PowerEnJoy

Requirement Analysis and Specification Document

A.Y 2016/2017

Francesco Tinarelli (matr:806146)

Marco Wenzel (matr:878021)

Versione 1.2

Index

1 Introduction

1.1 General Description

We will project and implement PowerEnJoy, it's a digital management system for a car-sharing service.
This system is based on mobile application and web application, there are two kind of different people that can use the system: Unregistered user or Registered user.
Unregistered user can only visit the web application and register himself on the system.
The system allows registered user to see the available cars in a specific zone or current position area using GPS and through this service is possible to make a reservation for up to 1 hour.
The user can unlock the reserved car logging on the web application and using "unlock car" button, the system provides this operation if the user is nearby to the reserved car.
The system registered some special safe area called: power grid area which is possible plug the car.
The system during the ride show the current charges money on the car screen.
More over the screen show power grid stations and permits to choose the path for the nearest grid station or ones specified by the user.
In addition to the functionality above, the system should incentivize the virtuous behaviours of the users, in fact, if the user provides to plug the car or in generally take care of the car, the system rewards him with discount.
There is no system before of this one.

1.2 Actors

- **Unregistered user:** a person no registered on system, he has to register on the system if he wants use system services.
- **Registered user:** a person registered on system, he can use all services and the system has personal data of him.

1.3 Goals

Unregistered User:

[G1]: Allow to register on the system

Registered User:

[G2]: Allow to log in on the system.

[G3]: Allow to find the locations of available cars within a certain distance from their current position or from a specified address.
[G4]: Allow to reserve a single car for up to 1 hour.
[G5]: Allow to pay a fee if he misses the reservation.
[G6]: Allow to tell the system she's nearby.
[G7]: Allow to see the current charges.
[G8]: Allow to reach grid stations location.
[G9]: Allow to receive a discount if the user has at least two passengers.
[G10]: Allow to receive a discount if the user left the car with no more than 50% battery empty.
[G11]: Allow to receive a discount if the user left the car in a Power grid station and takes care of plugging the car.
[G12]: Allow to receive a fee if the user left the car more than 3km from the nearest power station or left the car with more than 80% of the battery empty.
[G13]: Allow to see an e-mail sent by the system with the bill ride and payment resume, at the end of this one.
[G14]: Allow to user to pay immediately after at the end of the ride.

1.4 Glossary

Reservation: it's the ability to reserve an available car for up to 1 hour, provided only for registered user.
For reserve a car the user must choose the available car from the map and click on it.
For use this service, user must be logged on web/mobile app.
Reservation Timer: this timer count how much time remain for open the car.
Unlock Car Button: it's a button on the mobile app that allow the system to check the user position and the reservation for unlock the car.
Safe Area: every zone where it's possible park the car safely.
Power Grid Area: special safe area where it's possible park the car safely and user takes care of plugging the car into power grid if he wants.
Plugging Time: the time that the user has to plug the car before the system create the bill.
Bill: Amount of cost that the user must pay for the ride, discount and/or fee included.
System: it's the new system that we want implement. The system interacts with user for providing services, External Agency for payment, cars for receive ride data.

A piece of system software is installed into each car, so in this way the system and the cars can communicate and send data.

External Agency: It's an agency that provides all type of payments service, when user must pay a fee or pays a ride. Interact with system for receive the amount to subtract on user bank account.

User: he is a client of the service. His personal data is already memorized into the database system (already registered on system), the personal data inserted:

- Name
- Surname
- E-mail
- Birth Date
- Driving License number
- Credit card number

Discount: it's a reward for good behaviour.

Ride: service provided by the system that allow to engine ignites and start to charging the user for a given amount of money per minute.

Fee: amount of money that the user pays in addiction, fee can be assigned through missed reservation or bad behaviours.

Car: an automobile, in each car we have a screen and a GPS for provide services. Car send to the system the data of the ride.

1.5 Reference Documents

For creating this RASD document, we used:

- Specification Document: Requirement Engineering III.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- Example document: MyTaxiService.pdf

1.6 Overview

The document is composed by 3 part:

1. **Introduction:** brief introduction, specific Goals and the glossary, in this way we have an overview of main services provided and some nomenclature.
2. **Overall Description:** Constraints, Domain and text assumption for specify the world that we want study.
3. **Specific Requirements:** analysis of Consistency about domain assumption requirements and goals.

2 Overall Description

2.1 Domain assumptions

- All the GPS always give the right position.
- The GPS of the cars cannot be switched off.
- A user doesn't make a reservation until he pays last ride of fee.
- A user drive only if has got driving license. ok
- There is a software that allow to recognize the right number of passengers.
- System battery car always show the right battery.
- Each car is registered on system.
- Every car is the same type.
- If a user parks in a not-safe area zone and take a fee, he will pay this fee.
- The payment is provided by an external agency.
- When the user finishes the ride and the plugging time is over, the external agency provides an immediate payment if the user has enough money for pay.
- If the user doesn't have enough money on the bank count, the external agency notifies the system and this one notifies user with an e-mail.
- Power grid area location is registered on the system.
- Car state could be only available or not-available
- There aren't pause during the ride.
- If a car is left with more 80% of battery empty, the car state 'll change into not-available state and a maintenance guy will go to the place for plugging the car.
- If a user takes more than 1 discount he takes only the biggest one.
- If a car is recharging, its state is not-available until the battery left is almost 70%.
- There is a car crash insurance for every car.
- The potential discounts will be counted only after plugging time.
- The plugging time is 3 minutes.
- A car is nearby if the distance between this one and the user is less than 3 meters.
- The amount of the payments is always right.
- If a user takes a discount and a fee, at the end of the ride system counted only the fee.

2.2 Product Perspective

User External Interfaces:

The product consists in a web and mobile application both based on web.
Web application should be providing register system; a user can register on the system only via web application.
Moreover, it's possible for registered user to see the payment history.
The mobile application should be use to check the user position and unlock the car using specific button.
In addition, the mobile application permits to reserve a car in the same way than web application.
So, the use of mobile application is mandatory if they want ride a car.
Moreover, in each car there is a screen that permits to find the grid station locations and reach them through specified address or the nearest one.
The screen permits also to see the current ride charge.

System External Interfaces:

The system interacts with an external agency through its external interface for providing the payment service.
At the end of the ride system receives the amount of money that user has to pay, it sends this amount to external agency, external agency tries to do the transaction and notifies the system if the transaction is successes or is failed (user doesn't have enough money in bank account).

2.3 Constraints

Regulatory policies

The system must require to user the permission to get his position and he has to manage sensible data (position, private data, credit card number) respecting the privacy law. Furthermore, the system must not use this data to send SPAM e-mail respecting the privacy law.

Hardware Constraints

Each car must have GPS and a screen to show charging ride.
Mobile application must be installed on a smartphone that has enough space for this app and almost 3G connection.

Parallel Operation

The server supports parallel operations from different users who use different cars and different power grid stations.

3 Specific Requirements

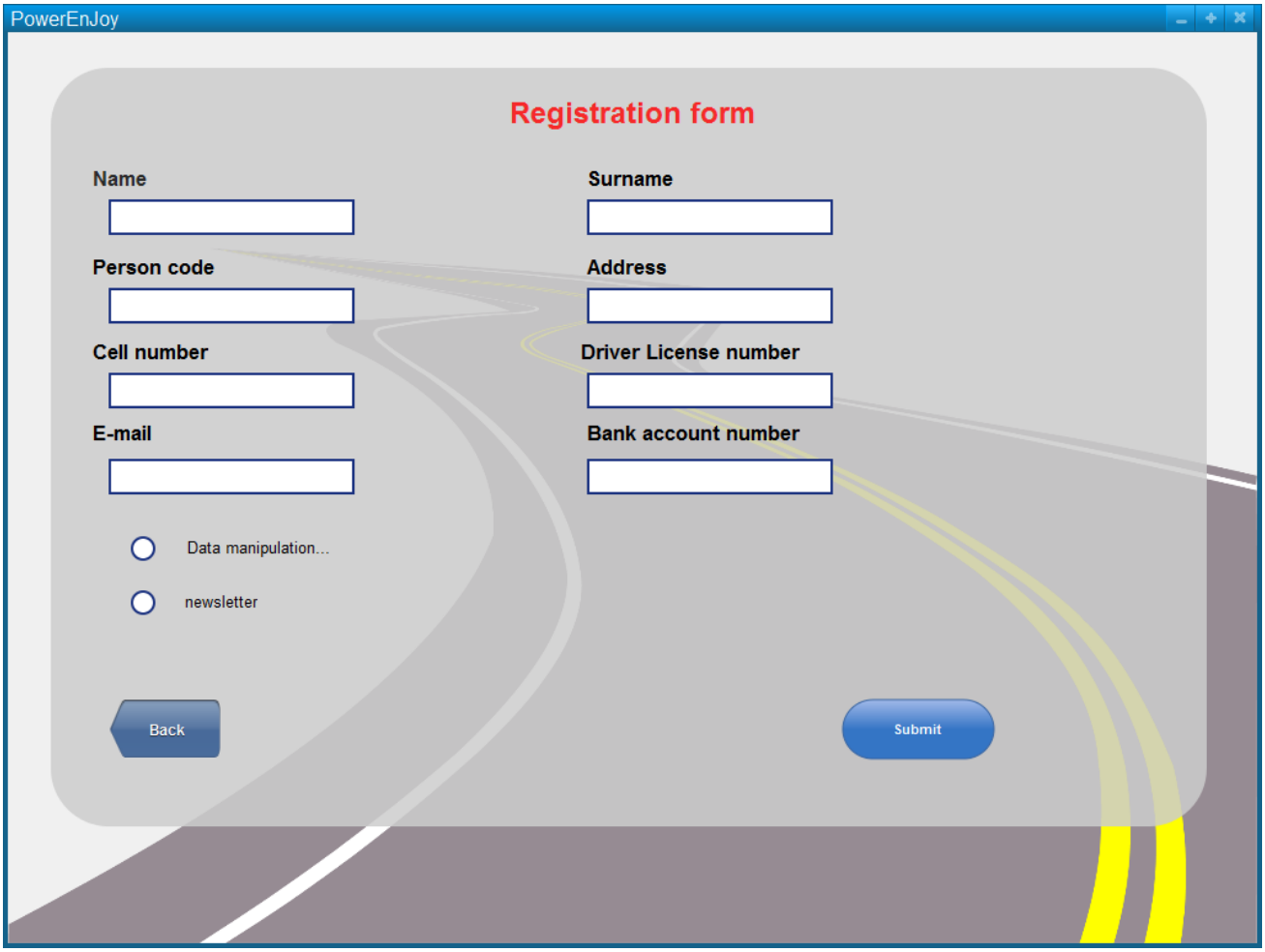
3.1 External interface requirements

User Web Application:

