R4] User can reserved just one car
  - [R5] Car can be reserved only by one user
- [G5] Allows users to unlock the reserved car and to ignite the engine
  - [R1] User must be already logged
  - [R2] The system must be able to recognize that the user that unlock the car is the same that reserve the car
  - [R3] User must unlock the car within one hour from the reservation
  - [R4] User must be next to the car and able to read and insert the car's code in the app in order to unlock the car
  - [R5] As soon as the engine ignites, the system starts charging the user
  - [R6] The system must show the current charge on the onboard desktop
  - [R7] The car must be driven only by the registered user that take the reservation
- [G6] Allows users to know the current battery level of each available cars
  - [R1] User must be already logged
  - [R2] The system must show the battery level of the car
- [G7] Allows users to know which are the "safe area"
  - [R1] User must be already logged or inside the car
  - [R2] The system must show which are the set of safe area in which the user can park and take the car
- [G8] Users should enable the money saving option in the car

  - [R1] User must be inside the car
  - [R2] User must be able to input his/her final destination
  - [R3] The system must be able to provides information about the station where to leave the car to get a discount
  - [R4] The system must know the availability of power plugs at each station
  - [R5] The system must be able to suggest the user the final station in order to ensure a uniform distribution of cars in the city

- [G9] Users should know where the power grid stations are
  - [R1] User must be already logged or inside the car
  - [R2] The system must be able to show the user where the power grid station are and if there are available power plugs

- [G10] Users can report if the reserved car has damage
  - [R1] User must be already logged
  - [R2] User must be able to report the damage of the car
  - [R3] The system must take care of the damage car and notify it to an available maintenance operator
  - [R4] As soon as the system receives the damage notify, it must let disappear the car from the list of the availables ones
  - [R5] The system must delete the reservation of the car by the user that notify the damage

- [G11] Allows users to cancel a reservation
  - [R1] User must be already logged
  - [R2] User must already done a reservation for a car
  - [R3] User must be able to cancel the reservation
  - [R4] The system must be able to reinsert the car in the list of the availables ones
  - [R5] User must cancel reservation within one hour from the reservation

- [G12] Allows users to have discount
  - [R1] User must be already registered
  - [R2] The system apply a discount of 10% when sensors on seats of the car pick up that there are more than two other passengers
  - [R3] The system apply a discount of 20% when the car is left with no more of 50% of the battery empty

  - [R4] The system apply a discount of 30% when the car is left at special parking areas and the user takes care of plugging the car into the power grid
  - [R5] The system must know if a car is plugged into the power grid
  - [R6] The system must know the percentage of battery of each car

- [G13] Allows users to end the ride
  - [R1] User must be already registered
  - [R2] User can end the ride if and only if the car is inside the safe area, otherwise he/she cannot lock the car
  - [R3] As soon as the user end the ride and lock the car, the system reinsert the car in the list of available car
  - [R4] When the ride is correctly end, the system stop to charge the user

3.1.3 Operator

- [G14] Allows operators to login in the system
  - [R1] Operator must enter the username and password that they receive after their hire to complete successfully the login process
  - [R2] Username and password insert during the login process must be correct
  - [R3] Wrong credentials will not let the operators enter in their account
- [G15] Allows operators to know all the informations of cars 
  - [R1] Operator must be already logged
  - [R2] The system must be able to show all the cars location in the city and the informations related
- [G16] Allows operators to unlock cars
  - [R1] Operators must be logged
  - [R2] The system must be able to recognize the origin of the request of unlocking
  - [R3] Operator must be next to the car
- [G17] Operators should receive a notification for incoming request of reparation
  - [R1] Operators must be logged

  - [R2] The system must send a notification to the nearest operator after a user report a damaged car
  - [R3] The nearest operator must take care of repairing the damaged car

- [G18] Operators must be able to report the done reparation
  - [R1] Operators must be logged
  - [R2] The system must be able to reinsert the car in the list of available car
  - [R3] The system should charge a fee to the last user which used the car

- [G19] Operators should receive a notification for moving car
  - [R1] Operators must be logged
  - [R2] The system must send a notification to the nearest operator after some user leaves the car at more than 3 km from the nearest power grid station
  - [R3] The system should charge 30% more on the last ride at the last user that used the car
  - [R4] The nearest operator must take care of moving the car
  - [R5] The car must be moved to a power grid station according to ensure a uniform distribution of cars in the city

- [G20] Operators should receive a notification for charging a car
  - [R1] Operators must be logged
  - [R2] The system must send a notification to the nearest operator after some user leaves the car with more than 80% of the battery empty.
  - [R3] The system should charge 30% more on the last ride at the last user that used the car
  - [R4] The nearest operator must take care of re-charge the car on-site
  - [R5] When a car is in charge, as soon as its battery is full, it appear with the available cars.

3.2 Non-functional requirements

3.2.1 Visitor interface

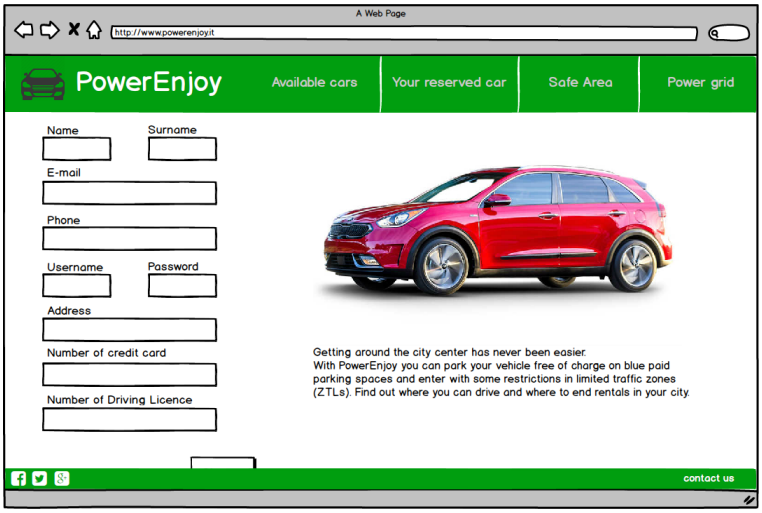


Figure 3.1: Web view for visitors

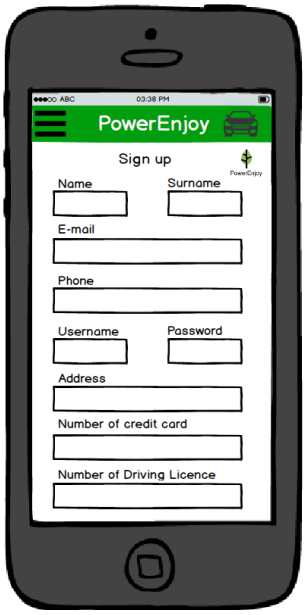
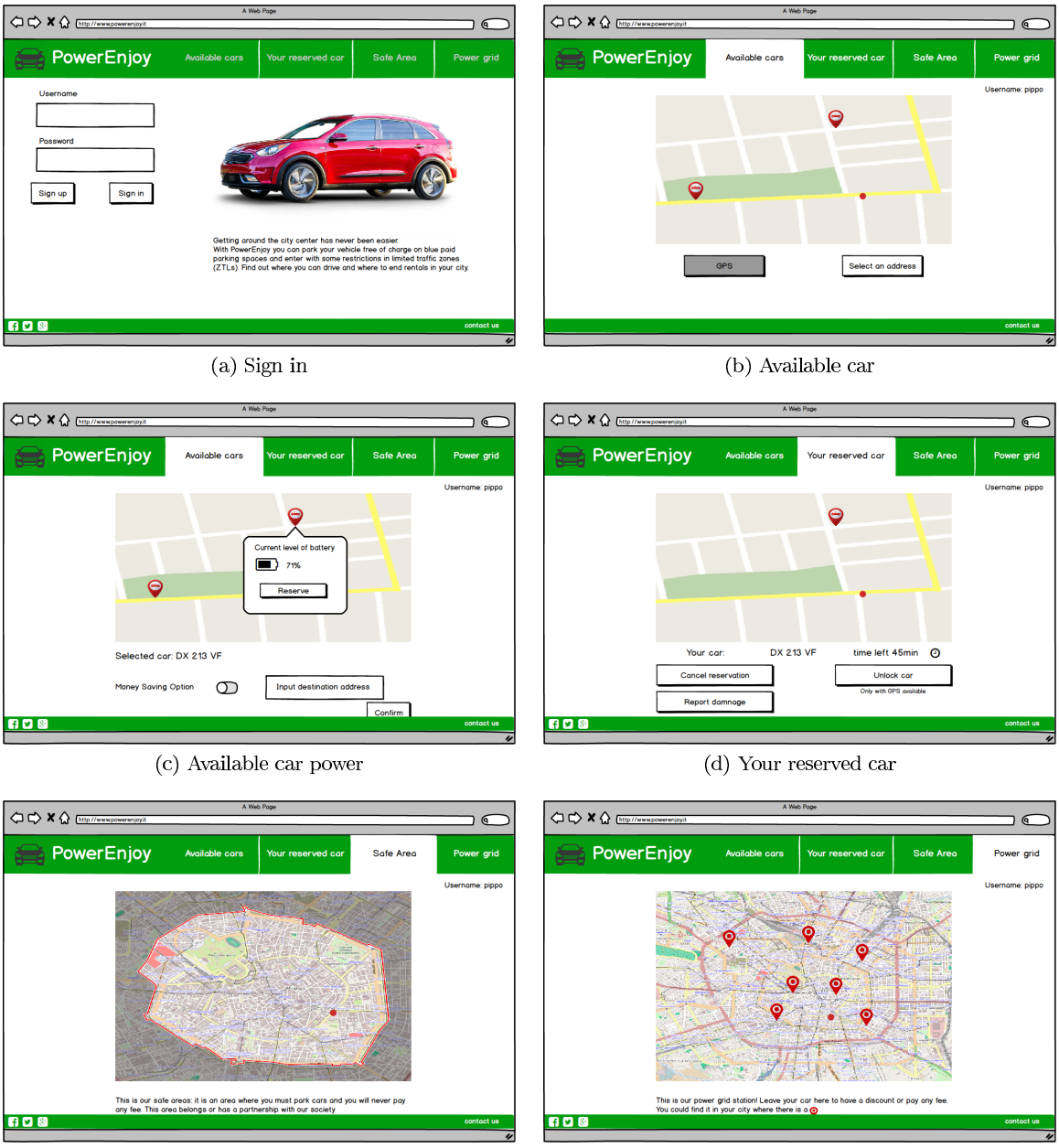


