POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
M. Sc. in Computer Science and Engineering
Dipartimento di Elettronica, Informatica e Bioingegneria

Power EnJoy

Software Engineering 2 - Project

RASD

Requirement Analysis and Specification Document

Version 1.0

Authors:
Andrea BATTISTELLO Matr: 873795
William DI LUIGI Matr: 864165

Academic Year 2016 - 2017

1. Introduction **4**
1.1. Purpose 4
1.2. Present system 4
1.3. Scope 5
1.3.1. A note on safe areas 5
1.4. Goals 5
1.5. Actors 6
1.6. Stakeholders 6
1.7. Definitions, acronyms, abbreviations 7
1.7.1. Definitions 7
1.7.2. Acronyms 7
1.7.3. Abbreviations 7
1.8. Reference documents 7
1.9. Overview 7

2. Overall description **9**
2.1. Product perspective 9
2.2. Product functions 9
2.3. User characteristics 10
2.4. Constraints 10
2.5. Assumptions and Dependencies 10
2.6. Future possible implementation 11

3. Specific requirements **13**
3.1. External Interface Requirements 13
3.1.1. User interfaces 13
3.1.2. Software interfaces 17
3.1.3. Hardware interfaces 18
3.1.4. Communication interfaces 18
3.2. Functional requirements 19
3.2.1. [G1] Build an effective and reliable electric car sharing system 19
3.2.2. [G2] Encourage drivers to behave correctly and with respect to the law. 20
3.2.3. [G3] Monitor cars and give assistance to drivers 21
3.3. Scenarios 21
3.3.1. Scenario 1 - User registration and login 21
3.3.2. Scenario 2 - Parking far from recharging station 22
3.3.3. Scenario 3 - Recharging the car after use 22
3.3.4. Scenario 4 - Reservation time expired 22
3.3.5. Scenario 5 - Using Power EnJoy with friends 22
3.3.6. Scenario 6 - Accidents 22
3.3.7. Scenario 7 - Cancel a reservation 23
3.3.8. Scenario 8 - Web app 23
3.4. Use cases 24

3.4.1. Use case diagram 24
3.4.2. Register account 24
3.4.3. View available cars through position 26
3.4.4. View available cars through specified address 27
3.4.5. Reserve car 28
3.4.6. Login 30
3.4.7. Cancel reservation 31
3.4.8. Pay 32
3.4.9. Unlock car 34
3.4.10. Start driving 35
3.4.11. Stop driving 35
3.4.12. Manage account 36
3.4.13. View reservation history 37
3.4.14. Call driver 37
3.4.15. Reimburse 38
3.4.16. Notify maintenance team 38
3.5. Class diagram 39
3.6. Performance requirements 40
3.7. Software system requirements 40
3.7.1. Reliability 40
3.7.2. Availability 40
3.7.3. Security 40
3.7.4. Maintainability 41
3.7.5. Portability 41
3.7.6. Documentation 41
3.7.7. User interface and human factors 41
3.8. Alloy model 42
3.8.1. Signatures 42
3.8.2. Facts 44
3.8.3. Predicates 46

Appendix **52**

1. Introduction

1.1. Purpose

Duckburg is a very large city, with a lot of people. Over the years, the number of people with a driving license has increased. This caused a lot of traffic, boosted air pollution, and made it really hard to find a parking spot.

The purpose of Power EnJoy is to provide an alternative to this. We want to create a commuting service that is highly available and reliable. In fact, the reason why people drive instead of using public transport is usually related to the fact that they don't want to wait for a bus / train (low availability) and they don't want to risk not getting home if there's no public transport available (low reliability) because of a strike, vehicle malfunctions, or just because it's late in the night.

1.2. Present system

In Duckburg, at the present time, there are already some services offering similar features. However, we identify the following weak points:

Customer service	Some car sharing services provide assistance to customers, but in some case it's a paid feature, and in some other cases the response time isn't immediate, or the assistance process is ineffective in some other ways.
Electric cars	Car sharing services usually employ, among others, petrol powered cars. To avoid polluting the city more than strictly necessary, we will only use electric cars.
Parking spots	Looking for a parking spot is still hard with some car sharing services. We want to make it easy, so we will offer dedicated parking spots for Power EnJoy cars, as well as letting our users park freely even on parking spots that would require payment.
Restricted traffic areas	A city like Duckburg, with a significant historical importance, typically has some areas restricted to traffic: this means that only certain vehicles are allowed inside. This policy is also used to reduce pollution in the most populated areas of the city. Just like normal cars, some car sharing services' cars also suffer from these bans. On the contrary, Power EnJoy cars will be allowed to access such areas.

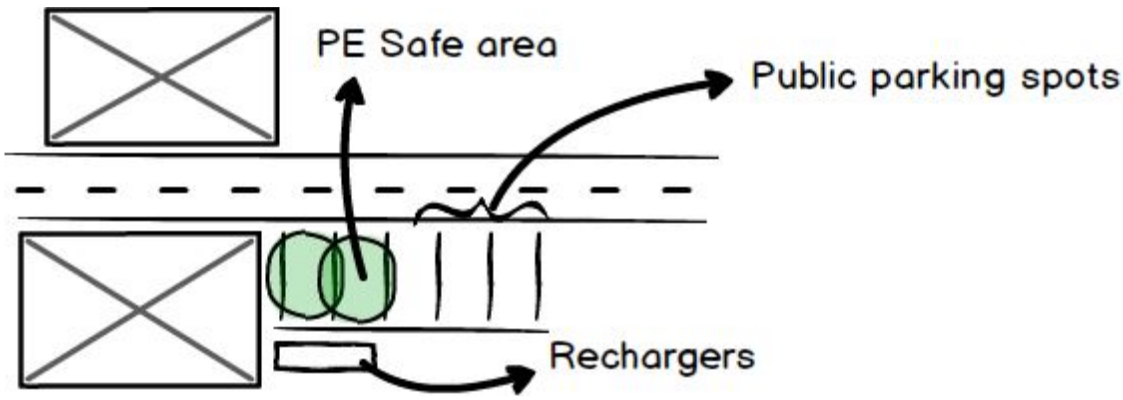
We also mention another option that would fulfill our purposes: the user can simply call a taxi. However, the cost of a taxi is typically much higher, and users may find themselves waiting for a long time (unless they booked the taxi in advance).

1.3. Scope

Power EnJoy is intended to be a service to people who need to commute from one point to another in a city. Thus, the scope of the service is a city. Moreover, the city will allocate a specific set of parking spots to be used by PE. These spots will be called *safe areas*, they will be dedicated to PE cars and will be equipped with electric chargers.

1.3.1. A note on safe areas

Power EnJoy Safe areas are intended to be dedicated parking spots only for PE cars. Every PE safe area will be equipped with proper rechargers and sockets, so that every driver should be inclined to park in one of these spots and connect the plug to recharge the car and get a discount.



1.4. Goals

[G1] Build an effective and reliable electric car sharing system.

- [G1.1] Allow every user to view available cars near or in a given position.
- [G1.2] Allow (activated) registered users to reserve available cars.
- [G1.3] Allow users to drive their reserved car.
- [G1.4] Allow users to cancel the reservation.
- [G1.5] Allow users to access to PE and manage their account.

[G2] Encourage drivers to behave correctly and with respect to the law.

[G3] Offer an easy-to-use system for call center operators to monitor cars and give assistance to drivers.

1.5. Actors

User	A person that interacts with Power EnJoy system using either the Mobile Application or the Web Application. Every User can be an Unregistered user or a Registered User.
Unregistered user	A User that has no active account in Power EnJoy system. They can register a new account or view (but not reserve nor use) a Power EnJoy car on the road.
Registered user	A User that already created an account in Power EnJoy system. The account can be either 'activated' if every informations provided during registration are valid or 'deactivated', if some of the informations are not valid or are pending confirmation.
Driver	A Driver is a Registered user that has an active reservation. The 'driver' status will be dropped as soon as the user stops driving.
Call center operator	Is an operator that answers to user calls and is able to manage the user's reservation, payment issues, accidents, malfunctions, car damages.
Governmental system	It's an external database service that checks the validity of the ID card and driving license of the driver.
Bank	It's an external system that takes care of checking the validity of the payment and assures reliable transactions and manages all eventual payment issues.
Maintenance operator	An operator from an external company that takes care of the maintenance of the car and charging on-site when the car has low battery.

1.6. Stakeholders

During the development of this project, we identified these main stakeholder, each of which has different needs and expectations about this car sharing service:

- *Local administration of Duckburg*: They commissioned this project and they are interested in a reliable service that should improve city transportation.
- *Private electric companies:* they are mainly interested in offering rechargeable spots and supplying car batteries.
- *Car manufacturing companies*: they see in this project an opportunity to relaunch electric cars in the city, thus creating market demand for their products.
- *Users:* they are mostly interested in a fast and reliable mean of transportation that can be used whenever they need it.

1.7. Definitions, acronyms, abbreviations

1.7.1. Definitions

- **Payment information** refers to either a bank account or a credit card number.
- **Device** indicates either a mobile phone / tablet running PEM or a browser that is using PWA.
- **Safe area** refers to any legal parking spot in the city, where any driver can leave the car without interfering with the traffic.
- **PE Safe area** is a safe area that can be used exclusively by PE cars.

1.7.2. Acronyms

- **PE** Power EnJoy system
- **PEM** Power EnJoy Mobile application
- **PEW** Power EnJoy Web application
- **PEA** Power EnJoy Administrator application
- **PEC** Power EnJoy Car application
- **SMS** Short Message Service
- **GPS** Global Positioning System

1.7.3. Abbreviations

- **[Gn]** n-th goal
- **[Dn]** n-th domain assumption
- **[Rn.m]** m-th requirement related to goal [Gn]

1.8. Reference documents

- [1] IEEE Software Engineering Standards Committee, “29148-2011 - Systems and software engineering — Life cycle processes — Requirements engineering”, 2011.
- [2] Payment Card Industry (PCI) Data Security Standard, v3.2, PCI Security Standards Council, LLC.
- [3] Response Times: The 3 Important Limits, JAKOB NIELSEN, 1993.
- [4] Software Engineering 2 course slides.
- [5] The assignment of *Power EnJoy*

1.9. Overview

This document is structured according to the IEEE Std 830-1998 for Software Requirements Specifications and present three sections:

- In the first section, this one, gives a general description of the document and defines the goals and the scope of the software to be developed, specifying also the actors and stakeholders interested in this project.

- The second section describes in more details what the product does and how it is structured, with a special attention to constraints and domain assumptions. It gives also a brief description of the types of user that are more likely to use this service and their characteristics.
- The third and last section specifies all the requirements that are necessary to fulfill the goals given the previous domain assumptions. In particular, it contains some practical scenarios and all the use cases, providing also additional UML diagrams to better explain critical issues. This section also provides a list of requirements that should be fulfilled in order to interface with the user and other external existing systems. Finally, the last part of this section uses an Alloy model to formalize a subset of the requirements, showing also some significant generated worlds.

2. Overall description

2.1. Product perspective

The product, Power EnJoy, is a car sharing service and a software system designed to provide said service. It consists of:

PE website	This website will be a central point that will provide a web identity to the PE service.
PE server	This component will handle all the operations that need to interact with the backend (e.g. read / write data about cars, users, reservations and so on). The component will be installed (and possibly replicated) on dedicated servers and will run indefinitely, responding to connections by clients.
Power EnJoy Mobile	This component is a client of the PE server. It will be distributed as either an Android app, an iPhone app or a Windows Mobile app. In order to install it, the users will need to visit their respective app store.
Power EnJoy Web	This component is another client of the PE server. It will be distributed directly on the PE website. In order to access it, users must have a fairly recent browser installed.
Power EnJoy Car	This is the software component that will be executed on the PE cars. It uses the Android environment and it is connected to a touch screen display.

2.2. Product functions

The features of this product are:

PE server	Expose (not necessarily public) APIs to create and authenticate users, access the list of cars, access the list of reservations and create new ones. Moreover, the cars themselves will connect to the server (e.g. to notify that the battery is running out). In addition, the server is also able to send and receive SMS messages, using them to communicate directly with the user.
Power EnJoy Mobile	The app will allow the creation of an account using a telephone number. The app will also allow users to log in. Once the user is logged in, the app allows to choose a car and reserve it. It also allows the user to browse their past reservations.
Power EnJoy Web	The website allows the same features available in the mobile application.

Power EnJoy Car	This component should send periodical information to PE server about its position and answer with similar informations to PE requests.

2.3. User characteristics

The target audience for this service is mostly the same as traditional public transportation services. Our users are people who need to commute from a point to another in the same city, e.g. to go to work or for leisure activities. Since the local legislation requires every user to have a driving license to use a car (and thus to legally be an adult), our service will also cater to legal adults who are equipped with a driving license.

This service does require specific knowledge and expertise from its users, since it would be highly risky to have an unexperienced user drive a car in a crowded city. Instead of training users, we will trust the state authority for motor vehicles and assume that every user with a valid driving license is qualified to use our service.

We identified two possible types of user:

Type A users

Users that are confident with the car sharing service and will systematically use PE cars for leisure and work activities.
This kind of users are usually prone to routine behaviour (that is not necessarily the most correct behaviour), they know all the locations of PE safe areas and usually plug in the car in the power grid to pay less. They rely on the availability of the service and they get very upset if no cars are available when they need it.

