Software Engineering 2 Project: PowerEnJoy

Requirements **A**nalysis and **S**pecification **D**ocument

Marco Ieni, Francesco Lamonaca, Marco Miglionico
Politecnico di Milano, A.A. 2016/2017

February 7, 2017
v1.1

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose is to develop a digital management system for a car-sharing service called PowerEnJoy that exclusively employs electric cars. The main goal is to let the users easily look for and reserve a vehicle.

Users must be able to find the locations of available cars within a certain distance from their current location or from a specified address and among the found vehicles, they must be able to reserve a single car for up to one hour before they pick it up.

Users must pay a fee if they do not pick up the car that they reserved and receive discounts if they have a virtuous behaviour, for example if they took at least two other passengers onto the car or if the car is left with no more than 50% of the battery empty.

In addition to this, users can enable the "money saving option" and, depending on their destination and the availability of power plugs, the system will communicate them a station where to leave the car to get a discount.

1.2 Scope

The user can interact with the system with a mobile application. Every car is equipped with a geo-localization system that allows to localize it through the GPS signal. The system uses also the GPS of the mobile phone of the user, so when it is near to the car he can unlock it if he reserved it.

In order to become a user and to access the service, a person has to sign up to it, providing their credentials and payment information. If the registration is successful the user will receive back a password that can be used to access the system and to unlock the car.

The system knows the position and the battery level of all the available electric-cars, so the user can choose which car to reserve based on these two factors. If a user books a car, the system reserves that car to that user therefore, for a certain period of time, that car will not be available to other users.

On board of the vehicle the user will find a tablet that can be used as a navigator, to see the amount of the current fee and in general to interact with the service. In particular, in the navigator and in the map of the mobile application, some zone of interests are highlighted: the safe areas. These special places are the the only areas in which the users can leave the car in order to end their reservation. In fact, the system stops charging the user as soon as the car is parked in a safe area and the user exits the car.

Inside some safe areas there are the power grid stations, where the user can recharge their car. If the user takes care of plugging the car into the power grid when he end his reservation, the system applies a discount of 30% on the last ride.

1.3 Definitions, acronyms, abbreviations

Non registered user: He/she is a common person who is not registered into the system and want to register.

Logged User: He/she is the person who has previously registered to the service and use its functionalities. In order to use he/she has to be logged into the system;

Banned User: He/she is a user that has been banned from the system;

Passengers: They are common people that don't need to be registered into the system;

Maintainer: He/she is an employee that is responsible to recharge and move the cars;

Car: The object the user can reserve. It is uniquely identified by its plate. It could be in one of three different states:

- Available: the car can be reserved.
- Booked: the car is reserved but who did it has not yet arrived.
- Busy: the car is reserved and who did it is in the car itself.

- Unavailable: the car cannot be reserved because it has some problems.

Each car has its own capacity and battery level.

Safe Area: An area where the user can park the car in order to end the rental. It is uniquely identified by a GPS coordinate and a range.

Bill/Fee: It is the amount of money that the user has to pay at the end of the ride;

Tablet: It is the tablet that is present in every car;

Power Grid Station: A spot where the user can park the car and charge it, in order to end the rental and obtain the relative discount. It is uniquely identified by a GPS coordinate.

On-Board Computer: The object that communicate with server, in order to update car status and calculate rental fee.

Ride: When a user reserves a car. It saves four different time events:

- Reservation Time: when the user reserved the selected car.
- Unlock Time: when the user opened the reserved car.
- Ignition Time: when the user starts the opened car.
- End Time: when the user parks and ends the reservation.

Fee Variator: Bonus and malus that can change the final fee amount.

There are 4 types of fee variator, 3 bonus and a malus:

- Passengers Bonus: if in a ride there are at least two passengers other the driver the final fee will be drecreased by 10%.
- Battery Bonus: at the end of a ride if the car has at least 50% power left the final fee will be decreased by 20%.
- Plug Bonus: at the end of a ride if the car has been plugged on a power grid station the final fee will be decreased by 30%.
- Far from a plug and with low battery Malus: at the end of a ride if the car has at more 20% power left and it is parked at more than 3 km from the nearest power grid station the final fee will be increased by 30%.

During a ride the user can unlock more than one fee variator, but only one for each type. If more than a fee variator is unlocked the final fee is calculated by applying a fee variator equal to the algebric sum of all the fee variators unlocked.

1.4 Reference Documents

This document refers to the project rules of the Software Engineering 2 project and to the IEEE Standard 830-1998 Recommended Practice for Software Requirements Specifications.

1.5 Overview

This document is the Requirement Analysis and Specification Document for the PowerEnJoy system. Its main task is to give a specification of the requirements that our system has to fulfil, using both natural language and models. It explains both the application domain and the system to be developed and it constitutes a baseline for the project planning and the estimation of size, cost and schedule.

Furthermore, it contains the description of the scenarios, the use cases that describe them, the constraints of our system, and the relationships with the external world. Finally it also formalize the specification of some features of the applications, using Alloy language.

The remainder of this document is composed by three chapters:

Chapter 2. Overall Description: This chapter contains the product perspective, his functions and constraints, some mockups of the mobile application, the actors and the assumptions. It also contains the most significant scenarios of our system. This chapter has been written in order to be read from both the customer and the users, therefore it does not contain technical terms.

Chapter 3. Specific Requirements: This chapter goes into the detail of functional and non-functional requirements, the alloy code and the user characteristics. The target of this chapter are the team members (project managers, developers, testers and so on), so it is not written in a way to be understood by the customer or the users.

Chapter 4. Models: This chapter contains the class diagram, some bpmns, some use cases and some sequence diagrams of all the system. The main

target of this chapter are the team members, but the customer could read it in order to clarify some mechanics of the system.

Chapter 2

Overall Description

2.1 Product prespective

The PowerEnJoy front end is composed of a mobile app and an app for the tablet of the cars. The system developed will not be integrated with other existing systems that managed a car-sharing service.

Also, the system will only be user based, this means no internal interface will be developed for now. For example, it will be possible to modify the safe areas and the power grid stations only by the database interface, so editing the tables directly. In the future some API will be provided, in order to permit the evolution of the features of the system.

The interaction with the car is managed by "HandyCar", a system that abstract all the operations related to the interaction with the car, like lock and unlock the car, know his battery level, the number of passengers and so on.

2.1.1 User Interfaces

All the interfaces shall be intuitive and user friendly and they should not require the reading of detailed documentation to be used.

Below are presented the mockup of the main screens of the Android mobile app version, in order to give an idea of the final product.

Login

When the user open the mobile app for the first time he/she will visualize the login screen, which is reported in figure 2.1. Here the user can insert his/her username and password couple if he/she has one, otherwise he/she

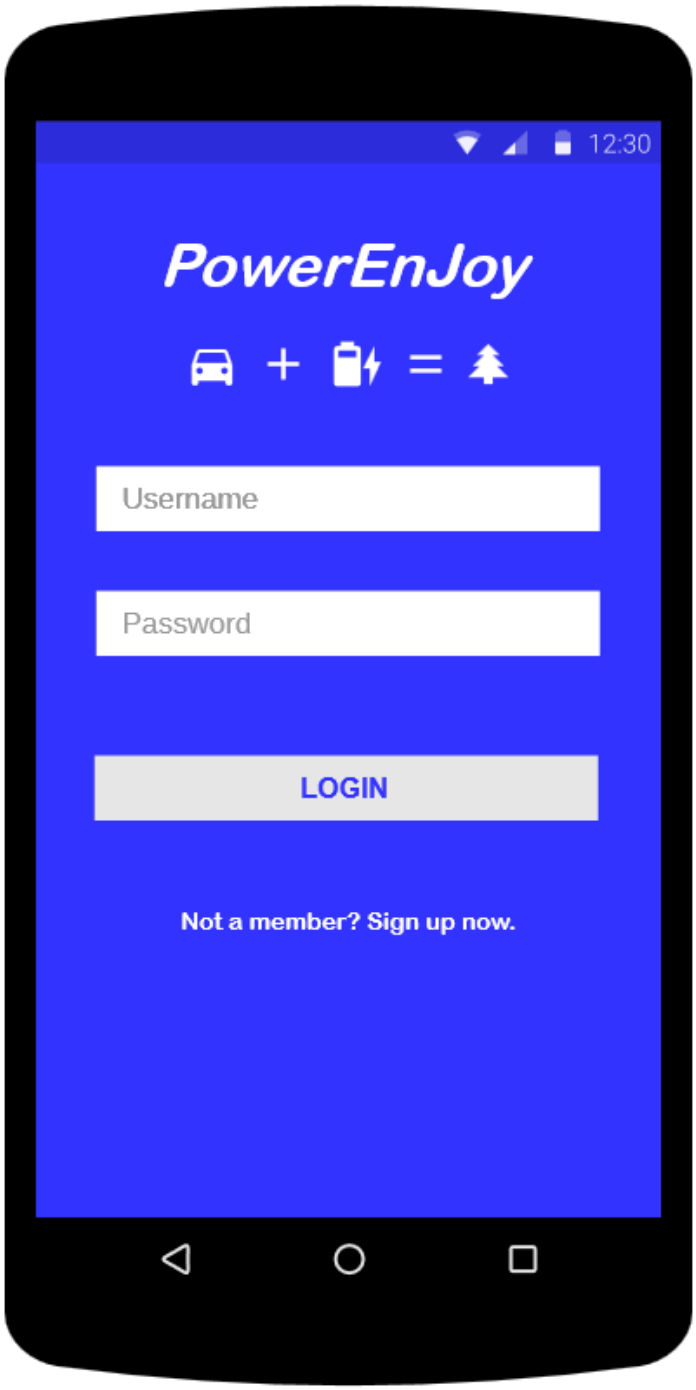


Figure 2.1: Login and registration page.

can register to the service. When the user does the login he/she will visualize the Main page.

Main page

In figure 2.2 is represented the first page that is shown to the user if it logged in to the system before.

From this page a user can search for the nearest available car having a quick look to their battery level, too.

The legend of the battery level is the following:

red: battery level from 1% to 19%;

yellow: battery level from 20% to 49%;

green: battery level from 50% to 100%.

Car selection

From the Main page, when the user selects a car a bottom sheets slides up from the bottom of the screen to reveal the car information and a marker will appear over the car that he/she selected. You can reserve the car by clicking on the blue bottom with the timer icon (a yes/no confirmation will appear). You can see this screen in figure 2.3.

Car unlock

After the reservation of the car, you can see how the screen changes in figure 2.4. The user can find indications to the car by pressing the white button that appears and when he/she is near to the car he/she can press the button with the key.

2.1.2 Hardware interfaces

In the system there are the following hardware interfaces:

Panic button: a button that is present in every car and can be pressed by the user to receive assistance from the PowerEnJoy call center. It can be used also to report that there is something wrong with some mechanical aspects of the car.

GPS: it's used to track the position of cars and users.

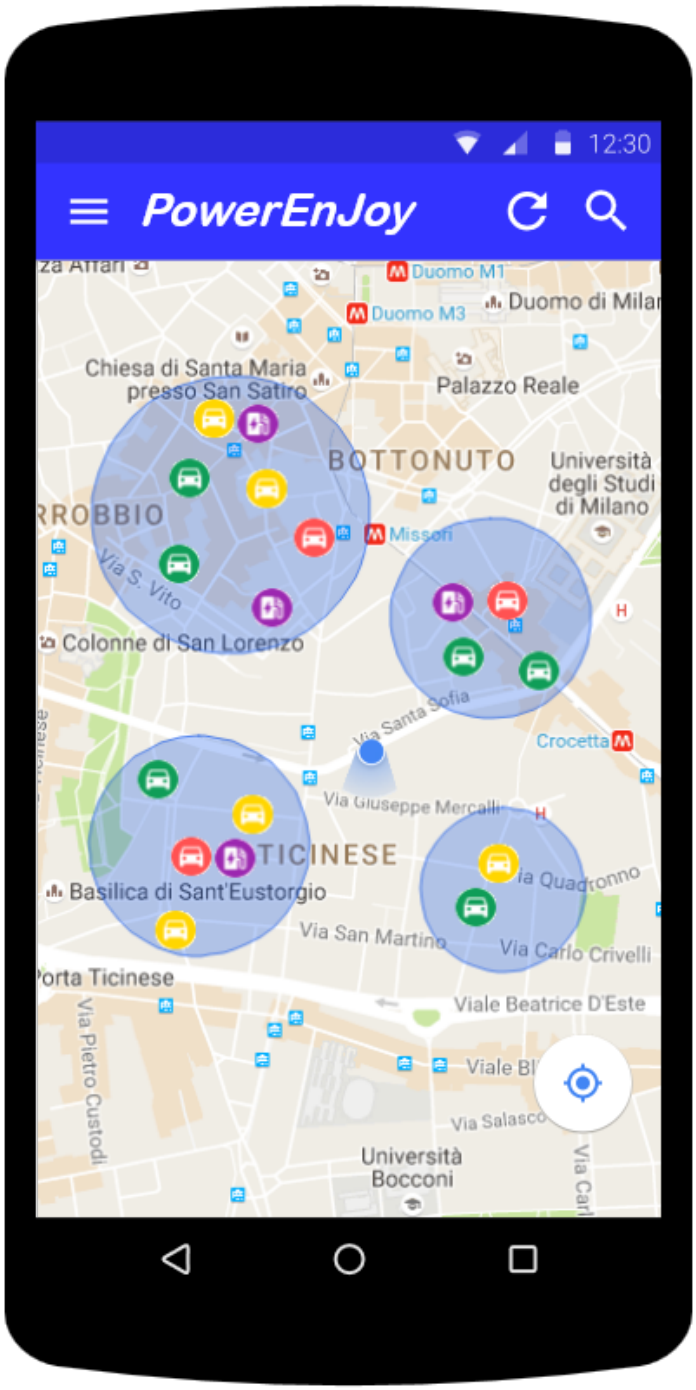


Figure 2.2: Main page.