Figure 3.2: Mobile view for visitors

3.2.2 User interface



(a) Sign in

(b) Available car

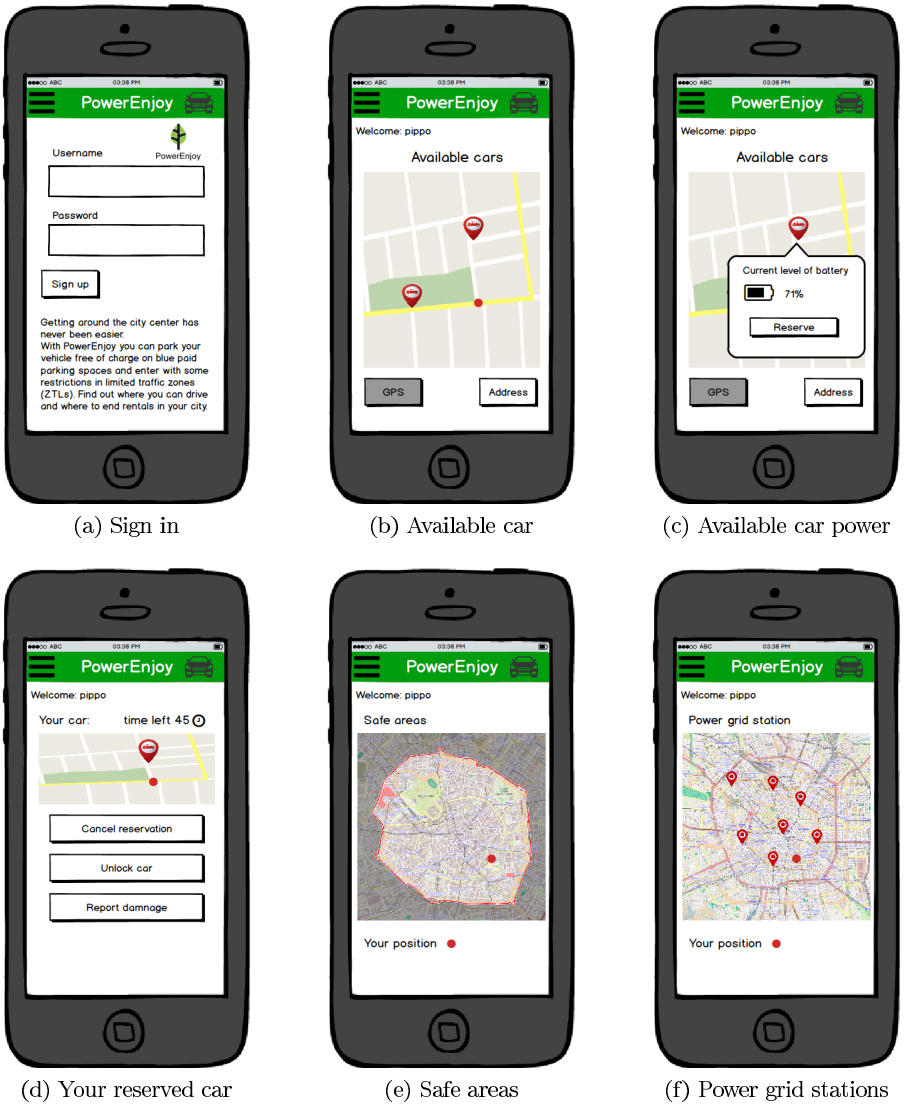
(c) Available car power

(d) Your reserved car

(e) Safe areas

(f) Power grid stations

Figure 3.3: User web view

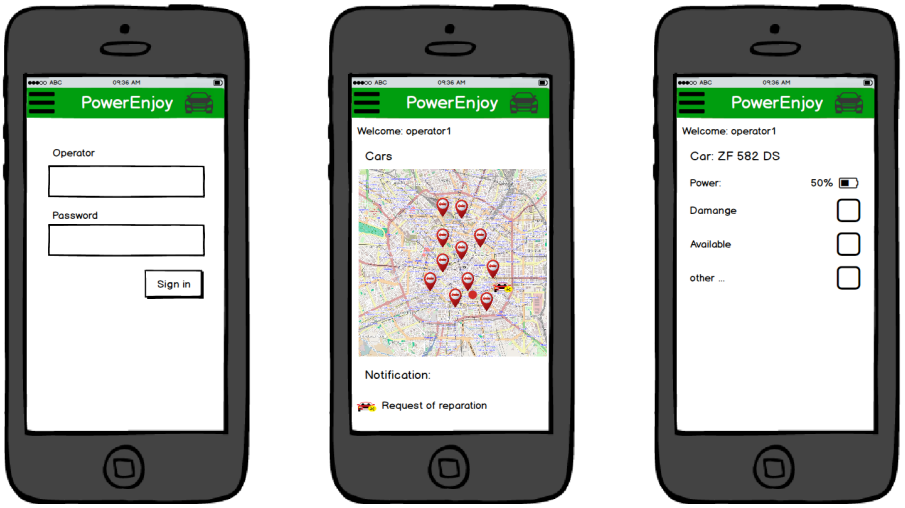


(a) Sign in (b) Available car (c) Available car power

(d) Your reserved car (e) Safe areas (f) Power grid stations

Figure 3.4: User mobile view

3.2.3 Operator interface



(a) Sign in (b) All cars (c) Single car

(d) Report operation done

Figure 3.5: Client mobile view

4

Scenario identifying

Here some possible scenarios of usage of this service

4.1 Scenario 1

Bill is coming back home and he finds a PowerEnjoy car near his home. He get some information about the service and he decides to sign up himself and inserts him credentials. After registration process, he receives an email with the password that will use for the login. Therefore Bill login into the system with the right credentials, enables the gps and reserves the car that he saw before. Then Bill goes to the car, unlocks it and joins his friends in city centre.

4.2 Scenario 2

Paolo is going shopping, when he arrives at the supermarket he sees a PowerEnjoy car. Paolo is already registered at the service, so he login in the app, enables the gps and reserve the car for the return trip. Meanwhile Paolo goes shopping but he has to queued up at the till. One hour from the reservation has already passed when Paolo exits the supermarket, so he has to pay a fee of 1 EUR and the system tags the car as available again and the reservation expires. Paolo has to look for another car.

4.3 Scenario 3

Riccardo, after login into the system, reserved a car near his home. He goes to the reserved car but he finds out that it's damage! Riccardo has to notify the damage (from the previous user) at the system through the application. The system will notify the damage car at the nearest operator and will let disappear the car from the list of available car. Riccardo has to look for another car

4.4 Scenario 4

Nicola goes out to dinner with friends of him. All Nicola's friends live in the same neighborhood, that it's some bus station far away from the restaurant. Instead Nicola lives in another district, so he decides to go with friends till their home and then reserve a PowerEnjoy car there. Therefore Nicola login into the system through the application, insert the position of his friends home and reserves a car. Nicola has to reach the car within one hour from the reservation.

4.5 Scenario 5

Filippo has to reserve a PowerEnjoy car for a quite long trip inside the city. He login into the system with his credentials through the web site and he finds out that there is an availablecar in proximity. But when he sees the law battery level, then he decides to reserve another car.

4.6 Scenario 6

Maria Chiara and her friends goes out in the night. It's late and public transportations are infrequents. So, Maria Chiara and her friends decide to rent a PowerEnjoy car to come back home. Since they are more than two passengers, system sensor detect them and the system will applies a 10% of discount on the last Maria Chiara's ride.

4.7 Scenario 7

Matteo has to go to the stadium to see a football match. He wants to go in a cheap way. So Matteo login into the system through the PowerEnjoy app and reserves a car. He reach the car and as soon as he engine ignites he selects on the car screen the money saving option and key the final address destination. The system provides informations about the station where to leave the car to get discount.

4.8 Scenario 8

Sara has to come back home and she lives at more than 3 km from the nearest power grid station. Sara uses PowerEnjoy to come back home and she park near her home. The system will charge 30% more on her last ride.

4.9 Scenario 9

Francesco rent a PowerEnjoy car to go to work. He's late, so Francesco parks the car with more than 80% of the battery empty because he has no time to

put in charge the car. The system will charge 30% more on his last ride to compensate for the cost required to re-charge the car on site by an operator.

4.10 Scenario 10

Stefania is at the University and when she has to come back home, starts rain. She decides to rent a PowerEnjoy car. When she arrives at her home, she parks the car with more than 50% of battery. For this reason, the system will apply a discount of 20% on her last ride.

4.11 Scenario 11

Marco has to go to visit a friend and he decides to go with PowerEnjoy service. Marco left the car at special parking areas and he takes care of plugging the car into the power grid. The system will apply a discount of 30% on his last ride.

4.12 Scenario 12

Luca is an operator. When he starts his work schedule he logs the system and looks if there are any cars that need reparations. At this moment he receives a notification from the system that there is a car which need to be recharged. Luca reaches the car, then he unlocks the car and charges it.

