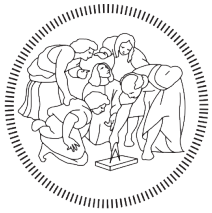


POLITECNICO
MILANO 1863

Requirements Analysis and Specifications Document for PowerEnJoy

Daniele Riva* Marco Sartini†

February 7, 2017

version 1.1

*matr. 875154
†matr. 877979

Contents

Revision History

Name	Date	Reason For Changes	Version
Marco + Daniele	13/11/2016	Starting	1.0
Marco + Daniele	07/02/2017	Final revision	1.1

1 Introduction

1.1 Purpose

Power EnJoy is a digital management system for a car-sharing service that exclusively employs electric cars. The system provides the basics car-sharing services plus other functionalities listed in this document. This is a full new system to be develop. This document describes all the parts of the system.

1.2 Document Conventions

In this document we will use the italic style when a specific concept is expressed with more than a single word. This document follows the IEEE standard about Software Requirements Specification. In particular it complies to:

- IEEE Standard for RASD Adapted from ISO/IEC/IEEE 29148 dated December 2011;
- IEEE Std 830, IEEE Recommended Practice for Software Requirements Specifications;
- IEEE Std 1233, IEEE Guide for Developing System Requirements Specifications;

.

1.3 Intended Audience and Reading Suggestions

This document is written with the purpose of specify the requirements of the project. This document is intended for users, developers, project managers and testers. Reading this document in its printed sequence will guarantee the best experience and understanding. Before proceed, please be sure to appreciate the proper meaning of special words listed in the Appendix A: Glossary.

1.4 Project Scope

In the era of sharing economy, a noticeable sector is the one related to cars. The digital management system *Power EnJoy* will be a useful system for shared electric cars users that will manage to book, pick up, drive and pay an available car in a city where the company will operate.

1.5 References

This SRS refers to the assignment provided by the Software Engineering 2 professors available on the private BeeP platform at Polimi and also in the GitHub repository http://github.com/marcosartini/PowerEnJoy as "Assignments AA 2016-2017.pdf" Other specifications may be found at the BeeP Forum in the project section.

2 Overall Description

2.1 Product Perspective

The system shown in this paper will implement an infrastructure that allows the company to care of its own cars, knowing where they are and their state. On the user side, this system allows the user to become a user, and allows a user to book a car in the location he prefer, it allows him to pick up the reserved car, have a trip, release the car and it will also take care of billing the user.

2.2 Product Functions

The goals of the system are better specified in separated sections belonging one the client-side and one the company-side.

2.2.1 Client-side

The system, related to the client:

G1. Allows potential users to sign up the system (information, credential and payments data);

G2. Allows users to sign in and use the system;

G3. Allows users to find the locations of available cars near their own current position or near a specified address;

G4. Allows users to book a car for up one hour in advance to pick it up;

G5. Allows users to tell the system they are nearby the car;

G6. Allows a user to see the amount of his current ride through a screen on the car;

G7. Allows users to notify the system a problem related to the car;

G8. Allows users to keep aside their car for a little time (max 3 hours, under specific payment);

2.2.2 Company-side

The system, related to the company:

G9. Allows to compute the amount of each user's ride [starts debit, stops debit, discount];

G10. Allows to manage the opening and closing of the cars;

G11. Allows to stimulate the users' virtuous behaviors applying discounts on their last ride as and deter the malicious of the users applying extra fees as stated in the business rule section (see 5.5 on page 32);

G12. Allows to check the status of the cars as a prospectus/summary;

G13. Allows the company to be notified if any *on-board system* detects a car failure;

G14. Allows to edit settings such as prices per minute, discounts amount, "etc.";

G15. Allows the company to bill users for services they benefited;

G16. Allows to edit stored information as cars, Power Grid stations and Safe Areas.

2.3 User Classes and Characteristics

This system will be used by the people who intend to drive a rented electric car, by the employee of the company and by the management of the company.

user client of the company, he provide his own data, he request a rental;

maintenance employee take care of problems, of discharged cars, etc;

management employee watch maps of the cities crowded of cars, checks revenue, etc.

payment handler it is in charge of collecting the amounts for services the user benefited. The company communicate it the debit of the service and it takes care of collect it, giving feedback of the outcome. It is his duty to insist to get the amount due, even proceeding to court.

2.4 Design and Implementation Constraints

The mobile functionality is inherent in the definition of the business, because the cars are moving, and users want to have the service immediately when they need it, wherever they are, without having to forcibly use a desktop computer from their office. Therefore, a mobile application developed for the majors mobile OS is required (at the moment they are Google Android and Apple iOS). Other users with different operating systems should be however able to interact with *Power EnJoy* using a browser and surfing the website version of the interface. Both interface versions only work if supported by a running

Internet connection. At the moment, the most widespread technology for geolocalization is GPS. It is extremely required that the system supports the simultaneous interaction of different users with it.

2.4.1 Regulatory policies

Authorization to access the location should be granted by the user to the system, in order to comply privacy laws. Denied authorization implies the user manually enter an address when required by the system.

2.5 User Documentation

The user will find a dedicated section in the website and in the application devoted to explanation of the functionalities of the system, so he will be able to proper operate. Will be also provided a manual reserved to the company-side, to show how to use the system.

2.6 Assumptions and Dependencies

2.6.1 Text Assumption

TA1. System is based on a mobile application and web site to interact with users who want to use *Power EnJoy* services;

TA2. A part of the system, called "on-board system" is mounted on every car and is the bridge between the central system and the client system;

TA3. We assume that passengers hop on and off the same way the driver.

TA4. We assume that the system generates a specific code when a user wants to tell the system he is nearby his requested car;

TA5. We assume that payment management is entrusted to a third party company, that receives withdrawal requests from our company;

TA6. We assume that a fine attributable to a user broken law is paid by the company and the company will force a refund by the user;

TA7. We assume that the management of procedures in case of accidents, damage to cars, disputes are handled by third party company and don't affect our system;

TA8. This is a brand new business, therefore no legacy system exists;

TA9. We assumed that discounts are not combinable;

TA10. We assume that the system checks the correct spelling of the data sent by the user in the registration step, inter alia: CF, driver license, payment data;

TA11. We assume that if a car is plugged into a power grid station within 4 (four) minutes since a rental is over, the action is due to the user who last rented the car;

2.6.2 Domain properties

D1. All the position detector system always give the right position;

D2. The position detector transceiver of the cars cannot be switched off, unless the car is out of order;

D3. All cars are registered in the system;

D4. When a new car is acquired by the company, the car is being registered in the system;

D5. When a car is out of order, the car is being removed from the system;

D6. Cars are equipped with a display able to receive data to show;

D7. The number of passengers is always nonnegative and less than the capacity of the car;

D8. A booked car has only one user and one driver a time;

D9. Identifying document and driver license of a user are verified by the company autonomously;

D10. The geographical positions of the power grid stations are known;

D11. The geographical positions of the safe areas are known;

D12. Cars are equipped with a device able to determine the number of passengers, and also able to make accessible/communicate that number;

D13. Cars are equipped with a device able to determinate the level of charge of the battery, and also able to make accessible/communicate that level;

D14. Cars are equipped with a device able to determinate the engine starts up and when it stops, and is also able to make accessible/communicate these events;

D15. Cars are equipped with a device able to lock and unlock the car afterwards the receiving of the respective command;

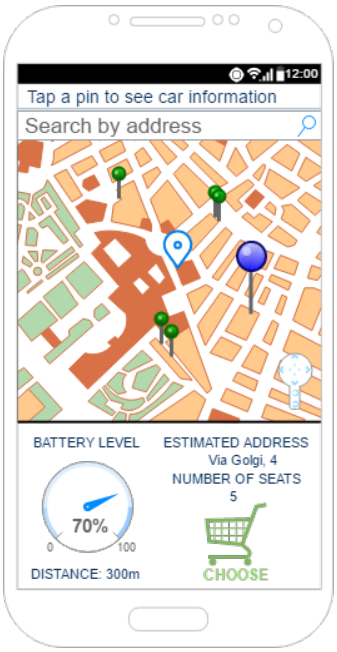
D16. Cars are equipped with a control unit able to communicate any mechanical and electric failure;

D17. All data provided by the sensors listed here above of each car are exact.

3 External Interface Requirements

3.1 User Interfaces

This is an example of a user interface screen for the mobile application.



In this case, when looking for a car, the user should see a map with the available cars around his position or the provided address. The user should be able to type an address using a text input field, should be able to choose a car tapping on the related pin, should be able to to read the battery level of the car and its real position, and also other characteristics. The web interface should follow the app style, and show the information in a similar way, taking into account the fact that the web site could be consulted either on large PCs screen or on small smartphones screens.

3.2 Hardware Interfaces

A particular hardware component needed by this system is the on-board device to be put on the cars to guarantee the interaction between the cars and the central system. This component should implement the communication of the data collected by the car sensors to the central system and should implement an interface to receive the information from

the central system. It may be based on the open source operating system Linux, adopting the more convenient distribution.

3.3 Software Interfaces

The system can be for the moment though as a client/server architecture with the user application as the client and the system logic as the server. To implement this kind of system, it is reasonable to develop it using the Java Enterprise Edition platform for the logic server and MySQL for the data management.

3.4 Communications Interfaces

Due to the evidence of the mobility feature, interaction with the system and the cars will have place over the Internet network, supported by the mobile phone-and-data network. Communications will adhere to a transportation protocol through the network which implement connection and confirmation features, as can be the TCP protocol. At the application layer the communication will stand on the HTTPS protocol.

4 System Features

This section is organized analyzing the functional requirements in system features.

4.1 Client registration

4.1.1 Description and Priority

The client registration represents the act of the subscription by the potential user to the system. He will provides all the information as:

- Name;
- Surname;
- Number of driver license;
- CF;
- e-mail address;
- Payment data (C/C or debit card or credit card or "anything else");
- Billing data;

A user is not allowed to sign up while he is yet signed up as the same physical/real person. The system, in case of positive registration, will communicate a password to the user, needed to sign in. The password will be generated in a consistence way to obtain a safe key (see 5.3 on page 32) This feature is fully required.

