PowerEnJoy
Requirements Analysis and Specification Document

Stefano Brandoli (mat. 875633)
Silvia Calcaterra (mat. 874887)
Samuele Conti (mat. 875708)

December 19, 2016

Version 1.1

Abstract

The Requirements Analysis and Specification Document (RASD) follows the redaction of the Feasibility Study document. This document aims to provide a model for the S2B which is aligned with the needs of the customers, to highlight the goals and limitations of the S2B, to describe some typical scenarios that will then lead to the specification of abstract use cases, which in turn will help to provide a complete description of the S2B in terms of functional and non-functional requirements.

This document leads stakeholders from some initial goals or objectives of the S2B to structured and more formal requirements.

The intended audience of this document are all the stakeholders which directly or indirectly have an interest on the S2B: the customers which are paying for the commission of this software, the final users that will use this software, the developers and designers that have to implement the S2B and the requirements analists that have to elicit the requirements. This document can also be used as a contractual basis between the customers and the developers.

Changelog v1.1
- Clarified the interaction with the cars: it was unclear that an Android operating system was present on board of every car
- Clarified Hardware Interfaces in section 2 and section 3
- Included the car's app in Hardware Limitations and Memory sections
- Changed provider for BillingService in Software Interfaces: Stripe provides better APIs documentation and a more straightforward implementation
- Clarified Non Functional Requirements and added new ones
- Renamed Actors section to Physical Actors
- Enlarged some diagrams for better visibility
- Refrased some sentences to be clearer
- Changed assumptions: precised T[3], D[3], D[4], D[15]; removed old D[23]
- Added Alloy syntax highlighting for better readability
- Fixed some typos, added a missing tool

Contents

1 Introduction

1.1 Purpose

The aim of the project is to develop *PowerEnJoy*, a digital management system for a car-sharing service that exclusively employs electric cars.

As every regular car-sharing service, the system shall guarantee some basic functionalities: a user can locate and reserve an electric car, after having registered and logged into the system. The S2B will discourage a user to make a reservation that won't be respected, since in this case the reserved car will not be used and this may cause a business damage to our customers on long terms. A user can communicate to the system with a mobile web application that he is near a reserved car to allow its unlock. During his ride on the reserved car, the user will pay a given amount of money per minute, while informed of the current charges on the screen on board of that car; the user is stop being charged when the car is parked in one of the predefined safe areas and the car will automatically be locked (if possible) after the exit of the user from the reserved car.

In addition to standard car-sharing services, the system aims to incentivize users which use properly the cars and to optimize the distribution of the cars in the city. This incentives are done by providing discounts on the last ride made by the user. A user will obtain a discount when he shares the car he reserved with other passengers, when he leaves specified battery capacity in the car at the end of a reservation and when he plugs the car into a power plug inside a power safe area in order to recharge it. These discounts aim to reduce the maintenance costs and time for our customers, to guarantee that future users will be able to use a car with enough battery capacity and to encourage users to use the car sharing service again. On the contrary, unworthy behaviours from the users that may prejudice the future users' experience will cause an extra charge to pay in addition to the original bill of the last reservation.

Finally, in order to guarantee a uniform distribution of cars in the city and let the user save money, the users can access a money saving option by accepting as destination a power safe area dictated by the system.

1.2 Goals

The overall, high level objectives which provide the motivations for our system in the world are the followings:

- **G1: Allow users to find available cars and reserve them**.
  This is one of the main goals to be fulfilled by the S2B. All the operations to achieve this goal will be done by final users through a mobile web application.

- **G2: Allow users to use the reserved cars inside the city area**.
  This goal aims to guarantee to users the usage of reserved cars and the access to the reserved cars through a mobile web application.

- **G3: Guarantee a uniform distribution of the cars in the city area**.
  This goal aims to exploit the usage of the cars by the users as much as possible and may also indirectly incentivize new users to register to the PowerEnjoy service.