5

UML models

5.1 Use case diagram

5.1.1 Visitor

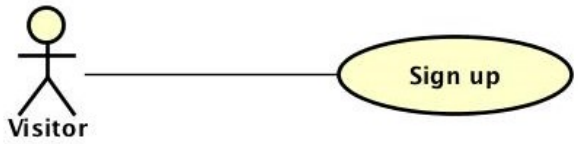


Figure 5.1: Visitor use case diagram

5.1.2 User

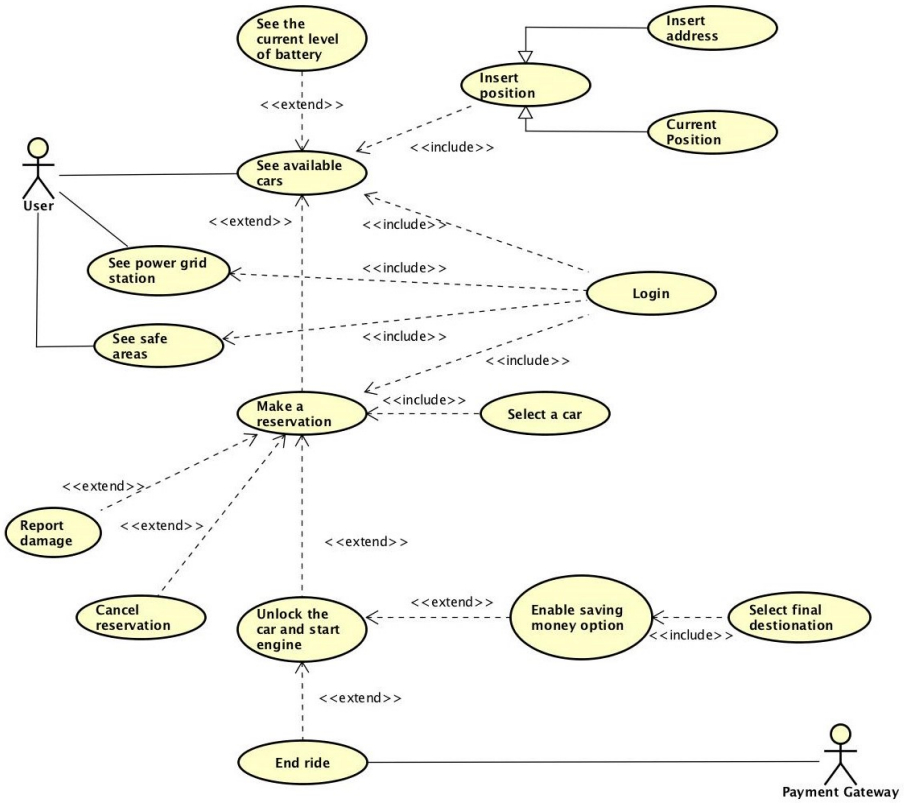


Figure 5.2: User use case diagram

5.1.3 Operator

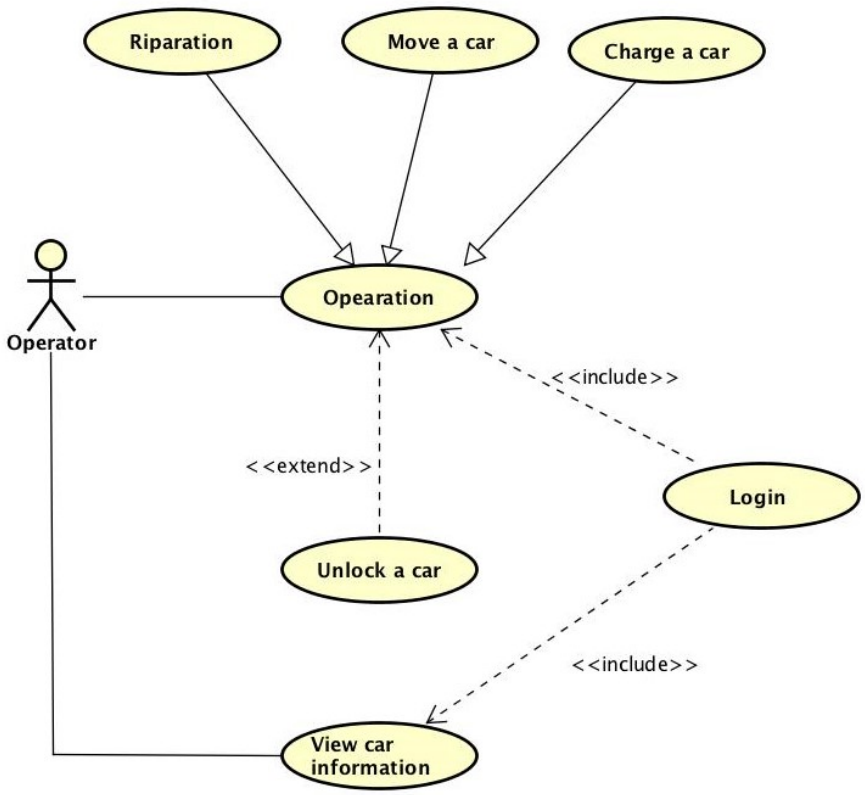


Figure 5.3: Operator use case diagram

5.2 Use case description

5.2.1 Visitor

Sign up	
Actor	Visitor
Goal	G1
Input condition	No input condition
Event flow	1. Visitor go on the webpage and click on "sign up". 2. Visitor fill in at least all the mandatory fields. 3. Visitor click on the "Confirm" button.
Output condition	Registration successfully done
Exception	1. Visitor has already an account. 2. At least one field is not correct.

5.2.2 User

Login	
Actor	User
Goal	G2
Input condition	No input condition
Event flow	1. User goes on the webpage and clicks on login. 2. User inserts email and password. 3. User clicks on "Sign in" button.
Output condition	The system verifies the credential and if correct redirect the user the personal page.
Exception	The credential are not correct and the system shows an error message.

Search a car	
Actor	User
Goal	G3
Input condition	User must be logged.
Event flow	1. The system let the user choose between searching available cars by inserting an address or by using the GPS position. 2. User must choose one of the two modality.
Output condition	The system shows a page in which there is a map with all the available car near the position chosen. At that point the user could see the current battery level of each car after clicks on a specific car.
Exception	1. User inserts an address that doesn't exist. 2. User decides to use the GPS position without enabling the GPS.

Report a damage	
Actor	User
Goal	G10
Input condition	User must have reserved a car.
Event flow	User reports a damage of the car that he/she has reserved.
Output condition	The system notifies the nearest operator the damage of the car. The system tags the car as not available.
Exception	No exception

Reserve a car	
Actor	User
Goal	G4
Input condition	User must be logged.
Event flow	1. Before select a car, user could filter available cars by position as descripted in "Search a car". In this case the system shows him/her the nearest cars. 2. User selects a car from the list of available car. 3. The system shows to the user all the information about the car like the battery level. 4. After that the user click on the "Confirm" button to make the reservation. 5. The system delete the car from the list of available cars. 6. The system start the timer of one hour.
Output condition	The user has made a reservation of the selected car.
Exception	If a car is not picked-up within one hour from the reservation the user has to pay a fee of 1€ and the car is tagged has available.

Cancel a reservation	
Actor	User
Goal	G11
Input condition	User must have reserved a car.
Event flow	1. User goes to the personal page in the section of reserved car. 2. User clicks on "Delete reservation". 3. The system deletes the reservation. 4. The system tags the car as available.
Output condition	The reservation has been deleted and the car is available.
Exception	No exception

Unlock a car and start a ride	
Actor	User
Goal	G5
Input condition	The user must have reserved a car.
Event flow	1. User go to the personal page in the section of reserved car. 2. If the car doesn't have any damage the user has to insert an identifier of the car and click on the "Unlock" button. 3. The system checks if the car that the user want to unlock has been reserved by the user. 4. If the code inserted is correct the system will unlock the car. 5. User starts a ride.
Output condition	The system starts counting the amount.
Exception	1. User reports a damage. 2. If the code inserted doesn't match the car reserved by the user the system doesn't unlocks the car.

End ride	
Actor	User
Goal	G13
Input condition	No input condition.
Event flow	1. User parks the car in a safe area. 2. The system stops counting the amount. 3. The system locks the car. 4. The system tags the car as available.
Output condition	The system, through the external payment gateway, charges the user with the total cost of the ride taking care of eventual discounts. If the system detects that the car is parked at more than 3km from the nearest power grid station, it sends a notification to the nearest operator and charges 30% more the user.
Exception	No exception

Enable money saving option	
Actor	User
Goal	G8
Input condition	User must have unlocked a car.
Event flow	1. User selects the money saving option. 2. The system shows a form for the destination of the ride. 3. User has to insert the final destination. 4. The system shows the station where the user has to leave the car to get a discount.
Output condition	No output condition
Exception	The destination address is not correct.

5.2.3 Operator

Login	
Actor	Operator
Goal	G14
Input condition	No input condition
Evelnt flow	1. Operator goes on the webpage and clicks on login. 2. Operator inserts email and password.
Oputput condition	The system verify the credential and if correct redirect the registered user the personal page.
Exception	The credential are not correct and the system notify it to operator and shows the login page.

Operation	
Actor	Operator
Goal	G16, G18
Input condition	The operator has to be logged
Event flow	1. The system sends a notification to the nearest operator with the operation that he has to do. 2. The operator as soon as receives the notification reaches the car and unlocks it for doing the operation. 3. As soon as the operator finishes the operation notifies the system that the operation has been done.
Output condition	The car is added to the available cars
Exception	No exception