- **Log in**

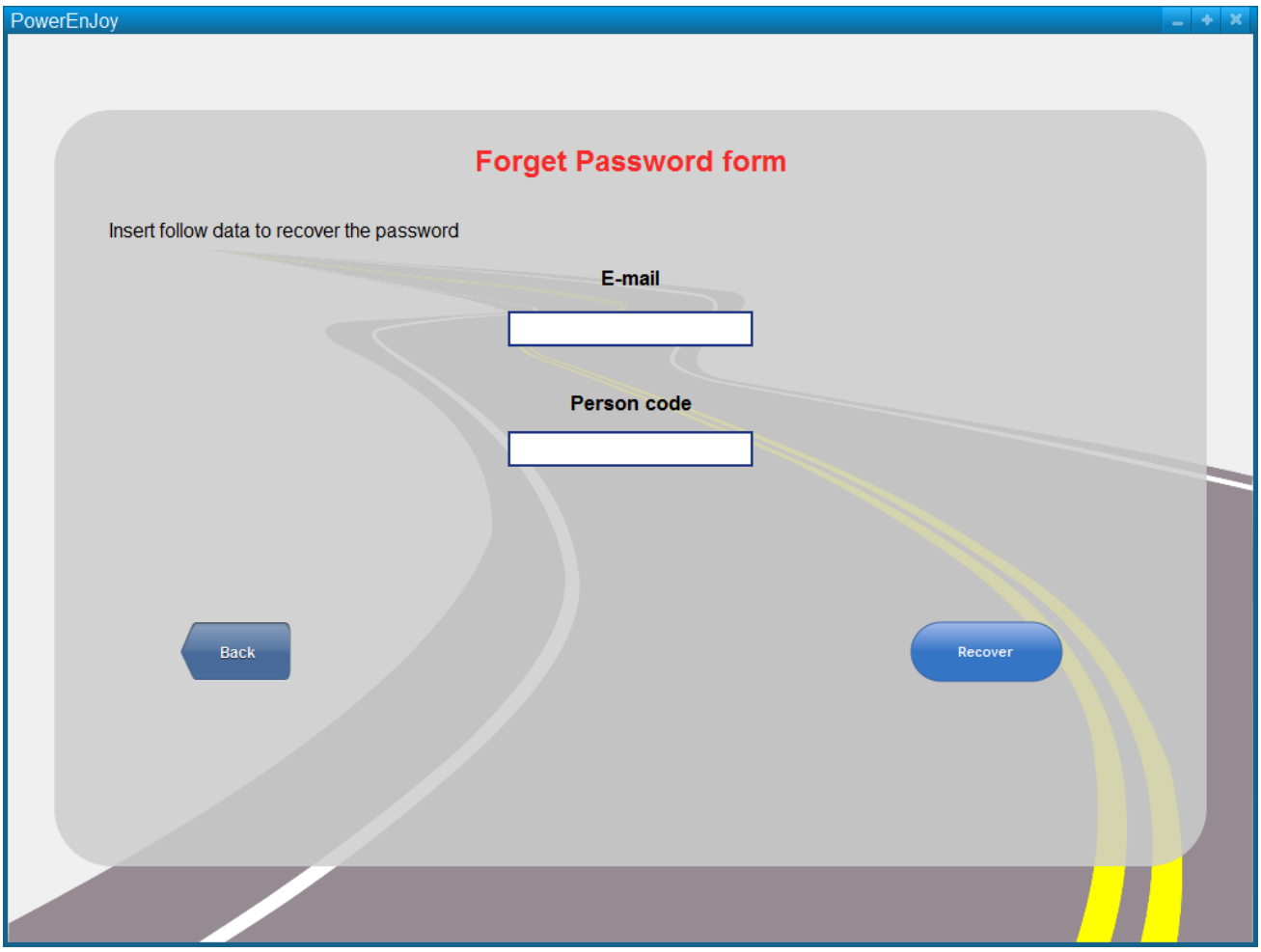
As we said Log in required only e-mail and the password generated by the system when user was registered.

- **Register**



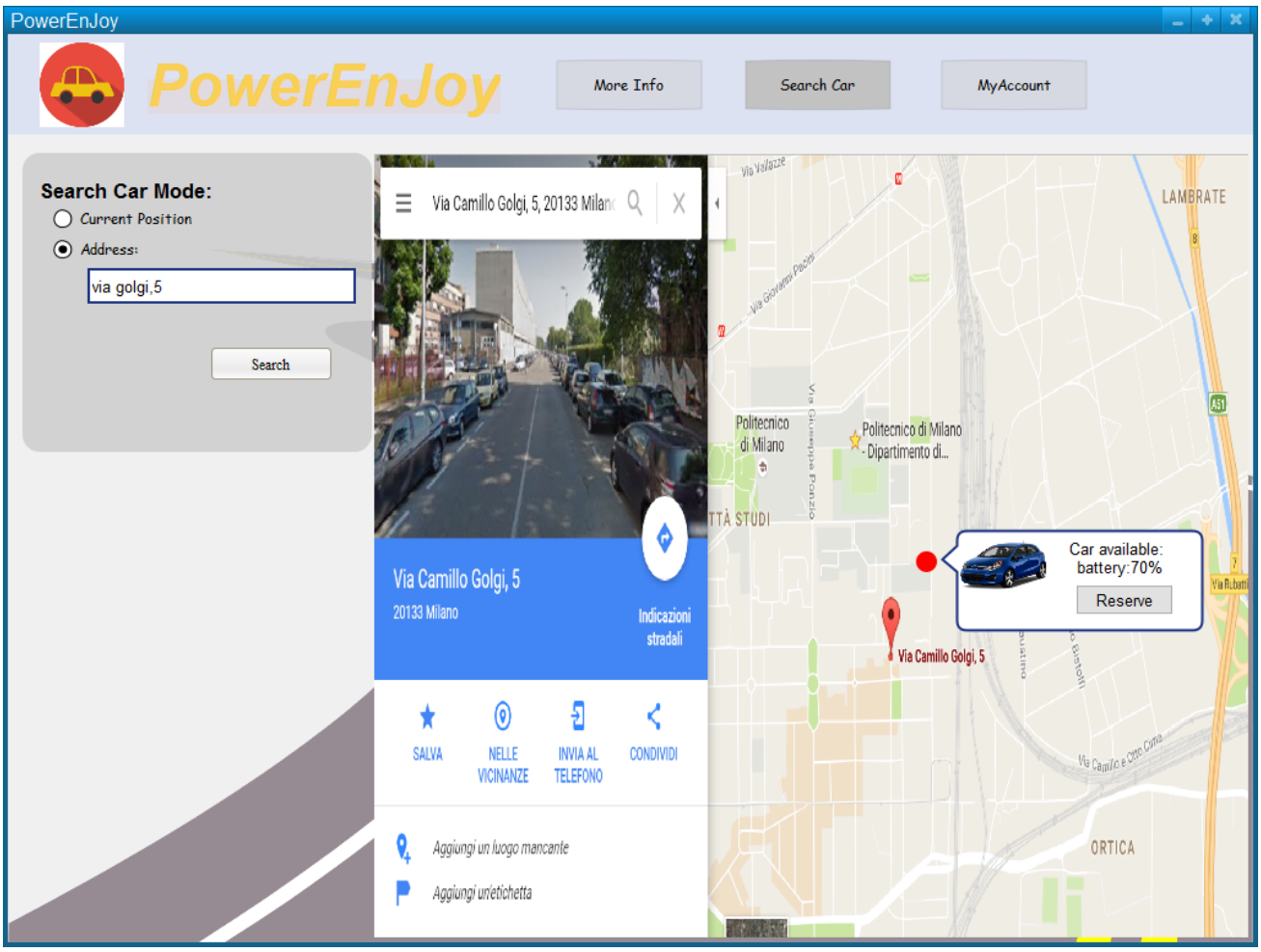
This functionality is provided only via web app.