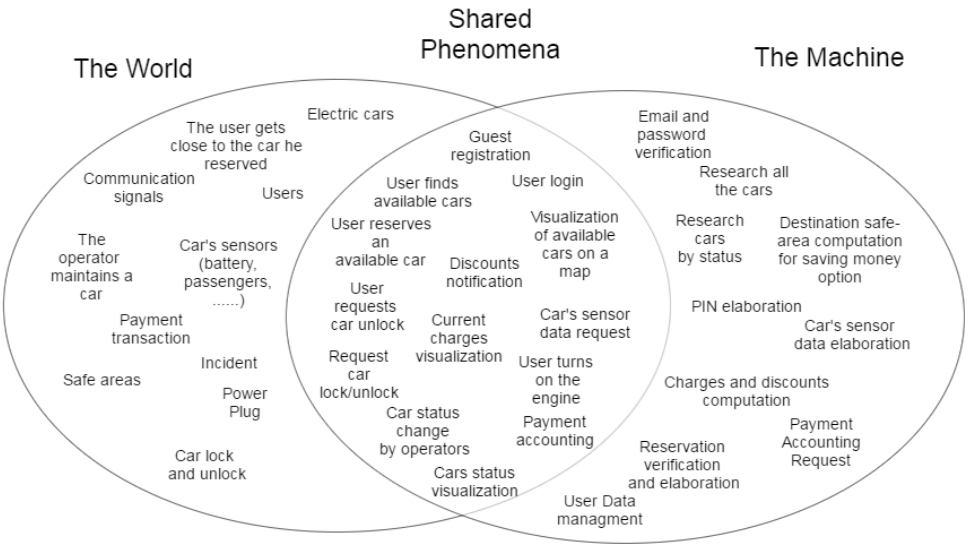
- **G4: Incentivize users to use properly the reserved cars.**
  This goal aims to incentivize users to use properly the fleet of cars provided by the company in order to reduce the maintenance time and maintenance costs of the cars.

All the goals contribute together to guarantee the main goal for PowerEnjoy: **the cars provided by the company should be used as much as possible by the users**.

1.3 Scope

In order to make a first analysis of the application domain and to define the boundaries of our S2B, we have followed the concepts and ideas provided in the paper by M. Jackson & P. Zave on "The World and the Machine".

The application domain can be divided in three main parts: the World, the Shared Phenomena and the Machine. In the World there are the goals of our S2B, the domain assumptions and all the objects and people of the reality. The Machine is our S2B. In the Machine there are all the tasks and programs of our S2B. These programs can be built from scratch or can rely partially on interactions with external services. The Shared Phenomena define the boundaries of our S2B, the interactions between our S2B and the World. In the Shared Phenomena there are all the requirements of our S2B that we need to capture. Every requirement is the result of an interaction that starts from the World and has an effect on the Machine or viceversa. The following is the result of this analysis on our S2B:



1.4 Actual System

Our customers don't have by now a software system for the management of the car-sharing service which they would like to provide, so we assume that the S2B will not incorporate any previous legacy software and will be developed from scratch. However our system will interact with external services to support part of its functionalities. More details about external interfaces will be explained in the Section 2 and Section 3 of this document.

1.5 Physical Actors

The main physical actors involved in the interactions with our S2B are the following:

- **Guest:** all the guest users can only see the introductory login page and decide to register to the system by following the registration procedure.
- **User:** these users, after a successful login, can access to all the main functionalities of the S2B, among which there are: research of available cars inside the city area, reservation of an available car, unlocking of a reserved car, usage of the reserved car.
- **Operator:** every operator can login into the system, find on a map of the city area the unavailable cars, choose a car to maintain and set that car as available again after having managed to fix it.

1.6 Definitions, Acronyms, Abbreviations

1.6.1 Definitions

- **Actor:** class of people or systems that interact with the S2B.
- **Credentials:** email and password.
- **Guest:** physical person who is not registered to the PowerEnjoy system, his credentials are not known by the S2B. A Guest can become a user after a successful registration procedure.
- **User:** physical person who is registered to the PowerEnjoy system, his credentials are known by the S2B.
- **Operator:** worker of the PowerEnjoy company encharged to do the maintenance of the cars when needed.
- **Car:** an electric car of the PowerEnjoy company.
- **Safe Area:** private area owned by the company. Each safe area is made of a number of parking slots.
- **Power safe area:** it's a special kind of safe area where each parking slot has a dedicated power plug that can be used to recharge a car.
- **City area:** the city area is the set of all the GPS coordinates within the boundaries of the city border.
- **Environment:** the world in which reside the goals and domain assumptions of this software to be.
- **PIN:** unique password received by the user from the S2B at the end of the registration procedure. This password will be then used by the user to unlock a car he has reserved. This password should not be confused with the password used by the user to perform the login.
- **Car failure:** mechanical problem occurring in the car or problem preventing a normal usage of the car to a future user. Examples of car failures are: battery damaged, battery empty, engine broken, flat tires, prevented car locking because of open doors and failures caused by incidents involving the specified car.
- **Battery (Battery Array):** is the energy tank of the car.
- **Service battery:** battery inside the car which provides the energy needed for the functioning of all the electrical equipments in that car. This battery can work even when the battery array of the car is empty. The service battery provides energy to guarantee the lock/unlock of the car, the ignition of the engine, the work of the ECU and so on.