Type B users

Users that almost never used car sharing services, are not confident with it and only needs to get to their destination as soon as possible.
They are mostly interested in usable and user-friendly interfaces. They rely on effective assistance if anything goes wrong.

2.4. Constraints

- Since the users will drive cars, they need to oblige to the road laws.
- Police needs to be able to attribute fines to the users that infringed the law.

2.5. Assumptions and Dependencies

We will make the following domain assumptions:

[D1] PE cars are equipped with a GPS system.

[D2] PE cars are equipped with an Internet connection.

[D3] PE cars always maintain a stable connection to PE.

[D4] PE cars are equipped with sensors to detect how many people are in the car.

[D5] PE cars are equipped with sensors to measure the percentage of battery left.

[D6] PE is equipped with a system capable of sending SMSs and emails.

[D7] GPS systems always find a position with an error of maximum 10 meters.

[D8] Payments are managed by an external system. If the payment didn't work, that same system will take care of it.

[D9] A call center operator must always be available to help the driver in case of accidents, payment issues, malfunction reports.

[D10] PE safe areas consist of a set of parking spots predefined by PE.

[D11] PE safe areas are equipped with charging plugs.

[D12] PE safe areas are equipped with a sensor that is able to detect if a car is parked there or not.

[D13] PE safe areas are assumed to be intended for PE cars only, not for other cars. This means that PE depends on the local administration being willing to reserve parking spots.

[D14] PE depends on an uninterrupted supply of electric current.

[D15] Every driver must have a mobile phone switched on in order to use the car.

[D16] The maintenance of the cars is due to an external company and is done periodically, in order to minimize the risk of breakages.

[D17] The same external company (that handles maintenance) is also responsible for moving cars to/from safe areas in order to keep them in charge: an operator will receive a notification and will move the car and plug it in the power grid.

2.6. Future possible implementation

- Since we store "samples" for each position in time of the car for each ride, the system could analyze this data in order to model the typical car behaviour and be able to detect malfunctions (e.g. unusual battery draining, application of energy saving policies, fraud).
- Since the users don't have to pay immediately, the system could further incentivize good behavior by crediting bonus money (or time) onto the monthly bill based on, for example, whether the user usually respects speed limits or not, usually parks in PE safe areas and plugs the charger to the car or not and so on.
- Display the number of free parkings in each PE safe area on the car display.

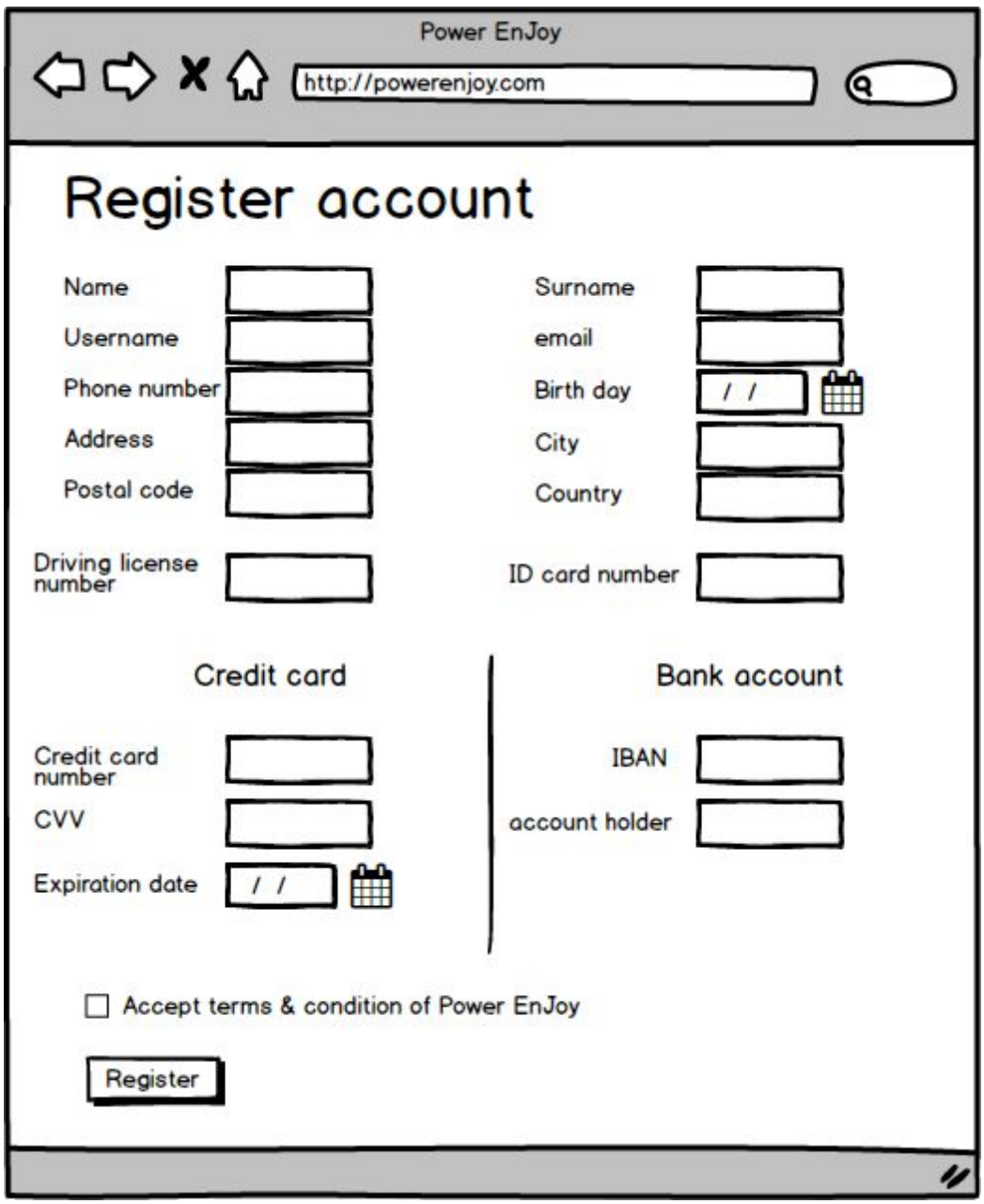
- Integrate the authentication system with existing OpenID providers (e.g. social networks like Facebook or Google+) in order to simplify the registration and log in procedure.
- Introduce the money saving option with which a user can get an additional discount by parking in a spot chosen by PE system to improve car distribution among the city.
- Integrate the assistance service with social networks such as Facebook and Twitter to provide a closer relationship with the user and spread news or status information about the service.