- **Forget Password**



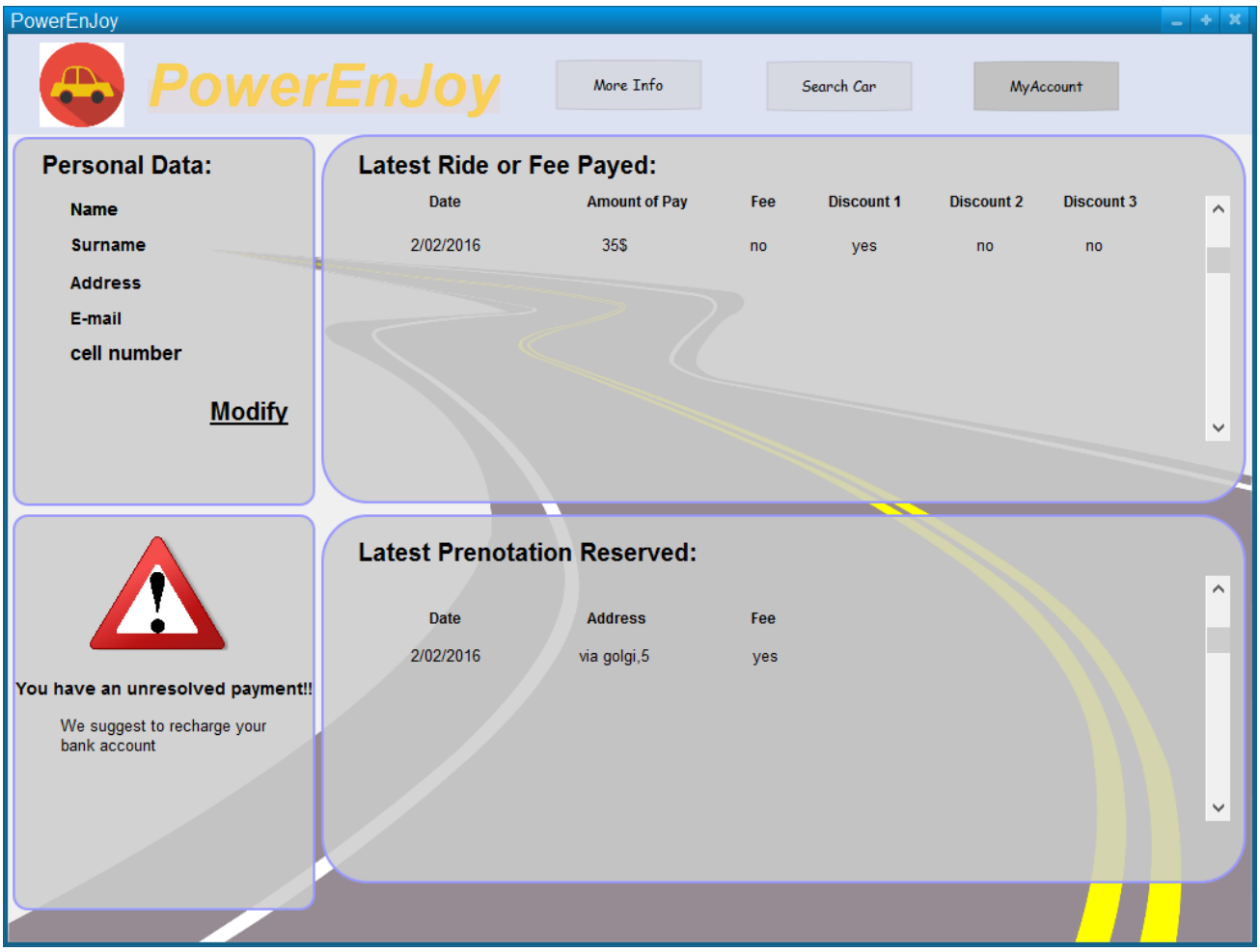
This functionality is provided only via web app.

- **Search available car and reserve**



This mock up represents the service: Search available cars near user position or near specified address, through this functionality is possible reserve a car click on the button like in the mock up.

- **See all Bill and Payments**



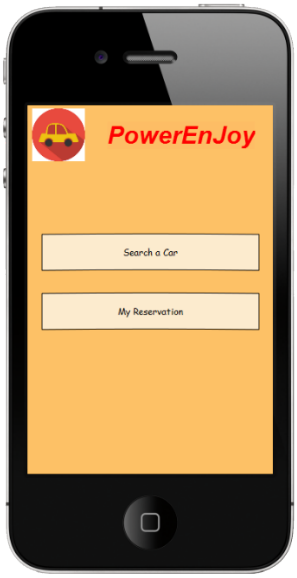
This functionality is provided only via web app.
This mock up shows the system service that permits to see the bills of the latest rides and through this user see how discount is reached.
This particular mock up shows the case which the user doesn't have enough money on bank account, the external agency that provides payments service notifies the system and this one notifies the user.

User Mobile Application

- **Log in**

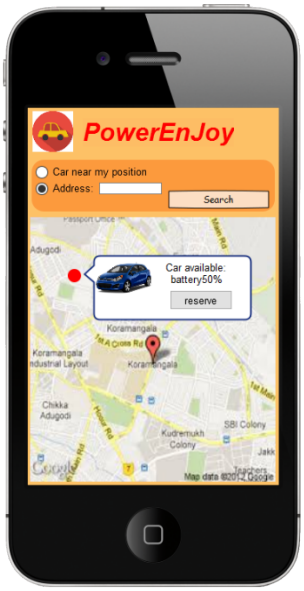
This is the log in service via mobile application, as we already said mobile application don't provide register service and forget password service.

- **Menu**



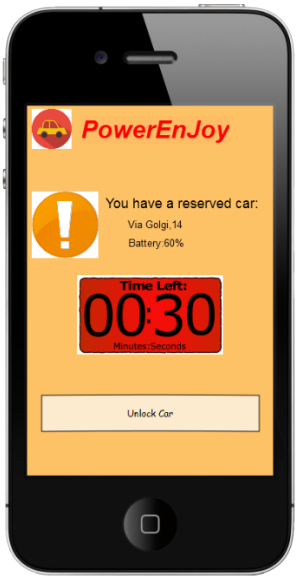
This screen don't provide any service but we think it's important to demonstrate that the mobile application is thin and provide only necessary service to control or make reservation.

- **Search a car**



The same service provides via Web Application.

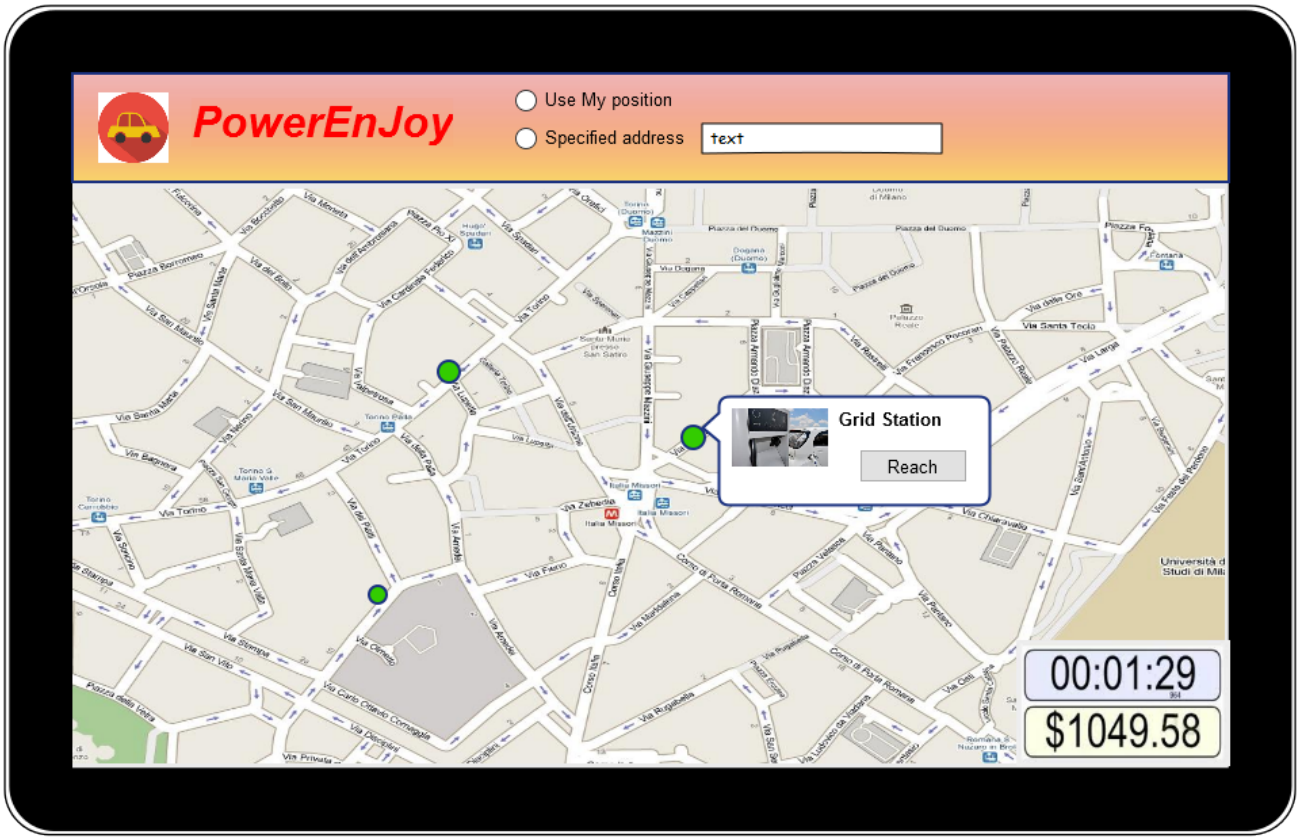
- **My reservation**



This functionality is provided only via mobile app.
This screen permits to show if this particular user has a reservation, address and battery of reserved car and the time left until the reservation expired.
Unlock car button permits to check the closeness (near 5m) and the reservation data.