4.1.2 Stimulus/Response Sequences

Potential user may register inserting his information in the form of the application or the website.

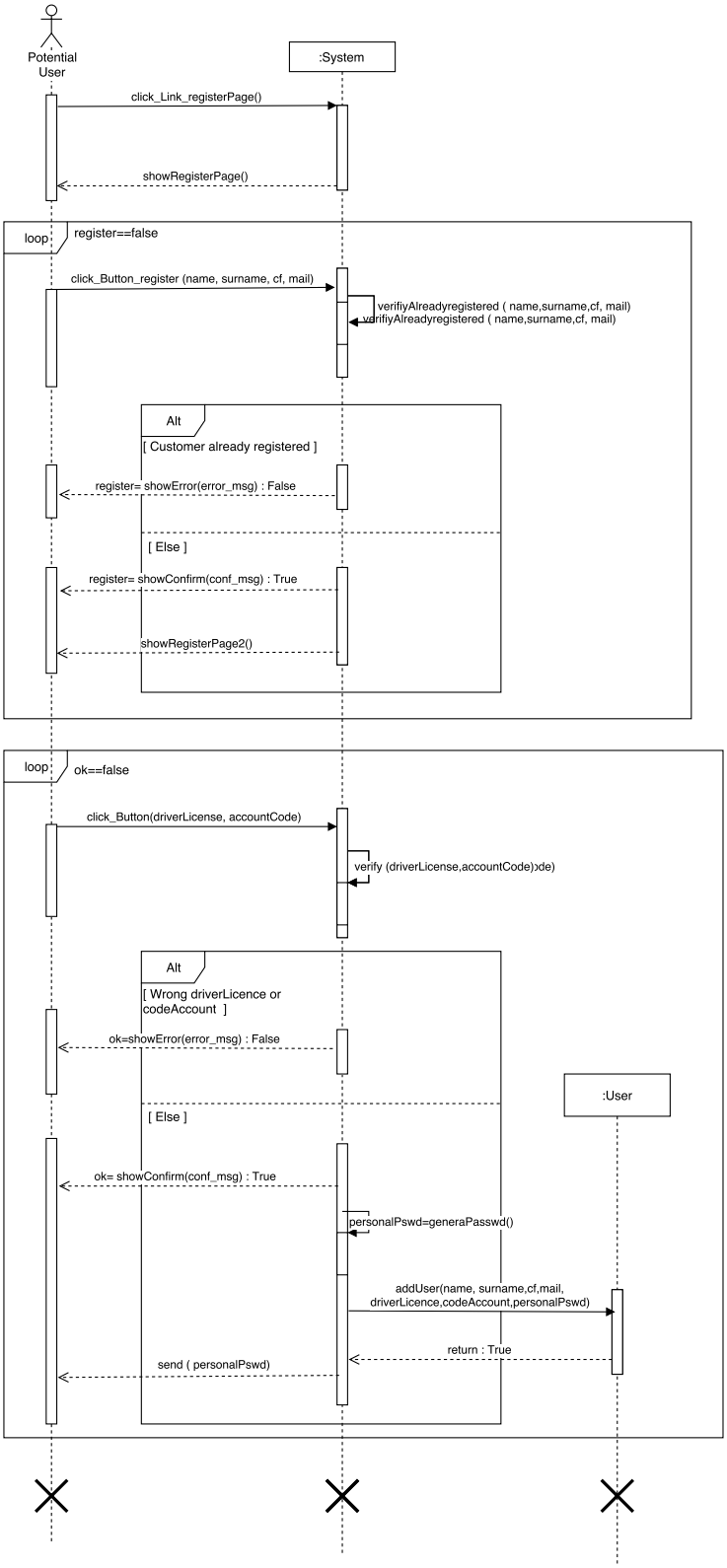
Use case

The use case related to this feature (see diagram 6.2.1 on page 35) is shown in the following table:

Use case	Registration
Actors	Potential user
Entry conditions	—
Flow of events	A potential user, to become a user, has to enter in the registration form on the application or website and insert his personal information (name, surname, cf, mail, code account and driver license). If information are correct and user is not already registered, he receives a password to login in the system and a notification that confirms his registration.
Exit conditions	System generates a password for the new user and add his in its database.
Exceptions	Information submitted by the user are not correct or the user is already registered. System rejects the registration request and notifies the user about.

Sequence diagram

Here a representation of the events that are involved in the step of registration.


Potential User
:System
click_Link_registerPage()
showRegisterPage()
loop
register==false
click_Button_register (name, surname, cf, mail)
verifiyAlreadyregistered (name,surname,cf, mail)
verifiyAlreadyregistered (name,surname,cf, mail)
Alt
[Customer already registered]
register= showError(error_msg) : False
[Else]
register= showConfirm(conf_msg) : True
showRegisterPage2()
loop
ok==false
click_Button(driverLicense, accountCode)
verify (driverLicense,accountCode)ode)
Alt
[Wrong driverLicence or codeAccount]
ok=showError(error_msg) : False
:User
[Else]
ok= showConfirm(conf_msg) : True
personalPswd=generaPasswd()
addUser(name, surname,cf,mail, driverLicence,codeAccount,personalPswd)
return : True
send (personalPswd)

4.1.3 Functional Requirements

R1. System "should/is able to" check if a person has already been registered;

R2. System should register a person only if he/she is not already been registered;

R3. System should verify if driver license information are valid and correct;

R4. System should verify if payment information are valid and correct;

R5. System should register a person only if all provided data are correct;

R6. System should generate and send a password to the user only if the registration is allowed;

4.2 Signing in and usage of the system

4.2.1 Description and Priority

This feature allows the user to sign in the system and be able to request the services. The user should identify himself by the e-mail and password associated to him by the system. (The e-mail inserted in the registration step and the password provided by the system.)

4.2.2 Stimulus/Response Sequences

A registered user has to sign in the system providing e-mail and password (see diagram 6.2.1 on page 35).

Use case	Login
Actors	Registered user (user)
Entry conditions	User must be registered in the system
Flow of events	The user enters the application or the website and provides, in a dedicated form, the e-mail submitted at the registration step and his associated password. Systems checks for e-mail and password. If the data are correct, the user becomes authenticated.
Exit conditions	The user is signed in the system and he is allowed to use the company services.
Exceptions	Data submitted correspond with no registered user. System rejects sining in. User is notified of that.

4.2.3 Functional Requirements

R7. System should be able to verify if user's password is correct;

R8. System should only let user sign in if the password and the e-mail are correct (=well-spelled and associated to that user);

R9. user can use company services only if he is logged on to the system.

4.3 Searching for a car

4.3.1 Description and Priority

Providing a location in the environment, the system has to show the cars available in a provided distance. The cars available can be located in parkings (actually safe areas), plugged or unplugged to a power grid station, that are available for the reservation for the rental. This feature is essential for the way the system is thought.

4.3.2 Stimulus/Response Sequences

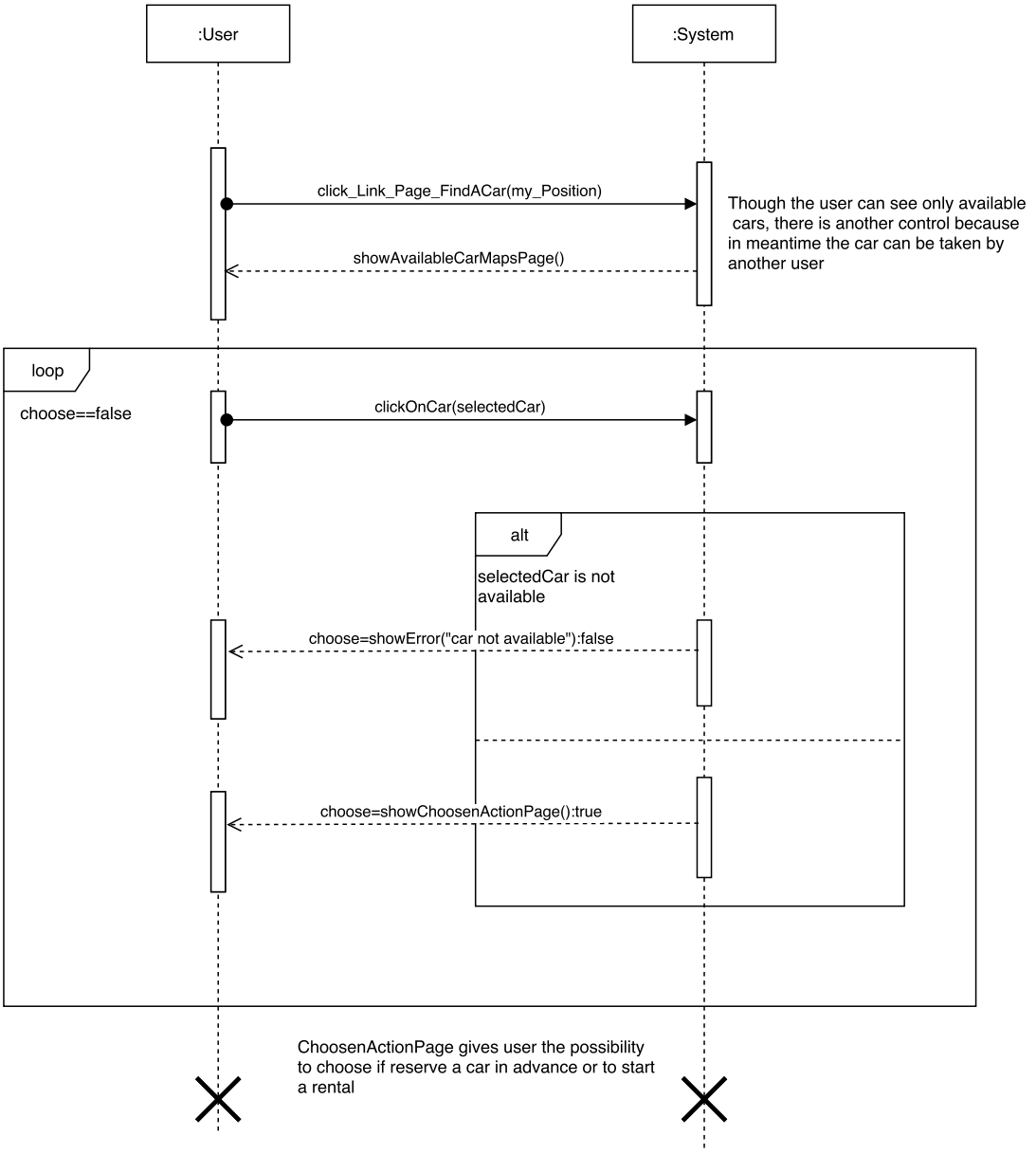
Use case

An authenticated user may wish to find a car near his position or near a certain address (see diagram 6.2.1 on page 35).