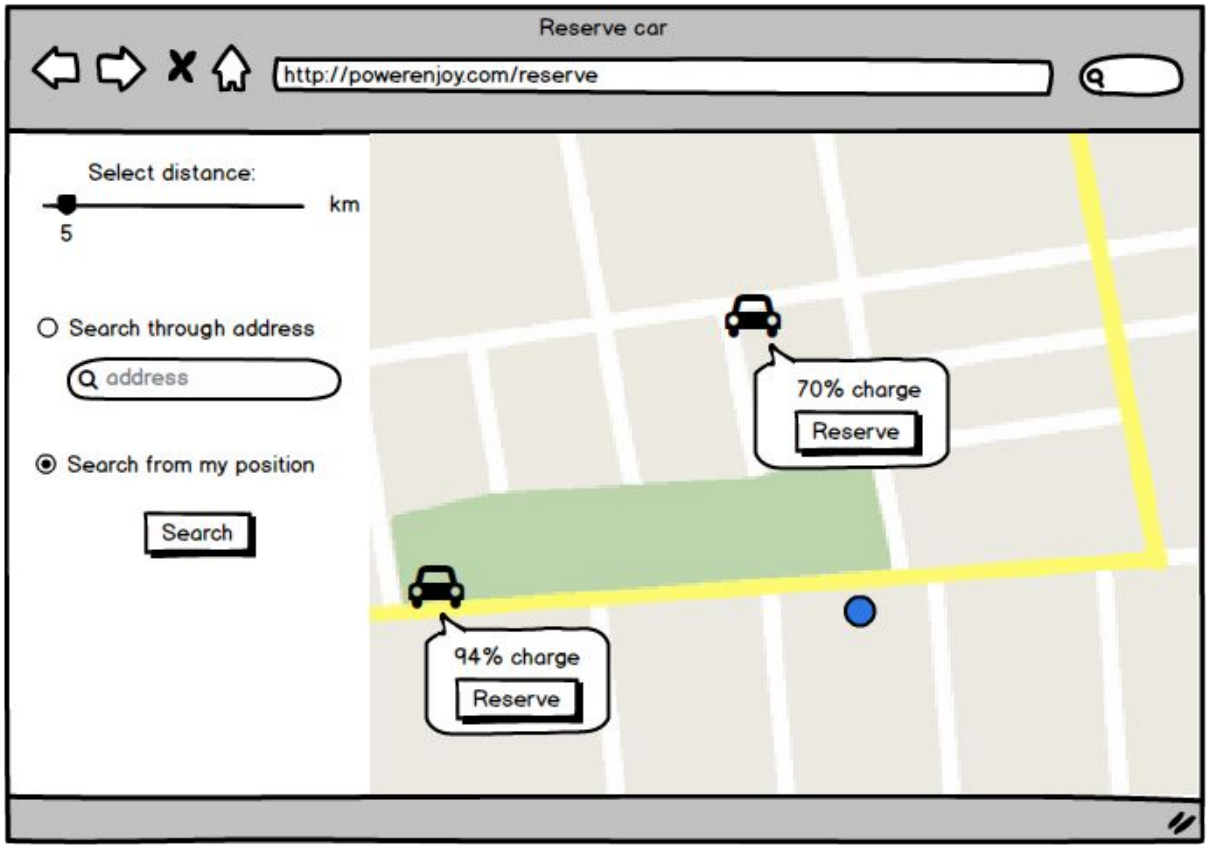
3. Specific requirements

3.1. External Interface Requirements

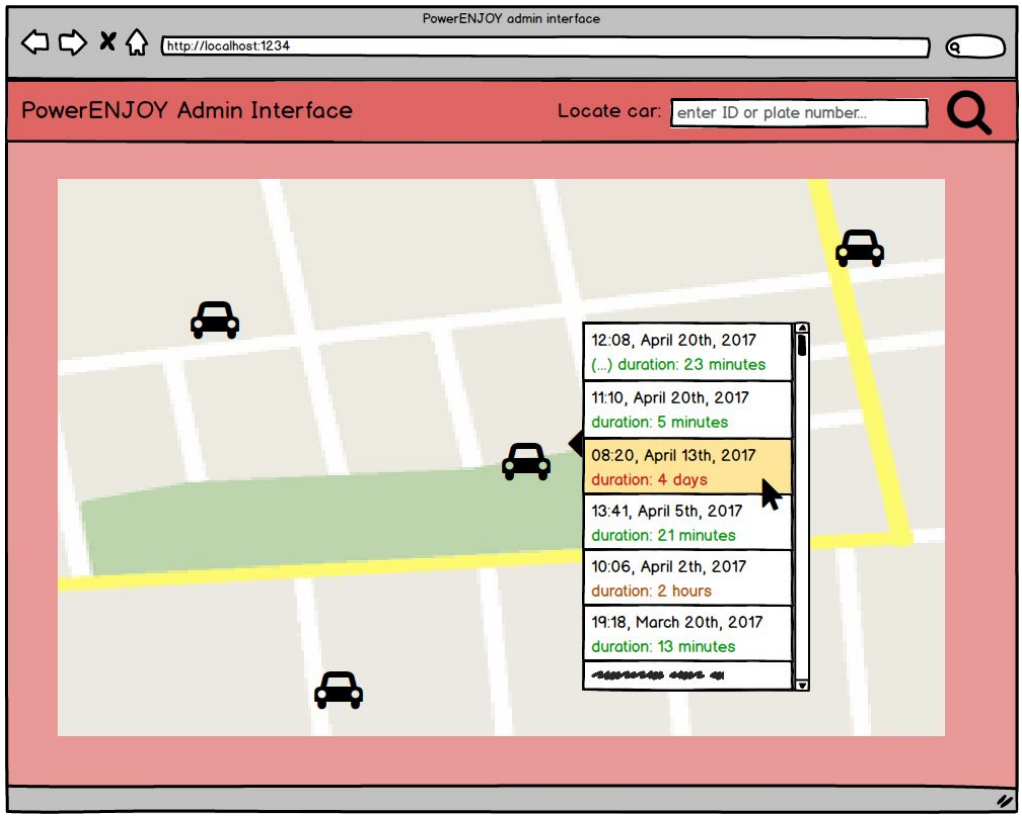
3.1.1. User interfaces

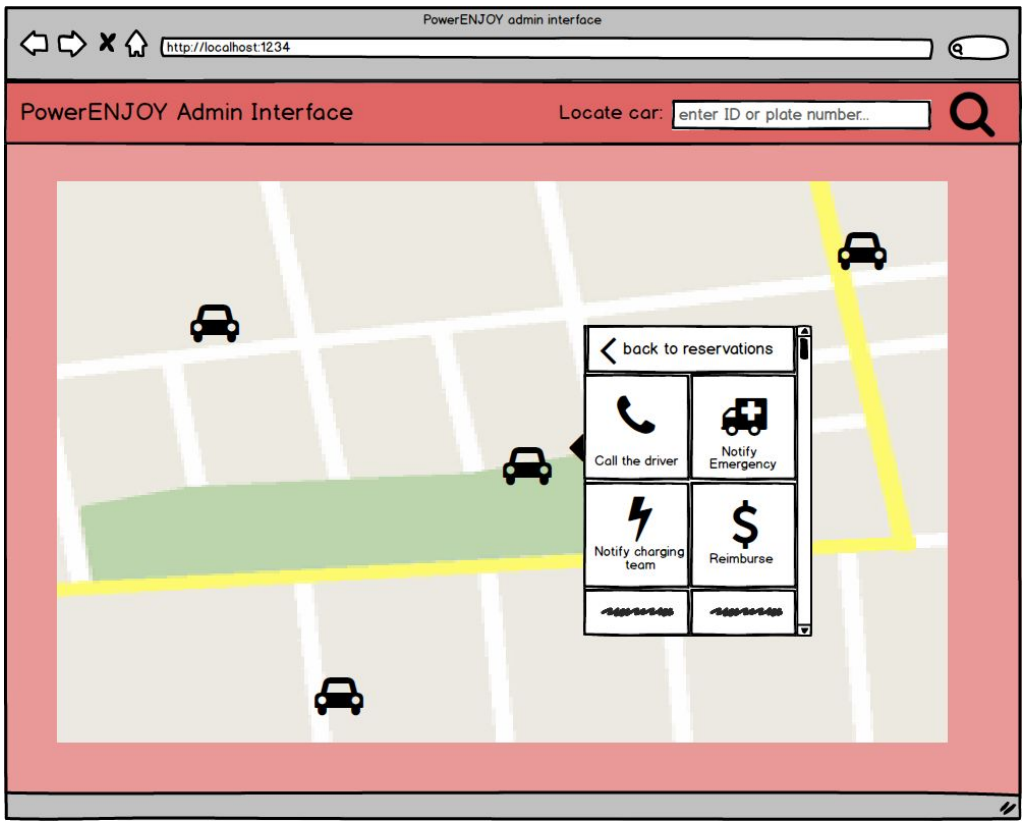
Power EnJoy Web application:



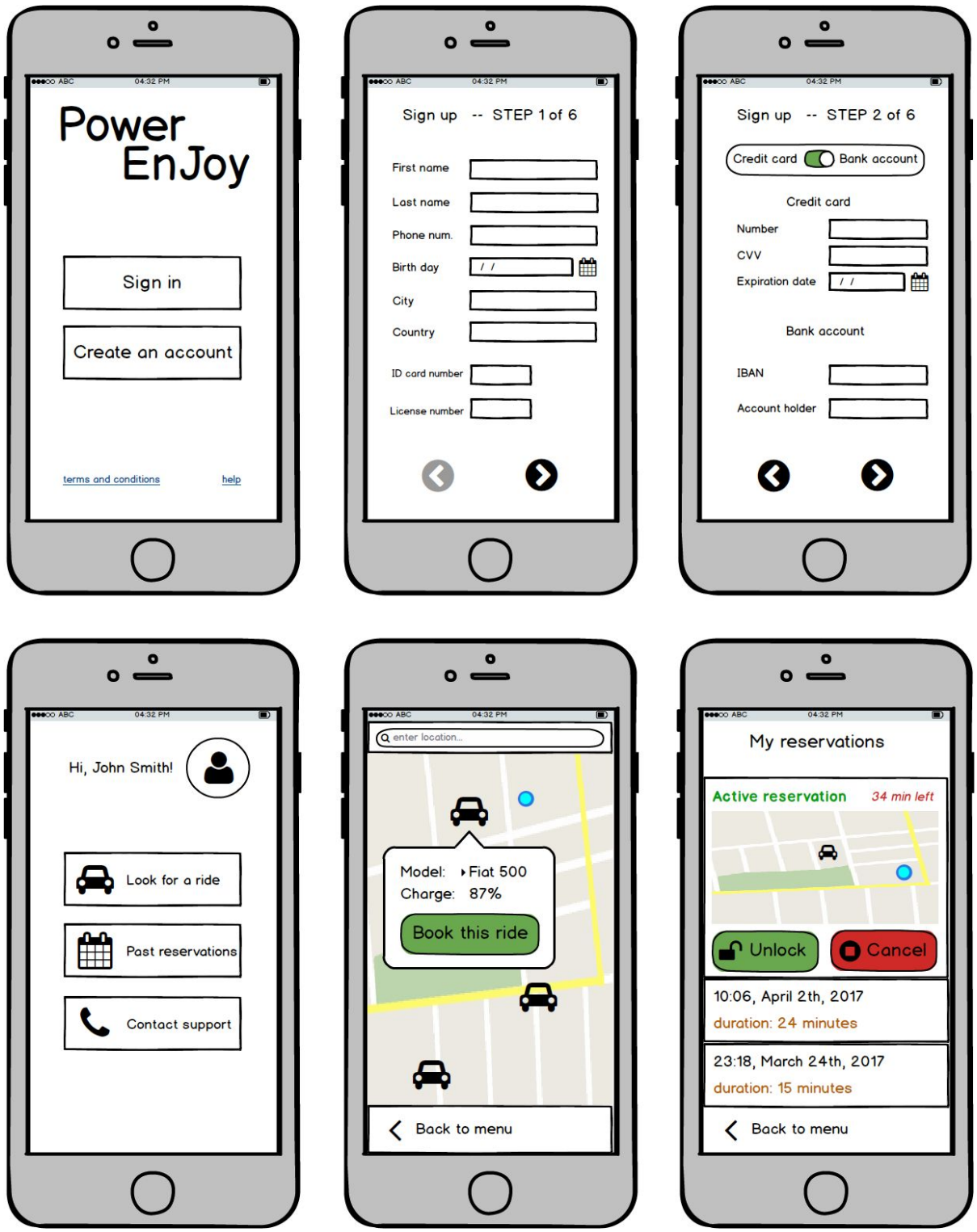

Reserve car
http://powerenjoy.com/reserve
Select distance:
km
5
Search through address
address
Search from my position
Search
70% charge
Reserve
94% charge
Reserve

Power EnJoy Administration panel:

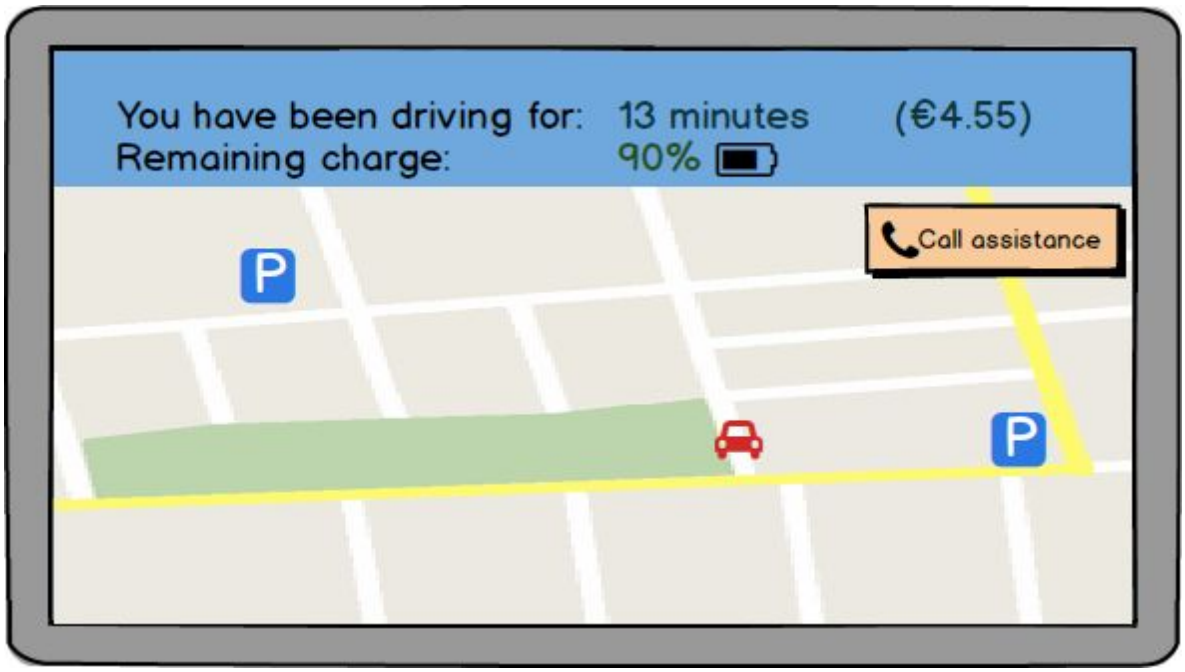




Power EnJoy Mobile application:



Power EnJoy Car application:



3.1.2. Software interfaces

PE employs the following APIs:

Maps API	In order to access geographical data, show maps, search for locations, compute distances, and so on, we will acquire a license to use an external API such as: Google Maps API, or OpenStreetMap API.
Geolocation API	In order to recognize the location of the user we will use some API, not necessarily the same for each device. For the PEW application, a possible choice is navigator.geolocation. For the PEM application, a possible choice is android.location or Google Location Services.
Message queue	In order to handle notifications (e.g. to notify the maintenance team that a PE car needs to be charged, or to notify a call center operator that a user is requesting help, and so on) we will use a message queue such as: RabbitMQ, Apache ActiveMQ and many others.

Moreover, PE exposes the following APIs:

Car availability	This API offers a (rate-limited) JSON endpoint that, for a given position, lists all the cars that are within a 3 km radius. This API can

	be used by third-party services which want to suggest commuting options to their users. For example, an Android app for booking movie theater tickets might show which trains / buses / PE cars are available near their users (or near the theater), thus allowing them to consider many different ways of commuting between their home and the theater without forcing them to download PEM or to use PEW.
PE status	This API will reply to queries with statistics on the availability of the PE system (e.g. 99.97% uptime) and on the availability of other PE-owned APIs.

3.1.3. Hardware interfaces

PE employs the following hardware APIs:

On-board touchscreen computer	Each car will be equipped with a touchscreen computer that will be used to connect to the Internet and send information (like the battery status, the number of people in the car, and so on) and events (like “started driving”, “request for assistance”, and so on) to PE. This touchscreen computer will be compatible with Android, so that we will be able to install our software on it.
Compatible electric plugs	Each PE safe area will be equipped with a plug that is compatible with the cars’ sockets. Since a car can be collected at a different location than the one where it will be left at, it’s wise to just adopt one type of plug/socket (or more, as long as they are all compatible with each other). Adapters could also be provided in some way (e.g. on the car dashboard) in order to account for technological advancements that could lead to better (backwards incompatible) plugs/sockets.

3.1.4. Communication interfaces

Internet	In order to work correctly, the PE server, as well as the PEW and PEM applications, need to be connected to the Internet. Moreover, in order to use the Internet to communicate, all these components will employ the HTTP (or, at least, TCP/IP) protocol. The PE server will also need to be able to send and receive emails, so it will also employ the SMTP and POP3 protocols.
SMS	The PE server will also need to be able to communicate via SMS, to support users that don’t have constant Internet access.

3.2. Functional requirements

3.2.1. [G1] Build an effective and reliable electric car sharing system

[G1.1] View available cars nearby or in a given position

[R1.1] PEM and PEW should be able to retrieve device position either with a GPS antenna or by using browser geolocalization after obtaining the user's permission.

[R1.2] PE should be able to find the GPS coordinate given a specific address.

[R1.3] Every car must respond with their GPS coordinate when it is asked by PE.

[R1.4] The user should be able to specify the range of the area where to find available cars.

[R1.5] The user should be able to visualize the battery level of available cars.

[R1.6] Each car must have a fixed minimum battery level in order to be considered available.

[G1.2] Reserve available cars

[R1.7] Each user should have only one active reservation at a time.

[R1.8] All reservations must be related to only one car and each car can be bound to at most one reservation at a time.

[R1.9] When the user reserves a car, it automatically becomes reserved and won't be visible to other users until the reservation is cancelled or he/she stops driving.

[R1.10] Each reservation should expire after a fixed amount of time (e.g. 1 hour). If the user forgets to cancel the reservation within that deadline, user should incur in a penalty fee.

[R1.11] A car should be available only if it has at least 20% battery full.

[G1.3] Drive the reserved car

[R1.12] Every car should be able to unlock every time they receive an unlock message by PE.

[R1.13] The user should be able to unlock the car either via SMS (e.g. with an 'UNLOCK' message) or PEM.

[R1.14] The car should lock the door after a fixed amount of time (e.g. 30 seconds) they have been unlocked. If the doors are still open after that time, the car will lock as soon as the doors closes.

[R1.15] The user should be able to stop the ride either via SMS (e.g. with a ‘STOP’ message) or PEM

[R1.16] The user should be able to pay for all his rides monthly.

[R1.17] The car should be able to detect whether it is on a PE safe area or not.

[R1.18] The car should be able to detect the number of passengers it is carrying.

[R1.19] User should be aware of current charge when driving through a screen in the car.

[R1.20] PE should start charging the user either when the car engine ignites or when a fixed amount of time (e.g. 10 minutes) is passed from the first unlock made by the user.

[G1.4] Cancel the reservation

[R1.21] User should be able to cancel a pending reservation only before he/she unlocks the car for the first time

[R1.22] User should be able to view his/her pending reservation.