Figure 2.3: Car selection.

Figure 2.4: Car unlock.

HandyCar Board: a board to which all the actuators and sensors of the car are connected. The sensors system that detects how many passengers there are in the car it's connected to this board too, so the board can memorize the maximum number of passengers of the current ride.

2.1.3 Software interfaces

The mobile app will be developed both for IOS and Android, so the system will use the API of these operating systems to access to the GPS of the smartphone and maps.

Furthermore, the HandyCar Board provides some simple API, in order to control the board remotely through the Internet.

The other software interfaces that we have are the external payment gateway and the motorization gateway.

2.2 Actors and Product Functions

2.2.1 Non Registered User

A Non Registered User is someone who has not been logged yet to the system. His/Her provided functions are:

Registration: a non registered user registers to PowerEnJoy service providing the following data:

- name;
- surname;
- username;
- e-mail address;
- date of birth;
- sex;
- country of birth;
- phone number;
- province of birth;
- tax id code;
- document information, that has to be validated;
- residential address;

- living address;
- payment information.

The user will receive back a password that can be used to access the system.

Login: a user accesses the system using the e-mail address they provided in the registration and the password they received.

2.2.2 User

A user is someone who has been logged to the system. He or she can access to the following functionalities:

Edit profile data: a user updates their data, to keep them updated for example.

Delete account: a user deletes his account.

Look for a car with user position: a user looks for a car basing the research on his or her devices position.

Look for a car with a specified address: a user looks for a car basing the research on a specific address.

Reserve a car: a user reserves a car that he or she has found in a previous research. He or she has up to one hour to reach the car and unlock it, otherwise the reservation would expire and he or she has to pay a fee of 1 EUR.

Unlock a car: a user unlocks a car that he or she has previously reserved at maximum one hour before. This option will be available to the user only if the GPS of his or her device is recognized at maximum at 6 meters from the car.

Start the car: a user that has unlocked the car starts the engine of the car by inserting his password in the tablet of the car

Know the current fee: a user looks at the tablet of the car and reads the current fee. He or she can do this in every moment when he or she is in the vehicle, because the current fee is always displayed.

Recharge the car: a user connects the car to a plug of a power grid station.

Enable saving money option: a user enables the money saving option by clicking on the corresponding button in the tablet interface. At this point, he or she can input his or her final destination and the system provides information about the station where to leave the car in order 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.

End the ride: the system stops charging the user as soon as the car is parked in a safe area and the user exits the car.

2.2.3 Maintainer

A maintainer is an employee that has to charge the cars with a battery level under 25% and to repair the broken components (electronic or mechanical) of the cars. He has also to return all the cars left outside a safe area in a safe one. He or she can access to the following functionalities:

Find an unavailable car: the maintainer searches for an unavailable car to fix.

Unlock a car: as for the user, the maintainer can unlock a car. Unlike the user he or she can unlock each available and unavailable car, without reserve it.

Call the garage: if the maintainer cannot fix the car he or she can call the garage in order to asks for a tow truck.

2.3 User Characteristics

In order to access and use the service, user must have a device with Android or IOS with the PowerEnJoy app installed on it. In addition, the device needs to have access to Internet and the GPS enabled.

PowerEnJoy aims to reach any person who needs to share a car, regardless his cultural or instruction level, so everyone with a basic knowledge of mobile applications will be able to access easily the service.

The user has to be registered and logged into the system in to the system in order to access his functions.

2.4 Constraints

Regulatory policies: It is user responsibility to ensure that the use of the system complies with the local laws and policies. The system must ask the user for the permission to acquire, store and process personal data. Furthermore, the system must offer to the user the possibility to delete all the personal data.

Hardware limitations: PowerEnJoy application will require a device with Android 4.0.3+ or IOS 8+ to work. In addition to this, the device must have access to the Internet and the GPS must be enabled when the user wants to unlock the car or when he wants to do a research of a car based on their position.

Reliability requirements: The system must have a minimum availability of 99.9%.

Safety and security considerations: All the data about the user, including their position and routes, must be kept private. Therefore, these data must not be accessible from external subjects (e.g. advertising agencies).

2.5 Assumption and Dependencies

2.5.1 Text Assumptions

1. the mobile application can work also without GPS enabled because the user can manually insert the address of the starting point;
2. the system track the position of the cars and of the users;
3. we assume that the validity of the driving license are checked by an external system;
4. the payments are handled by an external system;
5. the car has a tablet that display the current bill the user has to pay;
6. when a car is parked in a power grid station the user can start plugging the car within 10 minutes and during this time the company won't plug it;
7. the discounts are cumulative

8. the unusable (without battery) or almost unusable cars that are not in a power grid station or that can't reach a power station with the current battery will be charge on site by the maintainers of the company;
9. the user is able to search for a car within a range of 1 km from his/her position or from a specified address;
10. we assume that once the user unlocks the car he/she has up to 3 minutes to enter otherwise the car will be locked again;
11. we assume that the system stops charging the user only when the car is parked in a safe area and he/she choses to end the ride;
12. we assume that the user can decide to end the ride only when he/she is a safe area;
13. the maximum duration of a single ride is 24 hours;
14. we assume that if the user parks the car in an unsafe area he/she will continue to pay the normal amount of money per minute of the ride until the maximum duration of a ride is reached;
15. we assume that the cars can be in 4 different states: available, busy, booked, unavailable;
16. the user can reserve only one car at the same time;
17. the car parked in an unsafe area will be taken to a safe area by the company maintainers;
18. the person who rented the car is responsible for everything that could happen to the car (all the damages and all the types of fine)
19. the system detects the number of passengers on the car;
20. the system detects the battery level of the car;
21. the system detects if the car is on charging or not;
22. the system detects if the engine of the car is ignited or not;
23. the system can connect the car with the remote server;
24. the user can chose how much he/she wants to recharge the car, there is not a minimum;

25. the discounts applied to the final price based on the battery level, refers to the full charge of the car and not to the charge that the car had when the user picked it up;

26. we assume that if the user recharge the car and does not end the ride he/she will not obtain the discount;

27. if the user cannot pay his bill he/she will be banned until the bill will be paid;

2.5.2 Domain Assumptions

1. all the GPS give always the right position;

2. the GPS of the car can't be switched off;

3. the password given to the user is unique;

4. the user unlocks the correct car;

5. the user always has the mobile app installed on his/her phone;

6. the car is exactly in the specific position in which it was at the moment of the booking from the user;

7. the discounts are correctly applied;

8. the power grid stations are always working and can't be damaged;

9. the money saving option is always available and work correctly;

10. each car is unique;

11. a car can be in only one zone at the same time and this is the real zone;

12. the cars update correctly their status (available, busy, booked, unavailable);

13. the car plates are unique;

14. the amount of money charged is always correct and can't be negative;

15. the number of people on the car detected by the system is always correct;

16. the system detects the battery level correctly;

17. the system detects correctly if the car is on charging or not;
18. the system detects correctly if the engine is ignited or not;
19. the number of passengers is positive;
20. the number of passengers never exceeds the capacity of the car.

2.6 Scenarios

Here we will present some possible scenarios in which the service is involved.

Scenario 1

Al has to come back home and there are no more bus available. He opens the PowerEnjoy app on his smartphone. The app, using the phone GPS, shows Al all the cars close to him. Al finds a car 50m close and reserves it. Once he reaches it he asks to the system to open it. Once he enters he submit his password on car the tablet present on the car and starts the ride. Al find the safe area nearest his house, parks the car there and ends the ride. The system calculate the bill and charges it to Al's account.

Scenario 2

Al, John and Jack have to go to the airport. They meet at Al's house. Al finds and books a car. They reach the car and enter into it. Car sensor counts two passengers excluding the driver. When the ride ends a discount of 10% will be applied before the system charges the fee to Al's account.

Scenario 3

Similar to scenario 2 Al, John and Jack go to the airport with the same car. Once they enter into the car, Al selects the money saving option before they start moving. The system shows him the position of the power grid stations near their destination and Al picks up one. Once Al parks and connects the car to the power grid a discount of 40% will applied to the total bill (10% because there were two passenger excluding the diver and 30% because Al connected the car to the power grid).

Scenario 4

Similar to scenario 2 Al, John and Jack go to the airport with the same car and Al selects the money saving option before they start moving. However they are late and Al does not connect the car to the power grid even though he has parked in the power grid station. At the end only a discount of 10% will be applied, because there were two passengers other the driver.

Scenario 5

John and Jack reserve a car to go to the mall. There is not a safe area where to park near the mall so they park in the mall parking. The system will warn them that there it is not possible to end the rent. John and Jack can leave the car but the car is still reserved by them and they continue to pay the normal fee. Once they end shopping they return to the car and come back home, looking for a safe area where to end the rent.

Scenario 6

Carl decide to have an excursion and reserves a car to go to the lake. There is not a safe area near the lake so he parks near it without ending the rent. At the end of the day he decides to come back home with the train leaving the car there without ending the rent. 24 hours after the rent started, the system will end it and it will charge to Carl the full amount of rent fee with an additional fine.

Scenario 7

Carl, with his friend Joe and Debby, has an appointment through the city and they decide to rent a car. At some point he realizes the car has only 10% power left and so he parks it in the nearest safe area and ends the rent. The system will charge Carl's account the final bill increased of 20%. This because he gained a discount of 10% taking two passengers on the car and a fine of 30% because he did not connect the car to the power grid and left it with 10% of battery.

Chapter 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces must satisfy the following UI constraints:

Mobile App

- The iOS version must adhere to the iOS Human Interface Guide-lines
- The Android version must follow Android design guidelines;
- If the user did not logged in before in his/her current device the app must show him/her the login page;
- From the lateral panel that appears when the user presses the menu button he/she must be able to change his/her data and to see his/her status (so for example if he/she has pending payment) and to reach an help page,;
- In the help page of the mobile app there is a button that when it is pressed the user can call the PowerEnJoy help center if he/she needs further help respect to what is written in the help page;
- The refresh button can be pressed by the user in order to update the current available cars, including their battery level;
- UI controls and views must be suitable for the input interface and the screen size.

Server back-end

- The server back-end must be configurable by means of a configuration text file.

Tablet on-board

- The tablet must let the user enable the money saving option. After that the user must be able to insert his/her destination.
- The interface must follow Android design guidelines;

Common to car tablet and Mobile app

- The interface must offer the possibility to choose the language used between the available ones;
- The interface must follow Android design guidelines;
- The compilation of data fields has to be made with suitable controls like multiple choice, date picker or text field in order to simplify the user experience;
- If the user press on a power grid station, the app has to tell him how many free plugs there are there.

3.1.2 Hardware Interfaces

Panic button: When the panic button is pressed the HandyCar board must send a message to the system to notify that in that car there is something wrong. The message must contain an ID that identifies the car from which it was pressed. When the system receives a message from a panic button the call center of PowerEnJoy calls the driver of that car to provide assistance to that user. After the user exits the car, the operator of PowerEnJoy call center can assign to that car the state of "unavailable".

3.1.3 Software Interfaces

- The mobile app must run on both Android 4.0.3+ and IOS 8+.
- The payment gateway and the motorization gateway has to communicate via HTTPS in order to ensure security.

- Both payment gateway and motorization gateway expect a POST on their server in order to begin a transaction and the payload must be in JSON format. Every transition is very simple, because after the post they will respond to us with another POST, where the payload is composed by a "yes/no" plus further optional information.

In the next subsections we will see more in detail the services offered by the Payment gateway and the motorization gateway.

Payment gateway

The payment gateway offers to our system these two services:

- check if the payment information of a user are valid;
- execute a payment.

For the first point, the response will be "yes" if the information are valid and "no" otherwise.

For what regards the second one, the response will be "yes" if the payment was executed correctly and "no" otherwise.

Motorization gateway

The motorization gateway offers the following two services:

- check if the driving licence and the document matches with the other data provided from the user (except the payment information);
- forward a fine that the company received to the motorization.

For the first service, the response will be "yes" if the information are valid and "no" otherwise.

For the second point, the system must send the content of the fine, the information of the car to what the fine was addressed and the driving licence number of the user that was driving it. The response from the motorization is not expected, so if it arrives it will be ignored.