Car screen

- Reach grid station and see current charging money



These functionalities are provided only via car screen, these permits to choose grid station path and see the current charging money.

All of these mock up images represents only a few possible screen of web, mobile application, in this document miss the confirm screen for reservation and unlock car.
We have decided to insert the most representative screen.

3.2 Functional Requirements

[G1]: Allow to register on the system

[R1]: The system check if the input datas are correct and of those one are not already exist.

[Domain Assumptions]: An user drive only if has got driving license.

[G2]: Allow to log in on the system.

[R2]: The system must be able to check if the password and username are correct.

[G3]: Allow to find the locations of available cars within a certain distance from their current position or from a specified address.

[R3]: The system checks the syntactical correctness of the input address

[R4]: System must show only the available car into 2km.

[R5]: System must capture the current position user if he chooses this option.

[R6]: For each available car, the system must show how much battery has the car.

[Domain Assumptions]:

All the GPS always give the right position.

The GPS of the cars cannot be switched off.

System battery car always show the right battery.

Each car is registered on system.

Car state could be only available or not-available.

[G4]: Allow to reserve a single car for up to 1 hour.

[R7]: The system must show the reservation timer.

[R8]: The system must change the car state whenever a car is reserved.

[R9]: The system must forbids more reservation at the same time from the same user.

[G5]: Allow to pay a fee if he misses the reservation.

[R10]: The system must add a fee on the user bank account.

[R11]: The system must change the car state from not-available to available.

[R12]: The system must allow user makes a new reservation.

[Domain Assumptions]:

A user doesn’t make a reservation until he pays last ride of fee.

Car state could be only available or not-available.

//e.a. assumptions?

[G6]: Allow to tell the system she's nearby.

[R13]: The system must capture the user position when he wants unlock the car.

[R14]: The system must forbid the car unlocking if the distance between user position and the car is not near or he doesn't have a reservation for that car.

[R15]: The system must stop the reservation timer

[Domain Assumptions]:

All the GPS always give the right position.

The GPS of the cars cannot be switched off.

A car is nearby if the distance between this one and the user is less than 3 meters.

[G7]: Allow to see the current charges

[R16]: The system must charge money when the motor ignites.

[R17]: The system must notify in real time how much the client is paying for the ride.

[R18]: The system stops charging when the engine is shut down.

[Domain Assumptions]:

There are no pause during the ride.

The potential discounts will be counted only after plugging time.

[G8]: Allow to reach grid stations location.

[R19]: The system during the ride visualize the position of grid station to the map (through the car screen).

[R20]: The system must calculate the path to the nearest grid station or grid station chosen by the user.

[Domain Assumptions]:

Power grid area location is registered on the system.

[G9]: Allow to recive a discount if the user has at least two passengers.

[R21]: The system must apply a discount of 10% for the last ride if there aren't any other discount.

[R22]: The system must be able to recognize if there are passengers.

[Domain Assumptions]:

There is a software that allow to recognize the right number of passengers.

If a user takes more than 1 discount he takes only the biggest one.

The potential discounts will be counted only after plugging time.

If a user takes a discount and a fee, at the end of the ride system counted only the fee.

[G10]: Allow to receive a discount if the user left the car with no more than 50% battery empty.

[R23]: The system must read the state of the battery.

[R24]: The system apply a discount of 20% if battery left is more than 50%.

[Domain Assumptions]:

If a user takes more than 1 discount he takes only the biggest one.

The potential discounts will be counted only after plugging time.

If a user takes a discount and a fee, at the end of the ride system counted only the fee.

System battery car always show the right battery.

[G11]: Allow to receive a discount if the user left the car in a Power grid station and takes care of plugging the car.

[R25]: The system must recognize if a car is parked in a power grid station

[R26]: The system applies a discount of 30% if a car is left in a power grid station and user takes care of plugging the car.

[Domain Assumptions]:

If a user takes more than 1 discount he takes only the biggest one.

The potential discounts will be counted only after plugging time.

If a user takes a discount and a fee, at the end of the ride system counted only the fee.

Power grid area location is registered on the system.

If a car is recharging, its state is not-available until the battery left is almost 70%.

The plugging time is 3 minutes.

[G12]: Allow to receive a fee if the user left the car more than 3km from the nearest power station or left the car with more than 80% of the battery empty.

[R27]: The system must calculate the distance from the nearest power grid station.

[R28]: The system must read the state of the battery.

[R29]: The system must apply a fee of 30% on last ride if user lefts the car more than 3km from the nearest power station or left the car with more than 80% of the battery empty.

[Domain Assumptions]:

If a car is left with more 80% of battery empty, the car state ’ll change into not-available state and a maintenance guy will go to the place for plugging the car.

If a user takes a discount and a fee, at the end of the ride system counted only the fee.

All the GPS always give the right position.

The GPS of the cars cannot be switched off.

[G13]: Allow to see an e-mail sent by the system with the bill ride and payment resume, at the end of this one.

[R30]: The system must send an e-mail containing the bill at the end of the plugging time and the resume of the bank transaction.

[G14]: Allow to user to pay immediately after at the end of the ride.

[R31]: The system must send the amount of money that user pays to external agency.

[Domain Assumptions]:

The payment is provided by an external agency.

When the user finishes the ride and the plugging time is over, the external agency provides an immediate payment if the user has enough money for pay.

If the user doesn’t have enough money on the bank count, the external agency notifies the system and this one notifies user with an e-mail.

3.3 Non-Functional Requirements

Performance Requirements

There is no specific non-functional requirements required but the part of the system installed on car must be show the user ride charge it's a critical service that test system performance, so, the system must be reactive and able to show in a faster way the charging.

Software System Attributes

Reliability: This system has to notify system owner if something goes wrong (system unexpected behaviour) and/or user try to cheat.

Availability and Maintainability: This system permits user to make a reservation 24h, so the system theoretically is always up; in case of maintenance time or repair time the system notifies user through warning screen on web application.

Security: The system has to protect user personal data with specified architecture and application that specified in Design Document.

Portability: The system is also composed by a mobile application that permits user to have a full experience, mobile application is a focal part of PowerEnJoy services because that permits to control and makes ride (as we already said).

This application is compatible with the major mobile platform system (Android, IOS).

4 Scenario identifying

Here some possible scenarios of usage of this application.

Scenario 1

Bob have a driver license but he doesn`t have any car because he had to move only in the city and thinks that buy a car is useless.
He decides to register PowerEnjoy. He accedes the site, choose a username but it is still use, so he decides another not still in use.
Bob write down in the form also is personal code, email, telephone number, address, and in the end his credit card number for all the payments.
He submits the form and then, after a few minutes, he receives, in the email writes in the form, the password for login the app.

Scenario 2

Mr Flanagan is a registered user in PowerEnJoy. He doesn`t log PowerEnJoy app until one
month and he forgives his password.
Flanagan so enters the site and clicks on recover the password.
Flanagan fills the form with the person code use in the registration and, after a minute he receives back the password to login.
After the login Flanagan would like to look his history about the reservations and payments so, he clicks on the respective part of the site.

Scenario 3

Miss White is a user of PowerEnJoy and she has a 21 years old daughter calls Martha.
Miss White is thinking that is daughter use her username and password for use the PowerEnJoy cars.
She decides to log for watch the history of reservation and the relatives payments.
She writes the right username and a wrong password the first time, the second time she remembers the right password and so she login.
She clicks on the part of the site that allows her to see her history of payments and she discovers that there are no strange payments.
Miss White is so happy.

Scenario 4

Anna needs to go to the Ikea for buy some furniture.
Before she enters in the Ikea she decides to book a PowerEnJoy car. She logs in the mobile app and search near her position if there are any car.
Fortunately, there is a car near her available, and she books it.
She goes to the Ikea and while she is paying in the pay desk she receive an e-mail from
the PowerEnJoy. The e-mail says that she has to pay 1 euro of fee because she books a car for more than one hour, and also the car return available for all.
The bank also sends an e-mail of the successful payment of the 1 euro.
Oh, no she stays too much at the Ikea! Fortunately, anyone take the car, so she books it again and returns to home.