Use case	Find available car
Actors	Authenticated user
Entry conditions	— (User authenticated, ma lo è per forza dallo usecase)
Flow of events	User selects the way to locate the center of the circle where to search for available cars: by writing an address or by providing his geographical location (e.g. with GPS). System locates on a *user side map* the available cars in the requested area and let the user select the desired car. Pin on the map contains level of battery information for every car.
Exit conditions	After user selects a car, system shows 2 possible actions: reserve the selected car or start immediately a rental.
Exceptions	—

Sequence diagram

Here a representation of the events that are involved in the step of choosing a car.



4.3.3 Functional Requirements

R10. System must be able to know user's position;

R11. System must be able to know the position of the car;

R12. System must be able to identify a specified address.

R13. System must be able to show the state (location and battery charge level) of all the available cars near (1 Km) the user's position;

R14. System should let the user select a car to be used in the services.

4.4 Rental reservation

4.4.1 Description and Priority

This is the key feature of the system, probably the most used one. An authenticated user can choose a car as in the previous feature, and then he requires the reservation. Therefore the system has to reserve the car required, confirm the reservation, start the timer that measure one hour from the reservation...

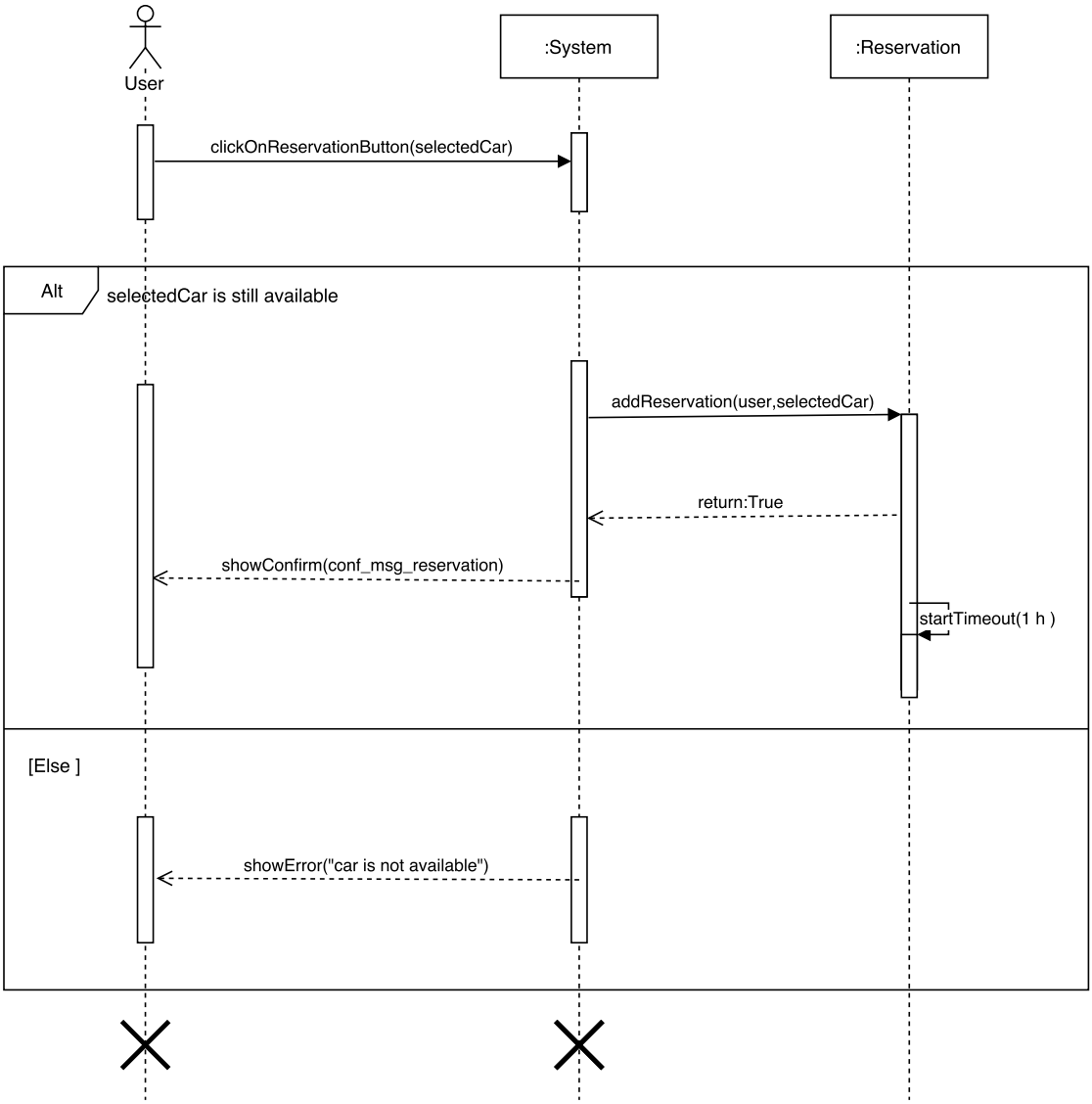
4.4.2 Stimulus/Response Sequences

Use case

(see diagram 6.2.1 on page 35)

Use case	Reserve a car
Actors	Authenticated user
Entry conditions	A car has to be previously selected (read "Coming from *Searching for a car*")
Flow of events	The user decides to reserve the selected car, so he let it be known to the system. The reservation expires within 1 (one) hour.
Exit conditions	The reserved car is marked as unavailable to other reservations in the next hour.
Exceptions	If the car is being selected by another user who more rapidly complete the reservation request, the selected car results unavailable, and the user is redirect to the selection step.

Sequence diagram



4.4.3 Functional Requirements

R15. user can book only one car a time;

R16. user can book for rental at most one car in a defined period of time (six hours) without pick it up;

R17. user can choose the car to be booked (see previous goal);

R18. System should make a (the booked) car unavailable for other rental requests;

R19. System should notify a confirmation of reservation to the user;

4.5 Proximity to the car

4.5.1 Description and Priority

The user, in the moment he wants to pick up the reserved car, will communicate with the system to manifest his nearness to the reserved car. So the system, stated the proximity, could continue with the other tasks such as unlock the car, stop the timer and so on as seen in the other goals/features. This feature requires special care on the security aspect as described in the non -functional section of this document (see 5.3 on page 32) .

4.5.2 Stimulus/Response Sequences

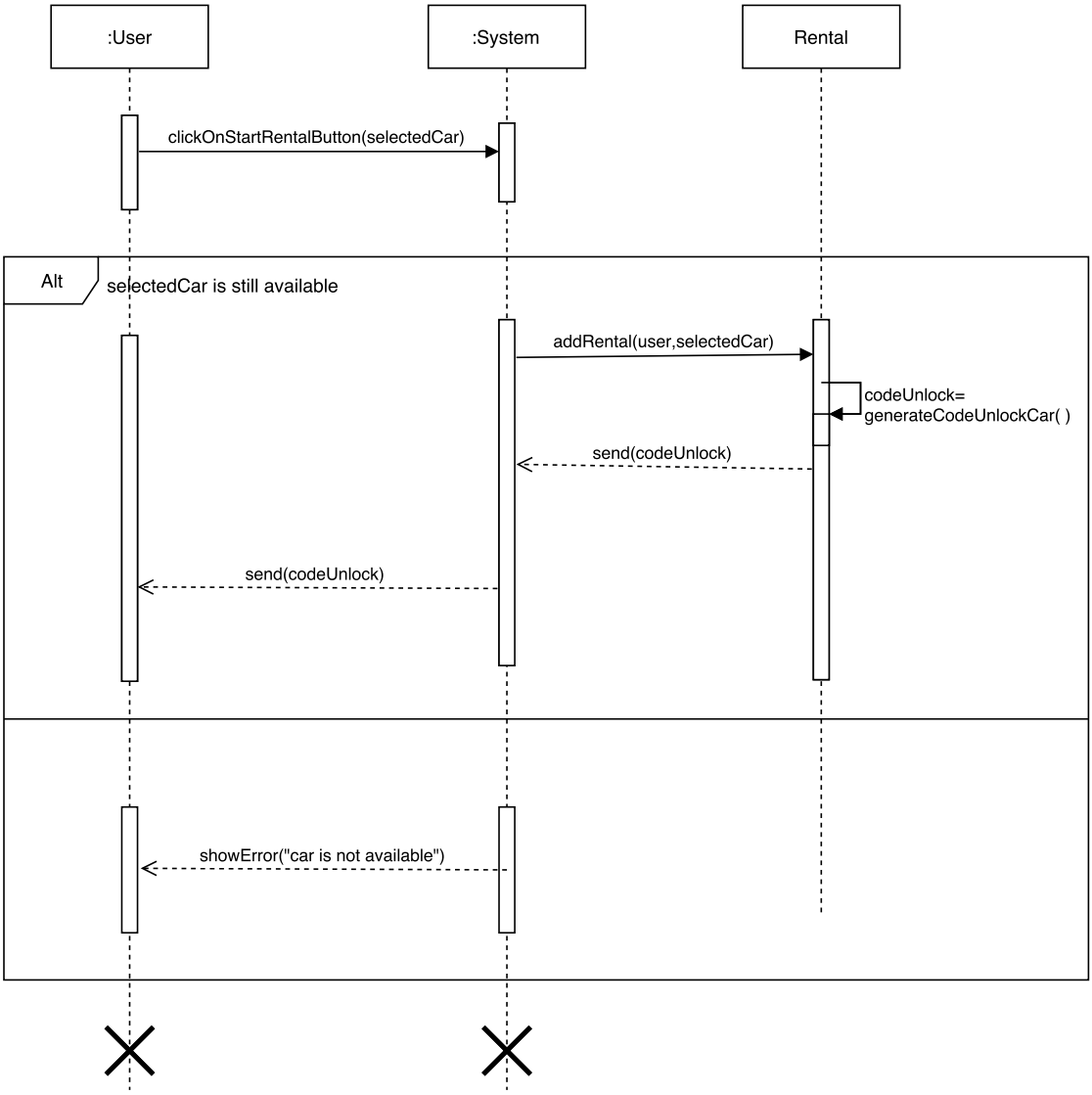
Use case

(see diagram 6.2.1 on page 35)