3.1.4 Communication Interfaces

The mobile app, the tablet in the car, the HandyCar Board and the remote system itself communicate with each other through HTTPS requests (port 443). In this way we ensure that everything that is sent on the internet is encrypted.

3.2 Goals and Functional Requirements

Assuming that the domain properties stipulated in the paragraph [2.5] hold, here are presented all the goal to achieve and the requirements, grouped under each goal from which they are derived.

1. [G1] Allows the user to reserve a car for up to an hour before he/she picks it up:

    - [R1] The system must turn the state of the car from reserved to available if the user did not unlock it within an hour since the moment of the reservation;
    - [R2] The system must turn the state of the car from available to reserved after the user reserves it.

2. [G2] Allows the user to find the closest cars with respect to his/her position:

    - [R1] The system must be able to check the position of the user through the GPS position of his mobile phone;
    - [R2] The system must be able to show all the available cars, with their current level of battery, within a range of 1 km from the position of the user;
    - [R3] If there are not any available cars within a range of 1 km from the position of the user the system must notify the user.

3. [G3] Allows the user to search for a car in a specific location:

    - [R1] The system must be able to identify a specified location;
    - [R2] The system must be able to show all the available cars, with their current level of battery, within a range of 1 km from that specific location;
    - [R3] If there are not any available cars within a range of 1 km from that specific location the system must notify the user.

4. [G4] Allows to unlock a car only to the user who currently reserved it:

    - [R1] The system must be able to check if the user who wants to unlock the car is the same who reserved it;

5. [G5] Allows the user to end the ride only in specific areas called safe areas:

    - [R1] The system must be able to show to the user the position of the safe areas on the map;
    - [R2] The system must notify the user when he/she is inside a safe area;
    - [R3] The system must end the reservation only when the car is parked in a safe area and the user exits the vehicle;
    - [R4] When the reservation ends the system must change the status of the car in available and lock the car.

6. [G6] Allows the user to unlock the vehicle using the mobile app when the distance between his/her mobile phone and the car is less than 6 meters:

    - [R1] The system must be able to calculate the distance between the car and the mobile phone of the user using the GPS positions;
    - [R2] The system must show to the user the option to unlock the car at least when the distance between the car and the mobile phone of the user is less than 6 meters;
    - [R3] The system must be able to unlock the cars remotely;
    - [R4] The system must be notified when the user enter inside the car;
    - [R5] The system must be able to lock the car again if the user doesn't enter inside the car within 3 minutes.

7. [G7] Allows the user to know the current fee he/she has to pay;

    - [R1] The system must be able to calculate the current fee with respect to the amount of money per minute defined by the company;
    - [R2] The system must be able to show on the tablet present on the car the current fee that the user has to pay.

8. [G8] Allows the user to recharge a car in power grid stations:

    - [R1] The system must be able to show to the user the position of the power grid station;

- [R2] The system must be able to show to the user the number of free plugs in every power grid station.

9. [G9] Does not allow user who has not paid the last fee to use the system, banning them until that fee will be paid:

    - [R1] The system must allow only the users who has not been banned to search for a car;
    - [R2] The system must allow only the users who has not been banned to reserve a car;
    - [R3] The system must be able to know the result of the payment operation;
    - [R4] The system must ban a user who has a pending fee;
    - [R5] The system must be able to mark a fee as paid if and only if the payment operation is successful;
    - [R6] The system must be able to remove the ban from the user when his/her pending fee has been extinguished.

10. [G10] Allows the user to enable the saving money option:

    - [R1] The system must allow the user to enable the saving mode option in every moment from when he/she starts ride;
    - [R2] The system must be able to recognize the best power grid to station where to park a specified car based on its position and the distribution of the other cars of PowerEnJoy;
    - [R3] The system must be able to provide information about the station where to leave the car to obtain a discount;
    - [R4] The system must ask the user his/her destination;
    - [R5] The system must be able to check the availability of power plug in a specific station.

11. [G11] Keeps the electric cars working, with an high level of battery and well distributed:

    - [R1] There must be some maintainers who repair every broken car after at maximum two working days after the time they broke;
    - [R2] There must be some maintainers who recharge the cars after at maximum one working day after the time they discharged.

- [R3] The system must be able to identify a lack of cars in some parts of the city.
- [R4] If there is a lack of cars in some parts of the city, the system has to notify the maintainers.

12. [G12] Stimulates the virtuous behaviours of the users:

    - [R1] The system must be able to detect that there are at least two passengers (not including the driver) on the car and in this case it applies a discount of 10% on the final fee of that ride;
    - [R2] The system must be able to detect if the battery level at the end of the ride is greater or equal to the 50% of the total battery level and in this case it applies a discount of 20% on the final fee of that ride;
    - [R3] The system must be able to detect if the car is in a recharging station area and it is plugged to the power grid when the reservation ends, in this case it applies a discount of 30% on the final fee of that ride;
    - [R4] The system must be able to detect if the battery level at the end of the ride is less than 20% of the total battery level, in this case the system applies a raise of 30% on the final fee of that ride;
    - [R5] The system must be able to detect if the car is left more than 3 Km away from the nearest recharging station, in this case the system applies a raise of 30% on the final fee of that ride.

13. [G13] Allows the user to register to the system if he/she provides valid personal and payment information:

    - [R1] The system must be able to acquire all the informations for the registration (name, surname, email address, username, birth date, driving license and payment information);
    - [R2] The system must be able to check if all the mandatory fields has been completed with valid data;
    - [R3] The system must be able to check the validity of the driving license through an external service;
    - [R4] The system must be able to check the validity of the payment information through an external service.

14. [G14] Allows only the registered users to log in to the system:

- [R1] The system must be able to check if the username and password provided are correct;
- [R2] The system must allow the user login if and only if the provided username and the password are correct.

3.3 Performance Requirements

We assume that the response time of the mobile app is almost zero, i.e. the interface do not cause any delay in the interaction with the user, so the speed of the whole system depends only on user's internet connection.

The functions of reserving a car and unlock it shall be completed in less than 4 seconds from the back-end perspective. The 90% of the request shall be processed in less then 6 seconds, while all the requests shall be completed in less then 11 seconds.

The request of showing the available cars is the most frequent function executed in our system, so there must be a structure in memory that keeps track of them, in order to not compute them every time a user opens the app.

Performance must be high to guarantee a high usability under the conditions of at least 7500 users connected simultaneously.

3.4 Software System Attributes

3.4.1 Reliability

The data should always be accessible and they should also be duplicated in case of a system fault to prevent data losses.

3.4.2 Availability

The system shall have an availability of 99.9%, because a user could strongly rely on this system as his main transport, so it is important to provide an high availability. Also for this reason, the interruption of the service, due to upgrade or maintenance interventions, must be announced at least 5 days before its interruption.

3.4.3 Security

- All the communications between the system and his components must be encrypted using SSL protocol.

- The system must refuse all attempts of establishing an insecure communication channel (e.g. simple HTTP).
- Users' passwords must not be stored in plain text, but they must be hashed and salted. In addition to this, the hash algorithm used must be specified design to hash passwords.
- If a user fails to insert a password more than 4 times, then he has to insert a captcha to try again.
- The system shall use a secure authentication protocol.
- The system shall record all the reservations and rides of the user in order to ensure "non-repudiation";

3.4.4 Maintainability

- The application must be strongly documented to help future and current developers to maintain and apply changes to the system easily.
- The application must only use well documented and maintained APIs.
- The source code will be written using a Version Control System with remote hosting, in order to have backups of every state of the application.
- The system will receive log files about failures of the application containing information in order to identify the criticality to work on the next patch.
- The structure of the system shall allow as much as possible to be changed without the introduction of degradation.

3.4.5 Portability

The mobile application should be compatible with Android 4.0.3+ and IOS 8+, while the back-end should be compatible to all major hardware and software platform present on the market. The version of Android and IOS are such as to ensures to reach more than the 97% of the users that use these OS, while Windows Mobile is not supported because there are not enough users that use it to justify the costs.

The user must be able to install the app from Play Store and Apple Store.

For what regards the back-end, there must not be host-dependent dependencies.

3.5 Alloy

3.5.1 Code

```
open util/boolean

//Signatures

sig Array{}

sig DateTime {
time: one Int
}{
time >= 0
}

abstract sig CarState{}
one sig Available extends CarState{}
one sig Unavailable extends CarState{}
one sig Reserved extends CarState{}
one sig Busy extends CarState{}

sig Position{
latitude: one Int,
longitude: one Int
}

sig ElectricCar{
licencePlate: one Array,
batteryLevel: one Int,
currentPosition: one Position,
currentState: one CarState,
seatsNumber: one Int,
engineOn: one Bool
}{
batteryLevel >= 0
batteryLevel <= 100
seatsNumber > 0
}

sig SafeArea{
center: one Position,
radius: one Int
```

```
40 }

42 sig PowerGridStation{
   location: one Position,
44 plugs: set Plug
   }{
46 #plugs > 0
   }
48
   sig Plug{
50 pluggedCar: lone ElectricCar
   }
52
   sig User{
54 taxCode: one Array,
   email: one Array,
56 password: one Array,
   name: one Array,
58 surname: one Array,
   birthDate: one DateTime,
60 birthPlace: one Array
   }
62
   abstract sig FeeVariator{
64 feeVariatorPercentage: one Int
   }{
66 feeVariatorPercentage <= 100
   }
68 one sig PassengersDiscount extends FeeVariator{
   minPassengersNumber: one Int
70 }{
   minPassengersNumber > 0
72 }
   one sig PlugDiscount extends FeeVariator{}
74 one sig BatteryDiscount extends FeeVariator{
   minBatteryDiscount: one Int
76 }{
   minBatteryDiscount <= 100
78 minBatteryDiscount >= 0
   }
80 one sig FarorLowonBatteryMalus extends FeeVariator{
   maxBatteryLevel: one Int,
82 minDistancefromPGS: one Int
```

```
}{
maxBatteryLevel <= 100
maxBatteryLevel >= 0
minDistancefromPGS > 0
}

sig Ride{
user: one User,
car: one ElectricCar,
reservationTime: one DateTime,
unlockTime: lone DateTime,
ignitionTime: lone DateTime,
endTime: lone DateTime,
maxPassengersNumber: one Int,
feeVariator: set FeeVariator
}{
#unlockTime = 1 implies reservationTime.time < unlockTime.time
#ignitionTime = 1 implies unlockTime.time <= ignitionTime.time
    && #unlockTime = 1
#endTime = 1 implies ignitionTime.time <= endTime.time
    && #ignitionTime = 1
maxPassengersNumber >= 0
maxPassengersNumber <= car.seatsNumber
}


//Facts

fact no2CarsWithTheSamePlate{
no disjoint c1, c2: ElectricCar | c1.licencePlate = c2.licencePlate
}

fact no2SimilarUsers{
no disjoint u1, u2: User | u1.email = u2.email or u1.taxCode = u2.taxCode
    or u1.password = u2.password
}

fact aStartedCarisBusy{
all c: ElectricCar | c.engineOn = True implies c.currentState = Busy
}

fact aReservedorBusyCarHasAlwaysARide {
all c: ElectricCar |(c.currentState in Reserved + Busy) implies
```