Scenario 5

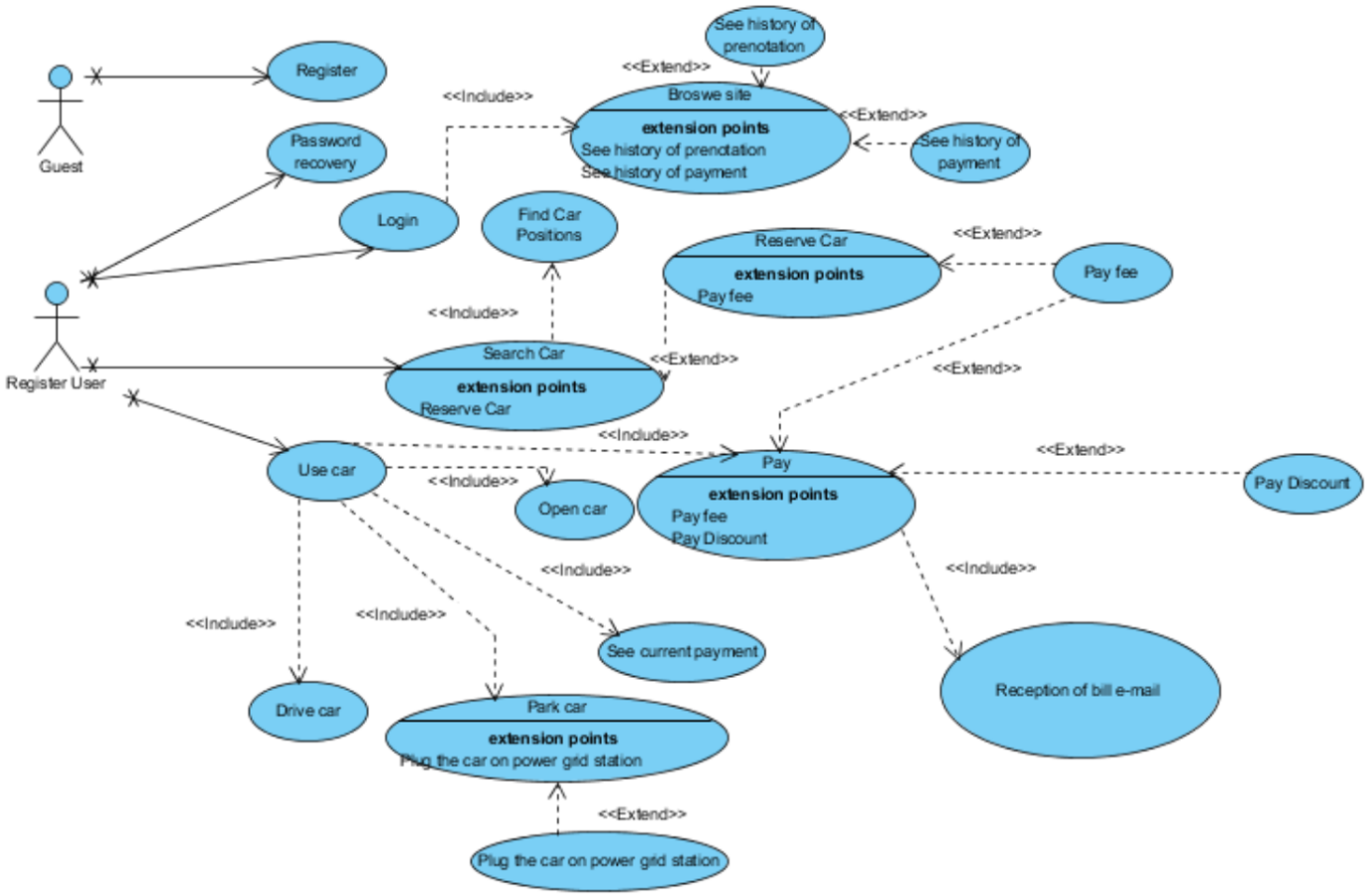
Gina and Marika are best friends and this evening they have an Halloween party.
They don`t have cars but Gina is a PowerEnJoy`s user, so they decide to book a car.
She logs, searches for a car with her smartphone, and books a car under her house.
Marika reaches Gina`s house, and after a few minutes, they open the car and start driving for the party.
About twenty minutes later they reach the party place and so they start to search for a park.
Unfortunately, they see on the map of the screen, that the park they find is very far from a power grid station.
After 3 minutes the e-mails from the bank and the PowerEnJoy arrives and the bill is very huge because there is a 30% fee for the more than 3km distance from nearest power grid, and there wasn`t the discount of the two people in the machine because if there are fee and discount, only the fee is take into consideration.
So, with money less they go to the party, knowing that they will drink less.

Scenario 6

Mark has an appointment with the dentist this afternoon. He doesn't have the car because
his wife takes it for shopping.
Mark decides to books a PowerEnJoy car. Mark logs into the mobile app, search for a car near him and reserves a car with 100% the battery very near him!
When Mark is near the car, he opens the car with the mobile app, ignites the engine and starts to drive.
When Mark is near to the dentist sees on the map of the car, that near the dentist there is a power grid station, so he decides to park there.
Before getting of the car, Mark sees on the car screen the amount of payment for the trip, but he knows that there will be a discount of 50% because Mark will recharge the car and he left the car with 70% battery full.
Mark plugs the car and goes to the dentist. After 3 minutes, he receive an e-mail from his bank that says the payment for PowerEnJoy goes fine and also an e-mail from PowerEnJoy for the bill.
Mark is surprise for the bill because the discount is only of 30%, but Mark now remembers that only the bigger discount is take.

5 Uml models

5.1 Use case diagram



5.2 Use case description

In this paragraph, the main use cases will be described. These use cases can be derived from the scenarios and the use case diagram.

User registration

Name: user registration.

Actors: guest user.

Entry conditions: there are no entry conditions.

Flow of events:

- The guest enters the web site and click on the registration button.
- The guest fill the form with his name, surname, person code, bank code, driver license number, e-mail, address, phone number and then click on finish registration.
- The system control if exists the person code, the bank code and the driver license number.
- The system send the user a password.
- The guest receives the e-mail with the password.

Exit conditions: The guest now is a register user and can login the system on the web app and also the mobile app.

Exceptions:

- The system advises the guest if person code, bank code or driver license number are wrong and give the possibility to rewrite them after the clicks of finish registration.
- The system advises the guest if person code, driver license number or e-mail are already in use after the clicks of finish registration. If one of them is in use the guest is already register and can`t do the registration.
- If the guest after receiving the password don`t make first login for the next day, the registration will be eliminated.

Password recovery

Name: Password recovery.

Actors: Registered user.

Entry conditions: The user must be register to PowerEnJoy.

Flow of events:

- The user enters the site web app and click on the recovery password button.
- The user fill the form with the person code and the e-mail and click on recovery button.
- The system control e-mail and person code.
- The system send the user his password.

Exit conditions: The system can now log the mobile app or web app.

Exceptions: The system advises the user if the e-mail or the person code are wrong and give the possibility to rewrite them after pushing the recovery button.

Login the system

Name: Login the system.

Actors: Registered user.

Entry conditions: The user must be register on PowerEnJoy.

Flow of events:

- The user enters in the site by mobile or web app.
- The user fills the white space of the login with his e-mail and relative password and click on login.
- The system controls the insert data.
- The system sends the user on the search car page of the site.
- The user can browse the site and visit the pages of reservations history and payments history by clicking on my account.

Exit conditions: Now the user is logged and can reserve a car.

Exceptions: The system deny the user to log if e-mail or password are wrong but do the possibility to rewrite them after click on login.

Searching a car

Name: Searching a car.

Actors: Registered user.

Entry conditions:

- The user must be logged on the system by mobile or web app.
- The system is on the search car space at the login.

Flow of events:

- The user can choose two search mode: current position or address
- If user click current position, the map let user see the position of the available car near him.

- If user click address, the user can write an address in a space where would like to search a car. The system will screen on the map the right address and the available car near the address choose.
- The user can choose a car clicking on it on the map.
- The system show the situation of the battery of the clicked car.
- The system control if the user has pay the last ride.
- The user can reserve the clicked car by push the button reserve car.

Exit conditions:

- If a car is reserved the system makes start a timer of 60 minutes on the mobile app.
- If a car is reserved the system deny the user to reserve another car.
- If a car is reserved, a button “unlock car” begins available on the mobile app.

Exceptions:

- If the system notices that the user doesn`t have pay the last ride, the reservation button is denied.
- If the system doesn`t know the address fill by the user, the system notifies them.
- If a car begins unavailable before the reservation of the user, the system notifies the user of the error and redirects the user on the map.

User use the car

Name: User use the car

Actors: Registered user

Entry conditions: The user must have done a reservation of a car.

Flow of events:

- The user must login at the mobile app.
- The user can click on the timer part of the site for watch the remaining time until the reservation expires.
- The user can click unlock car button on the mobile app.
- The system controls the current position of the user and the position of the reserved car, and if the distance between the two position is less or equal of 5 meters the system unlocks the car.
- The user opens the car and ignites the engine
- The system turns on the screen on board and notifies in real time the current payment, the position of the car in a map on the screen and the positions of nearest power grid station or the position of the chosen one by the user.
- The user stops the engine and park the car
- When all the passengers get off the car and the door are all close, the system lock the car.
- If the user park near a power grid station, he can plug in the car for recharging it.
- After 3 minutes since the system lock the car, the system computes the payment with the fee or the discount and sends it to the payment handler.
- The system sends an e-mail of the bill to the user.
- The user receives the e-mail of the bill by PoweeEnJoy and an e-mail from the payment handler that says if the user has enough money to pay or not.