[G1.5] Allow users to access to PE and manage their account.

[R1.23] PE should verify all the data changed by the user

[R1.24] User should be able to:

- Change password
- Change/update driving license and ID card
- Change phone number
- View pending reservation (if any) and all past reservation
- View the bills to pay and/or already paid
- Change payment informations

[R1.25] Users should be able to register and create an account.

[R1.26] Users should be able to log in with a previously created account

3.2.2. [G2] Encourage drivers to behave correctly and with respect to the law.

[R2.1] PE should be able to retrieve driver information for a specified car in a given time.

[R2.2] PE should be able to verify the correctness of the driving license and ID card of a driver (e.g. using a governmental system)

[R2.3] PE should apply some discounts on the ride in this situations:
- The driver is carrying at least 2 passengers
- The driver leaves the car with at least 50% of battery full.
- The driver parks in a PE safe area and plugs the car in the power grid

[R2.4] PE should apply some penalty fees on the ride in this situations:
- The driver parks the car far from the nearest PE safe area (e.g. the nearest PE safe area is at least 3 km far)
- The driver leaves the car with less than 20% of battery full.

[R2.5] PE should periodically check the validity of the documents presented by each user and forbid user to reserve cars with out-of-date documents.

3.2.3. [G3] Monitor cars and give assistance to drivers

[R3.1] Call center operators should be able to:
- View the position of all the cars on road
- Notify maintenance operators
- Reimburse the driver
- Call the driver
- Notify police/ambulance of emergencies.
- View the reservation history of each car.

3.3. Scenarios

In this section we provide several different scenarios, each of which should clarify how the user interacts with the system in practical situations.

3.3.1. Scenario 1 - User registration and login

Robert is a tech fan and always wants to try out new things. He just discovered PE and immediately downloaded PEM. Next, he inserted both his credentials, payment information and the numbers of his driving license and his ID card. Robert is also asked to read and accept the terms and conditions. He follows the procedure and after a while he receives an email with a password that he must use to access PE system. Robert can now log in with his email and password and search for available cars.

3.3.2. Scenario 2 - Parking far from recharging station

Anna is a university student and currently studies abroad. It’s been a long time since she had last seen her parents so she decided to surprise them by getting home. She landed in the nearest airport and took a train to the city. She used a PE car to get home. Unfortunately, her parents' house is far from the city center and the nearest charging station is 3.5km away. Hence, when she parked the car and finished her run, she received the check with an additional 30% charge to compensate for the cost required to re-charge the car on-site.

3.3.3. Scenario 3 - Recharging the car after use

Bob is an environmentalist and takes very seriously the issue of climate change. To get around the city he usually uses his electric car but unfortunately he forgot to charge the battery the day before. Hence he used PEM and reserved a car. When he arrived close to the car 15 minutes later, he opened PEM and he sent a notification to the system to unlock the car. Next, he used the car and got to his workplace. He noticed on the car display that the battery level was under 50%, so he parked close to a recharging station and he plugged the car into the power grid. Soon later, he received the check on his mobile phone with a 30% discount and an acknowledgment for his good action.

3.3.4. Scenario 4 - Reservation time expired

Carl is always in a hurry and hates losing any minute of his precious time. For this reason, he reserved a car through the mobile application while he was still at work, in order to be ready to pick it up 30 minutes later and get home early that day. Unfortunately his boss called him for an unexpected meeting that lasted 2 hours. Carl forgot to cancel his previous reservation, so the system charged him a fee and made the reserved car available again.

3.3.5. Scenario 5 - Using Power EnJoy with friends

Dave enjoys going out on Saturday nights and comes back home as late as possible with his two friends, Mario and John. Neither he nor his friends have their own car, so they usually use public transportation to get to the pubs. Unfortunately, public transportation service in the city is not available late at night. Therefore, when they wanted to get home, Dave used PEM to reserve the closest car available. When they got in the car, the car detected the additional passengers and informed Dave that a discount would be applied. Dave brought each of his friends at their home and parked the car in a safe area. Soon after he get out the car, he received his check and a notification of the applied discount.

3.3.6. Scenario 6 - Accidents

Fred the Trouble Maker is always very unlucky. He signed in the system with his mobile phone and picked up a car to get home. He was happily driving down main street when a truck turned suddenly and struck the car. Fred, luckily undamaged, immediately phoned the PE call center and reported the accident. After a while, Simon the operator came by and helped Fred with all the required procedure.

3.3.7. Scenario 7 - Cancel a reservation

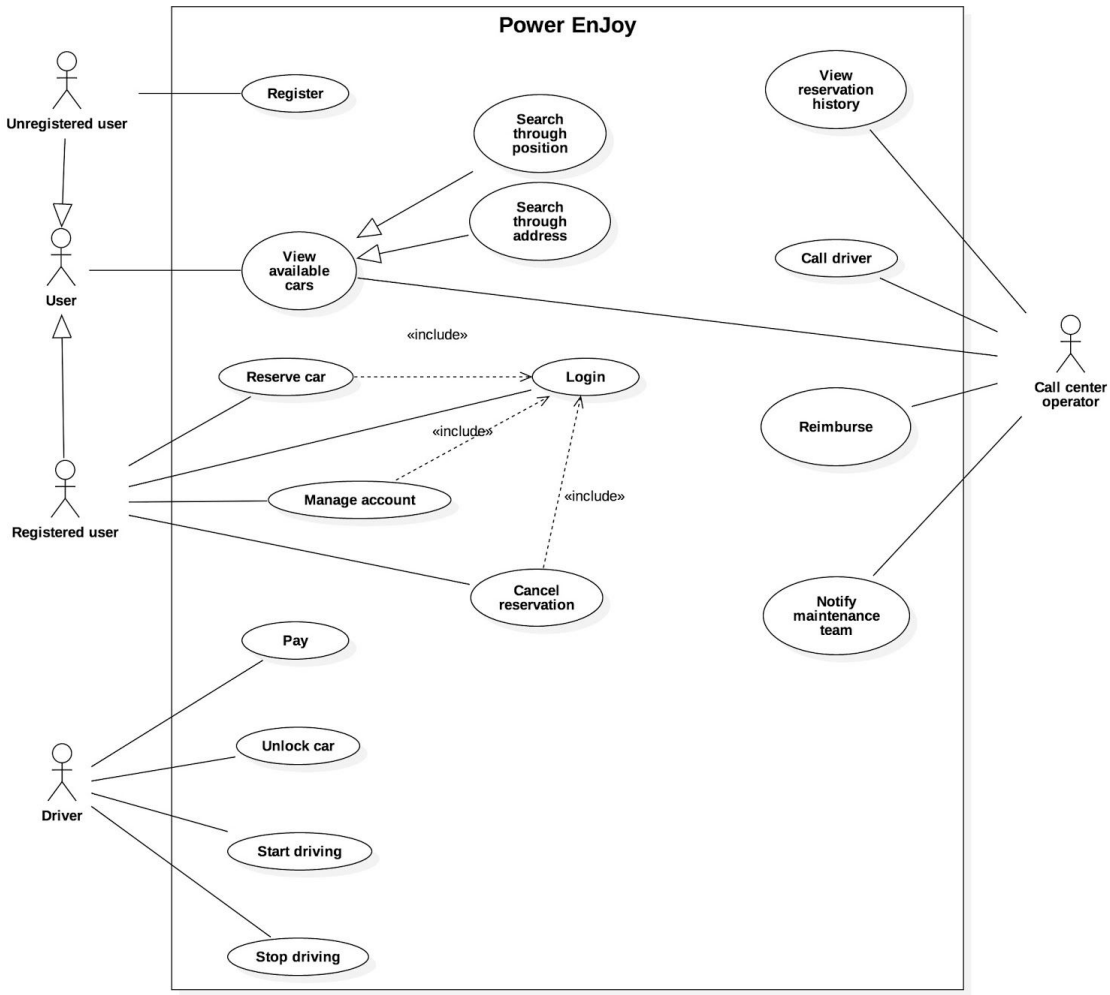
Elaine is always very busy. This time, she planned to go shopping in the town centre, so she reserved a car with PEM. Soon after she did the reservation, though, she remembered that she had to finish a report within that evening, hence she used PEM again to cancel her reservation. The car got available again and no fee was charged to Elaine because she canceled the reservation within a certain fixed time (e.g. 1 hour).

3.3.8. Scenario 8 - Web app

George is out of mobile data on his phone. Fortunately, he is at home and can log in to PE system through his computer. He found a nearby car available and reserved it. Then, he reached the selected car and sent an SMS to PE using his phone. After about a minute, the car unlocked and George used it.

3.4. Use cases

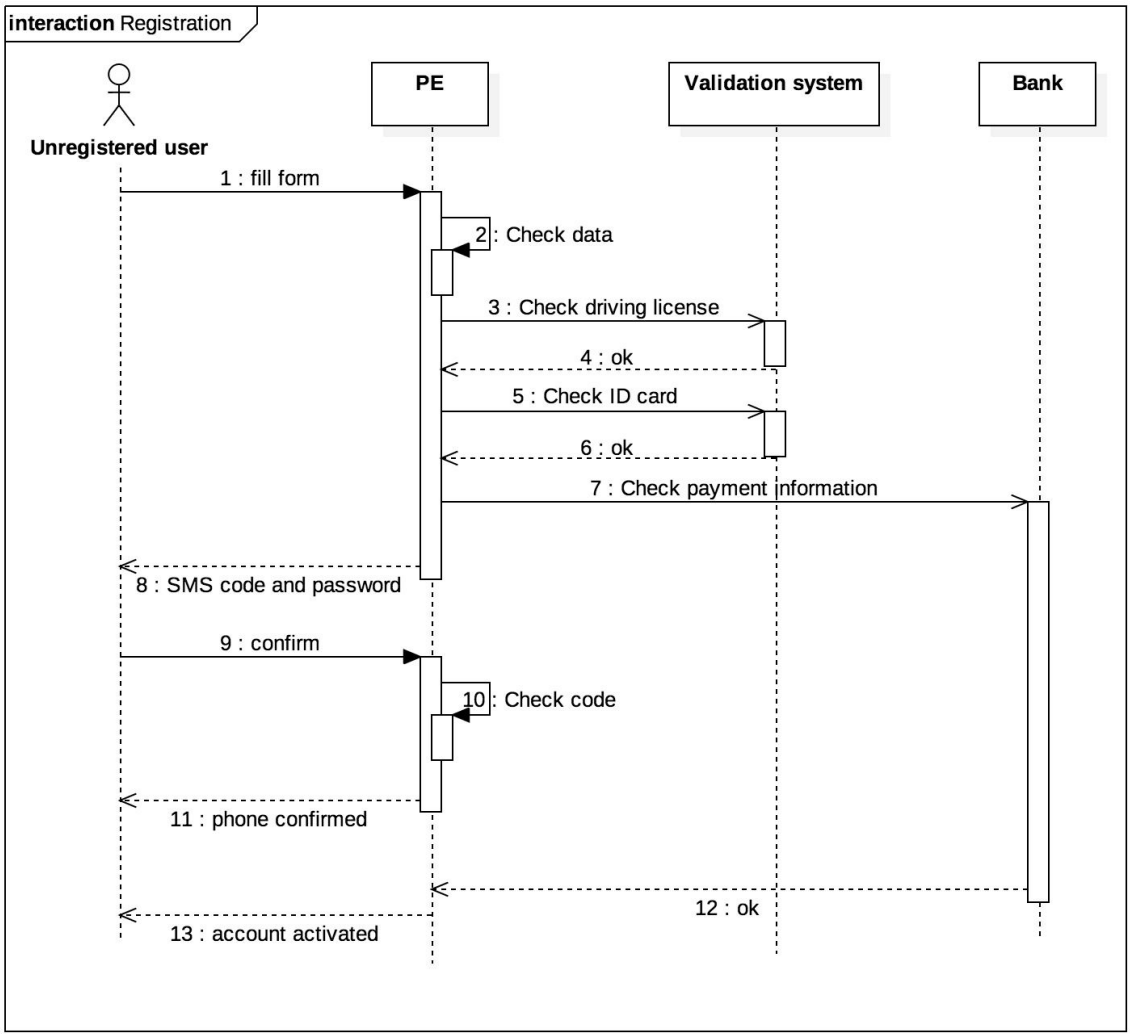
3.4.1. Use case diagram



3.4.2. Register account

Name	Register account
Actors	Unregistered user
Related goals	G1.5
Entry condition	The User wants to create an account and use PE service.
Flow of events	1. User accesses the registration form on his device 2. User fills in the form with required information (name, surname, birthday, email, username, address, phone number, driving license number, ID card number, payment

	information) 3. User must agree with the Terms and Conditions of the service 4. User submits data to PE server 5. PE checks the validity of the inserted data with the help of a governmental validation system and a bank. 6. PE creates a new account sends a code via SMS to User. 7. User inserts the code received via SMS and submits. 8. PE checks the code, generates a password and sends the generated password to User via SMS. 9. As soon as the password is generated and PE receives a confirmation from the bank, the account is activated and a notification is sent to User.
Exit condition	User inserted valid data that will be stored in a database.
Exceptions	● If some information provided by the User are invalid, the whole procedure stops and an error message is returned to the device. Then User can restart the procedure from the beginning and insert correct data. ● If the procedure is interrupted before its completion (connection lost, breakdowns, system errors...) the transaction will roll back and no modifications are done in PE system. ● If User choose to pay with bank account but the bank rejects the payment, a message will be sent to User notifying him to change the payment information to activate the account.
Special requirements	● *Payment information* refers to a bank account or a credit card number. Such a payment method must ensure that any abuse by the User (fines, damages, misappropriation, etc.) can be properly indebted.