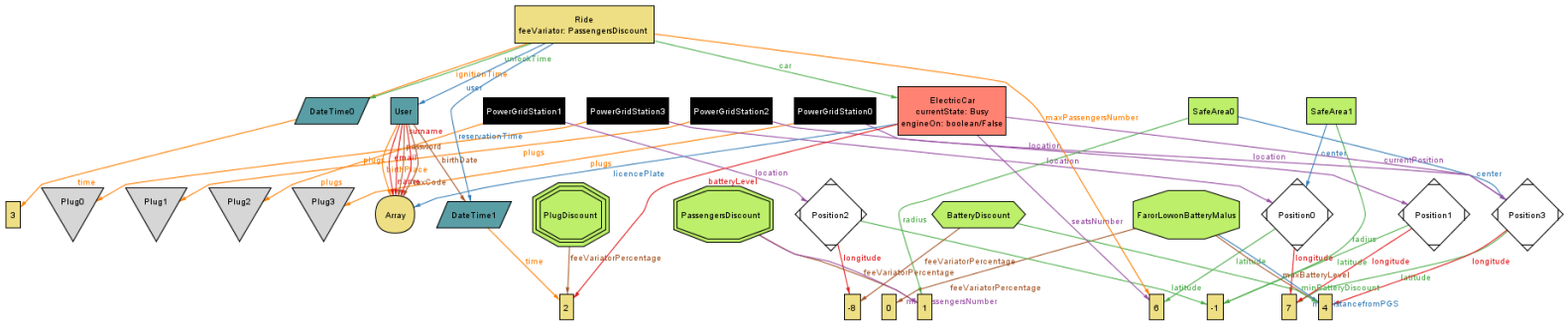
```
126 one r: Ride | r.car = c && rideisOngoing[r]
    }
128
    fact anOngoingRideHasACarReservedOrBusyAndViceVersa{
130 all r: Ride | #r.unlockTime = 0 && #r.endTime = 0
        implies r.car.currentState = Reserved
132 all r: Ride | #r.unlockTime = 1 && #r.endTime = 0
        implies r.car.currentState = Busy
134 all c: ElectricCar | c.currentState in (Busy + Reserved)
        implies one r: Ride | rideisOngoing[r]
136 }

138 fact passengerDiscountUnlocksCorrectly{
    all r: Ride, b: PassengersDiscount | r.maxPassengersNumber >=
140     b.minPassengersNumber iff b in r.feeVariator
    }
142
    fact noPassengersGreaterthanSeatsNumber{
144 no r: Ride | r.maxPassengersNumber > r.car.seatsNumber
    }
146
    fact no2PGSinTheSameSpot{
148 no disjoint psg1, psg2: PowerGridStation | psg1.location = psg2.location
    }
150
    fact eachPGSinaSA{
152 all psg: PowerGridStation | one sa: SafeArea |
        positionInSA[psg.location, sa]
154 }

156 fact eachPlugHasAPSG{
    all pl: Plug | one psg: PowerGridStation | pl in psg.plugs
158 }

160 fact eachBusyPlugHasACarPlugged{
    all psg: PowerGridStation | #psg.plugs.pluggedCar = 1 implies
162     psg.location = psg.plugs.pluggedCar.currentPosition
    }
164
    //Predicates
166
    pred positionInSA[p: Position, sa: SafeArea]{
168 plus[mul[sub[p.latitude, sa.center.latitude],
```

```
        sub[p.latitude, sa.center.latitude]],
170     mul[sub[p.latitude, sa.center.latitude],
        sub[p.latitude, sa.center.latitude]]]
172     <= mul[sa.radius, sa.radius]
    }
174
    pred haveCarOrUserinCommon[r1: Ride, r2: Ride]{
176 r1.user = r2.user or r1.car = r2.car
    }
178
    pred rideisOngoing[r: Ride]{
180 #r.endTime = 0
    }
182
    pred reserveACar[ui: User, uf: User,
184     ci: ElectricCar, cf: ElectricCar, r: Ride]{
    no ride: Ride | (ride.user = ui or ride.car = ci) && rideisOngoing[ride]
186
    r.car = cf && rideisOngoing[r] && cf.currentState = Reserved && r.user =
188     uf && rideIsConsecutive[r, ci]
    }
190
    pred endARide[ui, uf: User, ci, cf: ElectricCar, ri, rf: Ride]{
192 ri.car = ci && ri.user = ui && rideisOngoing[ri]

194 rf.car = cf && rf.user = uf && not rideisOngoing[rf]
    }
196
    pred rideIsConsecutive[r: Ride, c: ElectricCar]{
198 all ride: Ride | ride.car = c implies
        ride.endTime.time < r.reservationTime.time
200 }

202 //Functions
    fun busyPlug[psg: PowerGridStation]: Int{
204 #{pl: Plug| (pl in psg.plugs) && (#pl.pluggedCar = 1)}
    }
206
    fun carParkedIn[p: Position]: Int{
208 #{c: ElectricCar | c.currentState in {Available + Unavailable}
        && c.currentPosition = p}
210 }
```

```
212 //Assertion

214 assert noUnavailableCarCanBeRent{
    no r: Ride | r.car.currentState = Unavailable && rideisOngoing[r]
216 }

218 assert twoOngoingRidesImpliesTwoCarsAndTwoUsers{
    all disjoint r1, r2: Ride | rideisOngoing[r1] && rideisOngoing[r2] implies
220     r1.user != r2.user && r1.car != r2.car
    }
222
    assert allRidesAreConsecutive{
224 all disjoint r1, r2: Ride | haveCarOrUserinCommon[r1, r2] implies
        rideisOngoing[r1] && r2.endTime.time < r1.reservationTime.time or
226     rideisOngoing[r2] && r1.endTime.time < r2.reservationTime.time or
        !(rideisOngoing[r1] or rideisOngoing[r2]) && (r2.endTime.time <
228     r1.reservationTime.time or r1.endTime.time < r2.reservationTime.time)
    }
230
    assert unavailableCarCannotStart{
232 all c: ElectricCar | c.currentState = Unavailable implies c.engineOn = Fals
    }
234
    assert noPlugInTwoPSG{
236 no disjoint psg1, psg2: PowerGridStation, pl: Plug | pl in psg1.plugs
        && pl in psg2.plugs
238 }

240 pred show() {}

242 run show for 4 but exactly 1 Ride, exactly 1 ElectricCar
```

3.5.2 World Example

Below there is one possible solution of the Alloy logic model.

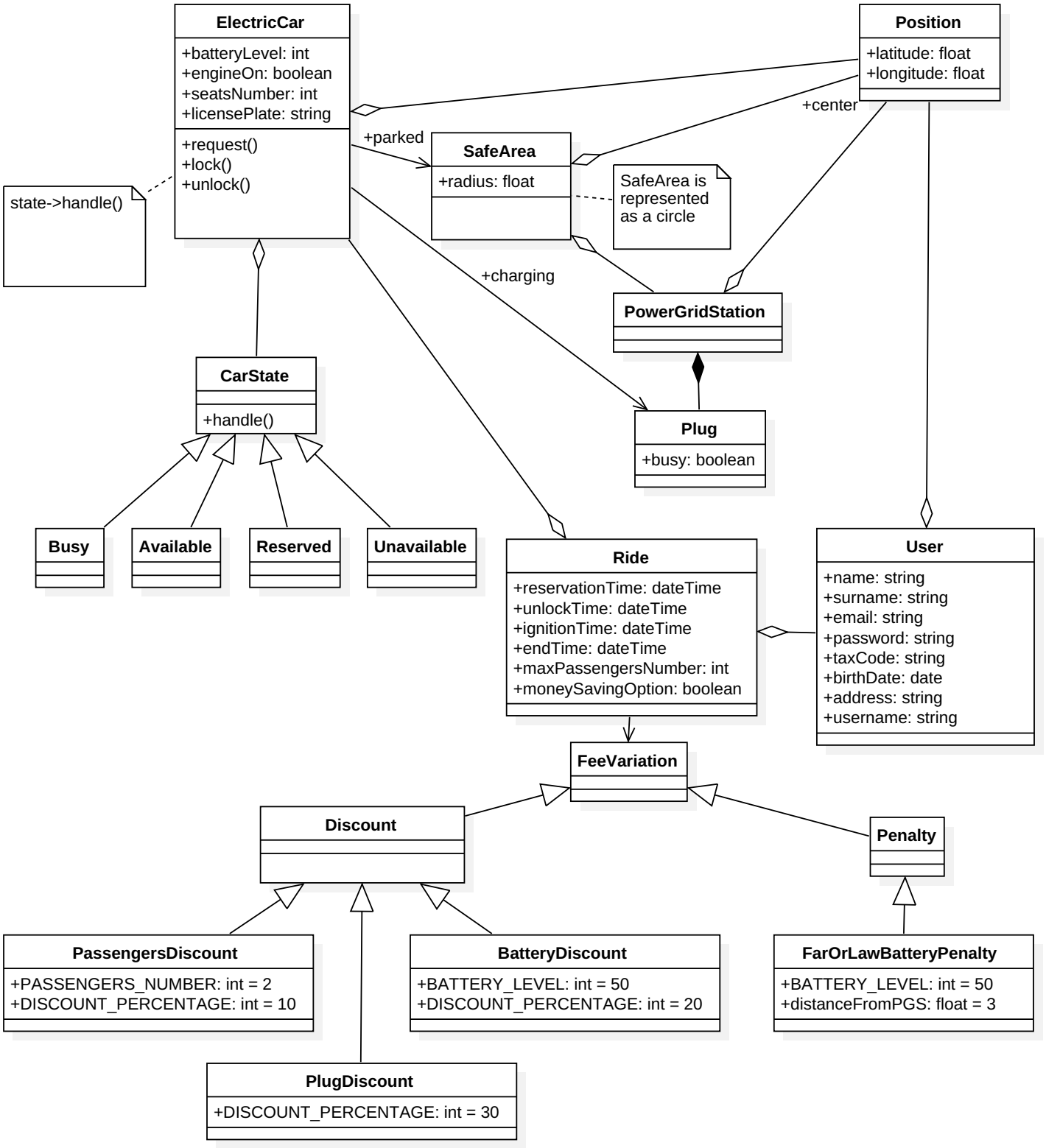

Ride
feeVariator: PassengersDiscount
unlockTime
ignitionTime
user
car
DateTime0
User
PowerGridStation1
PowerGridStation3
PowerGridStation2
PowerGridStation0
ElectricCar
currentState: Busy
engineOn: boolean/False
maxPassengersNumber
SafeArea0
SafeArea1
location
center
currentPosition
time
plugs
licencePlate
batteryLevel
radius
seatsNumber
latitude
longitude
3
Plug0
Plug1
Plug2
Plug3
Array
DateTime1
PlugDiscount
PassengersDiscount
Position2
BatteryDiscount
FarorLowonBatteryMalus
Position0
Position1
Position3
feeVariatorPercentage
2
-8
0
1
6
-1
7
-4

Chapter 4

Models

4.1 Class Diagram

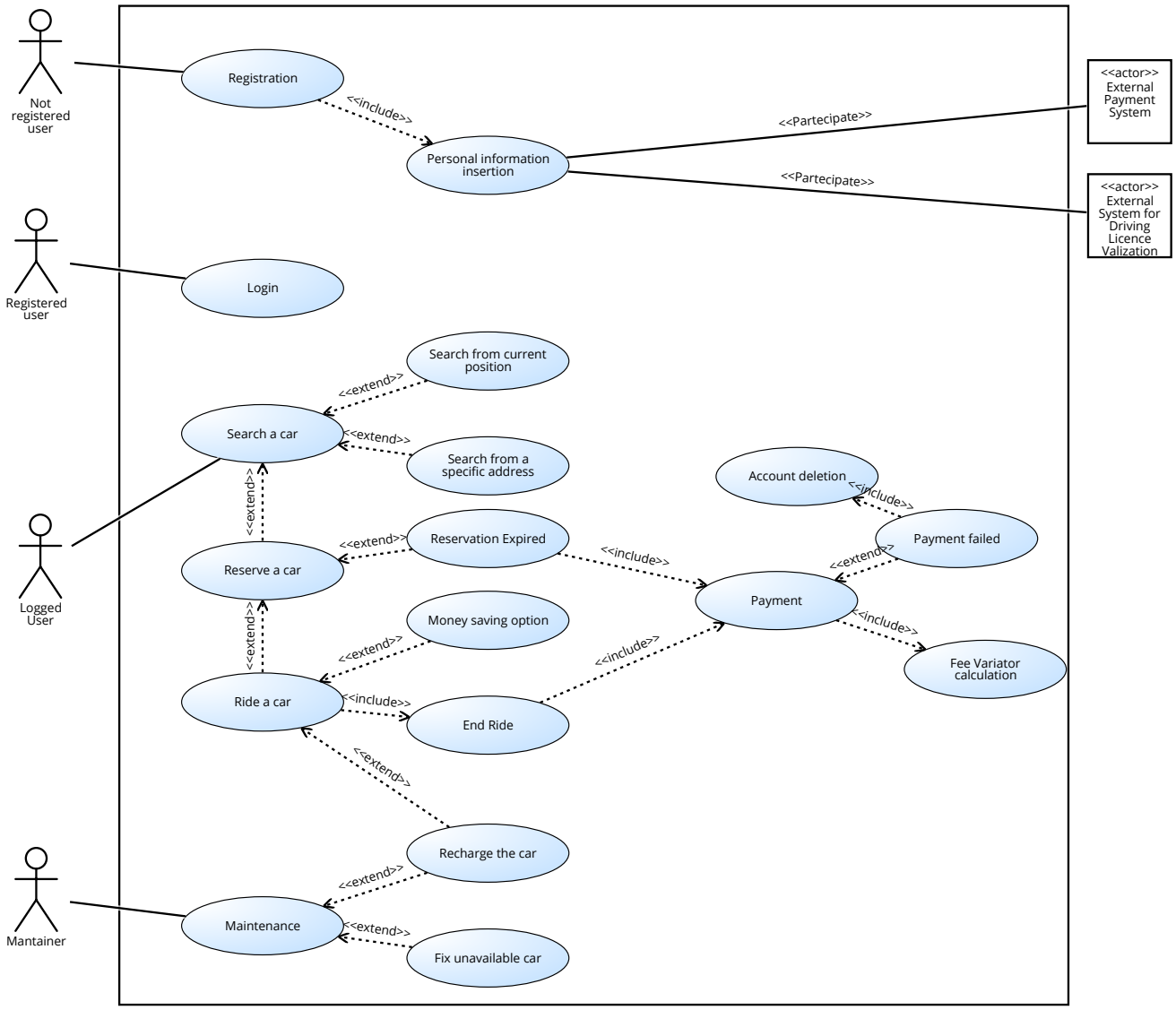
Below you can find the class diagram of the system.