Use case	Start a rental and notify proximity at the car
Actors	Authenticated user
Entry conditions	A car has to be previously selected or reserved
Flow of events	A user is ready to start a rental, so he notify the system he is nearby the car. The system provide the user a way to unlock the car (eg. a QR code, a NFC tag). The user interacts with the car reader and get the car unlocked, so he can enter and start driving.
Exit conditions	The car is unlocked
Exceptions	—

Sequence diagram

Here a representation of the events that are involved in the step of unlocking a car and start driving.



4.5.3 Functional Requirements

R20. System should provide a way to receive the proximity information from the user;

R21. System should be able to analyze the proximity information and check for truth;

R22. System should acknowledge the user if the check fails.

4.6 Monitoring ride costs

4.6.1 Description and Priority

A user seems to like to know how much is the amount to pay in real time. So the system will show the amount of the ride up to date, "minute after minute". This cost will represent only the time-affected fraction of the price, so without any discount nor extra-fees.

4.6.2 Stimulus/Response Sequences

4.6.3 Functional Requirements

R23. System should be able to notify the amount of the ride to the user;

R24. System should be able to compute the amount of the ride;

4.7 Keeping aside a car

4.7.1 Description and Priority

It is possible that a user who booked a car needs to do something out of the car in a little period of time (at most 3 hours) and he wants to be sure the car still remains under his control. He can pay for a *keeping aside* service that let him to just don't lose the car in that small period. He will pay the price of the service. The user has to ask for the keeping before his current reservation expires.

4.7.2 Stimulus/Response Sequences

Use cases

Use case	Keep aside your car
Actors	Authenticated user
Entry conditions	User is currently using a rented car (the current rental is not expired/ended)
Flow of events	While a user is driving a car, he needs to stop and leave the car for a while to do something or for a commitment. Before power the engine off, the user can *keep aside* his car. Using the application or the website, he can activate the keeping aside function, so the car remains unavailable to other users for up 3 (three) hours
Exit conditions	The car is marked as unavailable until he users picks it up again or interrupts the *keeping aside*. System locks the car when nobody is inside the car. System starts to measure duration of the service when user actives it.
Exceptions	—

Coming back and pick-up again the car:

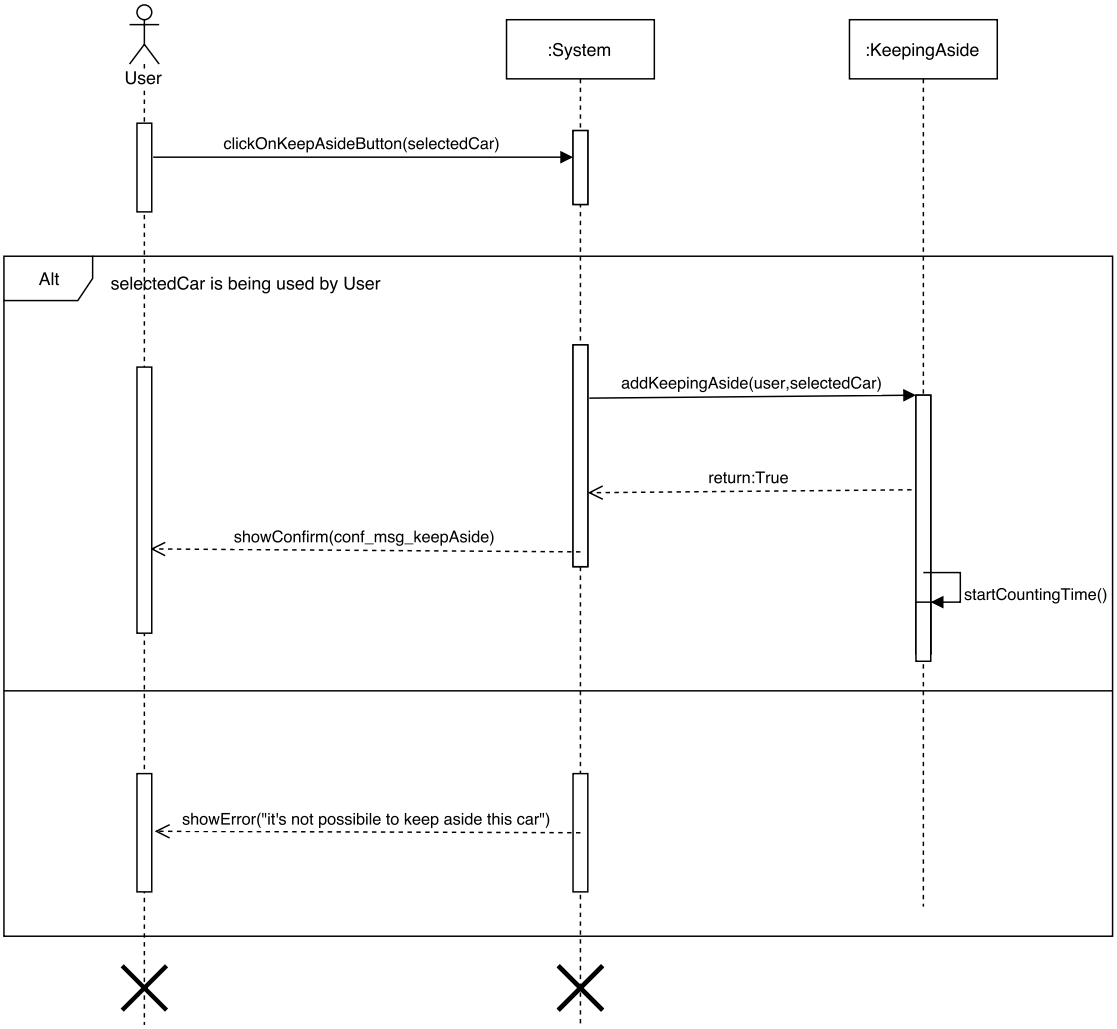
Use case	Continue rental
Actors	Authenticated user
Entry conditions	The car is kept aside
Flow of events	After keeping aside his car, user can unlock again the car using the same unlock code and continue his rental.
Exit conditions	System continues to charge the rental as if it has never stopped
Exceptions	—

Interrupt a keeping aside:

Use case	Interrupt a keeping aside
Actors	Authenticated user
Entry conditions	The user has to have keep aside his car before
Flow of events	If after keep aside his car the user realizes that the car is no longer required, he can interrupt the keeping aside reservation by the related command from the application or the website.
Exit conditions	The system will bill the user the benefited services. Also the system will mark the car as available.
Exceptions	—

Sequence diagram

Here a representation of the events that are involved in the step of keeping aside a car.



4.7.3 Functional Requirements

R25. System should allow to require a *keep aside reservation* only while the current car use is not over;

R26. System should not allow to keep aside a car for more than 3 (three) hours;

R27. System should start debit the user of keep aside cost;

R28. System should not mark as available a *kept aside* car to users other than the applicant.

4.8 Notify issues

4.8.1 Description and Priority

If a user notice a car issue or encounter problems during a ride, he will be able to notify the company so that an employee can take care of him and hopefully solve the issue. This feature is complementary, but for a complete service it needs to be developed.

4.8.2 Stimulus/Response Sequences

(see diagram 6.2.1 on page 35)

4.8.3 Functional Requirements

R29. System should be able to receive notifications;

R30. System should be able to store notifications;

R31. System should be able to report the status of a notification the user;

4.9 Computing amount of a ride

4.9.1 Description and Priority

4.9.2 Stimulus/Response Sequences

4.9.3 Functional Requirements

R32. System should start debit the user when the engine ignites;

R33. System should stops debit the user of *riding cost* when the engine stops and he exits the car;

R34. System should check the number of people present inside the car;

R35. System should be able to notify the end of debit to user.

4.10 Opening and closing cars - Unlocking and locking

4.10.1 Description and Priority

A car has to be unlocked when required by the nearby user associated in that moment to the car, and has to be locked when it is left or when it is parked during a *keep aside*.

4.10.2 Stimulus/Response Sequences

4.10.3 Functional Requirements

R36. System should being allowed to open the car by the *proximity check* (see 4.5 on page 22);

R37. System should generate a different unlock code for each reservation;

R38. System should lock the car only when nobody is inside the car and the engine is powered off;

R39. System should not lock the car while someone is inside the car;

R40. System should not leave an available car unlocked;

4.11 Applying discounts

4.11.1 Description and Priority

4.11.2 Stimulus/Response Sequences

4.11.3 Functional Requirements

R41. System should be able to compute a discount;

R42. System should check the number of passenger in the car;

R43. System should compute the amount of battery used;

R44. System should deduce the remaining battery amount;

R45. System should check if the car is parked and plugged in a power grid.

4.12 Applying extra fees

4.12.1 Description and Priority

4.12.2 Stimulus/Response Sequences

4.12.3 Functional Requirements

R46. System should be able to compute itself the overcharge;

R47. System should compute the distance from the nearest power grid station;

R48. notify...