5.3 Class diagram

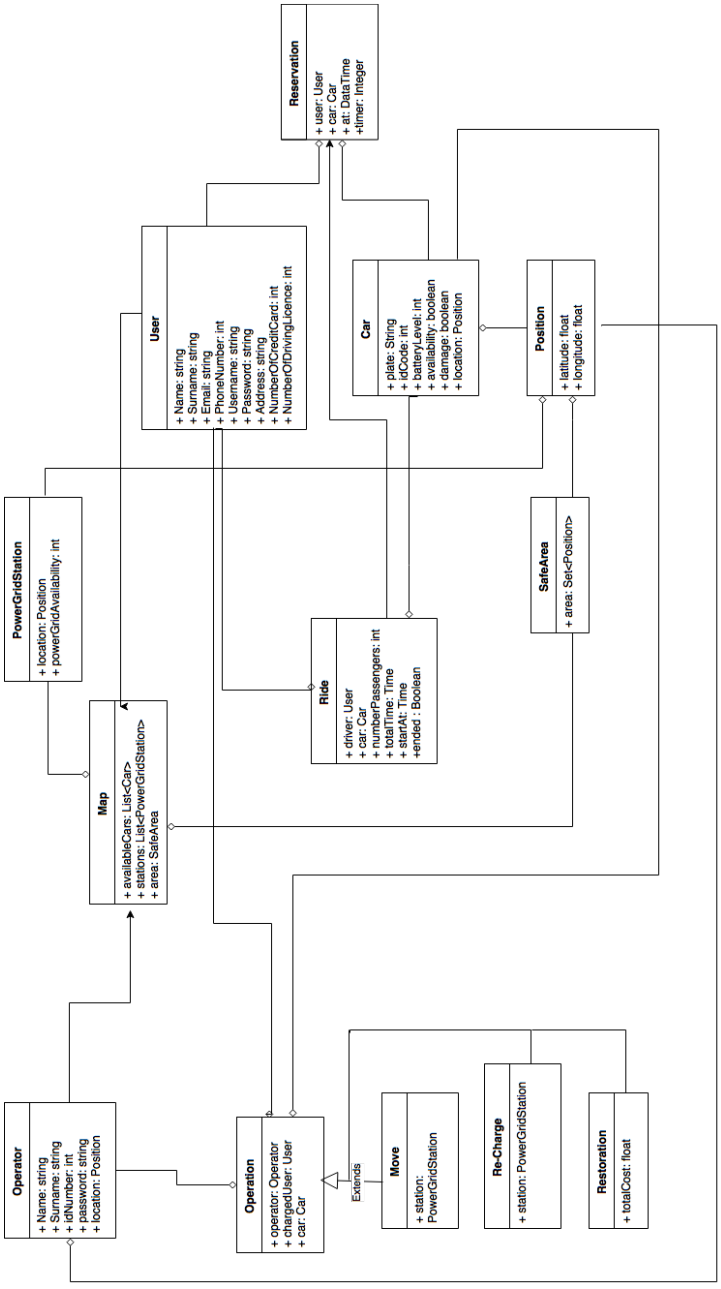


Figure 5.4: Class diagram

5.4 Sequence diagrams

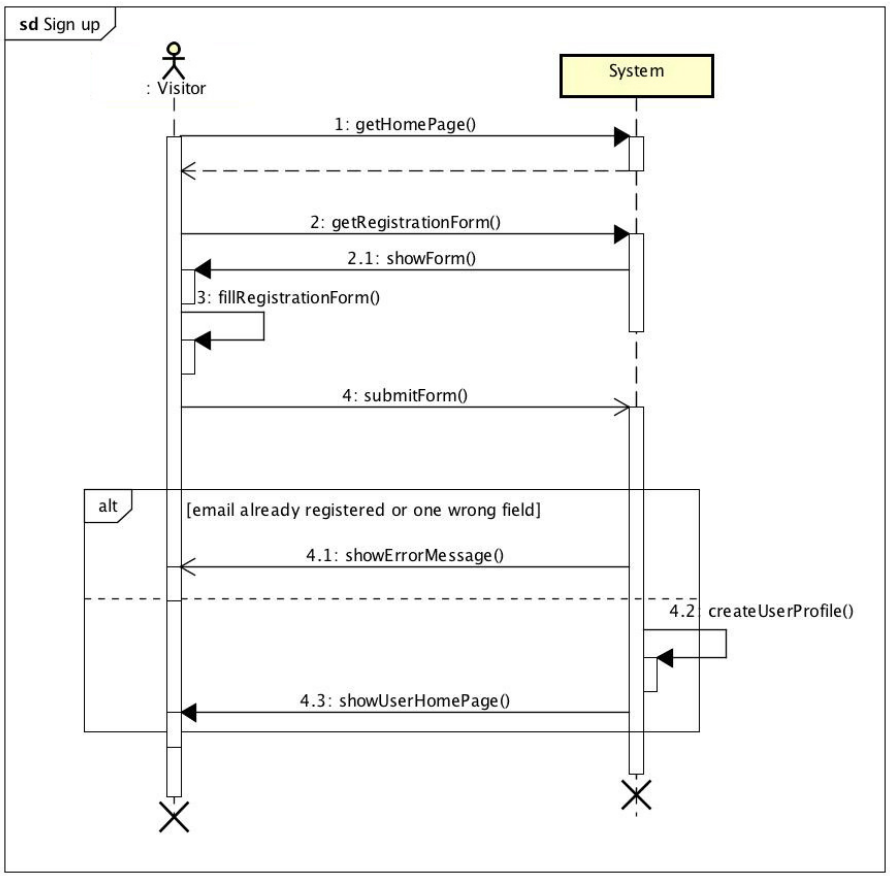


Figure 5.5: Sign up sequence diagrams

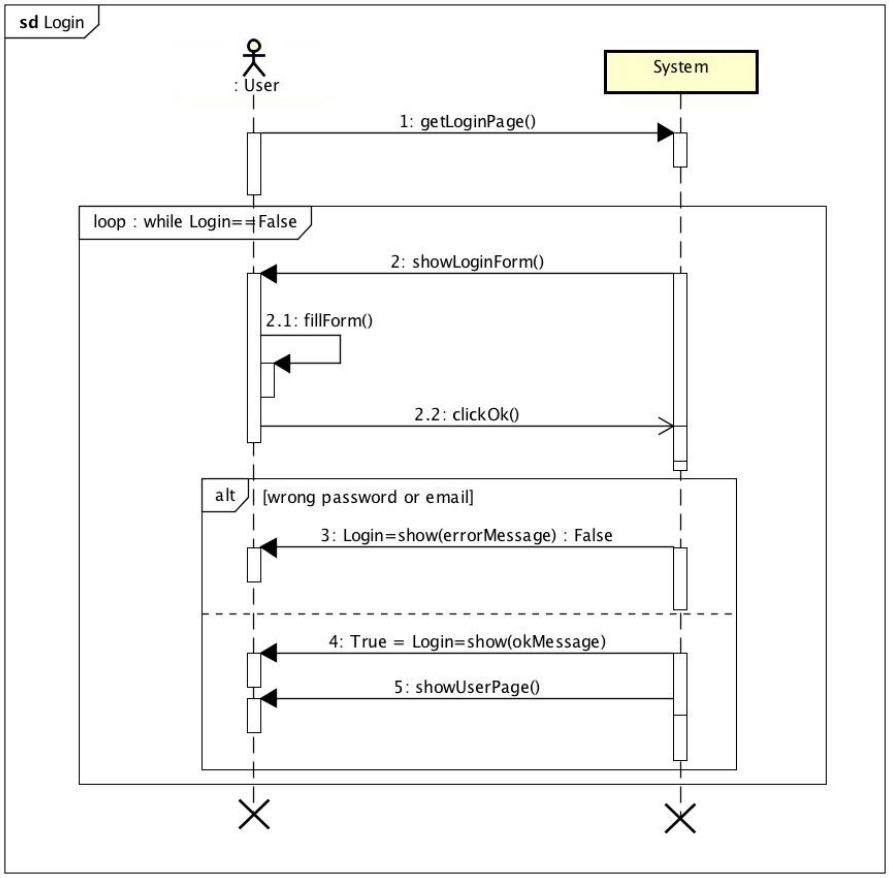


Figure 5.6: Login sequence diagrams

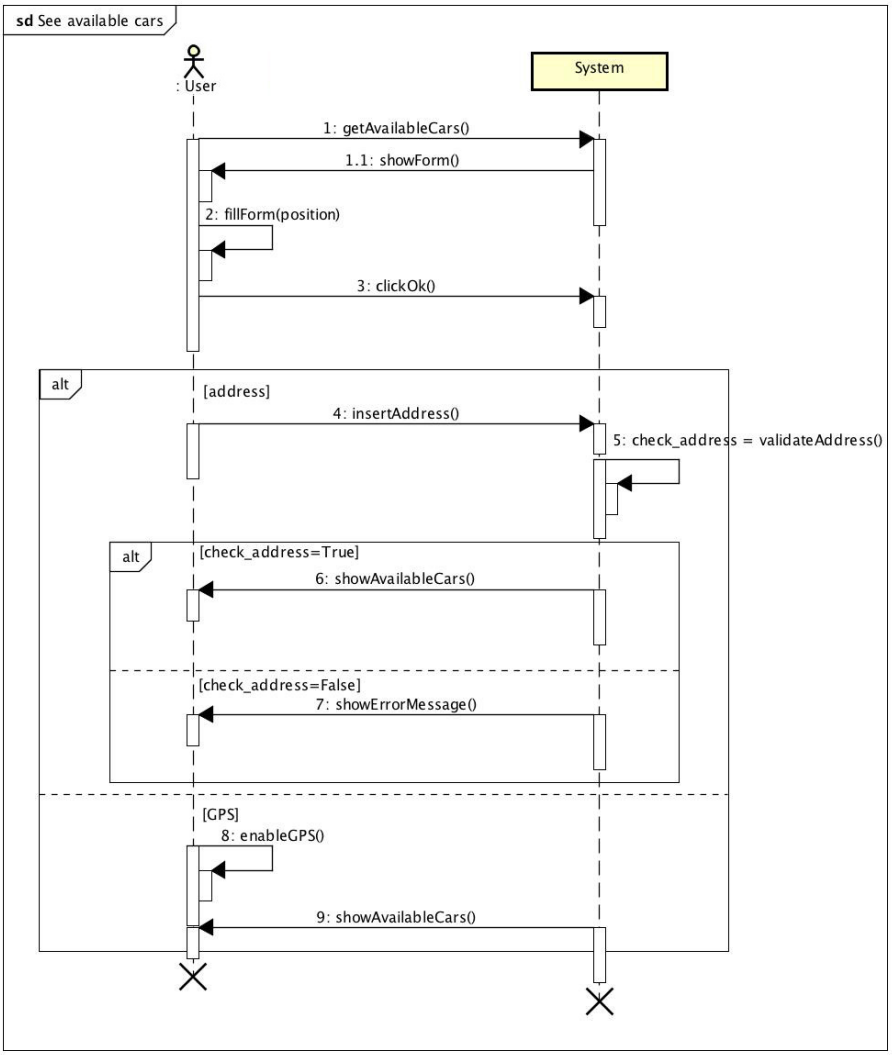


Figure 5.7: See available cars sequence diagrams

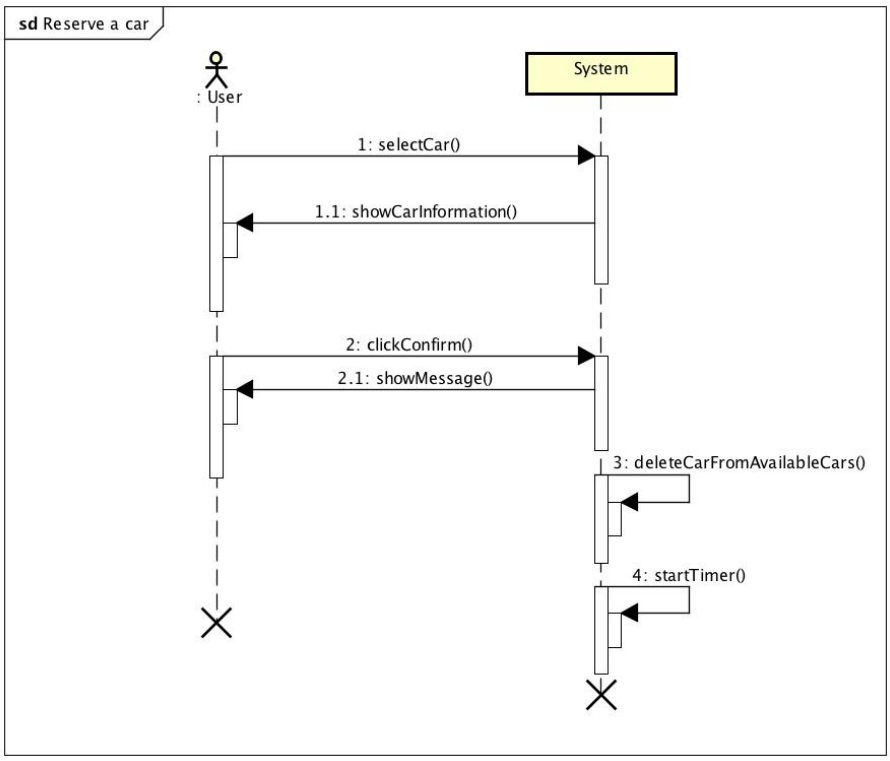


Figure 5.8: Reserve a car sequence diagrams

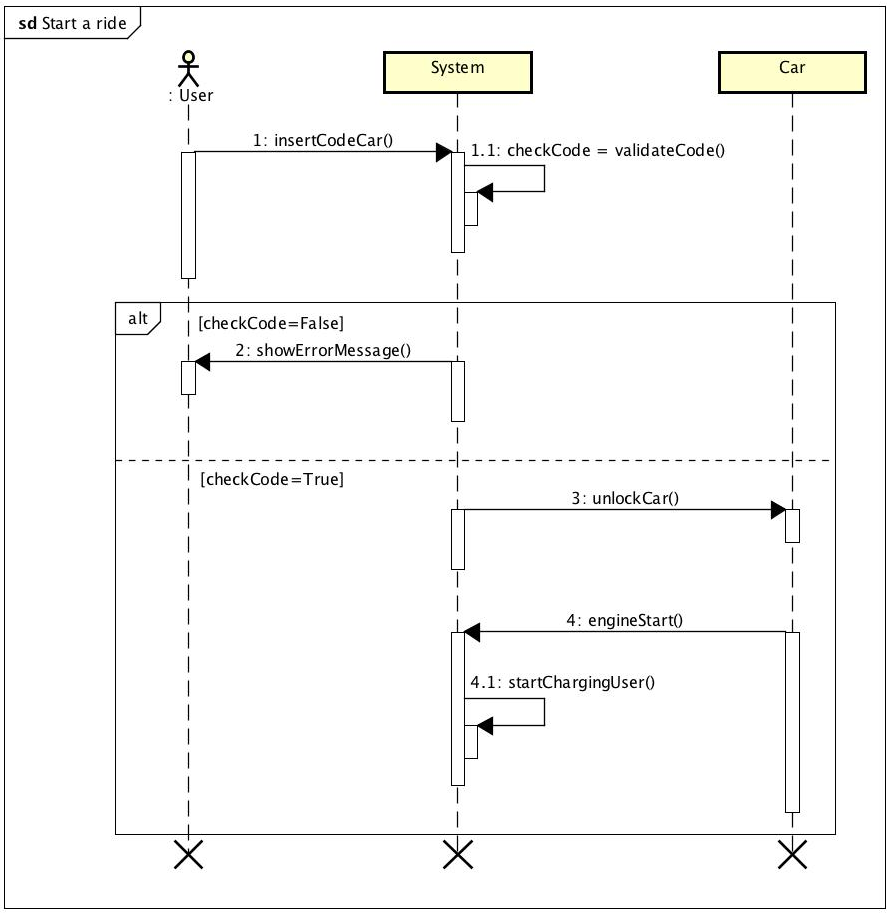


Figure 5.9: Start a ride sequence diagrams

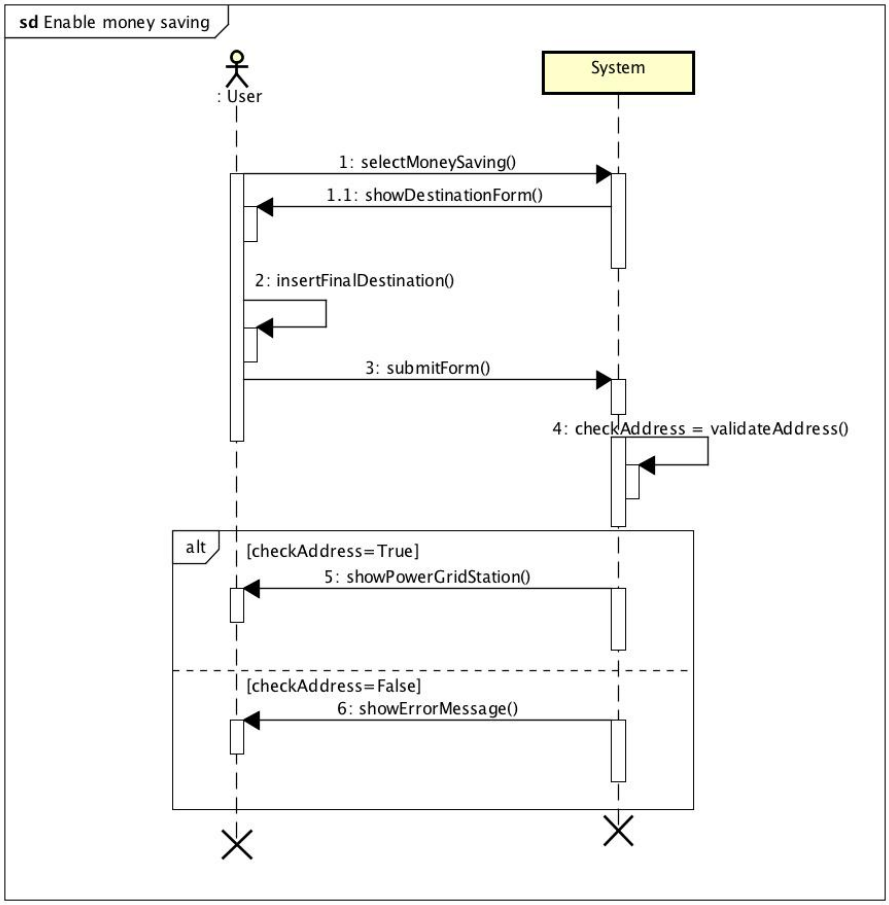


Figure 5.10: Enable money saving sequence diagrams

5.5 Activity diagrams

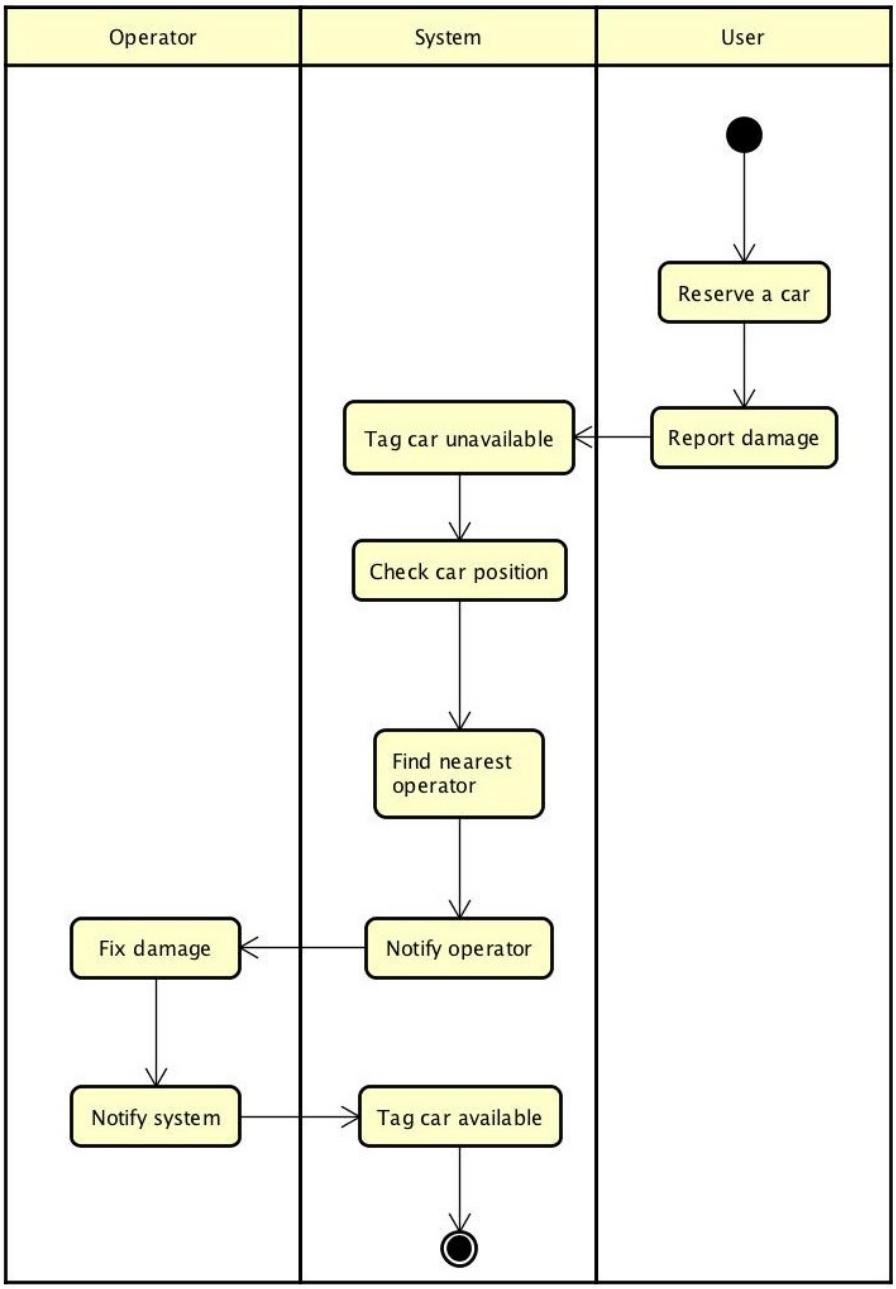


Figure 5.11: Report a damnage activity diagram

5.6 State chart diagram

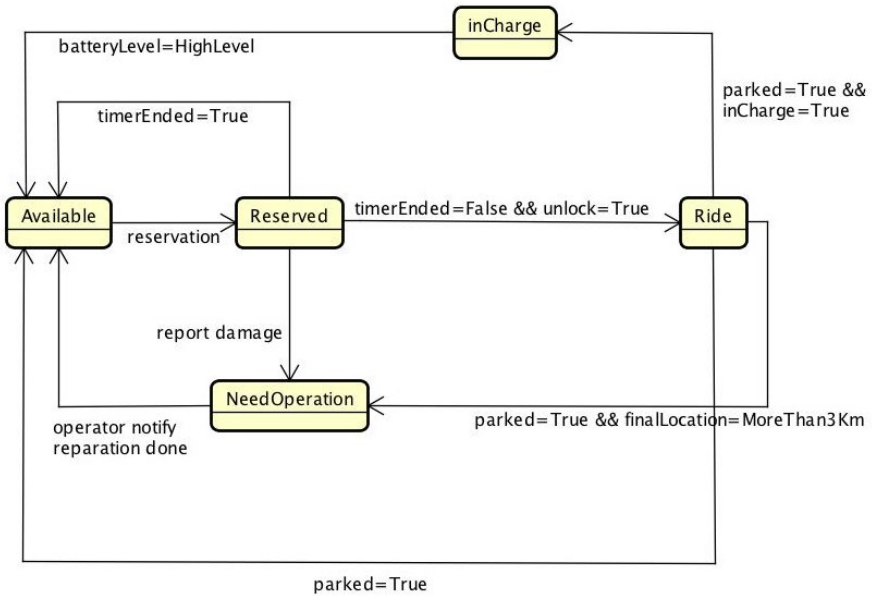


Figure 5.12: Car state chart diagram

6

Alloy modeling

6.1 Model

```
open util/boolean

//SIGNATURES

sig Code{}

sig Position{}

sig User{}

abstract sig BatteryLevel{}

one sig LowLevel extends BatteryLevel{} // battery level between 0% and 20%
one sig MediumLevel extends BatteryLevel{} // battery level between 21% and 99%
one sig HighLevel extends BatteryLevel{} // battery level equal to 100%
one sig MaxLevel extends BatteryLevel{} // battery level equal to 100%

sig Car{
   code: Code,
   position: Position,
   available: Bool,
   batteryLevel: BatteryLevel,
   inCharge: Bool
}{
   inCharge=True => batteryLevel!=MaxLevel
}

sig Ride{
   reservation: Reservation,
   driver: User,
   car: Car,
   numberPassenger: Int,
   ended: Bool,
   moreThan3km: Bool,
   inCharge: Bool,
   endBatteryLevel: BatteryLevel
}
{
   inCharge=True => moreThan3km=False
   moreThan3km=True =>  inCharge=False
   numberPassenger>=0 and numberPassenger<=4
}
```

```
sig Reservation{
    car: Car,
    user: User,
    timerEnded: Bool,
    deleted: Bool
}

sig Availability{}

sig PowerGridStation{
    position: Position,
    cars: set Car,
    availability: Availability
}

one sig SafeArea{
    area: set Position
}

sig Operator{}

sig Operation{
    operator: Operator,
    car: Car,
    ended: Bool
}

//FACTS

//cars in a power grid station are in charge or have just been charged
fact carInAPowerGridStation{
    all p:PowerGridStation | all c:Car | c in p.cars => ((c.inCharge=True or (c.inCharge=False and c.batteryLevel=MaxLevel)))
}

//one car could be at most in one power grid station
fact carInPowerGridStation{
    all c:Car | lone p:PowerGridStation | c in p.cars
}
```

```
//available cars must not be involved in an operation or in a ride or in charge
fact availableCars{
   all c:Car | c.available=True =>
         c.inCharge=False and (no o: Operation | c=o.car and o.ended=False)
            and (no r: Ride | c= r.car and r.ended=False)
}

//unavailable cars
fact unavailableCars{
   all c:Car| c.available=False =>
      (c.inCharge=True and ((no o: Operation | c=o.car and o.ended=False) and (no r: Ride | c= r.car and r.ended=False))) or