3.4.3. View available cars through position

Name	View available cars through position
Actors	Unregistered and Registered Users
Related goals	G1.1
Entry condition	The User wants to view the position and availability of the cars using the GPS tracking information.
Flow of events	1. User specifies maximum distance from his location. 2. If localization is disabled in User's device, a popup will appear asking to activate GPS. 3. GPS system detects User position and send position and distance information to PE. 4. PE search for available cars in this area and sends a list of nearby cars to device.

	5. The device receives these informations from PE and shows the position of the cars in the list over a map. Each car must provide informations on the remaining battery level.
Exit condition	User can see available car nearby and their battery level.
Exceptions	● If localization is not available on the device an error message is shown together with the advice to search available through a specified address instead. ● If no available car is found in the area, a dialog will pop up informing the User of this fact and recommending the User to repeat the process with a wider range.
Special requirements	-

3.4.4. View available cars through specified address

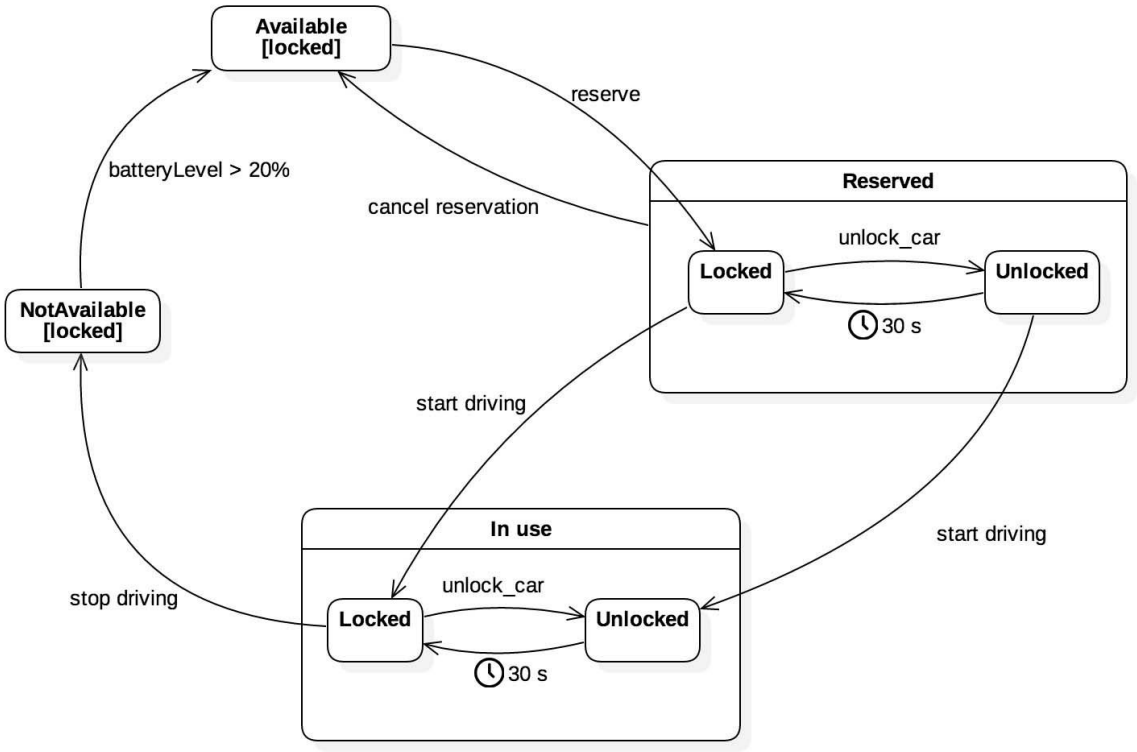
Name	View available cars through specified address
Actors	Unregistered and Registered Users
Related goals	G1.1
Entry condition	The User wants to view the position and availability of the cars specifying an address.
Flow of events	1. User inserts the address and the maximum distance in a form and submits it to PE. 2. PE receives the address. 3. PE elaborates the address and find the related GPS coordinate 4. PE search for available cars in this area and sends a list of nearby cars to device. 5. The device receives these informations from PE and shows the position of the cars in the list over a map. Each car must provide informations on the remaining battery level.
Exit condition	User can see available car in the specified location and their battery level.
Exceptions	● If PE can’t find GPS coordinate related to the specified address (wrong address, ambiguity) an error message is returned to device stating that no location was found related to specified address. User should then retry and type a correct address.

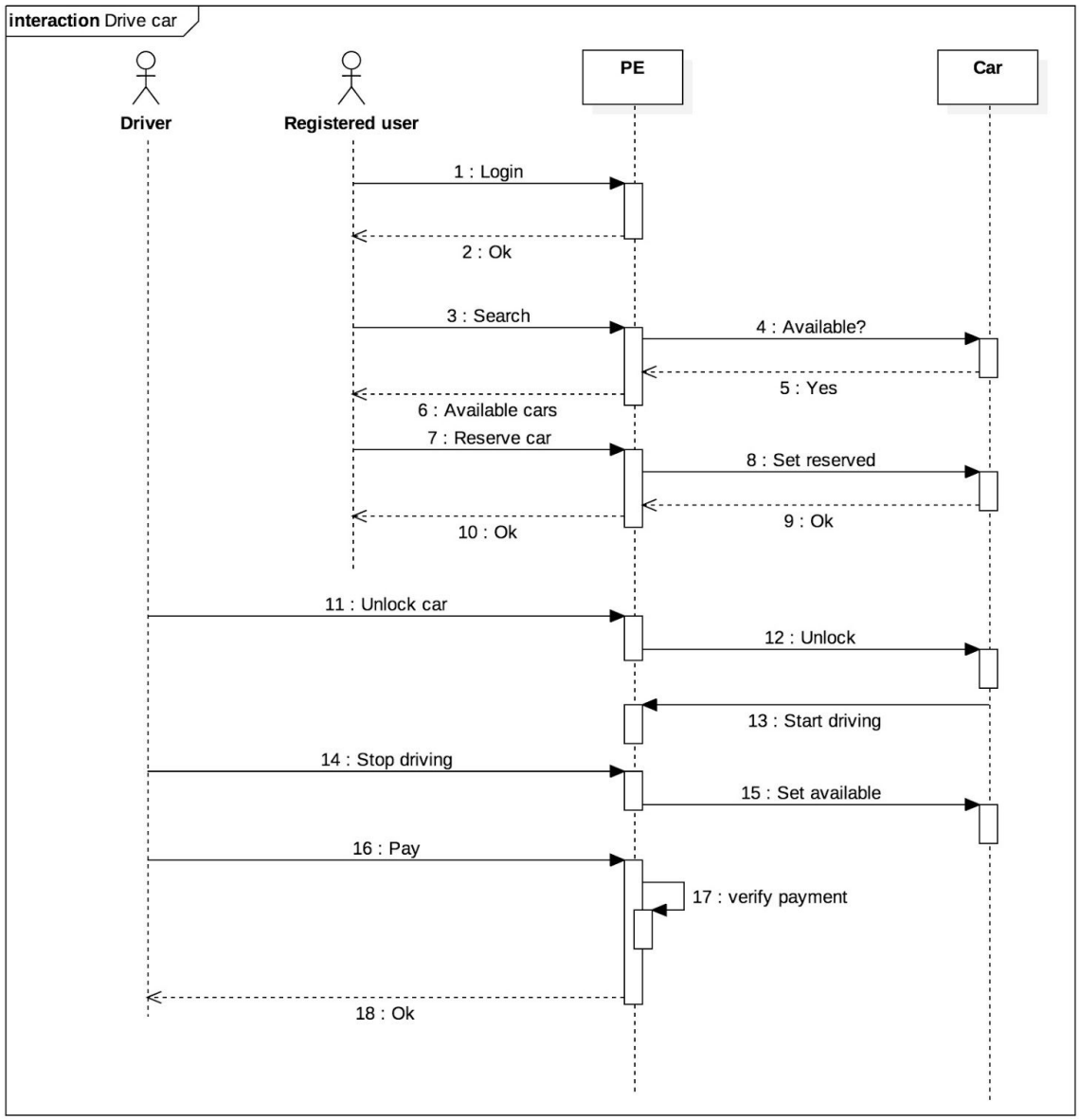
	● If no available car is found in the area, a dialog will pop up informing the User of this fact and recommending the User to repeat the process with a wider range.
Special requirements	-

3.4.5. Reserve car

Name	Reserve car
Actors	Registered User
Related goals	G1.2
Entry condition	Registered User must be logged in with a valid account.
Flow of events	1. User starts the searching procedure (refer to '*View available cars through position*' or '*View available cars through specified address*' use cases) 2. User selects the car to reserve from the map on his device 3. User selects 'reserve car' 4. A reservation request is sent to PE and a new reservation is stored in a database. The car is set to be unavailable to any other User. 5. PE responds with a confirmation message on User device.
Exit condition	User has reserved a car. He can unlock the car within 1 hour and no one else can reserve that car until the reservation is expired or he finished driving.
Exceptions	● If multiple requests arrive to PE at the same time, only one (the first to be processed) can be satisfied. The other requests will fail returning an error message to User.
Special requirements	● A reservation is granted only to one User. ● PE must be able to handle concurrent requests.

The following statechart shows how the state of a PE car change in time.





3.4.6. Login

Name	Login
Actors	Registered Users
Related goals	G1.5
Entry condition	The Registered User wants to log in on his device.

Flow of events	1. The Registered User accesses the PEM or PEW and is presented with a login page. 2. The user enters his credentials and confirms. 3. PE responds with either a confirmation message or an error (along with a new, blank, login page).
Exit condition	The user is logged in the PE system
Exceptions	● If the credentials are incorrect, the PE system lets the user try again. ● If the number of login attempts exceeds a fixed quota (e.g. 10 attempts) then the account is locked, and the user will have to contact the customer service to get it unlocked.
Special requirements	-

3.4.7. Cancel reservation

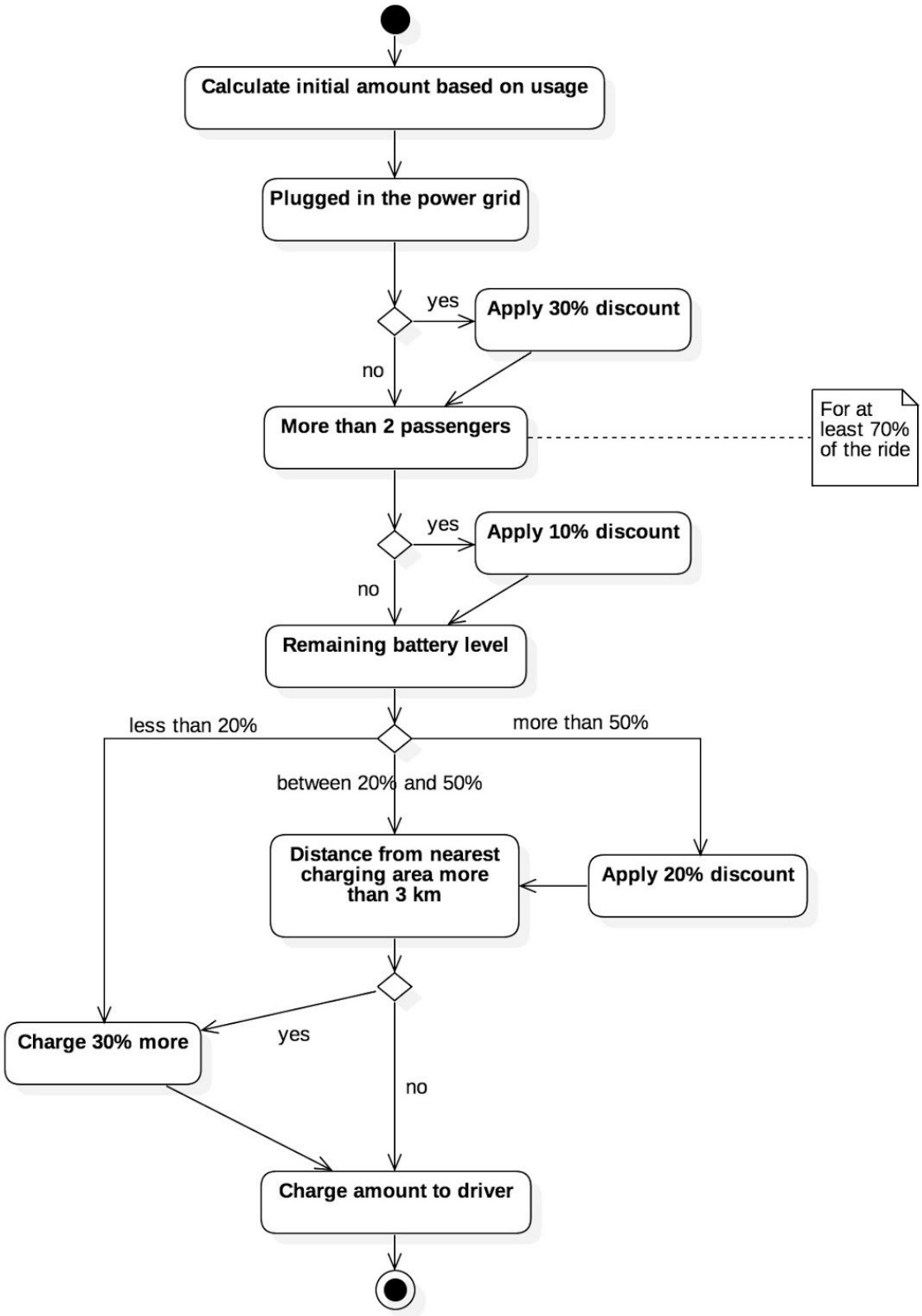
Name	Cancel reservation
Actors	Registered Users
Related goals	G1.4
Entry condition	The user must be logged in, and has to have at least 1 active reservation.
Flow of events	1. The user opens the PEW or PEM. 2. The user accesses their reservation list. 3. The user selects the reservation they want to cancel. 4. Once the reservation is selected, the system (PEW or PEM) will show what actions can be performed (e.g. modify a reservation, select a different car, and so on). 5. The user selects the “cancel reservation” action.
Exit condition	The reservation is canceled and the car is made available to other users.
Exceptions	● If the user is currently in in the middle of the ride, then the system will not let them cancel the reservation for the ride. They can only terminate the ride.
Special requirements	-