4.13 Overview and monitoring the services

4.13.1 Description and Priority

4.13.2 Stimulus/Response Sequences

(see diagram 6.2.1 on page 35)

Use case	View report
Actors	Management employee
Entry conditions	User has to be signed in
Flow of events	A manager wants to know the current position of the cars, the amount of money earned so far, statistics. He sign in the website and ask for the "View Report" function. The system shows him a map crowded by cars icons, and report and statistics
Exit conditions	—
Exceptions	—

4.13.3 Functional Requirements

R49. System should offer an interface for the state of each car;

R50. System should be able to get all the status data and positions at the current instant;

R51. System should be able to elaborate a human-readable diagram to show up that data;

R52. System should pin on a correspondent map the position of the cars;

4.14 Listening for failures

4.14.1 Description and Priority

4.14.2 Stimulus/Response Sequences

(see diagram 6.2.1 on page 35)

Use case	Watch problem status
Actors	Maintenance employee
Entry conditions	User has to be signed in
Flow of events	The user signs in the website and the system shows him a monitoring page. When a issue is notified, he can ask for details and take care of that issue.
Exit conditions	—
Exceptions	An issue occurs: the user is notified and he may attend and solve the issue.

Use case	Attend and solve issue
Actors	Maintenance employee
Entry conditions	User has to be signed in and has to previously being watching the problems status
Flow of events	The user receives a issue notification. He can ask for details and take care of that issue.
Exit conditions	The status of the issue is updated as the result of the maintenance employee action
Exceptions	—

4.14.3 Functional Requirements

R53. *On-board car system* should notify the main system when a failure occurs;

R54. Main system should listen for notification from any *on-board car system*;

4.15 Editing parameters

4.15.1 Description and Priority

4.15.2 Stimulus/Response Sequences

(see diagram 6.2.1 on page 35)

Use case	Edit settings
Actors	Management employee
Entry conditions	User has to be signed in
Flow of events	The user wants to update some parameters of the services
Exit conditions	The status of the issue is updated as the result of the maintenance employee action
Exceptions	—

4.15.3 Functional Requirements

R55. System should show the current parameters as *ride price* per minute, *keep aside* per minute, percentage of discounts;

R56. System should be able receive the new values to assign to the parameters;

4.16 Billing a user

4.16.1 Description and Priority

The company bill a user at the very end of the rental, when the user leaves the car definitively.

4.16.2 Stimulus/Response Sequences

4.16.3 Functional Requirements

R57. System should wait 5 minutes before proceed to bill a benefited service;

R58. System should sum the riding amount (if any), the keeping aside amount (if any), the penalty (if any);

R59. System should bill the user when the reservation expires without pick up or when engine stops and he exits the car and no *keep aside* is required;

R60. System should check where the car is parked/located when engine stops and the user exits the car;

R61. System should apply business rules about discounts;

R62. System should check the number of people present inside the car;

R63. System should notify the user about the invoice and the end of rental;

R64. System should send to the *payment handler* a withdrawal request;

R65. System should become aware of the transaction outcome.

5 Other Nonfunctional Requirements

5.1 Performance Requirements

5.2 Safety Requirements

A registered user is not allowed to register again while he is registered yet.

5.3 Security Requirements

The users' data need to be stored in a proper "database" to preserve them from any use not involved in the requirements listed in this document. Payment data are sensibles data which the company only wants to use to have its services payed. The password provided by the system to the user during the registration step should be longer than 10 chars and should contain at least a cipher, a special character, and a capital char. The way chose to state/verify the proximity of a user should guarantee 99% not to allow wrong deduction. [We suggest to solve this issue by NFC tags and QR code.]

5.4 Software Quality Attributes

5.5 Business Rules

Policies concerning the stimulus of users' virtuous behaviors. Price reduction or increasing on the last ride of the user as stated in the following list:

- –10% if the user took at least two passengers into the car;
- –20% if the car is left with no more than 50% of battery empty;
- –30% if the car is left at a special parking areas and user plugs the car into the power grid;
- +30% if the car is left more than 3Km from the nearest power grid station or with more than 80% of battery empty;

Discounts cannot be combined: in case of multiple satisfied conditions, the best (greatest) one will be applied. Extra fees conditions comply to the logical operator *OR*.

6 Other Requirements

6.1 Appendix A: Glossary

potential user is a person who may be interested in the service of rent a car and visits, as a guest does, the website or the application;

user is a registered user who gave his identification data (CF, name, surname, email, driver license number, payment data, billing data);

authenticated user is a user logged in the system who can access to all the services provided by the system to the users;

CF is the acronym of *Codice Fiscale*, the Italian correspondent of SSN – Social Security Number;

client refers generically to a beneficiary of the system such as the potential user, the user and the authenticated user;

safe areas are public parkings and company-owned parking places in both cases on a single layer. Parkings built as multiple layers perfectly overlapped are not considered safe areas;

reservation is a binding request that includes a car booked for a specified date and time, a booker user, provided up to one hour in advance;

car is a company owned vehicle with geographical position detector, lock/unlock system, passenger counter sensor, display for information related to the ride, engine-on detector, battery level status. Each car is identified by the license plate;

reserved car is a car associated to a user who selected and reserved it. That car is not available to other user other than the one associated.

rental indicates the set of services and actions from when the user picks up a car to when he leaves it;

ride refers to the specific part of the rental when the car is running around;

driver a person who drives the car;

passenger a person who is in the car but not the driver;

power grid station is an accessible device where a car can be plugged in (with a suitable cord) to recharge the battery;

maintenance concerns the taking care of cars that are damaged, discharged, with any sort of problems;

system is the whole developing architecture and application that will take care of the management of the services as shown in this document;

discount price reduction;

GPS stands for Global Positioning System, the technology used to geographically locate an object equipped with a special transceiver;

on-board system indicates the part of the system installed and running on the car which is responsible for interfacing the control unit of the car with the system. It allows communication of the values of the sensors on the car to the system, and allows the system to activate the actuators on the car;

NFC stands for Near Field Communication, a set of communication protocols that enable two electronic devices to establish communication by bringing them within 4 cm of each other;

QR code stands for Quick Response code. It is a two-dimensional bar code;

operator OR as the Boolean operator OR. Conditions linked with this disjunctive operator make the statement false if and only if all the conditions are false, otherwise the statement is true.

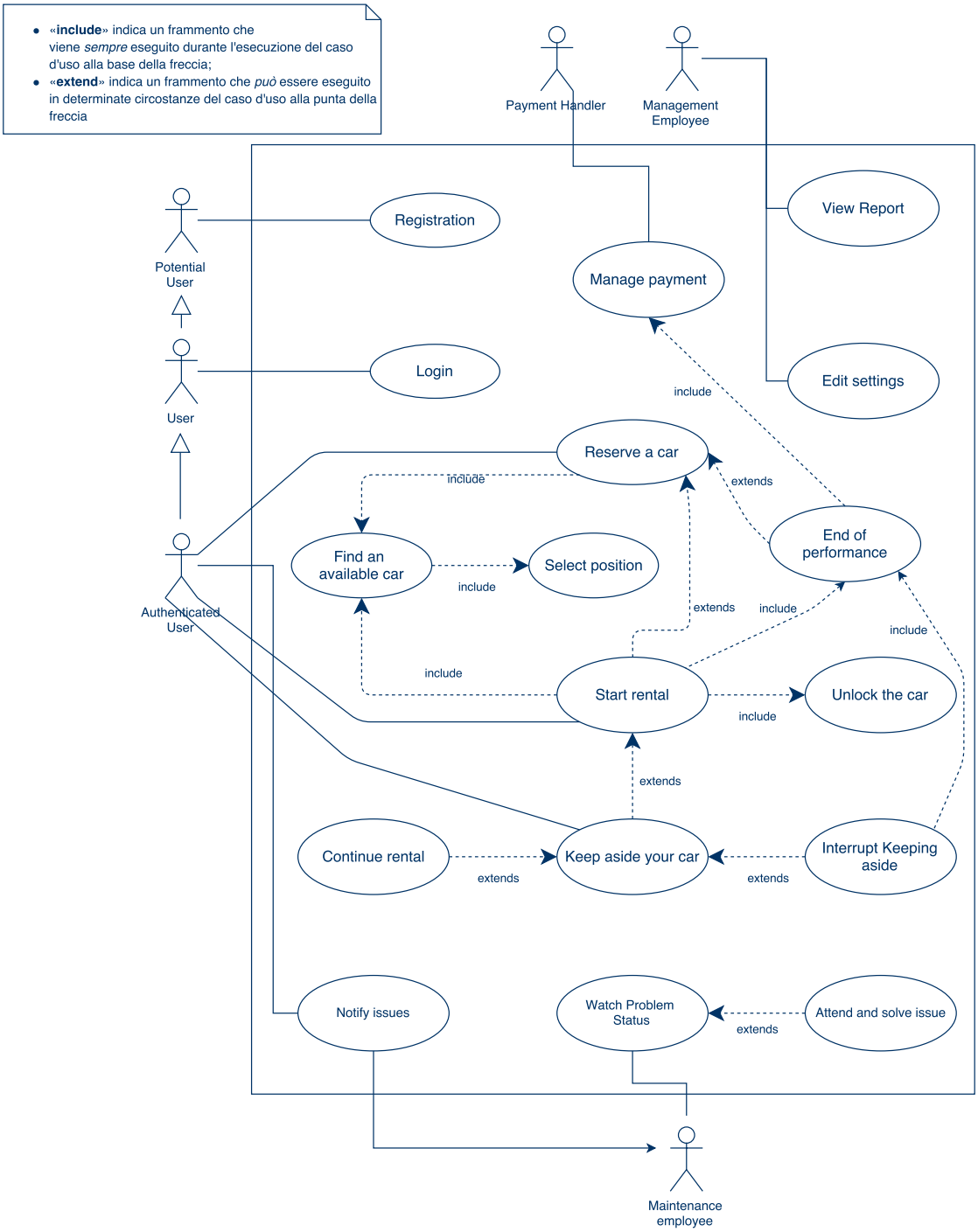
report a complete analysis and overview of the incomings, outgoings, revenues, statistics about the requested services;

payment handler third part service responsible for actuation of payments;

service refers to the renting cars business supplied by the company to the users.