ElectricCar
+batteryLevel: int
+engineOn: boolean
+seatsNumber: int
+licensePlate: string
+request()
+lock()
+unlock()
state->handle()
Position
+latitude: float
+longitude: float
+center
+parked
SafeArea
+radius: float
SafeArea is represented as a circle
+charging
PowerGridStation
CarState
+handle()
Plug
+busy: boolean
Busy
Available
Reserved
Unavailable
Ride
+reservationTime: dateTime
+unlockTime: dateTime
+ignitionTime: dateTime
+endTime: dateTime
+maxPassengersNumber: int
+moneySavingOption: boolean
User
+name: string
+surname: string
+email: string
+password: string
+taxCode: string
+birthDate: date
+address: string
+username: string
FeeVariation
Discount
Penalty
PassengersDiscount
+PASSENGERS_NUMBER: int = 2
+DISCOUNT_PERCENTAGE: int = 10
BatteryDiscount
+BATTERY_LEVEL: int = 50
+DISCOUNT_PERCENTAGE: int = 20
FarOrLawBatteryPenalty
+BATTERY_LEVEL: int = 50
+distanceFromPGS: float = 3
PlugDiscount
+DISCOUNT_PERCENTAGE: int = 30

4.2 Use Case Diagram

In this paragraph a complete Use Case diagram will be presented.

Use case PowerEnjoy



4.3 Use Case Description

In this paragraph all the 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: User, External system for Payments, External system for driving licence validation.
Entry conditions: User is not registered to the system.
Flow of events:

- The user opens the application and clicks on the button sign up in the homepage;
- The system shows him/her a registration form;
- The user fills the form with his/her data: name, surname, birth date, driving licence number, credit card number, username and password. Then he/she submits it;
- The system checks the correctness of the data inserted by the user and verifies the validity of the credit card and of the driving license through two external systems;
- If all the data are correct the system sends him/her a confirmation email of the registration.

Exit conditions: The user is successfully registered to the system and now can log in.
Exceptions:

- The information inserted by the user are not valid or some informations are missing. In this case the registration fails and the user is redirected to the registration form.
- The external system for credit card validation is not available. In this case the user cannot complete the registration until the problem will be fixed.

User log in

Name: User log in.
Actors: User.
Entry conditions: The user is registered to the system and he/she is not banned.
Flow of events:

- The user opens the application and clicks on the button log in in the homepage;
- The system shows him/her a form in which he/she can insert the username and the password;
- The user fills the form and then clicks on submit it;
- The system checks the correctness of the username and password;

Exit conditions: The user is successfully logged into the system and now he/she can access to all functionalities.
Exceptions:
The username and password inserted by the user are not correct. In this case the user is not logged in and he/she will be redirected to the login form.

Search Car

Name: Search Car + Search from current location + Search in a specific address.
Actors: User.
Entry conditions: The user is logged into the system and he/she is not banned.
Flow of events:

- The user chooses the option "Search Car";
- The system detects the user's position through the GPS and starts searching for cars within a range of 1 km from his/her position;
- The system allows the user to choose the option "Search in a specific address";
- If the user chooses "Search in a specific address" the system will ask him/her to insert the address and then it will verify the correctness of the inserted data;

- The system will shows on the map the available cars that satisfy the parameters, with their unique number and their current battery.

Exit conditions: The user can visualize on the map the available cars that fits the parameters.
Exceptions:

- The are no available cars that satisfy the parameters. In this case the user is notified of the unavailability of cars.
- The address inserted by the user is not correct. In this case the system will ask the user to insert it again.

Reserve Car

Name: Reserve Car.
Actors: User.
Entry conditions:

- The user is logged into the system and he/she is not banned;
- The user has used the "Search Car" function and has the results showed on the map;

Flow of events:

- The user selects one of the car that he/she can visualize on the map;
- The system shows the user all the informations about the selected car with an option to reserve it;
- The system checks if the car is still available and if the user has not any other reservation;
- If the car is still available the system sends the reservation to the car;
- The system notifies the user about the result of the operation;

Exit conditions: The car is reserved by the user.
Exceptions:

- The car is not available any more. In this case the user is notified and he/she is redirected to the map where he/she can choose another car.
- The user has already reserved another car. In this case the user is notified and redirected to the homepage.

Ride

Name: Ride.
Actors: User.
Entry conditions:

- The user is logged in;
- The user in not banned;
- The user that pick up the car is the same user that has reserved it.

Flow of events:

- The user reaches the location of the car following the directions given by the mobile app;
- The system shows on the mobile application of the user the option to unlock the car when he/she is within a range of 6 metres from the car;
- The user chooses to unlock the car;
- The system takes charge of the request and unlocks the car;
- The user enters into the car;
- The user inserts his/her personal password onto the tablet inside the car in order to start the ride;
- The user ignites the engine and the system is notified;
- The system starts computing the fee with a predefined amount of money per minute and shows it on the car screen.

Exit conditions: The user is now able to drive the car.
Exceptions:

- The user has unlocked the car but after 3 minutes he/she is not entered inside the car. In this case the car will be locked again;
- The user cannot unlock the car even if he/she is within a range of 6 metres from the car. In this case he/she will can use the help button to notify the problem to the system.

Reservation Expire

Name: Reservation Expire.
Actors: User, External system for payments.
Entry conditions:

- The user is logged into the system and he/she is not banned;
- The user has reserved a car;
- One hour has passed from the user's reservation.

Flow of events:

- The status of the car changes from reserved to available if the system verifies that the car has not been picked-up by the user;
- The system charges 1 euro to the user;
- The external system for payments takes charge of the request and invoke the **Payment** use case;
- The user is notified about the result of the operation.

Exit conditions: The user is now able to reserve a car again.
Exceptions:

- **Payment Failed** use case;
- The external system for payments is not available. In this case the payment will be performed as soon as the it will become available again.

Recharge Car

Name: Recharge Car.
Actors: User, Maintainer.
Entry conditions:

- The car is parked next to a power grid inside a special area;
- The car engine is off;

Flow of events:

- The driver goes out of the car and plugs the car in one of the power grid;

- The system is notified when the car is plugged into the power grid.

Exit conditions: The car has been recharged.
Exceptions:

- The car has been parked in an unsafe area or is not able to reach the nearest power grid station with the current battery. In this case the maintainer reach the area where the car has been parked, start charging the car on site and then he/she take it to a safe area. At this point the system is notified that the car has been charged by the maintainer and its status is changed from unavailable to available.
- The power grid has a problem and doesn't recharge the car. In this case the user can use the help button and notify the problem to the system.
- The user recharge the car and don't end the ride. In this case he/she can continue to use it but the Plug Bonus won't be calculated.

Maintenance

Name: Maintenance.
Actors: Maintainer.
Entry conditions:

- The car is unavailable;

Flow of events:

- The mainainer reached an unavailable car.
- The mainainer charges the car and fixes it.

Exit conditions: The system changes car status from unavailable to available.
Exceptions:

- The mainainer cannot fix the unavailable car. In this case he/she takes the car to the garage.

End Ride

Name: End Ride.
Actors: User.
Entry conditions:

- The car is inside a safe area;
- The user decides to end the ride.

Flow of events:

- The user turns off the engine of the car and clicks on the proper button to end the ride;
- The system is notified about the end of the ride;
- The system detects if the car has been parked in a safe area;
- The system waits until there are no more passengers on the car;
- The system stops charging the user;
- The system locks the car and changes his status from busy to available;
- The system waits 10 minutes before invoking the **Discount calculation** use case to give him the time to plug the car into a power grid;
- The system charges the calculated fee to the user;
- The system invoke the **Payment** use case.

Exit conditions:

- The car is available again;
- The user is able to reserve a car again.

Exceptions:

- The user leaves the car outside a safe area. In this case the user cannot end the ride and he/she will continue to pay until the limit time for a ride has been reached. In addiction a maintainer will take charge of moving the car to a safe area;
- The user turns off the car but one of the passengers or the user does not go out of the car within 3 minutes. In this case the system will start charging an additional fee per minute to the user;

- **Payment Failed** use case;
- The external system for payments is not available. In this case the payment will be performed as soon as the it will become available again;
- The user plugs the car to the power grid after 10 minutes from the end of the ride. In this case the discount will not be applied.

Discount calculation

Name: Discount calculation.
Actors: No actors but only the System.
Entry conditions: The user ends the ride.
Flow of events:

- The system checks through the sensor of the car the number of passenger, the level of battery, the GPS position and if the car is plugged to a power grid.
- The system applies the rightful fee variators.
- The system calculates the final fee that the user has to pay and shows it on the screen of the car.

Exit conditions: The user can visualize the final fee with all the fee variators applied.
Exceptions: The user plugs the car without ending the ride and decide to use it again after the recharge. In this case the plug bonus is not applied.

Payment

Name: Payment.
Actors: External payment system.
Entry conditions:

- The ride is ended or the reservation has expired;
- The eventual fee variators have been applied.

Flow of events:

- The system sends a request to the external system for payments with the credit card information of the user and the the final fee that the user has to pay;

- The external system for payments receive the requests and takes it in charge;
- The payment is performed.

Exit conditions: The bill has been paid.
Exceptions:

- **Payment Failed** use case;
- The external system for payments in not available. In this case the payment will be performed as soon as the it will become available again.

Payment failed

Name: Payment failed.
Actors: External payment system.
Entry conditions:

- The System has sent to the external payment system the request of charging the user with his/her current fee.
- The external payment system has tried to perform the payment.

Flow of events:

- The payment failed because the user has not enough money to pay the fee;
- The System is notified by the external payment system;
- The System bans the user from the system.

Exit conditions: The user is banned.
Exceptions: No exceptions.

Money Saving Option

Name: Money Saving Option.
Actors: User.
Entry conditions:

- The user is inside the car;

- The ride is not ended.

Flow of events:

- The user activates the Money Saving Option from the button present on the car;
- The system asks the user to insert his/her final destination;
- The user types his/her final destination;
- The system determines the power grid station where the user can leave the car by taking into account the distance from the user's destination and the available plugs in that station;
- The system shows the information about this power grid station on the screen of the car.

Exit conditions: The Money Saving Option is enabled.
Exceptions: The user inserts a destination that is not valid. In this case the system will ask the user to insert another destination.

4.4 Car Status Flow chart

The following diagram represent the status flow chart of the cars.