3.4.8. Pay

Name	Pay
Actors	Driver
Related goals	G1.3, G2
Entry condition	–
Flow of events	1. The User receives an SMS with the amount to pay after each ride. 2. Every month, PE checks the debit of each Users and sends the total amount to pay to the banking system. 3. The bank provides with the payment procedure. 4. The bank sends a confirmation of the payment to PE.
Exit condition	–
Exceptions	● If the User can't afford the check, PE must provide to deactivate the account until all the debt has been paid.
Special requirements	-

How to calculate the corresponding amount to pay:

An important aspect of the payment procedure is the due amount to pay and how to calculate it. In order to clarify this aspect, we put an activity diagram below showing the steps that the software should execute. We decided to apply the discounts sequentially instead of summing up the percentage (e.g. apply 30% and then 10% instead of applying the sum of 40%) because this way is it more profitable for our client. Note that the order in which the discounts are applied is not relevant. Also note that a 30% discount and a 30% additional charge does not cancel out in this way, resulting in a 'final' discount of nearly 9%, thus favouring the driver.


Calculate initial amount based on usage
Plugged in the power grid
yes
Apply 30% discount
no
More than 2 passengers
For at least 70% of the ride
yes
Apply 10% discount
no
Remaining battery level
less than 20%
more than 50%
between 20% and 50%
Distance from nearest charging area more than 3 km
Apply 20% discount
Charge 30% more
yes
no
Charge amount to driver

3.4.9. Unlock car

Name	Unlock car
Actors	Driver
Related goals	G1.3
Entry condition	User has reserved a car and wants to open it and drive.
Flow of events	Unlock via SMS: 1. User uses his mobile phone and sends a SMS with 'UNLOCK' text to PE. 2. PE receives the SMS and finds the reservation associated to that phone number 3. PE sends an unlock message to the associated car Unlock via PEM: 1. User opens PEM and goes to the page with his reservation details 2. User touches the "unlock" button 3. If PEM has access to GPS and it finds that the car position is more than 50m far from the current User position, a warning message will appear to ask for confirmation and prevent accidental mistakes. 4. PEM sends an unlock message to PE 5. PE receives the message and checks the reservation validity 6. PE sends an unlock message to the associated car
Exit condition	The car is unlocked for 30 seconds.
Exceptions	● If PE receives an SMS whose content is not recognized (e.g. "ulock") the SMS will be ignored and no actions will be made. ● If PE receives a correct SMS from a phone number not associated to any valid reservation, the SMS will be ignored and no actions will be made. ● If PE receives a message from PEM but the associated reservation is not valid for any reason, PE will send back an error message to PEM.
Special requirements	-

3.4.10. Start driving

Name	Start driving
Actors	Driver
Related goals	G1.3
Entry condition	User has unlocked and entered in the car
Flow of events	1. The Driver turns on the car with the start & stop button. 2. The car sends a message to PE and starts charging the Driver 3. PE receives the message and stores it in the database
Exit condition	The car is running and PE starts charging the user.
Exceptions	● If a fixed amount of time (e.g. 10 minutes) has been passed from the first unlock by the user, PE should begin charging the user as if he/she sent a start driving message.
Special requirements	-

3.4.11. Stop driving

Name	Stop driving
Actors	Driver
Related goals	G1.3
Entry condition	Driver has parked the car in a safe area and is off the car
Flow of events	Stop driving via SMS: 1. Driver sends a “STOP” text message to PE. 2. PE checks if the car is locked and plugged in the power grid. 3. PE calculates the amount to pay for the driver. 4. PE sets the car as available to other users. 5. PE sends back to the driver a confirmation SMS with the bill for his/her ride. Stop driving via PEM: 1. Driver open PEM and access to his current reservation. 2. Driver clicks on “stop ride” button on his device. 3. PEM sends a stop message to PE

	4. PE does all the required checks as for points 2, 3, 4 of previous “stop driving via SMS” situation. 5. PE sends back to the driver a confirmation SMS with the bill for his/her ride. The bill is also received and displayed in PEM.
Exit condition	Car is available to other users
Exceptions	● If the car is not locked when PE receives the stop message, the driver should receive an error message saying that the car must be locked before stop driving. ● If PE receives a message from a phone number that has no pending reservation, the message will be ignored.
Special requirements	-

3.4.12. Manage account

Name	Manage account
Actors	Registered users
Related goals	G1.5
Entry condition	No entry conditions
Flow of events	1. User logs in with his/her account using his/her device (either PEM or PEW) 2. User selects the personal page. 3. User can either: a. Change password b. Change/update driving license and ID card c. Change phone number d. View pending reservation (if any) and all past reservation e. View the bills to pay and/or already paid f. Change payment informations 4. As for points 3.b, 3.c and 3.f, PE must check the validity of changed informations as in the registration process. (view ‘Register’ use case for reference). The account should be unactivated until a confirmation of the correctness of data is received by PE. 5. PE stores changed values in a database.
Exit condition	No exit conditions
Exceptions	● If the values inserted by the user are not valid, the account should be unactivated until he/she provides

	correct informations.
Special requirements	-

3.4.13. View reservation history

Name	View reservation history
Actors	Call center operator
Related goals	G3
Entry condition	No entry condition
Flow of events	1. Operator opens PEA. 2. PEA shows all the cars on road in a map in real time. 3. Operator search a car by plate number or selects it from the map 4. PEA returns a list of all the past (or pending, if any) reservations related to that car.
Exit condition	No exit conditions
Exceptions	No exceptions
Special requirements	-

3.4.14. Call driver

Name	Call driver
Actors	Call center operator
Related goals	G3
Entry condition	Operator needs to call the driver of a specific ride.
Flow of events	1. Operator uses PEA in the same way as ‘view reservation history’ use case. 2. Operator clicks on the specific reservation and another panel should appear with the operations he/she can do with that reservation. 3. Operator clicks on ‘call driver’ 4. PEA starts a call on the phone number specified by the related user.

Exit condition	No exit conditions
Exceptions	No exceptions
Special requirements	-

3.4.15. Reimburse

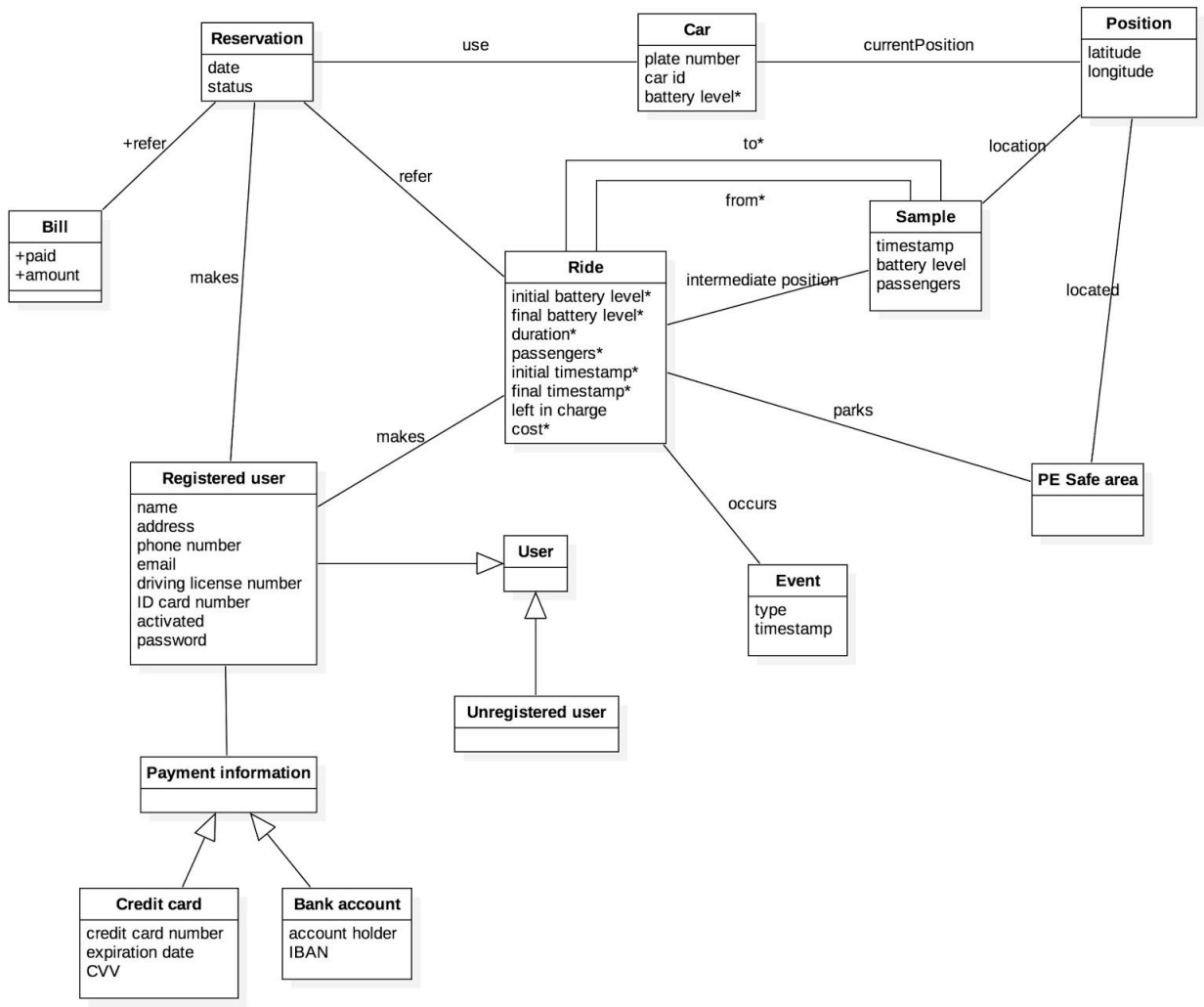
Name	Reimburse
Actors	Call center operator
Related goals	G3
Entry condition	Operator receives a call asking for a reimburse due to faults in the system.
Flow of events	1. Operator accesses the user's reservation on PEA. 2. Operator selects the reimburse option and specifies the amount of the reimburse. 3. PEA sends an email notification to accounting office for further checks on the validity of the reimburse 4. PEA adds the specified credit on user's bills
Exit condition	No exit conditions
Exceptions	● No exceptions
Special requirements	-

3.4.16. Notify maintenance team

Name	Notify maintenance team
Actors	Call center operator
Related goals	G3
Entry condition	Operator notices that a car has low battery or a user reports some damages to a car.
Flow of events	1. Operator selects the car on PEA 2. Operator selects 'notify maintenance team' option 3. Operator writes a brief description of what is the problem. 4. PEA will send an email to maintenance team with the location of the car and the brief description attached

Exit condition	Maintenance team will receive an email and will provide to solve the problem.
Exceptions	No exceptions
Special requirements	-

3.5. Class diagram



3.6. Performance requirements

PE needs to always be performant, at least to some extent, in order to satisfy our goal of building an effective car sharing service. There are a number of factors that could lead up to a degradation of PE's performance:

- In a city, it's not unusual to see strikes to the public transport service. If there's a sudden interruption to that service, we can expect that the number of connections to PEW and PEM will have a spike in frequency.
- Since we offer a publicly accessible API to check the availability of PE cars, we may suddenly find ourselves flooded with requests for similar reasons.

The reputation of PE would be damaged if a spike in connections caused PE users (or third-party apps that rely on our API) to notice server issues.

3.7. Software system requirements

3.7.1. Reliability

This service deals with sensible data, such as license number, ID number and credit cards. To assure that all the informations are authentic, PE should interface with governmental systems and banks to check the validity of the data inserted by the user.

3.7.2. Availability

PE users are going to rely on this service as an alternative to traditional public transportation services, so it's very important to guarantee an equal (if not higher) availability compared to those services. It's reasonable to assume that, every time public transportation services aren't available, there will be a spike in PE reservations.