      ((one o: Operation | c=o.car and o.ended=False) and (c.inCharge=False and (no r: Ride | c= r.car and r.ended=False))) or
      ((one r: Ride | c= r.car and r.ended=False) and (c.inCharge=False and (no o: Operation | c=o.car and o.ended=False)))
}


//available cars must have the battery level that isn't low
fact batteryLevelAvailableCars{
   all c: Car | (c.available=True) => c.batteryLevel!=LowLevel
}

//available cars must be in a safe area
fact availableCarsInASafeArea{
   all c: Car | (c.available=True) =>
         ( some s: SafeArea | c.position in s.area)
}

// car codes are unique
fact uniqueCodes{
   all c1,c2: Car | (c1 != c2) => c1.code != c2.code
}

// two cars cannot have the same position
fact twoCarWithSamePosition{
   all c1,c2: Car | (c1!=c2) => c1.position!=c2.position
}

// if a car is in charge, it exists one power grid station that contains the car
fact carsInCharging{
   all c: Car | c.inCharge=True =>
         (one p:PowerGridStation | c in p.cars)
}
```

```
//for each user exists at most one ride that is not ended
fact oneRideNotEnded{
   all u:User | lone r:Ride | r.driver=u and r.ended=False
}

//each reservation has at most one ride
fact reservationOneRide{
   all r:Reservation | lone r1:Ride | r1.reservation=r
}

//each deleted reservation has not a ride and his timer is not ended
fact deletedReservation{
   all r: Reservation | (r.timerEnded=False and
            (no r1: Ride | r=r1.reservation)) <=> r.deleted=True
}

//each ride has a reservation whose timer is not ended
fact {
   all r: Ride | r.reservation.timerEnded=False
}

// for each (ride, reservation) car and user must be the same
fact sameCarUser{
   all r: Ride | r.driver=r.reservation.user  and r.car=r.reservation.car
}

//each power grid station is in a safe area
fact powerGridStationInASafeArea{
   all p:PowerGridStation |
         one s:SafeArea | p.position in s.area
}

//two power grid stations cannot have the same position
fact twoPowerGridStationWithSamePosition{
   all p1,p2: PowerGridStation | (p1!=p2) => p1.position!=p2.position
}

//no operator without operation
fact NoOperationWithoutOperator{
   all o:Operator | one o2:Operation | o2.operator=o
}
```

```
//no availability without power grid station
fact NoAvailabilityWithoutPowerStation{
    all a:Availability| one  p:PowerGridStation  | a=p.availability
}

//no user without reservation
fact noUserWithoutReservation{
    all u:User | some r:Reservation | r.user=u
}

//each operator could have at most one operation that is not ended
fact oneOperationNotEnded{
    all o:Operator | lone o1:Operation | o1.operator=o and o1.ended=False
}

//each not ended operation must have different operator and different car
fact differentOperatorAndCars{
    no o1,o2:Operation | o1.ended=False and o2.ended=False and o1!=o2 and o1.operator=o2.operator and o1.car=o2.car
}

// one car could have at most one operation that is not ended
fact oneCareOneOperationNotEnded{
    all c:Car | lone o: Operation | o.car=c and o.ended=False
}