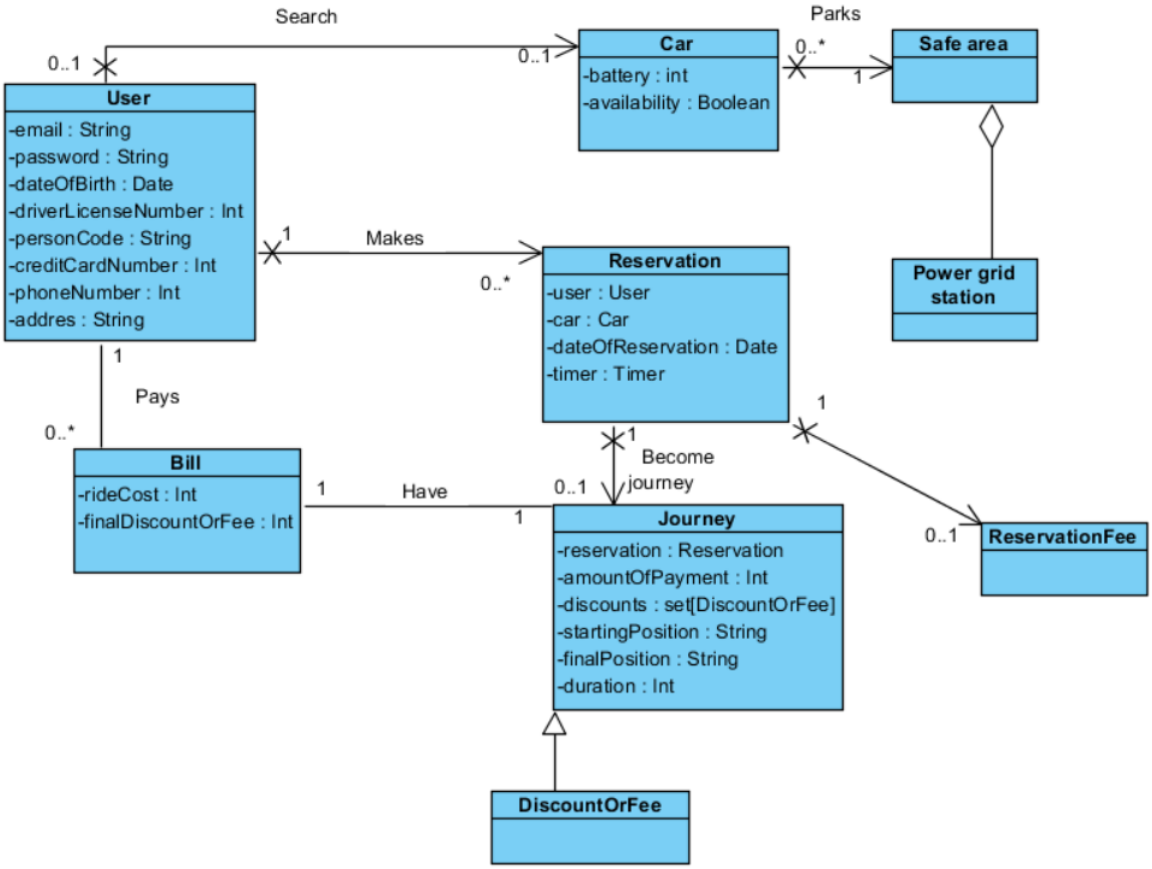
Exit conditions:

- If the user has pay the ride, allowed him to make another reservation.
- If the user not pay the ride, he can`t do another reservation until he pays it

Exceptions:

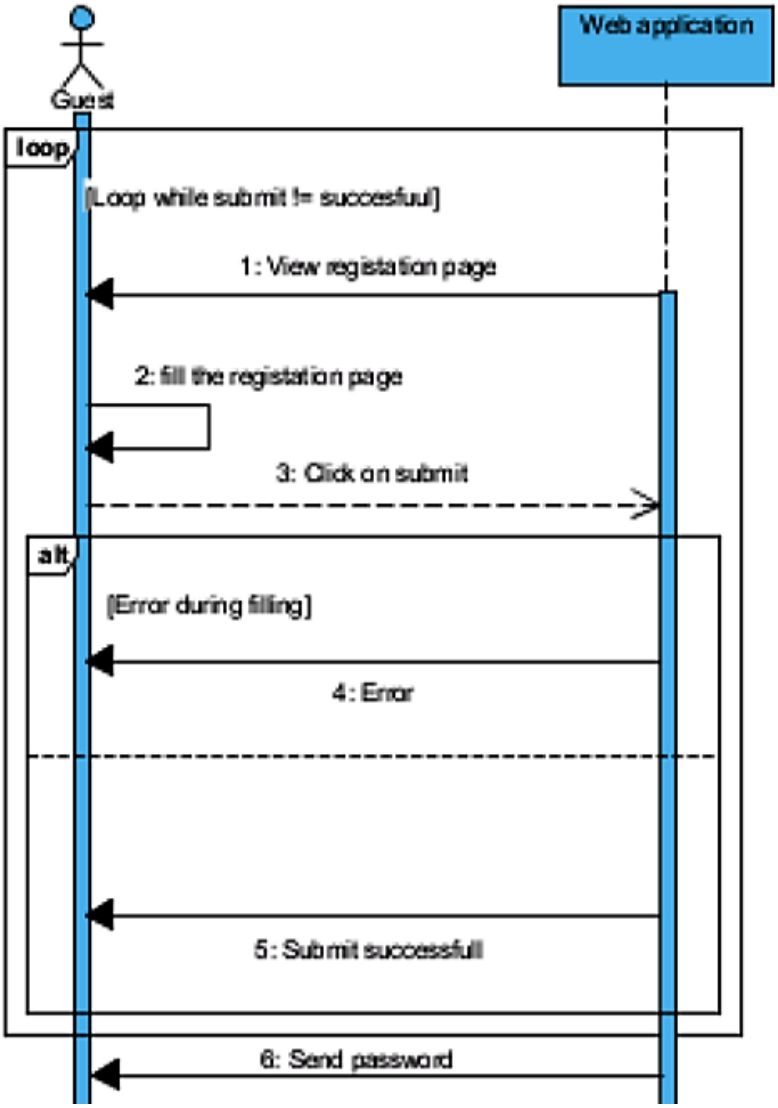
- The system disabled the unlock car button if the timer expires. The system sends a notification to the payment handler to make pay to the user the 1 Euro fee and send an e-mail to the user to being notify of the fee.