To guarantee a high availability, it's important to:

- Provision multiple servers, possibly with an auto scaling mechanism that will automatically start up more servers when the load increases. This can also help avoid downtime when doing maintenance: updates can be applied on individual servers while the others continue being active, and then the updated server can start serving users as another goes in maintenance mode.
- Backup data frequently. A possibility is to employ hot backups, if the DBMS offers such a feature.

3.7.3. Security

When dealing with sensitive data it's crucial to pay special attention to security, in order to avoid disclosing such data to the public. In fact, a data leak can lead to:

- Impersonation of the users: if attackers obtains user data, they can then use those data to access other services as that user (e.g. webmail).

- Money loss, from the point of view of the user: stolen credit card information can be used to transfer money and to buy things online.
- Loss of image. Oftentimes, when companies fail to protect user information and let attackers steal it, the loss of image causes users to not trust the company anymore.

To avoid this issue, we will ensure that the system complies to the PCI Data Security Standard [2].

3.7.4. Maintainability

This service will be subject to modifications and updates, either to improve functionalities, add new features or correct some errors. Given these assumptions, the system should be developed in order to minimize the cost of further modifications and be highly maintainable, by using proper design principles.

3.7.5. Portability

Every user should be able to access to all the functionalities offered by PE service by using either PEM or PEW + SMS.

3.7.6. Documentation

All PE applications (PEM, PEA, PEW and PEC) should have an accurate guide with all the instructions on how to use it properly, easily accessible from the main menu of the application.
Also, in order to facilitate the integration with external services, every exposed API should be well documented.

3.7.7. User interface and human factors

The interfaces that our users will use to access PE services will need to conform to accessibility and usability standards, in order to minimize cognitive stress.

Response times for example should be as low as possible (or at least: lower than the human reaction time) and, when we can't provide such immediate responses, we should provide continuous feedback in the form of a percent-done indicator.

The response time guidelines [3] have stayed unchanged for more than 40 years:

≤ 0.1 second	The user feels that they are directly manipulating the UI.
≤ 1 second	The user does not feel limited by the UI. For delays longer than 1 second it's best to clearly indicate that the system is working (e.g. by changing the shape of the cursor, or by adding a loading icon if on the mobile interface).

3.8. Alloy model

3.8.1. Signatures

```
/* Generic objects */
sig Time {}
sig Position {}

abstract sig Boolean {}
one sig True extends Boolean {}
one sig False extends Boolean {}


/* Battery status */
abstract sig Battery {}
one sig Low extends Battery {}
one sig High extends Battery {}


/* Possible states in which a car can be */
abstract sig CarState {}
one sig Available extends CarState {}
one sig Reserved extends CarState {}
one sig InUse extends CarState {}
one sig NotAvailable extends CarState {}

/* Event that may occur during a ride */
abstract sig Event {
      timestamp: one Time
}

/* Types of event that can occur during a ride */
one sig Unlock extends Event {}
one sig StartDriving extends Event {}
one sig StopDriving extends Event {}

/* Piece of information retrieved from a car during the ride */
sig Sample {
      location: one Position,
      timestamp: one Time
}

/* Safe area definition */
sig SafeArea{
      location: one Position
```

```
}

/* Types of payment */
sig Bill {}

abstract sig PaymentInformation {}
sig CreditCard extends PaymentInformation {}
sig BankAccount extends PaymentInformation {}


/* User definitions */
abstract sig User {}

sig UnregisteredUser extends User{}
sig RegisteredUser extends User {
      reservations: set Reservation,
      rides: set Ride,
      bills: set Bill,
      paymentType : one PaymentInformation
} {
 /* If the user has never reserved any car, no bills should be charged
*/
      #reservations = 0 implies #bills = 0
}



/* Car definition */
sig Car {
      currentPosition: one Position,
      state: one CarState,
      locked: one Boolean,
      batteryLevel : one Battery
} {
   /* If the battery level is Low, then the car should not be available
*/
      batteryLevel = Low implies state = NotAvailable

      /* If car is available then it must be locked */
      state = Available implies locked = True

      /* If car is not available then it must be locked */
      state = NotAvailable implies locked = True
}

/* Reservations */
```

```
sig Reservation {
	car: one Car,
	timestamp: one Time
}


/* A ride contains all the informations about where a driver goes */
sig Ride {
	car: one Car,
	from: lone Sample,
	to: lone Sample,
	parks: lone SafeArea,
	event: set Event,
	reservation: one Reservation,
	intermediatePosition: set Sample,
	initTimestamp: lone Time,
	finalTimestamp: lone Time
} {
	/* From position is set only if there is a start driving event */
	(#from > 0) <=> StartDriving in event

	/* To position is set only if there is a stop driving event */
	(#to > 0) <=> StopDriving in event

	/* Ride has samples only if a start driving event is present */
	(#intermediatePosition > 0) implies StartDriving in event

	/* User can't park before Stop Driving */
	#to > 0 <=> #parks > 0
}
```

3.8.2. Facts

```
/* Stop driving event is present only if there is at least one start
driving event */
fact StopDrivingOnlyIfStartDriving {
	all ride : Ride | StopDriving in ride.event implies StartDriving in
ride.event
}

/* Start driving event is present only if there is at least one unlock
event */
fact StartDrivingOnlyIfUnlock {
```

```
	all ride : Ride | StartDriving in ride.event implies Unlock in
ride.event
}

/* Only one car can be found in a given position */
fact OneCarInEachLocation {
	all disj c1, c2 : Car | c1.currentPosition != c2.currentPosition
}

/* User must reserve before ride a car */
fact ReservationBeforeRide {
	 all ride: Ride, ru : RegisteredUser | ride in ru.rides implies
ride.reservation in ru.reservations
}

/* Every Safe Area is located in a different position */
fact DifferentPositionForEachSafeArea {
	all disj sa1, sa2 : SafeArea | sa1.location != sa2.location
}

/* You can ride a car only once per reservation */
fact OneRidePerReservation {
	all disj r1, r2 : Ride | r1.reservation != r2.reservation
}

/* Every sample must be associated to a ride */
fact SampleMustBelongToRide {
	all s : Sample | some r : Ride | s in (r.from + r.to +
r.intermediatePosition)
}

/* Every bill must be associated only to one user */
fact OnlyOneUserPerBill {
	all r1, r2 : RegisteredUser |  r1 != r2 implies not (some b : Bill
| b in r1.bills and b in r2.bills)
	all b : Bill | (some regUser : RegisteredUser | b in regUser.bills)
}

/* Car can be unlocked only if there is at least one unlock event */
fact CarIsUnlockedIfThereIsAtLeastOneUnlockEvent {
	all c : Car | c.locked = False implies (some ride : Ride |
(ride.car = c and (Unlock in ride.event)))
}

/* User can stop driving only once */
fact AtMostOneStopDrivingPerRide {
```

```
	all ride : Ride | (lone ev : Event | (ev in ride.event and ev =
StopDriving))
}

/* Every event must belong to one ride */
fact EventMustBelongToRide {
	all ev : Event | (some ride : Ride | ev in ride.event)
}

/* Every reservation can be made only by one user */
fact OnlyOneUserPerReservation {
	all res : Reservation | (one usr : RegisteredUser | res in
usr.reservations)
}

/* The payment information must belong only to one user */
fact OnlyOneUserPerPayment {
	all pi : PaymentInformation | one usr : RegisteredUser | pi in
usr.paymentType
}

/* If the user parks in a safe area, the to field must be on the same
position of the safe area */
fact ParksPositionEqualToPosition {
	all ride : Ride | #ride.parks > 0 implies ride.parks.location =
ride.to.location
}
```

3.8.3. Predicates

```
pred show {}

/* This predicate shows how a reservation is made */
pred reserveCar[user, user' : RegisteredUser, c, c' : Car] {
	c.state = Available
		implies	(
		(one r : Reservation |
			not r in user.reservations
			 and r in user'.reservations
			 and r.car = c'
			 and user.reservations = (user'.reservations - r)
		)
		 and c'.state = Reserved
	)
}
```

```
/* This predicate shows what is going on when a user tries to unlock the
car */
pred unlockCar[ride, ride' : Ride] {
      Unlock not in ride.event
      Unlock in ride'.event
      ride.car = ride'.car
      one ru : RegisteredUser | ride.reservation in ru.reservations and
ride'.reservation in ru.reservations and ride in ru.rides and ride' in
ru.rides
}

/* This predicate explains what should happen when user starts driving
*/
pred startDriving[ride, ride' : Ride] {
      StartDriving not in ride.event
      StartDriving in ride'.event

      ride.car = ride'.car
      one ru : RegisteredUser | ride.reservation in ru.reservations and
ride'.reservation in ru.reservations and ride in ru.rides and ride' in
ru.rides
      #ride.from = 0
      #ride'.from = 1
      #ride.to = 0
}

/* This predicate clarify what we intend with a stop driving action */
pred stopDriving[ride, ride' : Ride] {
      one ev : Event |
            (
            not ev in ride.event
            and ev in ride'.event
            and ev = StopDriving
            )
      ride.car = ride'.car
      one ru : RegisteredUser | ride.reservation in ru.reservations and
ride'.reservation in ru.reservations and ride in ru.rides and ride' in
ru.rides
      ride.from = ride'.from
      #ride.to = 0
      #ride'.to = 1
}

/* Generates a world */
run show for 3
```

```
Executing "Run show for 3"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3039 vars. 267 primary vars. 4893 clauses. 6ms.
   . found. Predicate is consistent. 7ms.
```

```
/* Create a world for reserve car predicate */
run reserveCar for 3 but exactly 2 RegisteredUser
```

```
Executing "Run reserveCar for 3 but 2 RegisteredUser"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   3280 vars. 279 primary vars. 5406 clauses. 12ms.
   . found. Predicate is consistent. 5ms.
```

```
/* Create a world for unlock car predicate */
run unlockCar for 6 but exactly 2 Ride, 1 RegisteredUser, 2 Reservation,
2 Sample, 1 Car
```

```
Executing "Run unlockCar for 4 but 1 RegisteredUser"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   5636 vars. 444 primary vars. 9748 clauses. 10ms.
   . found. Predicate is consistent. 10ms.
```

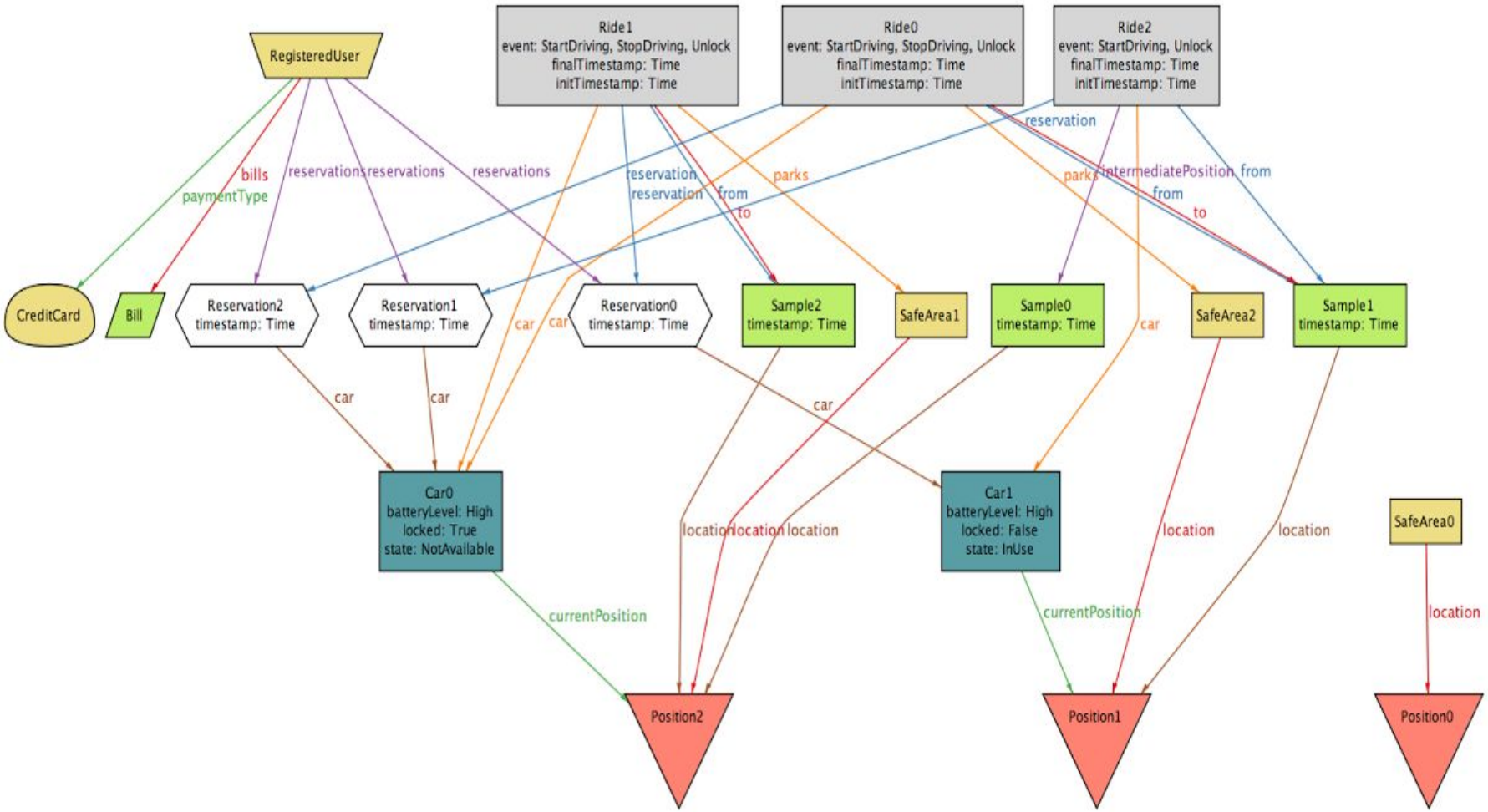
```
 /* Create a world for start driving predicate */
run startDriving for 6 but exactly 2 Ride, 1 RegisteredUser, 2
Reservation, 2 Sample, 1 Car
```