6.2 Appendix B: Analysis Models

6.2.1 Use cases diagram



6.2.2 Scenarios

Here some possible scenarios involved in the use of this system.

Registration

Wile E. Coyote got tired of chasing Beep Beep walking. He just found out there are rentable car that could fit to him. He decides to become better informed surfing the company's website and, convinced, wants to register. Fill in the form proposed with its data: name, surname, social security number, e-mail, driving license, credit card and billing information. Consent to the processing of personal data and consent to the company to withdraw the amounts charged of any services. Soon he will receive an email from the system with the password to log in the system and finally capture the spiteful roadrunner.

Unlock a car

Bugs Bunny is a registered user in Power EnJoy system. He finished the supply of carrots and does not want to remain without dinner, so he comes out the den and see that right there alongside there is a Power EnJoy car. So it hastens to approach, accesses the system via the application, selects the car and prompts to begin immediately a rental. The system sends the unlock code to be communicated to the machine sensor and since it is correct, the car opens, ready to move.

Reservation

Pluto wonders to move in Milan to places non served by public transport. He live out the city and he would like to use the train to get in the city and pick up a car of ours. He take the train and when he is less than an hour near to the car station, he sign in the system with the password he received at the registration step, then he ask for a car near the *Milano Nord Cadorna* railway station and select the car with the battery fully charged; he proceed booking the car. Once he arrives, he gets the car and goes its way.

Passengers discount

Topolino needs to go in a "ZTL" zone where only some types of vehicles are allowed. So he decide to opt for our electric cars, which are allowed in that area. He also wants to take with him his sister and her boyfriend. At the end of the ride, he will obtain a discount of 10% on the amount of the ride.

Plugging discount

Minnie is a very careful person. She sometimes needs to go from a part of the city to another, luckily both near a power grid station of our company. So when she has used the car, she parks it in the power grid and plugs it, so she can get a 30% discount.

Multiple discounts

Paperon de Paperoni is a regular user of Power Enjoy services. He knows very well the policies about discounts, but he never remembers that discounts are not combinable. So one day he decides to bring to the park Qui, Quo and Qua by car. They search a car using the application, they reserve it and they in 10 minutes are near the car and they pick it up. They arrive at the park, where is available a power grid station, and Paperon plugs the car into. He is sure to receive the discount for the number of passenger plus the discount for the recharging, but he will only get the second one (–30% because of power grid station).

Reservation expiring

Pippo wants to book a car for his travel. He registers to the site, provides all the infos required and book a car. He thinks he will arrive in an hour at the car station, but unfortunately his boss assigns him a very long task that committed him for two hours. So, one hour after the reservation, the system charges Pippo a 1EUR fee and the car will become again available.

Keeping aside

Minnie and Topolino are furnishing the house with the intention of getting married soon. They want to go and buy some small kitchen accessories to IKEA. The store is just outside the city, so they decide to get there by using one of our cars. Near their home there is a parked car available, therefore they start a rental. Arrived at the IKEA parking lot, they choose to keep aside the car, hoping to go out within 3 hours. So before turning off the engine Mickey enters the application and asks for a keep aside. Happy, they hop off the car, make purchases, and after well two hours they come back to the car. Through the application Topolino asks to unlock the car. They go home satisfied, and they park in a municipal parking lot.

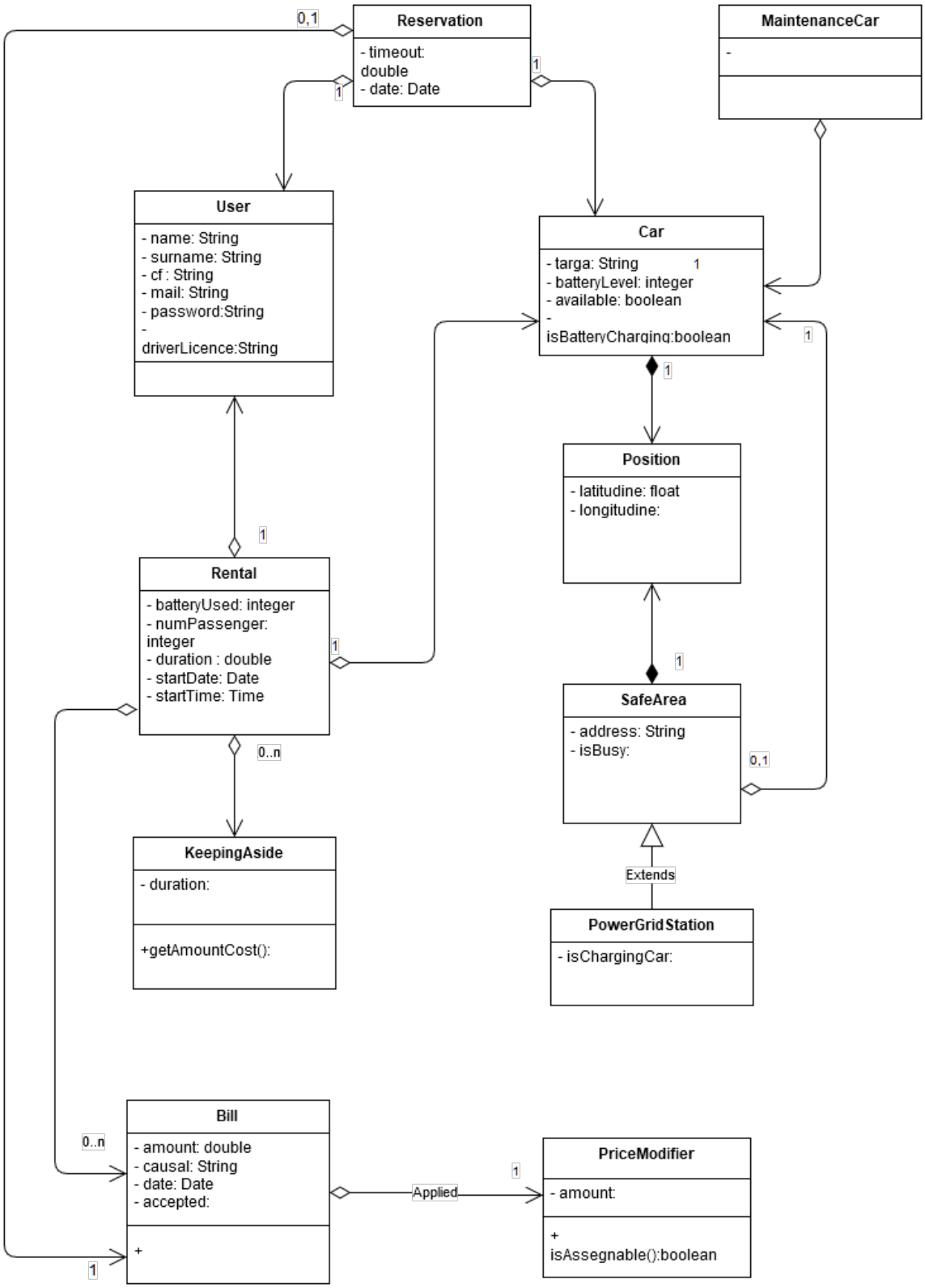
Notify issues

Daffy Duck is not very lucky... While driving, in summer with the windows down, it starts to rain. He tries to close the power windows of the car, but the controls do not respond. So he decides to report the problem to the company. Enter the application and writes a message reporting the problem. A maintenance employee will promptly take care of the issue.

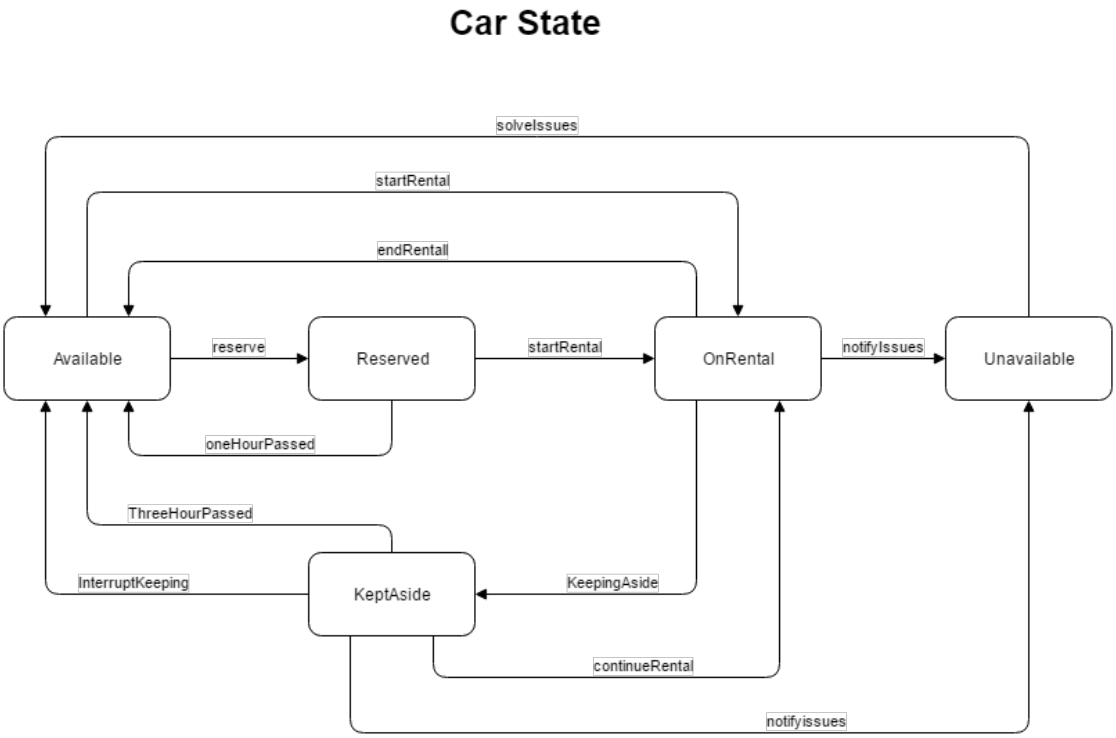
Edit settings

Yosemite Sam, a management employee of the company, receives an order from his boss about increasing the rate per minute of the ride to cope the increase in electricity costs. He enters in the editing parameters section of the website and changes just what he needs.