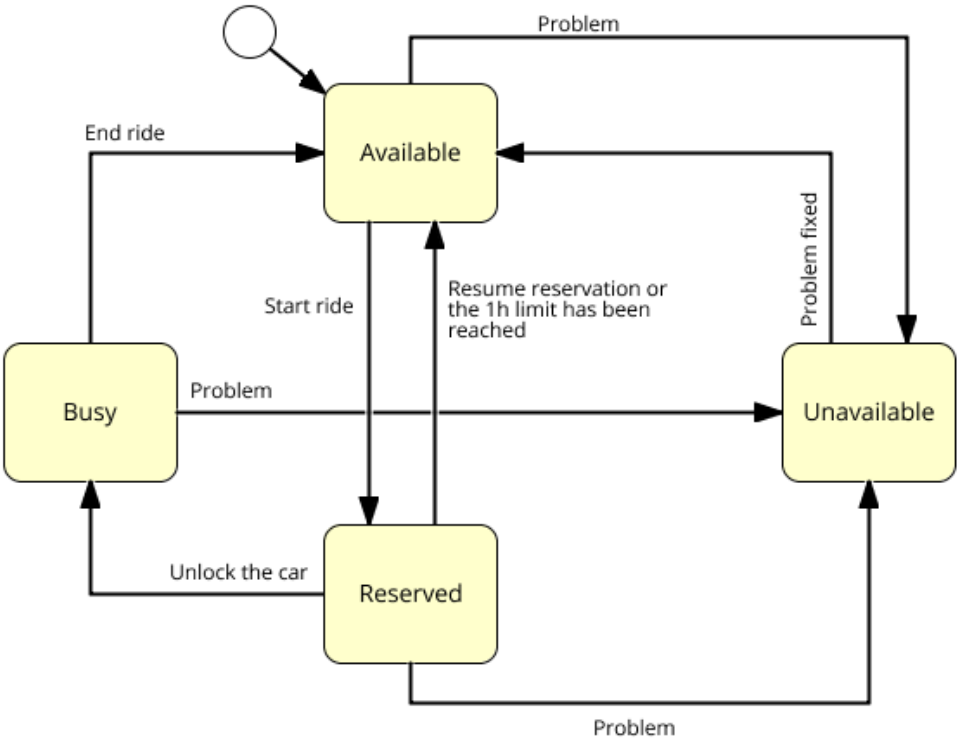
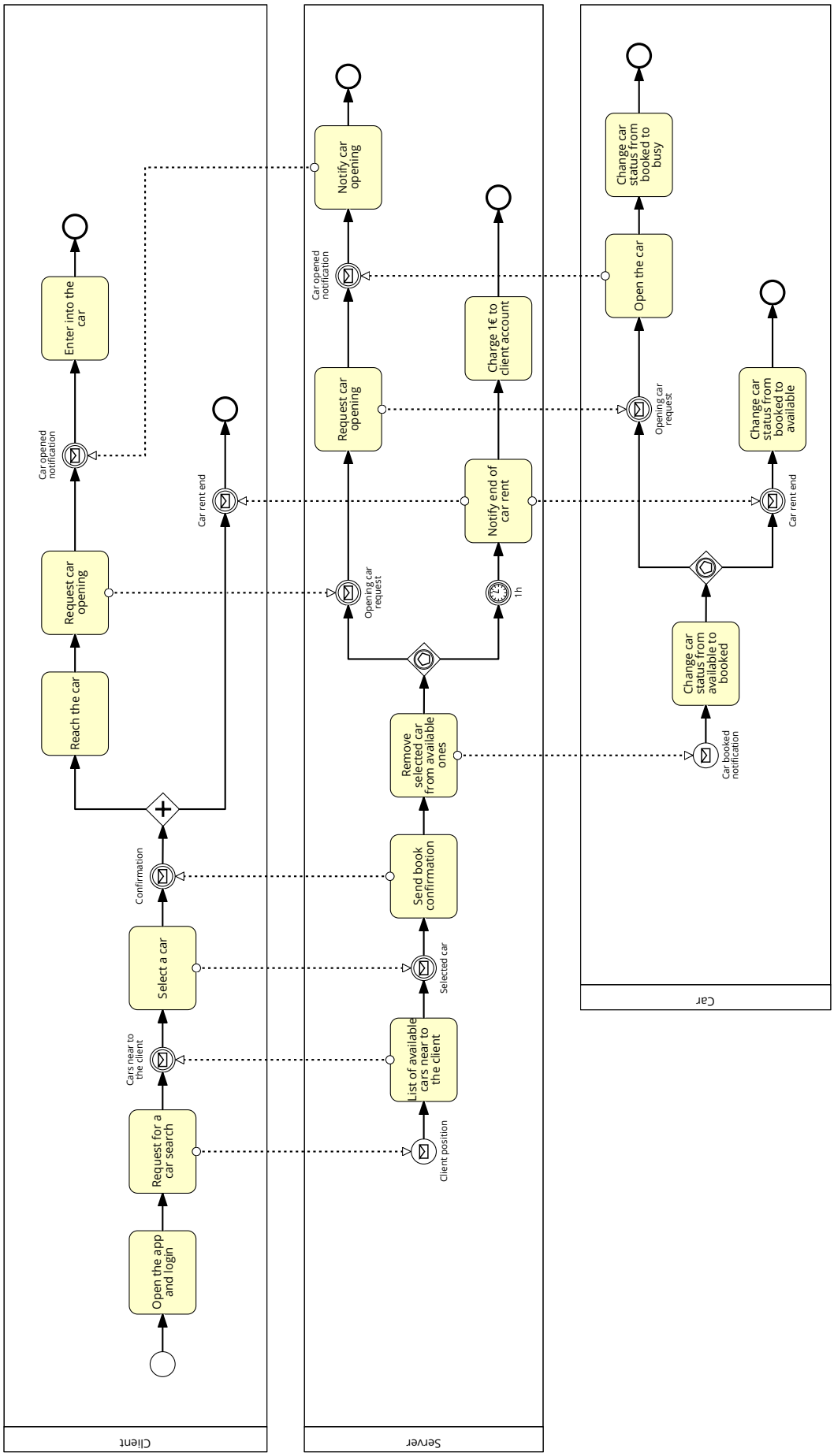


Figure 4.1: Car Status Flow chart.

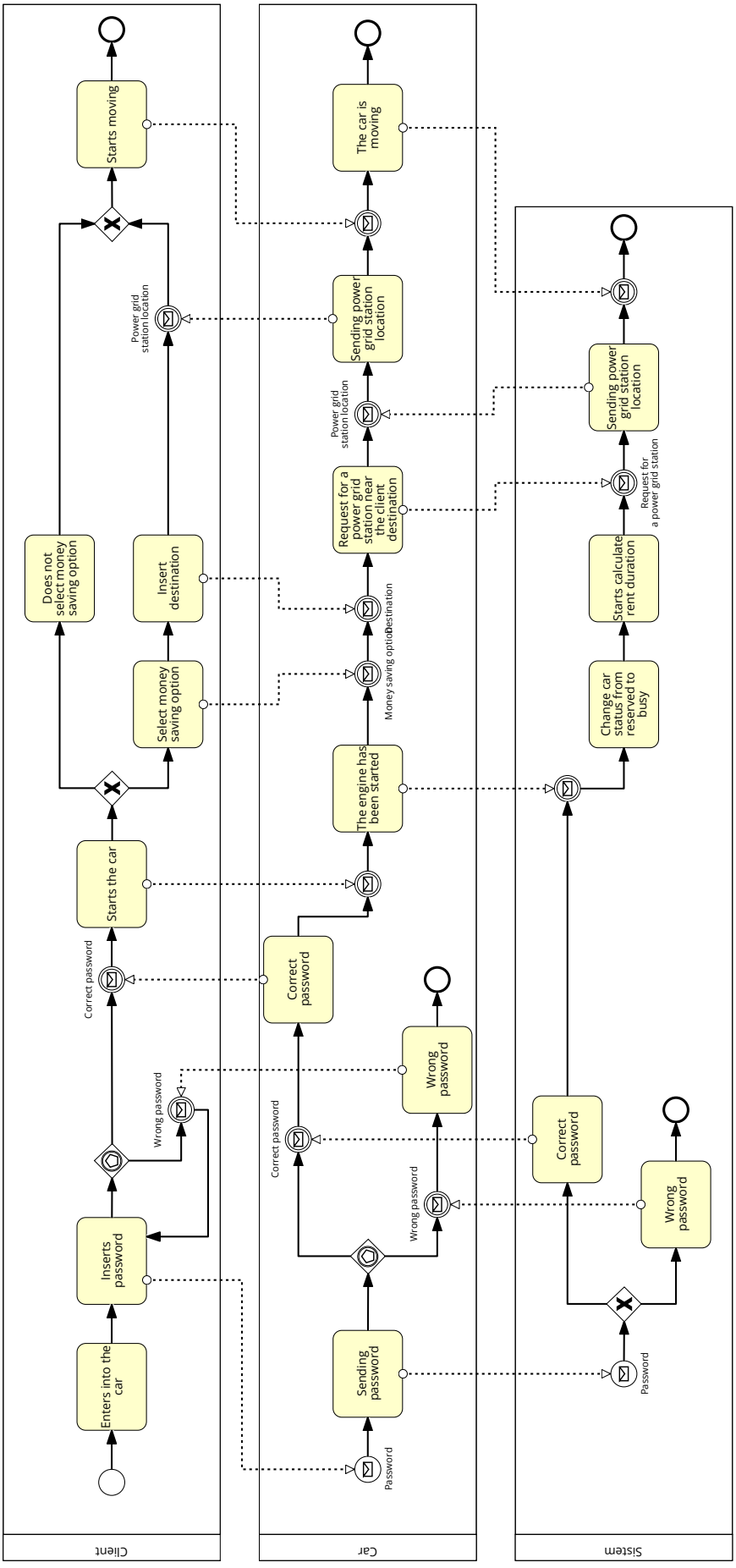
4.5 BPMN

The following three BPMN represent the whole rent. The first one follows the user from the login up to the moment when he or she enters into the car. The second one follows the user from the moment when he or she starts the car up to when he or she starts driving. The third one follows the user from the moment when he or she starts driving up to the rent end.