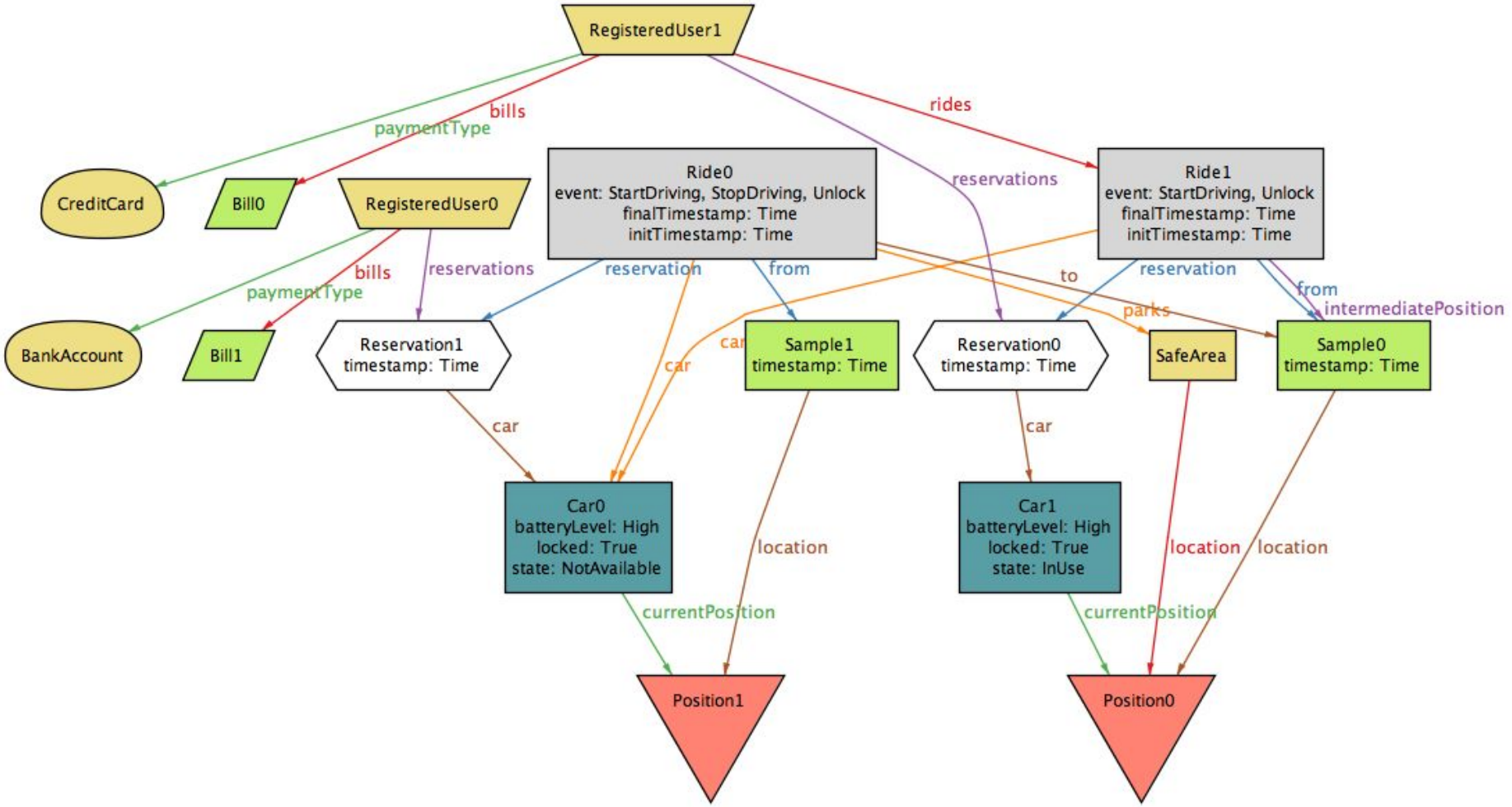
```
Executing "Run startDriving for 6 but exactly 2 Ride, 1 RegisteredUser,
2 Reservation, 2 Sample, 1 Car"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4860 vars. 365 primary vars. 8397 clauses. 51ms.
   . found. Predicate is consistent. 12ms.
```

```
 /* Create a world for stop driving predicate */
run stopDriving for 6 but exactly 2 Ride, 1 RegisteredUser, 2
Reservation, 2 Sample, 1 Car
```

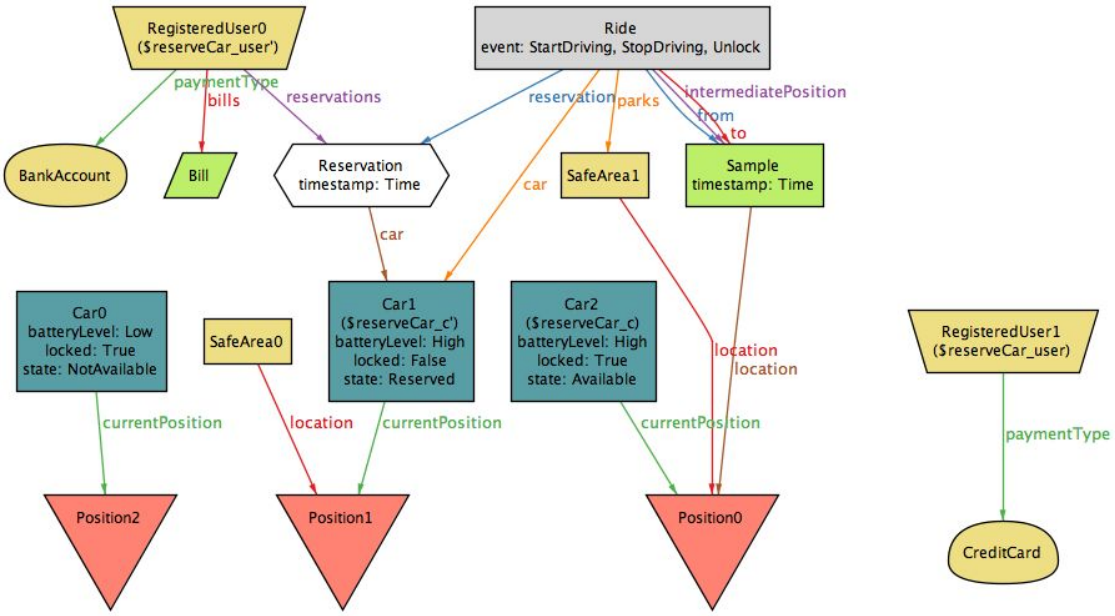
```
Executing "Run stopDriving for 6 but exactly 2 Ride, 1 RegisteredUser,
2 Reservation, 2 Sample, 1 Car"
   Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
   4872 vars. 365 primary vars. 8419 clauses. 51ms.
   . found. Predicate is consistent. 7ms.
```

In the next two pages, some examples of the show predicate are shown:

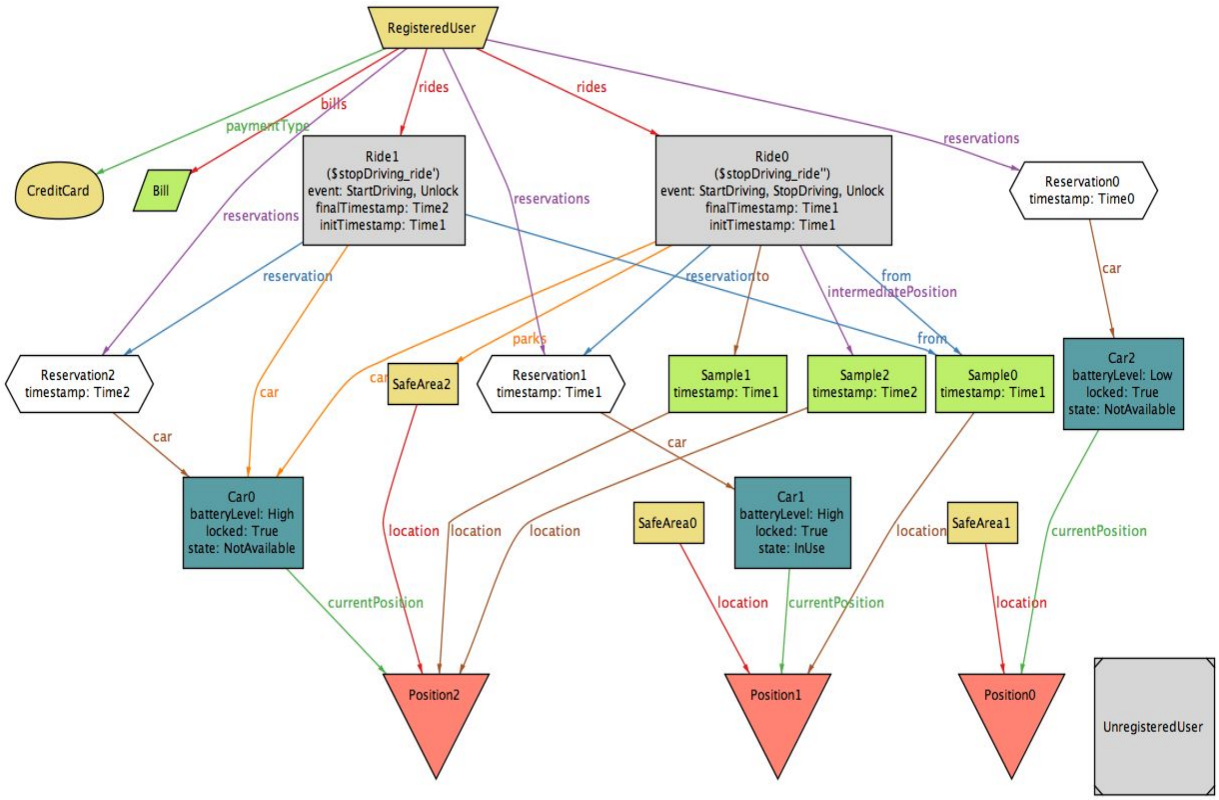

RegisteredUser
Ride1
event: StartDriving, StopDriving, Unlock
finalTimestamp: Time
initTimestamp: Time
Ride0
event: StartDriving, StopDriving, Unlock
finalTimestamp: Time
initTimestamp: Time
Ride2
event: StartDriving, Unlock
finalTimestamp: Time
initTimestamp: Time
CreditCard
Bill
Reservation2
timestamp: Time
Reservation1
timestamp: Time
Reservation0
timestamp: Time
Sample2
timestamp: Time
SafeArea1
Sample0
timestamp: Time
SafeArea2
Sample1
timestamp: Time
Car0
batteryLevel: High
locked: True
state: NotAvailable
Car1
batteryLevel: High
locked: False
state: InUse
SafeArea0
Position2
Position1
Position0
paymentType
bills
reservations
reservations
reservations
reservation
reservation
from
to
parks
parks
reservation
intermediatePosition
from
from
to
car
car
car
car
car
location
location
location
location
location
location
currentPosition
currentPosition
location


RegisteredUser1
paymentType
bills
rides
reservations
CreditCard
Bill0
RegisteredUser0
Ride0
event: StartDriving, StopDriving, Unlock
finalTimestamp: Time
initTimestamp: Time
Ride1
event: StartDriving, Unlock
finalTimestamp: Time
initTimestamp: Time
paymentType
bills
reservations
reservation
from
to
reservation
from
intermediatePosition
parks
BankAccount
Bill1
Reservation1
timestamp: Time
car
car
Sample1
timestamp: Time
Reservation0
timestamp: Time
SafeArea
Sample0
timestamp: Time
car
car
Car0
batteryLevel: High
locked: True
state: NotAvailable
Car1
batteryLevel: High
locked: True
state: InUse
location
location
location
currentPosition
currentPosition
Position1
Position0

This is an example of world generated by *unlockCar* predicate:



This example shows the result of *stopDriving* predicate:



Appendix

Used tools

- Google Docs (http://www.docs.google.com) for this document
- Star UML (http://staruml.io/) for the UML modelling, in particular use case, statechart, activity, class and sequence diagrams
- Alloy Model 4.2 (http://alloy.mit.edu/alloy) to generate the world and proving its consistency
- Balsamiq Mockup (http://balsamiq.com/products/mockups/) for the sketch on the user interfaces of this software system

Hours of work

The following are the total number of hours spent by each member of the group:

- Andrea Battistello: 35 h
- William di Luigi: 35 h

Revision history

Version	*Date*	*Revision description*	*Revision notes*
1.0	13-09-2016	Initial draft	-