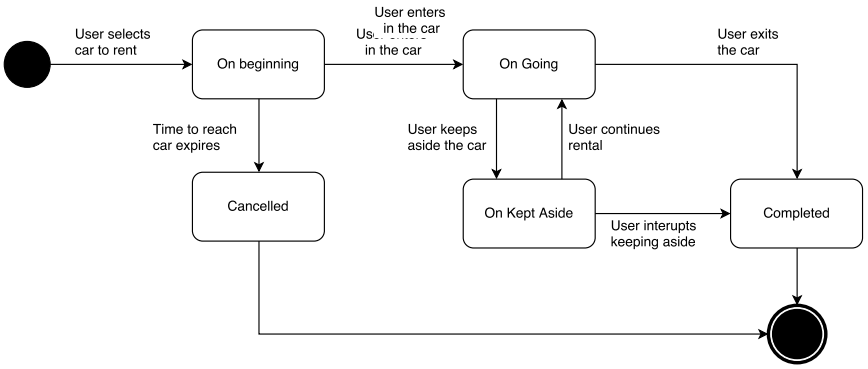
6.2.3 UML diagram



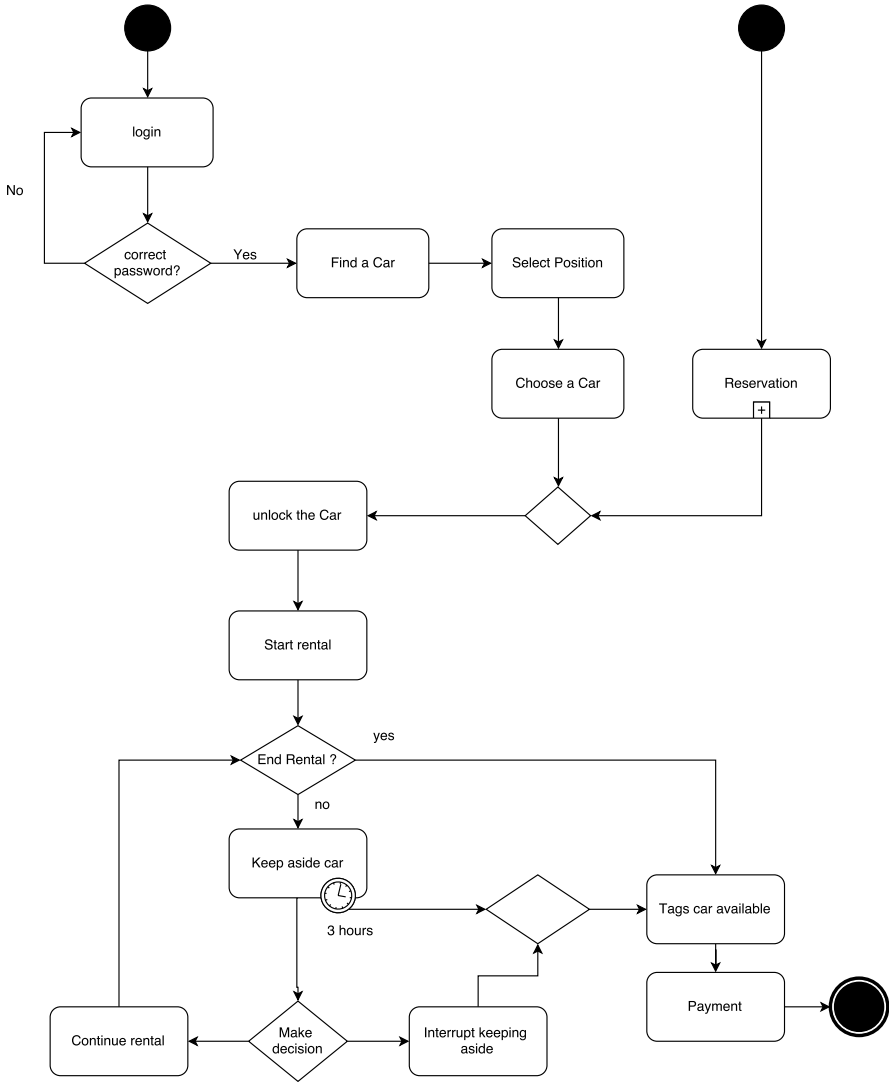
6.2.4 State and activity diagrams



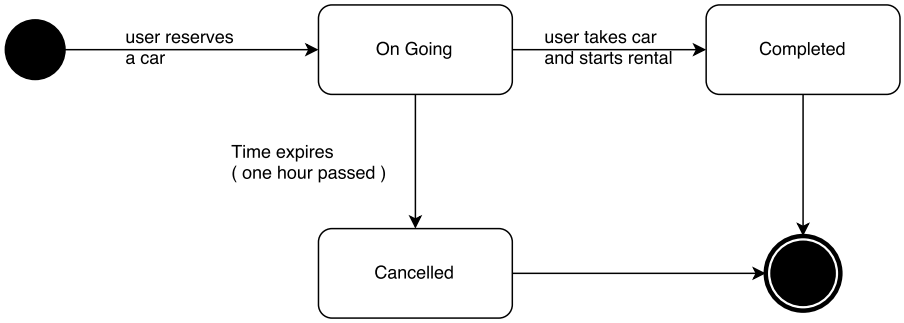
State Diagram Rental & Kept Aside



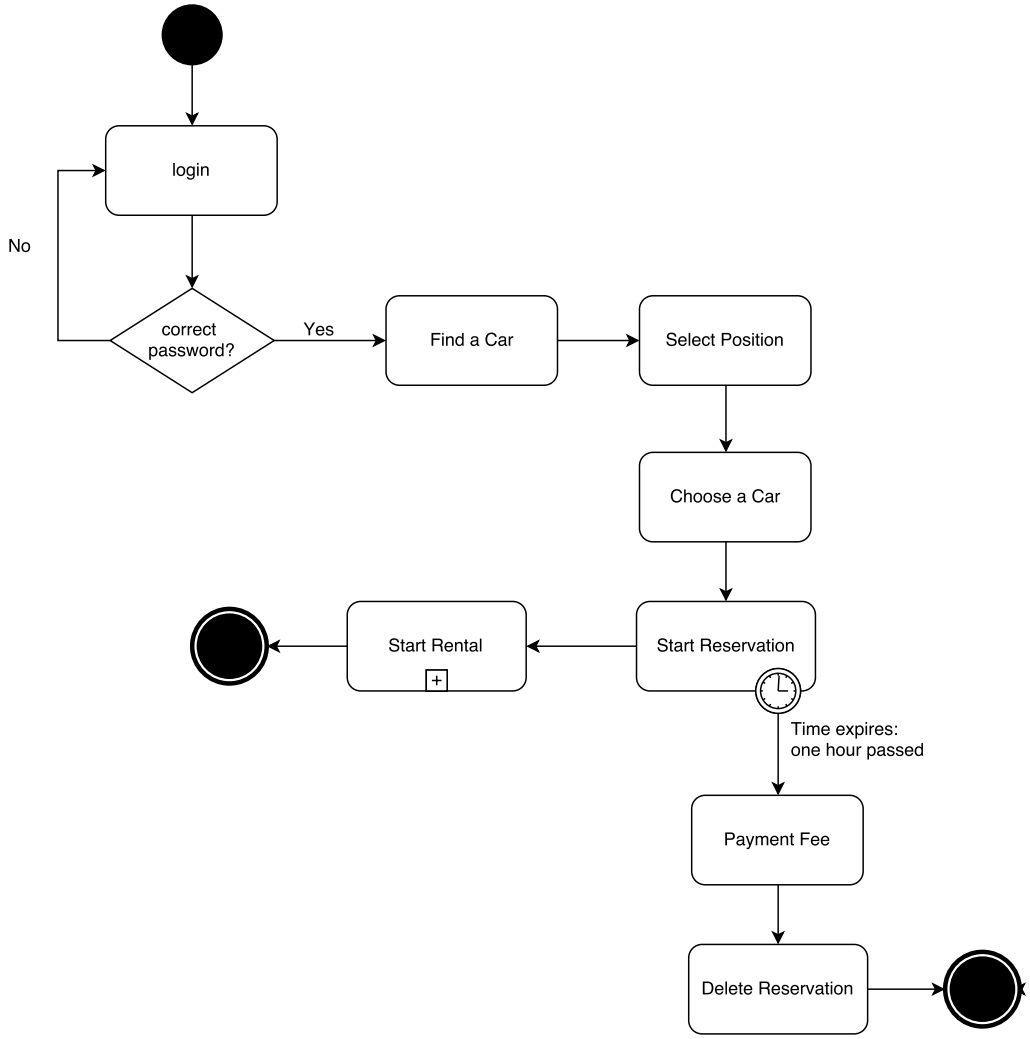
Activity Diagram Rental & Kept Aside



State Diagram Reservation



Activity Diagram Reservation



6.2.5 Alloy model

```
open util/boolean

one sig Company{
        cars: set Car,
        safeAreas: set SafeArea,
        users:set User
}

fact allUserBelongToCompany{
        no u: User | not (u in Company.users)
}


fact allCarsBelongToCompany{
        no c: Car | not (c in Company.cars)
}


fact allSafeAreaBelongToCompany{
        no s:SafeArea | not (s in Company.safeAreas)
}


sig Position{
        latitude: Int, //should be float
        longitude: Int //should be float
}


// CAR

sig Targa{}

sig Car {
        targa: Targa,
        position:  Position,
        available: Bool,
        isBatteryCharging:Bool,
        batteryLevel: Int,
        numberSeat: Int
}{
        batteryLevel>=0 and batteryLevel<=100
```

```
        numberSeat>=0 and numberSeat <5
}


fact noEqualTarga{
        no c1,c2:Car|( c1!=c2 and c1.targa=c2.targa )
}


// USER


sig DriverLicense{}
sig Password{}

sig User{
        driverLicense: DriverLicense,
        password: Password,
        signedIn: Bool// 0=no , 1 =yes
}

fact noSameUser{
        no u1,u2:User| u1.driverLicense=u2.driverLicense and u1!=u2
}


fact noSamePassword{
        no u1,u2:User| u1.password=u2.password and u1!=u2
}

fact noUserCanRentIfIsNotLogged{
        all r:Rental | r.user.signedIn=True
}

fact noUserCanReserveIfIsNotLogged{
        all r:Reservation | r.user.signedIn=True
}


// SAFE AREA

sig SafeArea{
        position: Position,
        isBusy: Bool
```

```
}


fact noDifferentAreaSamePosition
{
        no s1,s2:SafeArea  | (s1.position=s2.position and s1!=s2 )
}



fact noCarInSamePosition{
        no c1,c2:Car | (c1!=c2 and c1.position=c2.position)
}

fact areaIsFree{
        all s:SafeArea |s.isBusy=False iff(no c:Car | c.position=s.
    position)
}

fact areaIsBusy{
        all c:Car,s:SafeArea | c.position=s.position implies s.isBusy=
    True
}

//PowerGridStation

sig PowerGridStation extends SafeArea{
        isChargingCar: Bool
}


fact carIsCharging{
        all c:Car,pgs:PowerGridStation |
        ( c.isBatteryCharging=True and c.position=pgs.position  )
    implies pgs.isChargingCar=True
}

fact noCarNoChar{
        all p:PowerGridStation| p.isBusy=False implies p.isChargingCar=
    False
}

fact carIsChargingTwo{
        all c:Car,pgs:PowerGridStation |
```

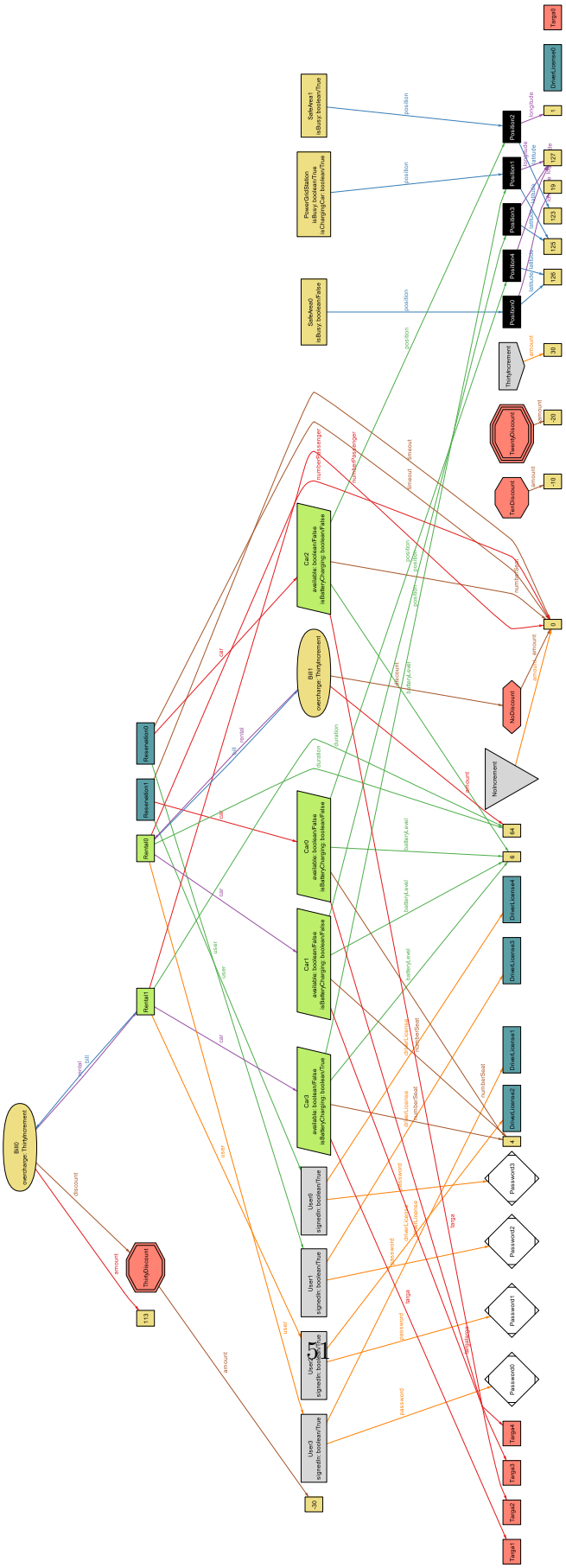
```
        (pgs.isChargingCar=True  and c.position=pgs.position  ) implies
    c.isBatteryCharging=True
}

fact noCarIsChargingOutOfPGS{
        all c:Car | c.isBatteryCharging=False iff (no p:PowerGridStation
     | c.position=p.position )
}


// Reservation


sig Reservation{
        timeout: Int,
        user:  User,
        car: Car,
//      bill: Bill
}{
        timeout>=0 and timeout<=60
}


fact allCarReservedAreUnavailable{
        all r:Reservation, c:Car | (c=r.car) implies c.available=False
}


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

sig Rental{
        car: Car,
        user: User,
        numberPassenger: one Int,
        duration:Int,
//      state : State,
        bill: Bill
}{
        numberPassenger>=0 and numberPassenger<car.numberSeat
        duration>0
}
```

```
//state of KeptAside

/*sig State{
        time:Int // time while car in one state or another of rental
}

sig OnGoing extends State{}

sig OnKeptAside extends State{}
{
        time>=0 and time<=18 // really is valued 180
}
*/


fact noSameUserOrCarOnRental{