5.3 Class diagram

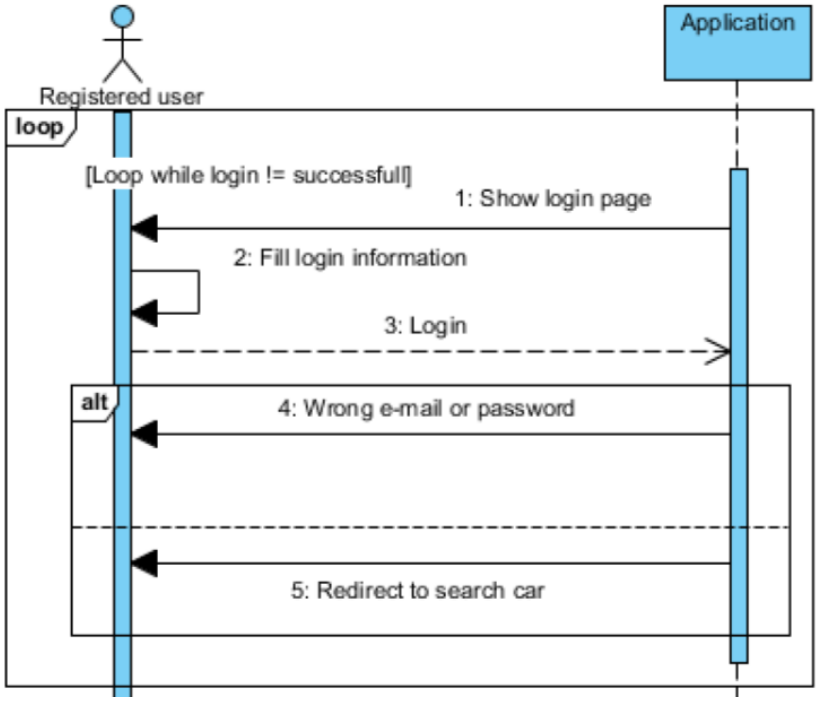


5.4 Sequence diagram

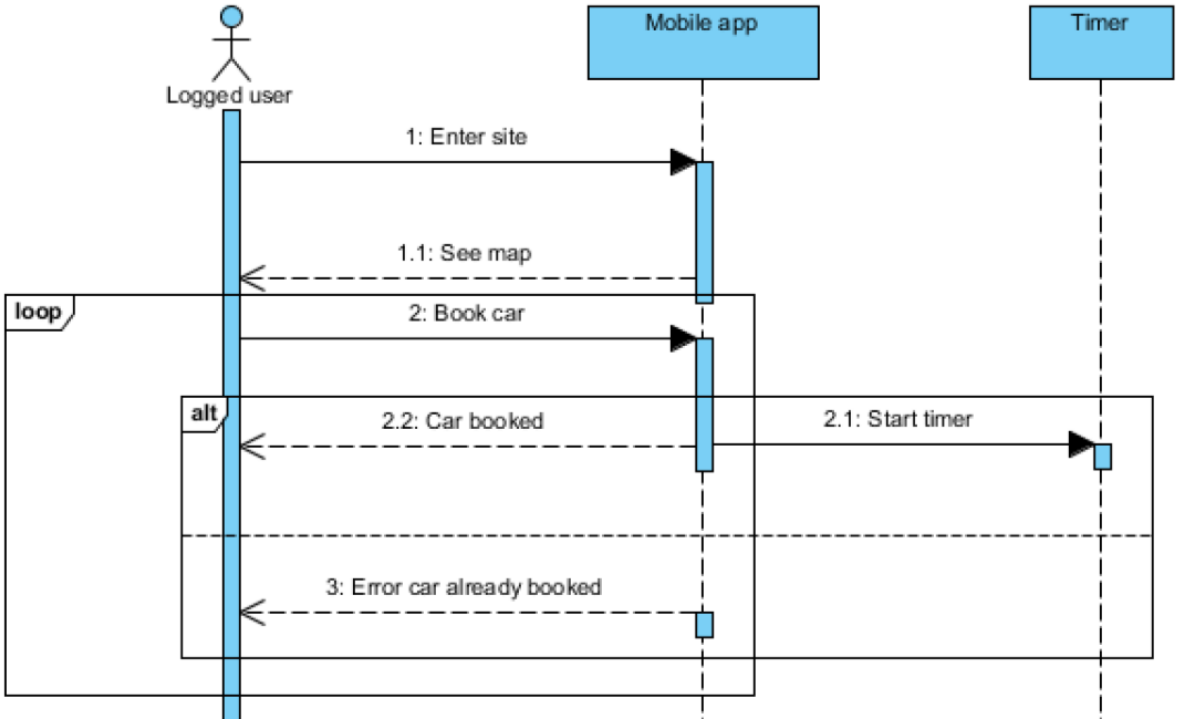
Registration sequence diagram



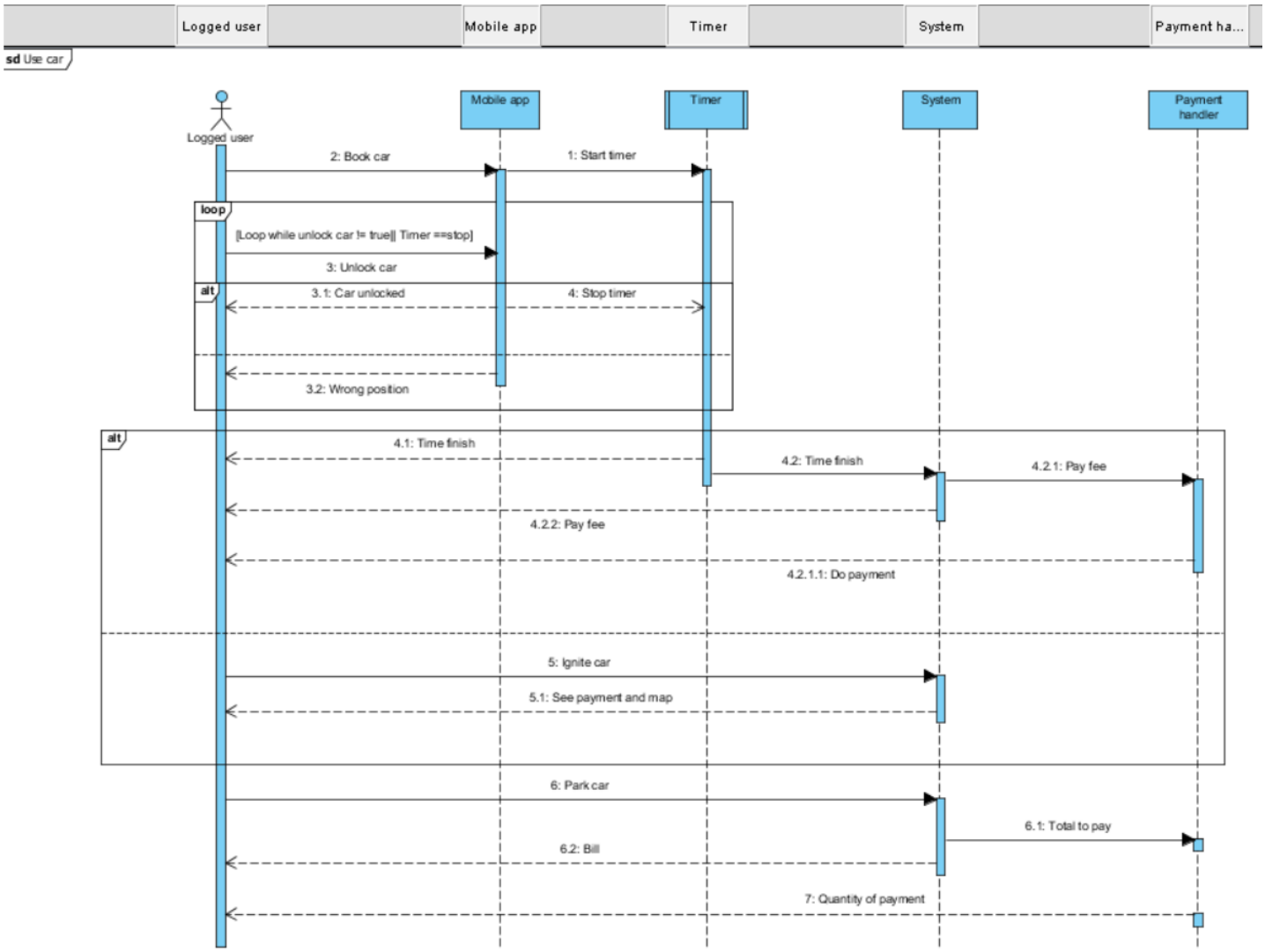
Login sequence diagram



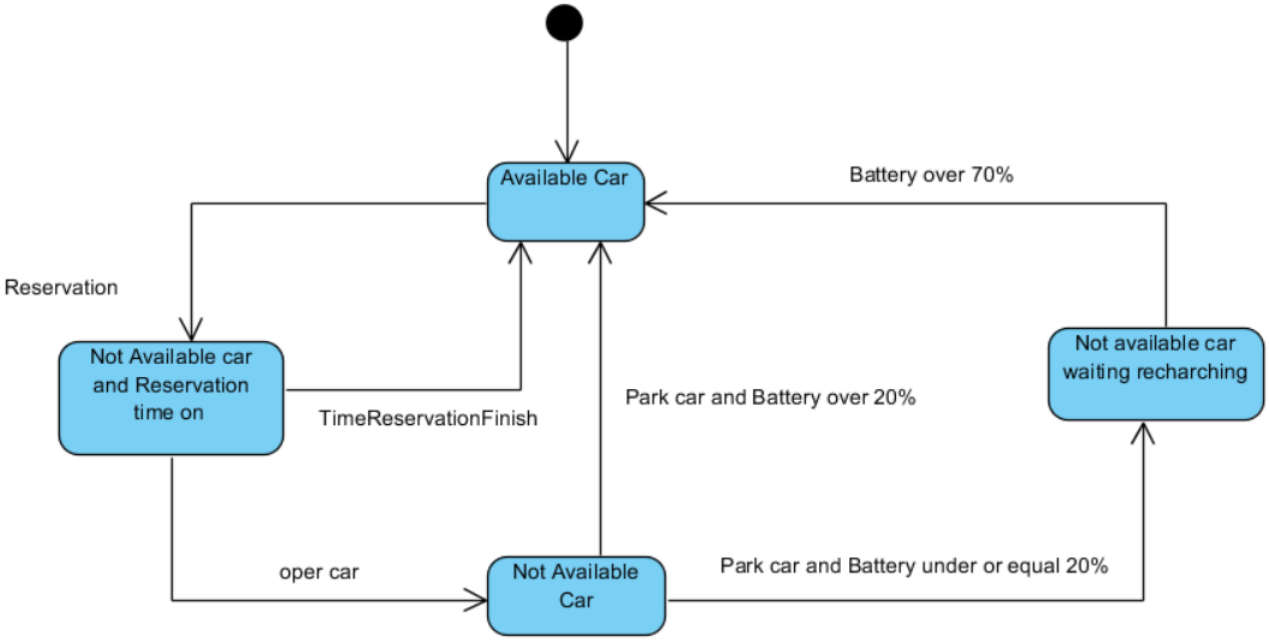
Book car sequence diagram



Use car sequence diagram



5.5 State Machine Diagram



6.1 Alloy Modeling

```
open util/boolean
sig Email{}
sig Password{}

sig PaymentHandler{
        emailToUser:set Email
}{      #PaymentHandler=1
        emailToUser=Bill.sendBill
}
abstract sig BatteryState{}
abstract sig TimeUnlock{}
abstract sig Availability{}
abstract sig Payment{}
abstract sig PluginFee{}
abstract sig PluginDiscount{}
abstract sig PassengerDiscount{}
abstract sig BatteryDiscount{}


sig Fee30 extends Payment{}
sig Discount30 extends Payment{}
sig Discount20 extends Payment{}
sig Discount10 extends Payment{}
sig NormalPay extends Payment{}

sig Upper20Battery extends BatteryState{}{#Upper20Battery=1}
sig Lower20Battery extends BatteryState{}{#Lower20Battery=1}

sig Available extends Availability{}{#Available=1}
sig NotAvailable extends Availability{}{#NotAvailable=1}

sig InTime extends TimeUnlock{}{#InTime=1}
sig OutTime extends TimeUnlock{}{#OutTime=1}

sig YesPassengersSensor extends PassengerDiscount{}{#YesPassengersSensor=1 }
sig NoPassengersSensor extends PassengerDiscount{}{#NoPassengersSensor=1}
```

```
sig BatteryUpper50 extends BatteryDiscount{}{#BatteryUpper50=1}
sig BatteryLower50 extends BatteryDiscount{}{#BatteryLower50=1}

sig ParkGridPowerStationAndPlugin extends PluginDiscount{}{#ParkGridPowerStationAndPlugin=1}
sig NotParkGridPowerStation extends PluginDiscount{}{#NotParkGridPowerStation= 1}

sig More3KmDistanceOrLow20Battery extends PluginFee{}{#More3KmDistanceOrLow20Battery=1}
sig Lower3kmDistanceAndMore20Battery extends PluginFee{}{#Lower3kmDistanceAndMore20Battery =1}

sig User{
        email: one Email,
        password :one Password,

}
sig Car{
        availability: one Availability,
        batteryState: one BatteryState
}

sig Reservation{
        driver:User,
        carDrive: Car,
        timeUnlock: one TimeUnlock

}

sig Journey{
        reservation : one Reservation,
        passengerDiscount: one PassengerDiscount,
        batteryDiscount: one BatteryDiscount,
        pluginFee : one PluginFee,
        pluginDiscount:one PluginDiscount
 }

sig Bill{
        journey: one Journey,
        payment: one Payment,
        sendBill:one Email,
        paymentToHandler:one PaymentHandler

//    ____Facts____

fact onlyRightSendBill{
        no b1:Bill|(b1.sendBill!= b1.journey.reservation.driver.email)
        }
//The system must generate password only when user wants  to register,no useless password
fact noAlonePassoword{
        no p1:Password|!(p1 in User.password)

}
```

```
// There is only one Email for one user
fact noAloneEmail{
	no m1:Email|!(m1 in User.email)
}
// An user can't reserve more than one car at the same type
fact noUserWithMoreReserv{
	no r1,r2:Reservation | r1!=r2 and (r1.driver=r2.driver or r1.carDrive=r2.carDrive)
}
// There aren't user with same passowrd or email, an user can register only one time
fact twoTypeUser{
	no u1,u2: User | u1!=u2 and (u1.password=u2.password or u1.email=u2.email)
}
// An user have a journey (ride car) if only if reservation time is not over
fact journeyOnlyTimeIsNotExpired{
	no j1: Journey | j1.reservation.timeUnlock=OutTime

}
// A reservation correspond only one journey, reservation is unique
fact notExistTwoSameJourney{
	no j1,j2: Journey | j1!=j2 and (j1.reservation=j2.reservation)
-


}
// If user unlock the car in time,he has a journey, nott exist a journey if the reservation time is over
fact eachReservationInTimeHasJourney{
	no r1:Reservation | r1.timeUnlock=OutTime && r1 in Journey.reservation
}
//when a user reserve a car,car state must change into not available, not exist reserved car with available state
fact reservedCarMustBeNotAvailable{
	no r1:Reservation | r1.carDrive.availability=Available
}
//It's not possible reserve a car with its battery is 20%
fact noReservedCarWithLowerBattery{
	no r1:Reservation | r1.carDrive.batteryState=Lower20Battery
}
// if a carBattery is lower than 20%,car state must be not available
fact lowerBatteryChangeState{
	all c1:Car | c1.batteryState=Lower20Battery  implies c1.availability=NotAvailable
}
// There aren't any cars with batteries upper than 20 and  not-available states that they are not a reserved cars
fact noOtherConditionToReservedCar{
	no c1:Car|c1.batteryState=Upper20Battery && c1.availability=NotAvailable && !(c1 in Reservation.carDrive)
}
```