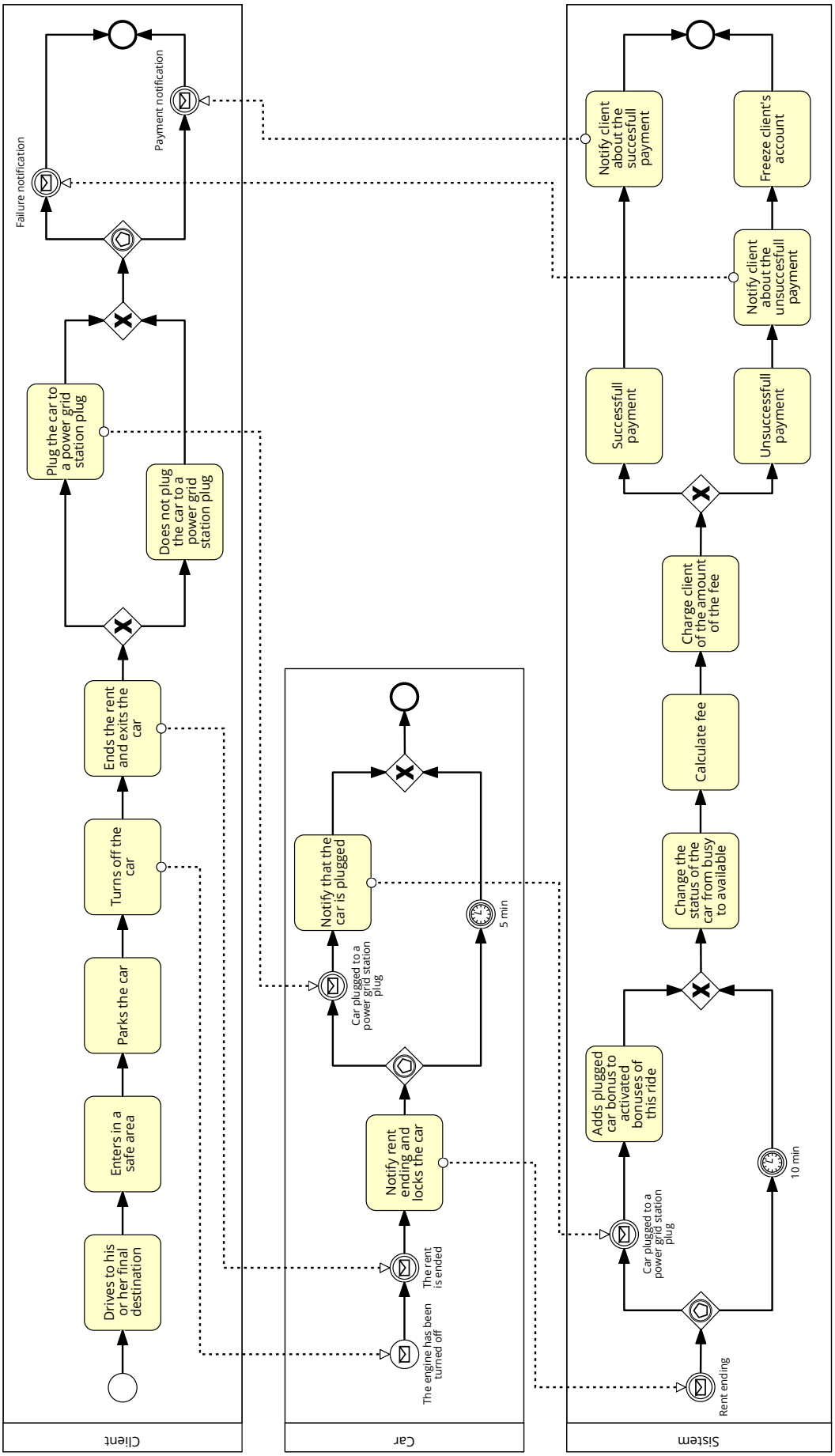
Rent Start BPMN



Rent BPMN

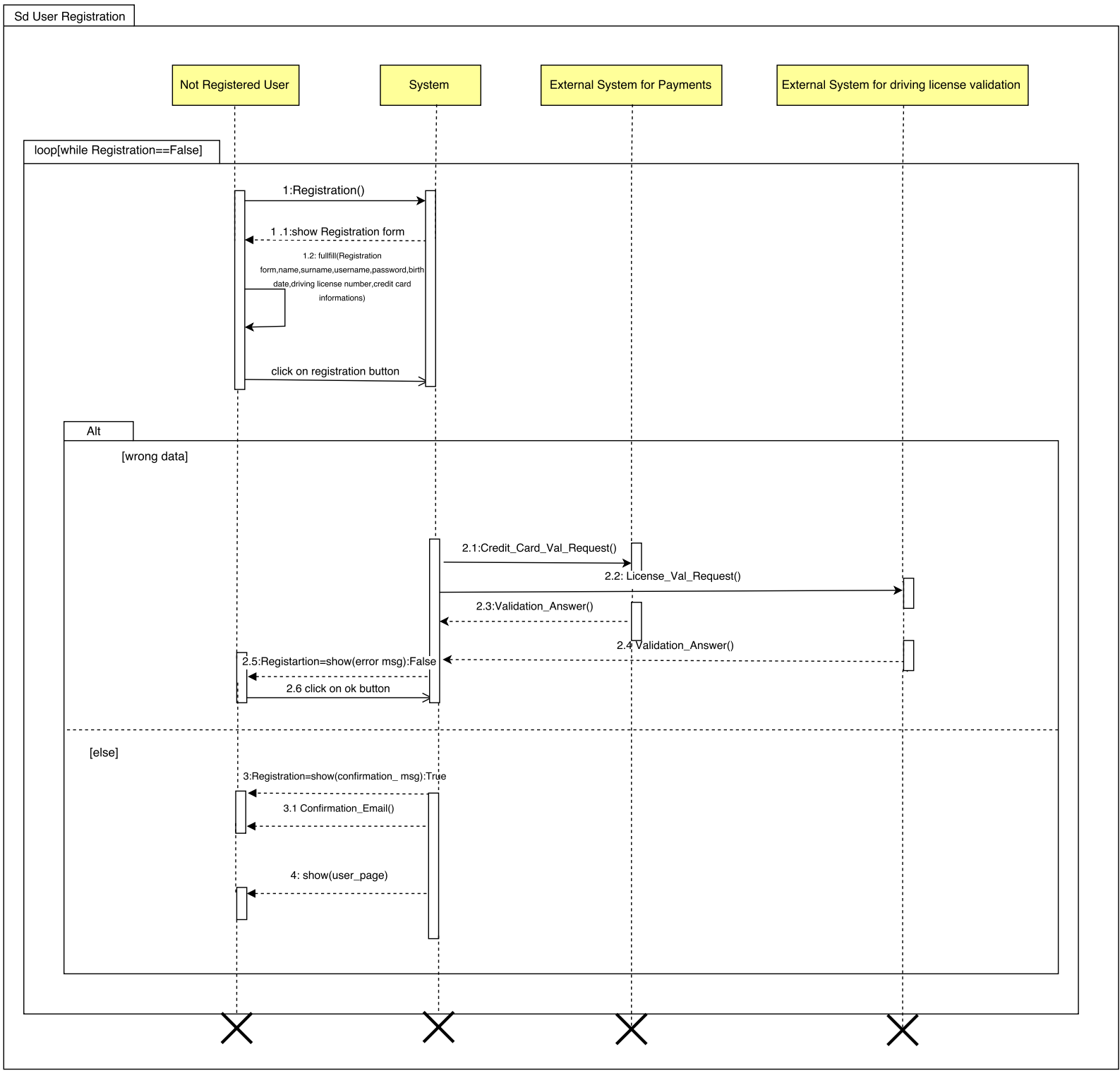

Client
Car
System
Starts moving
Enters into the car
Inserts password
Wrong password
Correct password
Starts the car
Does not select money saving option
Select money saving option
Insert destination
Power grid station location
The car is moving
Sending password
Password
Wrong password
Correct password
The engine has been started
Money saving option
Destination
Request for a power grid station near the client destination
Power grid station location
Sending power grid station location
Password
Wrong password
Correct password
Change car status from reserved to busy
Starts calculate rent duration
Request for a power grid station
Sending power grid station location

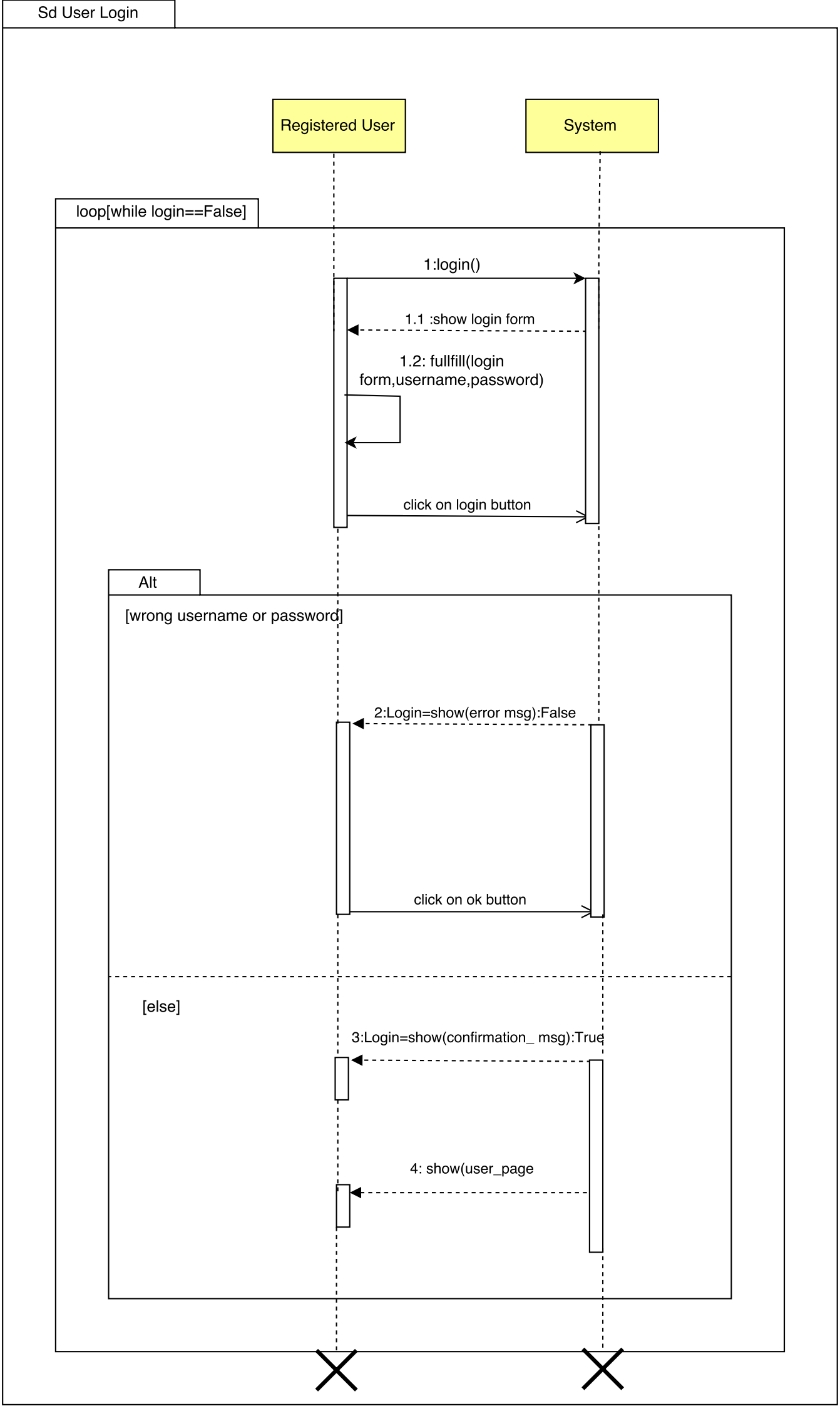
Rent Ending BPMN

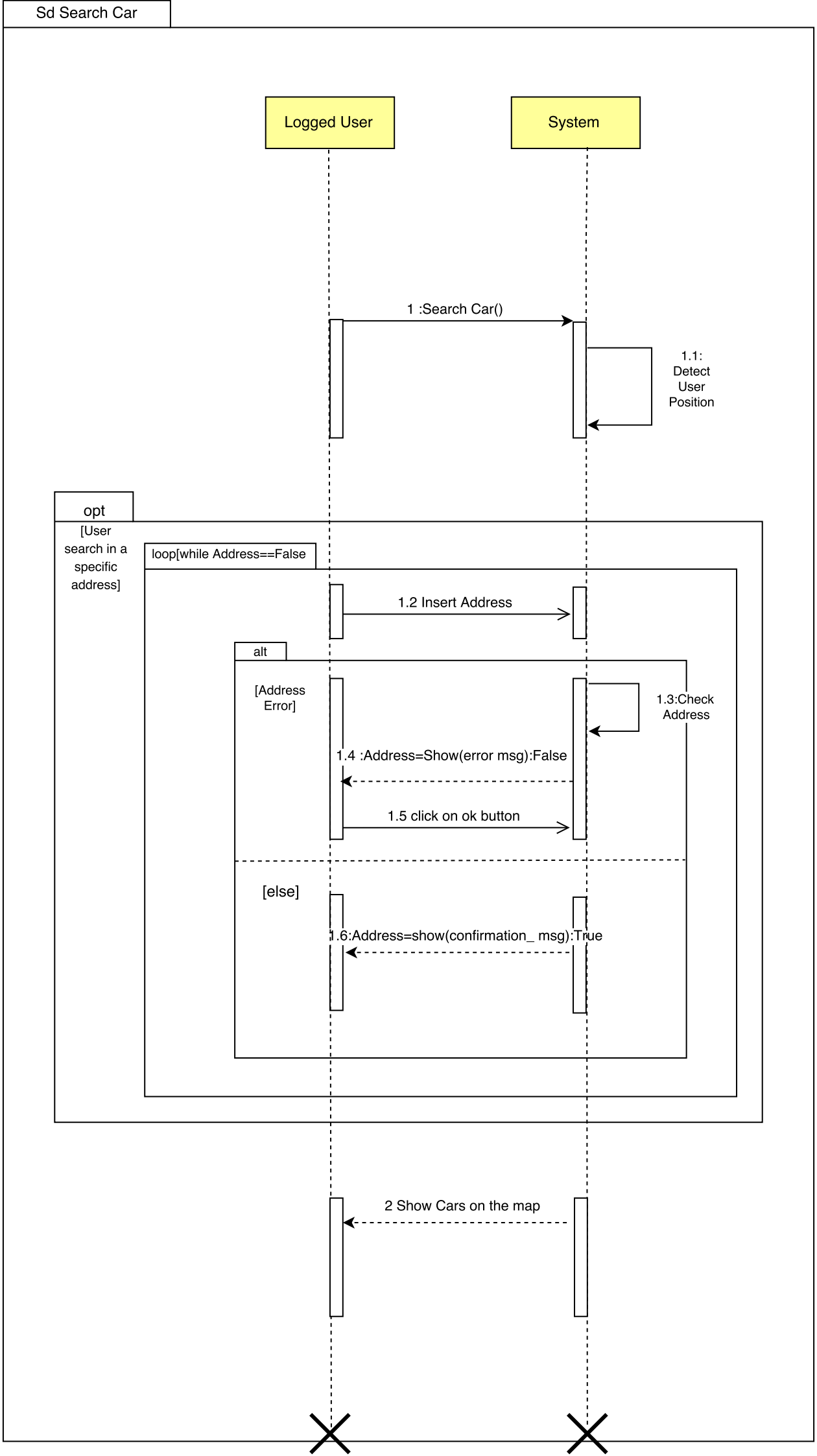


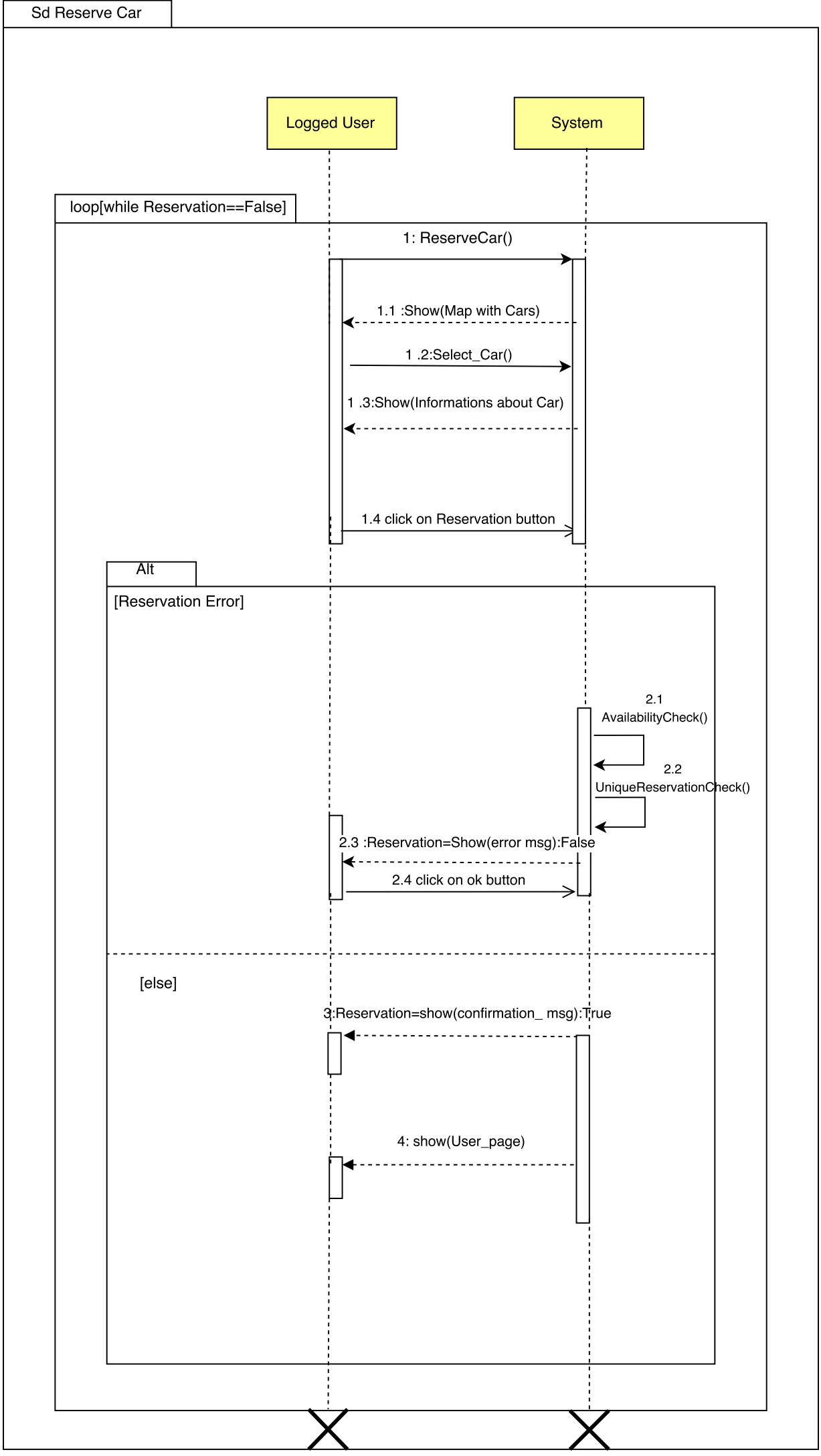
4.6 Sequence Diagram

In this paragraph some of the sequence diagram will be described. These diagrams can be derived from the use case descriptions.