//cars in a power grid station are in a safe are
fact carPowerGridSafeArea{
    all c:Car | all p:PowerGridStation | c in p.cars => (one s:SafeArea | c.position in s.area)
}

//ASSERTION

//check if different ride not already ended has different car
assert DifferentCarsForReservation{
    no r1,r2:Ride | ( r1.ended=False and r2.ended=False and r1.car=r2.car and r1!=r2)
}

check DifferentCarsForReservation

//check if different operation not already ended has different operator
assert DifferentOperationForOperator{
    no o1,o2:Operation | (o1!=o2 and o1.ended=False and o2.ended=False and o1.operator=o2.operator)
}

check DifferentOperationForOperator
```

```
//check if different operation not already ended has different cars
assert DifferentCarsForOperation{
   no o1,o2:Operation | (o1!=o2 and o1.ended=False and o2.ended=False and o1.car=o2.car)
}

check DifferentCarsForOperation

// check if all available cars are in the safe area
assert allCarInSafeArea{
   no c:Car | c.available=True and one s:SafeArea | c.position not in s.area
}
check allCarInSafeArea

//check that eac car has a unique code
assert noSameCode{
   no c1,c2:Car | c1!=c2 and c1.code=c2.code
}
check noSameCode

//check that doesn't exists a ride if the associated reservation is deleted
assert NoRideWithDeletedReservation{
   no r:Ride | r.reservation.deleted=True
}
check NoRideWithDeletedReservation


//PREDICATES

// user reserved a car but the timer is ended, and so make another reservation
pred timerEndedAndRide{
   one u:User | some r:Reservation | u=r.user and r.timerEnded=True and (one r1:Ride| r1.driver=u )
   #Ride=1
   #Reservation=2
   #User=1
   #Operator=0
}

run timerEndedAndRide



//user deletes a reservation
pred deleteReservation{
   one u:User | one r:Reservation | r.user=u and r.deleted=True
   #Ride=1
   #Reservation=2
   #User=1
   #Operator=0
}

run deleteReservation

//when users can have 30% of discount
pred  ThirtyPercentDiscount{
   one r:Ride | (r.ended=True and r.inCharge=True)
   #User=1
   #Ride=1
}

run ThirtyPercentDiscount

//when users can have 20% of discount
pred  TwentyPercentDiscount{
   one r:Ride | (r.ended=True and (r.car.batteryLevel=HighLevel or r.car.batteryLevel=MaxLevel)
            and r.inCharge=False)
   #User=1
   #Ride=1
}

run TwentyPercentDiscount

//when users can have 10% of discount
pred  TenPercentDiscount{
   all r:Ride | (r.ended=True and r.numberPassenger>=2 and
   (r.endBatteryLevel!=HighLevel and r.endBatteryLevel!=MaxLevel)
            and  r.inCharge=False)
   #User=1
   #Ride=1
}
```

```
//when users can have 10% of discount
pred  TenPercentDiscount{
   all r:Ride | (r.ended=True and r.numberPassenger>=2 and
   (r.endBatteryLevel!=HighLevel and r.endBatteryLevel!=MaxLevel)
              and  r.inCharge=False)
   #User=1
   #Ride=1
}

//run TenPercentDiscount

pred ThirtyPercentFee{
   all r:Ride | (r.ended=True and (r.endBatteryLevel=LowLevel or r.moreThan3km=True))
   #User=1
   #Ride=1
}

//run ThirtyPercentFee

pred show{

}

//run show
```