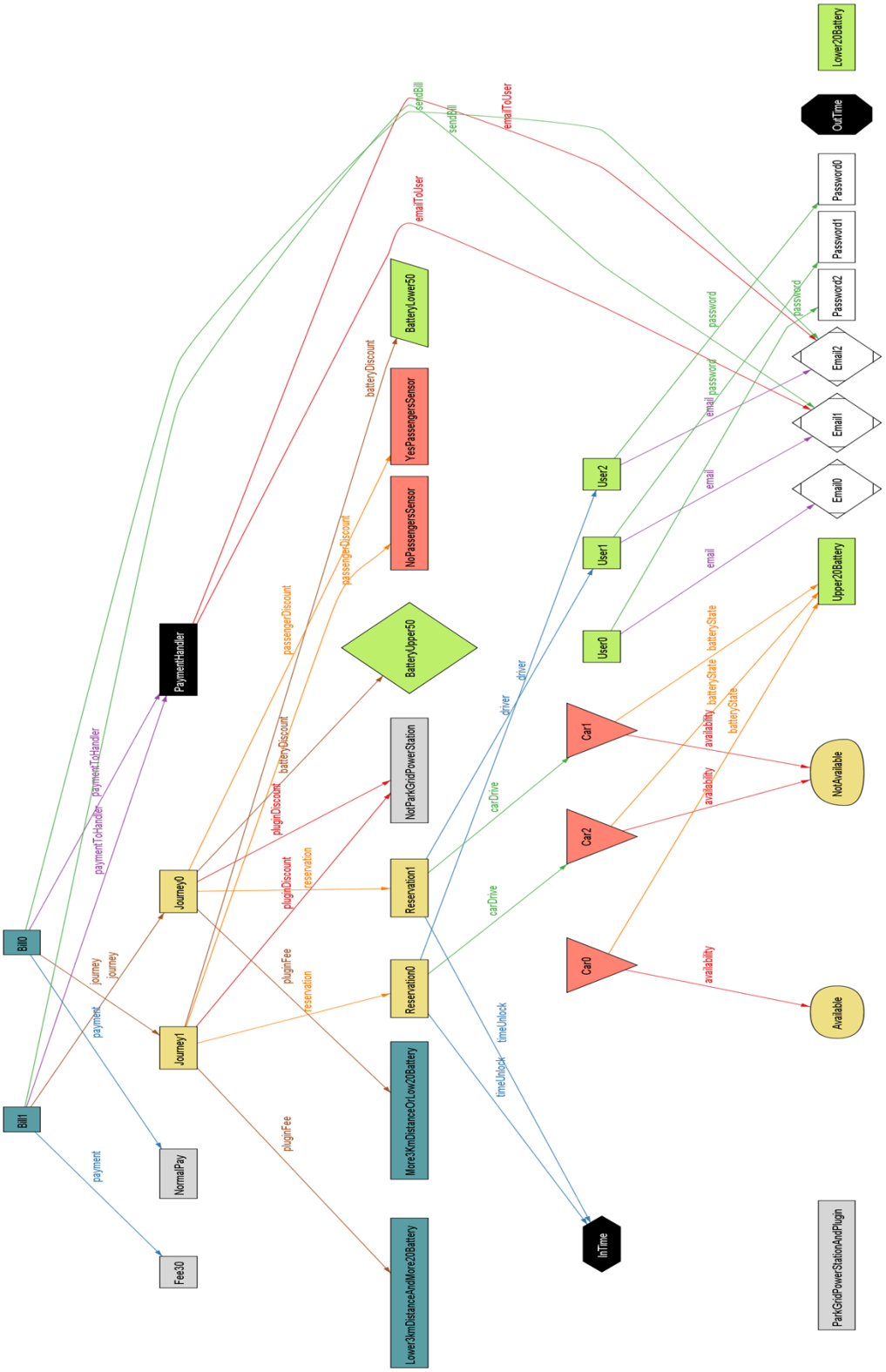
```
fact noBillWithMoreJourney{
        no b1,b2:Bill | b1!=b2 and (b1.journey=b2.journey)
}
// if user takes more than one discount,he will take only the bigger one
//if user takes a fee during the ride and other discount, the bill will count only the fee.
//in this cas we use fact with more condiction because all conditions refer to the same meaning
fact whatDiscountOrFee{
        all b1:Bill|b1.journey.passengerDiscount=NoPassengersSensor &&
         b1.journey.batteryDiscount=BatteryLower50 &&
         b1.journey.pluginDiscount=NotParkGridPowerStation &&
        b1.journey.pluginFee= Lower3kmDistanceAndMore20Battery=>b1.payment=NormalPay

        no b1:Bill|(b1.journey.passengerDiscount=YesPassengersSensor ||
         b1.journey.batteryDiscount=BatteryUpper50 ||
         b1.journey.pluginDiscount=ParkGridPowerStationAndPlugin ||
        b1.journey.pluginFee=More3KmDistanceOrLow20Battery) &&b1.payment=NormalPay

        no b1:Bill|b1.journey.pluginFee=More3KmDistanceOrLow20Battery &&
        b1.payment!=Fee30

        no b1:Bill|b1.journey.pluginFee= Lower3kmDistanceAndMore20Battery &&
        b1.journey.pluginDiscount=ParkGridPowerStationAndPlugin  &&
        b1.payment!=Discount30


        no b1:Bill|b1.journey.pluginFee= Lower3kmDistanceAndMore20Battery &&
         b1.journey.pluginDiscount=NotParkGridPowerStation &&
        b1.journey.batteryDiscount=BatteryUpper50 &&
         b1.payment!=Discount20

        no b1:Bill|b1.journey.pluginFee= Lower3kmDistanceAndMore20Battery &&
         b1.journey.pluginDiscount=NotParkGridPowerStation &&
         b1.journey.batteryDiscount=BatteryLower50 &&
         b1.journey.passengerDiscount=YesPassengersSensor &&
         b1.payment!=Discount10

        no j1:Journey|j1.pluginFee=More3KmDistanceOrLow20Battery &&
         j1.pluginDiscount=ParkGridPowerStationAndPlugin
}
```

```
fact onlyPassengerSensorUntilTheBill{
        no j1:Journey|!(j1 in Bill.journey) &&
        (j1.batteryDiscount=BatteryUpper50 ||
                j1.pluginDiscount=ParkGridPowerStationAndPlugin ||
                j1.pluginFee=More3KmDistanceOrLow20Battery)
}


//  ____Assertions____


assert NoJourneyWithAvailableCar{
        no j1:Journey| j1.reservation.carDrive.availability=Available
}

assert NoReservationWithNoBattery{
        no r1:Reservation|r1.carDrive.batteryState=Lower20Battery
}
check NoJourneyWithAvailableCar

check NoReservationWithNoBattery
pred show{

}
 run show for 7
```

```
Executing "Check NoJourneyWithAvailableCar"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  3604 vars. 237 primary vars. 6624 clauses. 32ms.
  No counterexample found. Assertion may be valid. 8ms.
```

```
Executing "Check NoReservationWithNoBattery"
  Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
  3581 vars. 237 primary vars. 6556 clauses. 28ms.
  No counterexample found. Assertion may be valid. 4ms.
```

World Generated



7.1 Used Tools

- Pencil
- Microsoft Word
- Visual Paradigm
- Paint
- Git Hub

7.2 Hours of Work

Francesco Tinarelli: 30h

Marco Wenzel: 30h