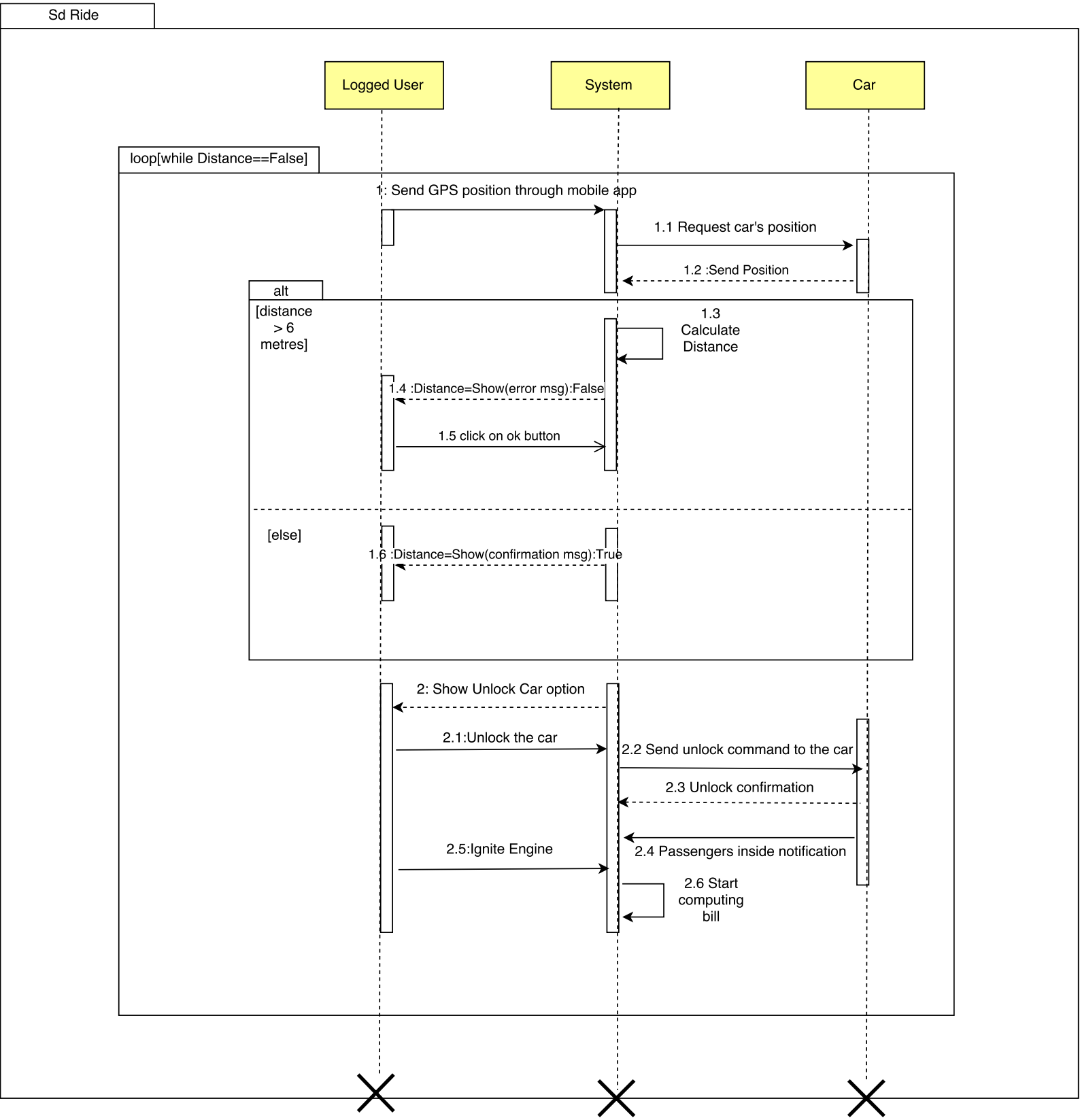

Sd User Registration
Not Registered User
System
External System for Payments
External System for driving license validation
loop[while Registration==False]
1:Registration()
1 .1:show Registration form
1.2: fullfill(Registration form,name,surname,username,password,birth date,driving license number,credit card informations)
click on registration button
Alt
[wrong data]
2.1:Credit_Card_Val_Request()
2.2: License_Val_Request()
2.3:Validation_Answer()
2.4 Validation_Answer()
2.5:Registartion=show(error msg):False
2.6 click on ok button
[else]
3:Registration=show(confirmation_ msg):True
3.1 Confirmation_Email()
4: show(user_page)

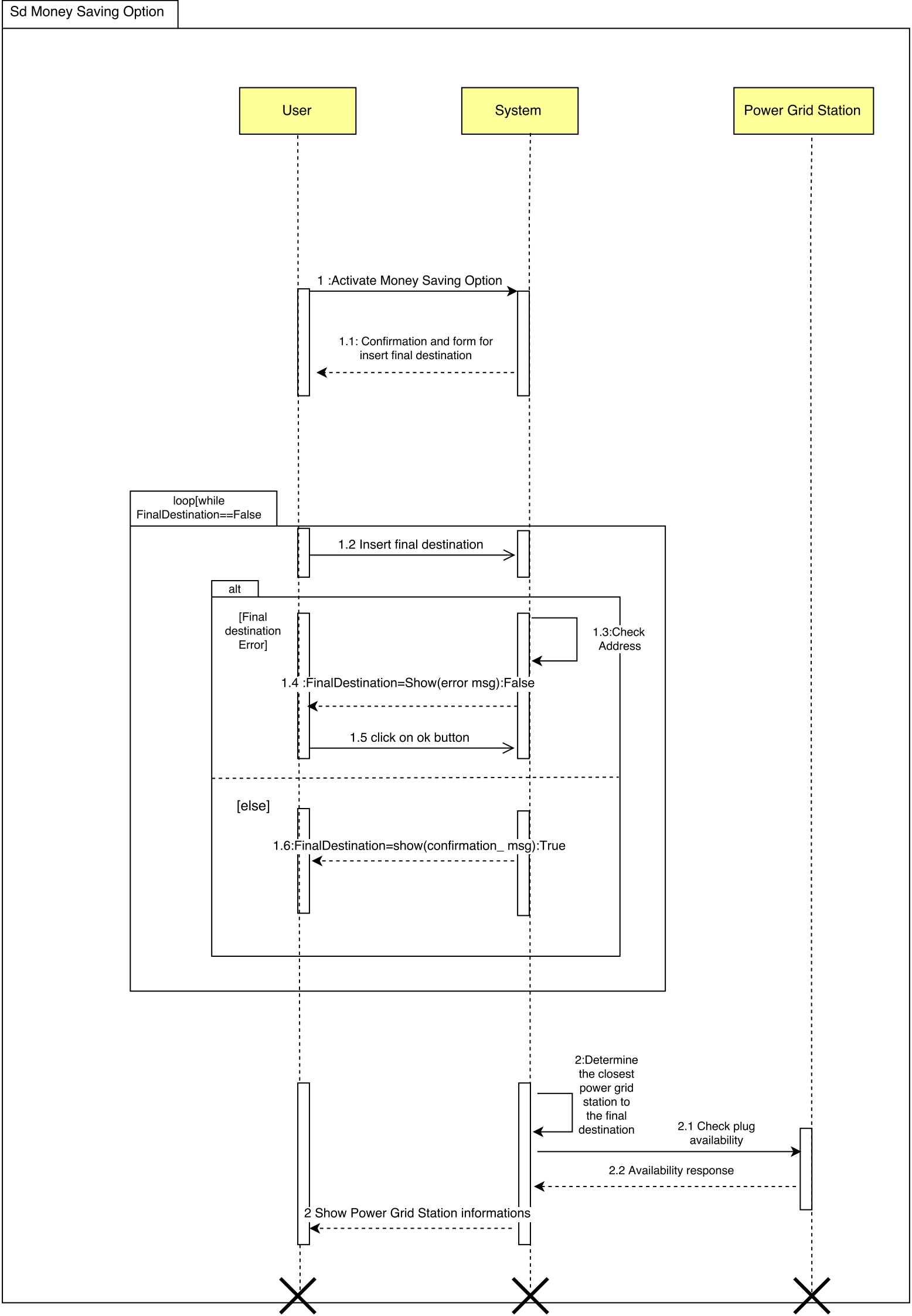

Sd User Login
Registered User
System
loop[while login==False]
1:login()
1.1 :show login form
1.2: fullfill(login form,username,password)
click on login button
Alt
[wrong username or password]
2:Login=show(error msg):False
click on ok button
[else]
3:Login=show(confirmation_ msg):True
4: show(user_page


Sd Search Car
Logged User
System
1 :Search Car()
1.1:
Detect
User
Position
opt
[User
search in a
specific
address]
loop[while Address==False
1.2 Insert Address
alt
[Address
Error]
1.3:Check
Address
1.4 :Address=Show(error msg):False
1.5 click on ok button
[else]
1.6:Address=show(confirmation_ msg):True
2 Show Cars on the map


Sd Reserve Car
Logged User
System
loop[while Reservation==False]
1: ReserveCar()
1.1 :Show(Map with Cars)
1 .2:Select_Car()
1 .3:Show(Informations about Car)
1.4 click on Reservation button
Alt
[Reservation Error]
2.1
AvailabilityCheck()
2.2
UniqueReservationCheck()
2.3 :Reservation=Show(error msg):False
2.4 click on ok button
[else]
3:Reservation=show(confirmation_ msg):True
4: show(User_page)

Text


Sd Ride
Logged User
System
Car
loop[while Distance==False]
1: Send GPS position through mobile app
1.1 Request car's position
1.2 :Send Position
alt
[distance > 6 metres]
1.3 Calculate Distance
1.4 :Distance=Show(error msg):False
1.5 click on ok button
[else]
1.6 :Distance=Show(confirmation msg):True
2: Show Unlock Car option
2.1:Unlock the car
2.2 Send unlock command to the car
2.3 Unlock confirmation
2.4 Passengers inside notification
2.5:Ignite Engine
2.6 Start computing bill


Sd Money Saving Option
User
System
Power Grid Station
1 :Activate Money Saving Option
1.1: Confirmation and form for insert final destination
loop[while FinalDestination==False
1.2 Insert final destination
alt
[Final destination Error]
1.3:Check Address
1.4 :FinalDestination=Show(error msg):False
1.5 click on ok button
[else]
1.6:FinalDestination=show(confirmation_ msg):True
2:Determine the closest power grid station to the final destination
2.1 Check plug availability
2.2 Availability response
2 Show Power Grid Station informations

Appendix A

Appendix

A.1 Used software and tools

- LaTeX [1], for typesetting this document.
- Texmaker[2], for the writing of this document.
- GitHub[3] for version control and distributed work.
- Evolus Pencil[4] for the mockups.
- StarUML[5] for the class diagram.
- Alloy Analyzer[6] used to build the model generated by the alloy code.
- Signavio Academic[7] for the use case diagram and for BPMN.
- GitHub desktop[8] used to collaborate in the team and to keep track of the changes.

A.2 Changelog

v1.1:

[1] https://www.latex-project.org/
[2] http://www.xm1math.net/texmaker/
[3] https://github.com/
[4] http://pencil.evolus.vn/
[5] http://staruml.io/
[6] http://alloy.mit.edu/alloy/
[7] http://academic.signavio.com/p/login
[8] https://desktop.github.com/

- added car status flow chart;
- corrected typos;
- changed some mock-ups.

v1.0:

- initial release.

A.3 Work hours

The statistics about commits and code contribution are available on the GitHub repository of the project[9]. Please keep in mind that some commits are the joined effort of two or all the components of the group. However, when this is the case, it is specified in the description of the commit.

These are our estimation of the work hours spent on this project:

- Marco Ieni: 38 hours
- Francesco Lamonaca: 40 hours
- Marco Miglionico: 36 hours

[9] `https://github.com/marcomiglionico94/Software-Engineering-2-Project`