6.2 Alloy result

13 commands were executed. The results are:
#1: No counterexample found. DifferentCarsForReservation may be valid.
#2: No counterexample found. DifferentOperationForOperator may be valid.
#3: No counterexample found. DifferentCarsForOperation may be valid.
#4: No counterexample found. allCarInSafeArea may be valid.
#5: No counterexample found. noSameCode may be valid.
#6: No counterexample found. NoRideWithDeletedReservation may be valid.
#7: Instance found. timerEndedAndRide is consistent.
#8: Instance found. deleteReservation is consistent.
#9: Instance found. ThirtyPercentDiscount is consistent.
#10: Instance found. TwentyPercentDiscount is consistent.
#11: Instance found. TenPercentDiscount is consistent.
#12: Instance found. ThirtyPercentFee is consistent.
#13: Instance found. show is consistent.

Figure 6.1: Alloy result

6.3 World generated

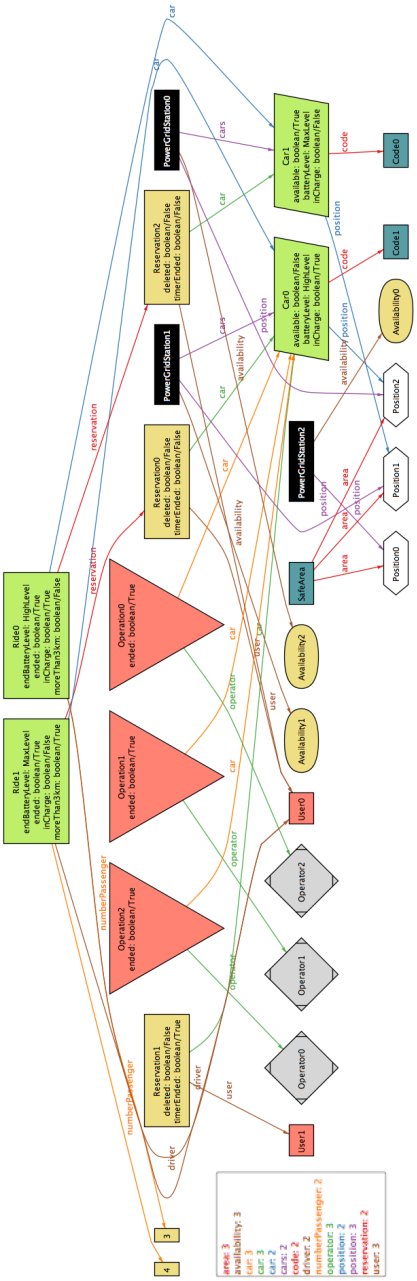


Figure 6.2: General world

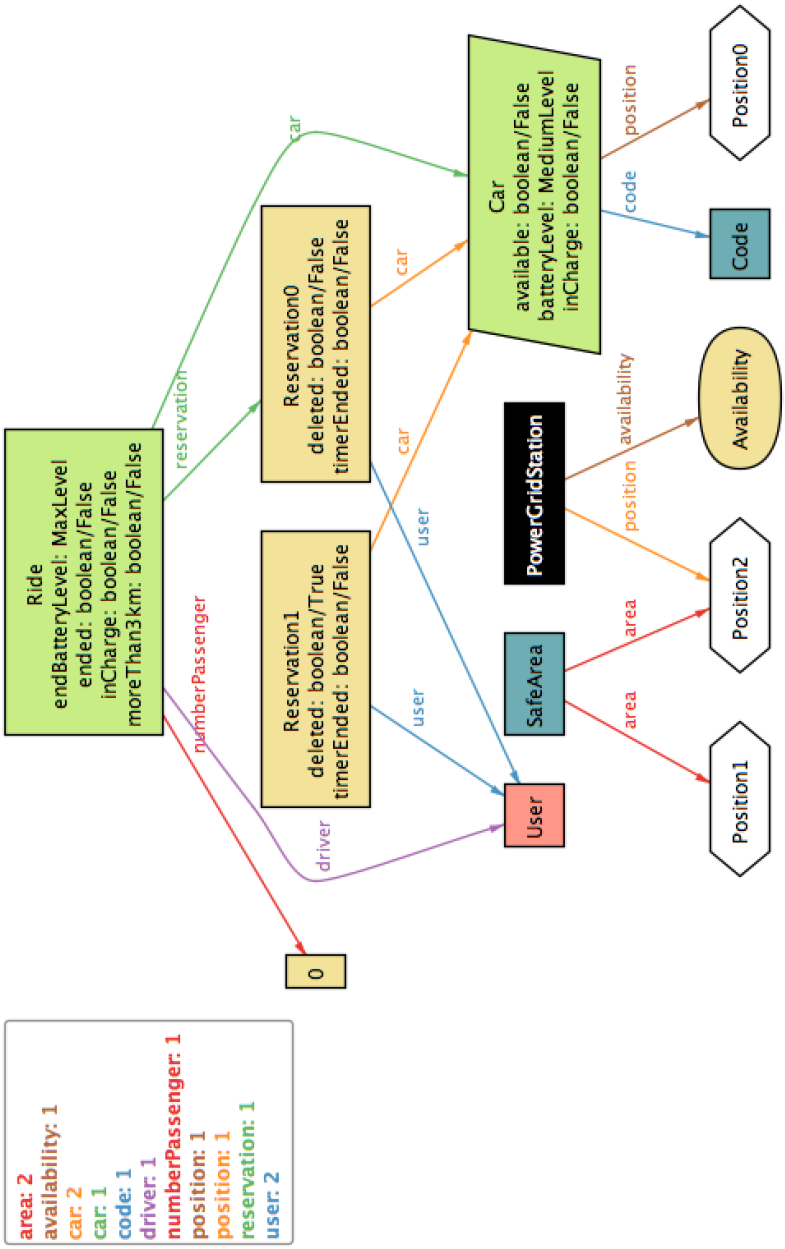


Figure 6.3: Deleted reservation

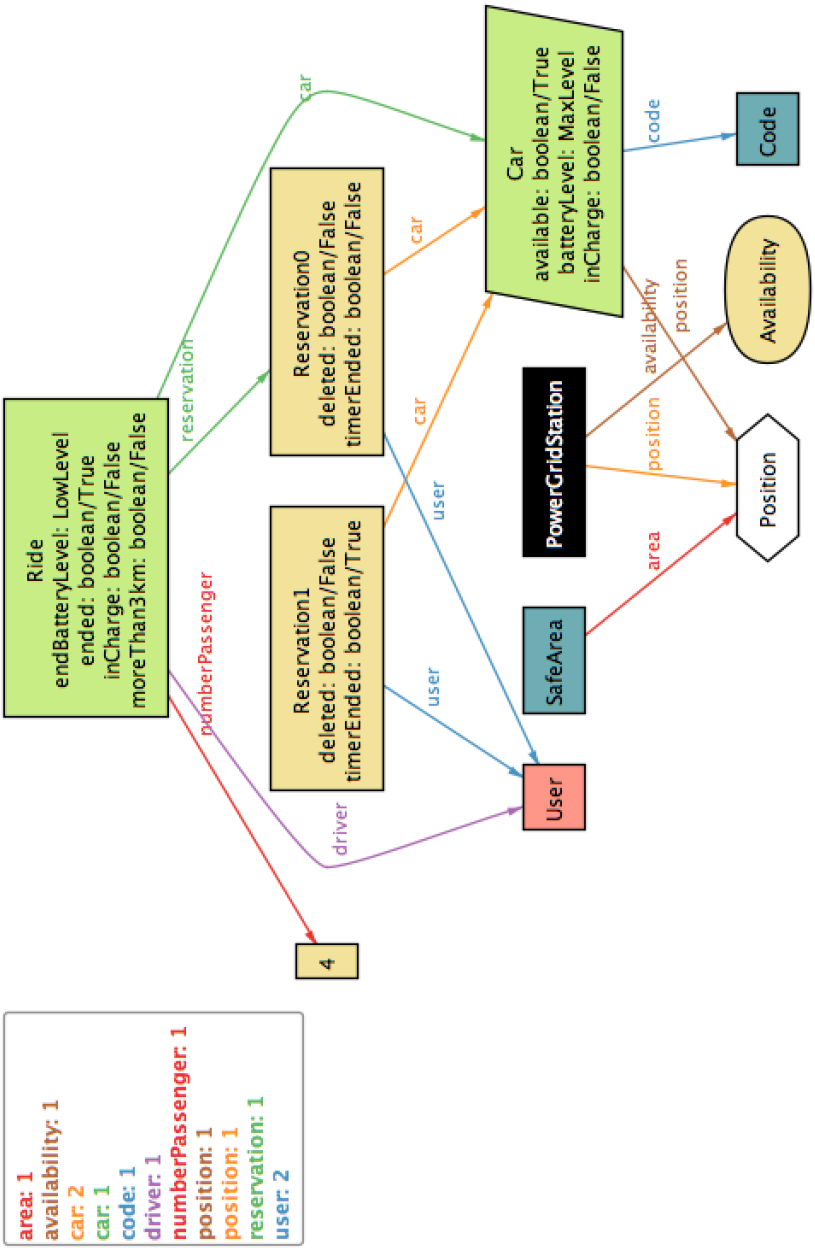


Figure 6.4: Time ended

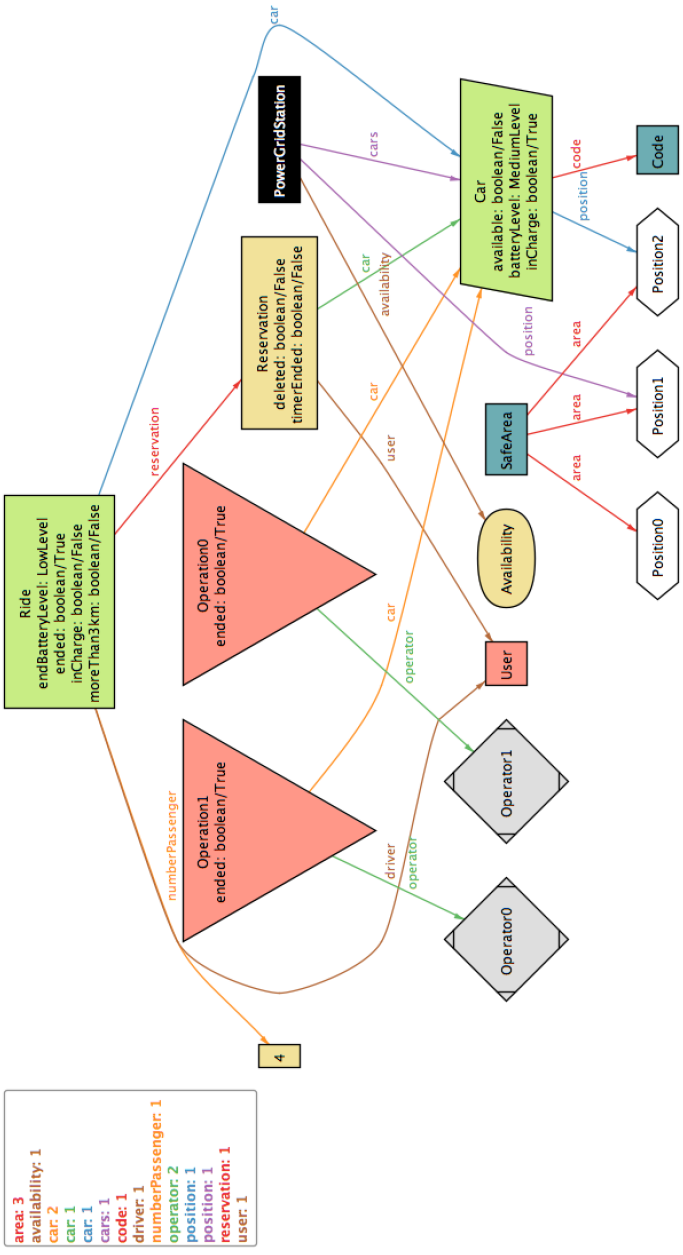


Figure 6.5: 10% discount

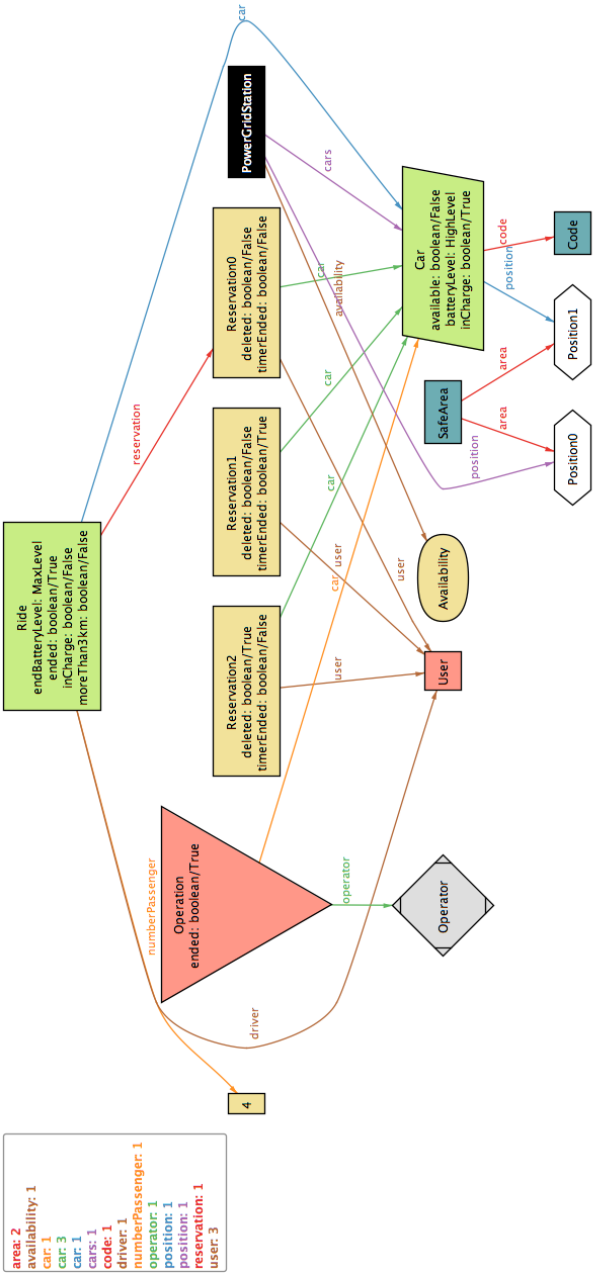


Figure 6.6: 20% discount

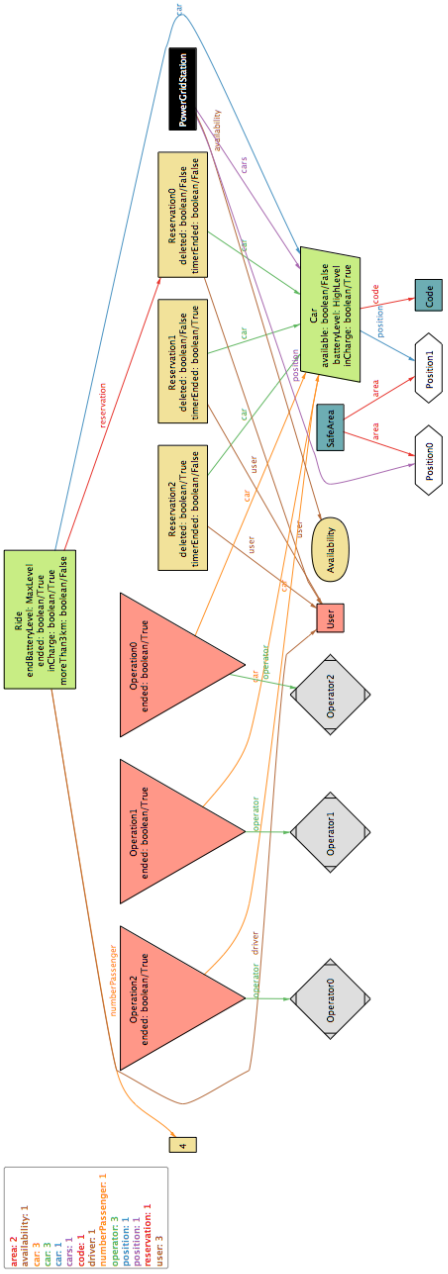


Figure 6.7: 30% discount

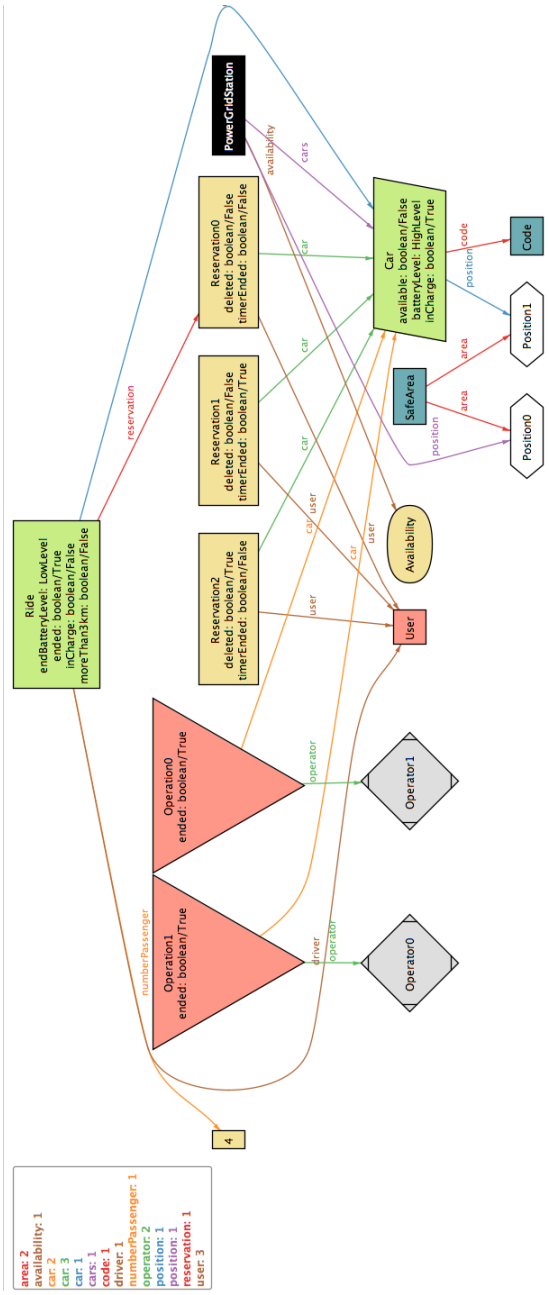


Figure 6.8: 30% fee

7

Used tools

The tools we used to create this RASD document are:

- Balsamiq: for mockup
- Draw.io (online): for Class Diagram
- Alloy Analizer 4.2: to prove the consistency of our model
- Github: for version controller
- Lyx: to format this document
- Astah: to create the Sequence Diagram and the Use Case Diagram

For redacting and writing this document we spent 30 hours per person