        no disjoint r1,r2:Rental| r1.user=r2.user or r1.car=r2.car
}

fact carNoRentalNoReserveAreAvailable{
        all  c:Car | c.available=True iff  (no rental:Rental, reserve:
    Reservation| rental.car=c or reserve.car=c)
}


fact carNotOnRentalAndReservationAtSameTime{
        no rental:Rental,reserve:Reservation |
        rental.car=reserve.car or rental.user=reserve.user
}


fact allCarOnRentalAreUnavailable{
        all r:Rental, c:Car | (c=r.car) implies c.available=False
}



//BILLS, DISCOUNT and OVERCHARGE


sig Bill{
        amount: one Int,
        discount:Discount,
        rental: Rental,
```

```
        overcharge:OverCharge
}{
        amount>0
}

abstract sig Discount{
        amount:Int
}{
        amount<=0
}

abstract sig OverCharge{
        amount:Int
}{
        amount>=0
}

one sig ThirtyIncrement extends OverCharge{}
{
        amount=30
}

one sig NoIncrement extends OverCharge{}
{
        amount=0
}

one sig NoDiscount extends Discount{}
{
        amount=0
}


one sig TenDiscount extends Discount{}
{
        amount=-10
}

sig TwentyDiscount extends Discount{}
{
        amount=-20
}

one sig ThirtyDiscount extends Discount{}
```

```
{
        amount=-30
}

fact noDiscountForUser{
        all rental:Rental |
        (rental.numberPassenger<2 and rental.car.batteryLevel<50 and
    rental.car.isBatteryCharging=False)
        implies rental. bill.discount=NoDiscount
}

fact discountPassenger{
        all rental:Rental |
        (rental.numberPassenger>=2 and rental.car.batteryLevel<50 and
    rental.car.isBatteryCharging=False)
        implies rental. bill.discount=TenDiscount
}

fact discountBattery{
        all rental:Rental|
        (rental.car.batteryLevel>50 and rental.car.isBatteryCharging=
    False)  implies rental.bill.discount=TwentyDiscount
}

fact discountCharging{
        all rental:Rental | rental.car.isBatteryCharging=True implies
    rental.bill.discount=ThirtyDiscount
}

fact payOvercharge{
        all rental:Rental |( rental.car.batteryLevel<=20 //or noPwgNear[
    rental.car]
        ) implies rental.bill.overcharge=ThirtyIncrement
}


fact payNoOvercharge{
        all rental:Rental |( rental.car.batteryLevel>=20 //or noPwgNear[
    rental.car]
        ) implies rental.bill.overcharge=NoIncrement
}


fact oneRentalOneBill{
```

```
        all disjoint r1,r2:Rental|r1.bill!=r2.bill
}

fact oneBillOneRental{
        all disjoint b1,b2:Bill|b1.rental!=b2.rental
}

fact billAndEquivalentRentall{
        all r:Rental,b:Bill | r.bill=b implies b.rental=r
}


pred userCanReserveOrRentCar[u:User,c:Car]
{
        all rental:Rental,reserve:Reservation | rental.user!=u and
    reserve.user!=u and rental.car!=c and reserve.car!=c
}

assert allCarOnRentalAreUnavailable{
        all rental:Rental | rental.car.available=False
}

assert allCarReservedAreUnavailable{
        all reserve:Reservation | reserve.car.available=False
}




run {} for 5 but 8 Int, exactly 4 Car,exactly 3 SafeArea,exactly 2
    Reservation

run userCanReserveOrRentCar

check allCarOnRentalAreUnavailable
check allCarReservedAreUnavailable

// # Instance. found. Predicate is consistent
// # Instance. found. Predicate is consistent

//No counterexample found. Assertion may be valid
//No counterexample found. Assertion may be valid
```



6.3 Appendix C: Work Hours

Day	Marco [h]	Daniele [h]
Saturday 29/10	3	3
Monday 31/10	3	3
Tuesday 1/11		1
Thursday 3/11		1.5
Friday 4/11	1	1
Saturday 5/11	4	1
Sunday 6/11	6	6
Monday 7/11	3	3
Tuesday 8/11	2	2
Wednesday 9/11	2	3
Thursday 10/11	2	2
Friday 11/11	3	4
Saturday 12/11	9	9
Sunday 13/11	10	10

6.4 Change log

On 07/02/2017 v1.1 we added the goal G16, we added the Interfaces specifications and we updated the activity and state diagrams.