- **Exit properly from a car:** the user goes out from a reserved car and by doing this action he has taken care that all the doors, trunk and windows in that car are closed properly.
- **Pick up a car:** the user unlocks his reserved car.
- **Park a car:** the car of the user is not moving and it is inside a parking slot.
- **Car stop:** a car stop occurs when a user wants to park temporarily his reserved car without terminating the reservation of that car.
- **Car ride:** time during which a car is reserved by a user.
- **Car screen:** display with touchscreen technology mounted on board of a car.
- **Car status:** the actual condition of a car of the company. The status of a car can be: *Available*, *Reserved*, *Unavailable*, *Maintenance*.
- **Reservation terminated correctly:** the reservation is terminated correctly by the system when the user has parked the reserved car in a safe area and exited properly from the car, thus not preventing its lock.
- **Penalty:** charge a user with more money on his last reservation bill.

1.6.2 Acronyms

- RASD = Requirements Analysis and Specification Document.
- DD = Design Document.
- S2B = Software to be.
- API = Application Programming Interface.
- ECU = Engine Control Unit.
- SDK = Software Development Kit.
- GPS = Global Positioning System.
- 3G = 3rd generation wireless mobile telecommunications technology.
- CAN = Controller Area Network.
- POS = Point Of Sale.

1.6.3 Abbreviations

- [Gi] = i-th goal.
- [Gi.y] = y-th subgoal of the i-th goal.
- [Ri.y] = y-th requirement of the i-th goal.
- [Ri.y.z] = z-th requirement of the y-th subgoal of the i-th goal.
- [RRi] = i-th requirement rationale.
- [Di] = i-th domain assumption.
- [Ti] = i-th text assumption.
- [UCi] = i-th use case.
- [NRi] = i-th non functional requirement.

1.7 Reference Documents

- Specification Document of PowerEnjoy.
- IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications: this document has been used for defining the general structure of the whole RASD (called SRS in the IEEE document).
- IEEE ISO/IEC/29148 - IEEE Standard On Requirement Engineering: this document has been used for some general concepts about requirement engineering.
- Jackson and Zave - "Deriving Specifications From Requirements": this document has been used to understand better where to put the boundaries of the S2B.
- Papers on the GreenMove project: to understand better the interface with the cars.
- "Maintenance Manual For Electric Cars" involved in a famous car sharing service: http://www.sharengo.eu/cs/wp-content/uploads/2014/05/Manuale-duso-e-manutenzione-Icar0.pdf : this document has been used to try to understand better the functioning of the electric cars for PowerEnjoy.

1.8 Overview

This document has been structured by following the IEEE Std 830-1998 with some minor modifications mainly devoted to highlight goals, scenarios and use cases. The whole document is divided in four sections.

- **Section 1: Introduction**, this section presents the main goals and relevant physical actors of the S2B. This section is audienced to all the stakeholders which have a legitimate interest in our system.
- **Section 2: Overall Description**, this section provides more informations about the boundaries of the S2B and the interactions between the S2B and external interfaces. It will also highlight the text and domain assumptions for the S2B. This part is audienced to all the stakeholders, in particular to our customers and users.
- **Section 3: Specific Requirements**, this section provides more informations about all the aspects in Section 2 and is mainly audienced to the developers of the S2B. It provides some mockups for the interfaces which are within the boundaries of the S2B, some informations about the external services used and also some scenarios of the usage of the software system with related use cases and UML diagrams. In this section all the goals of the S2B will be investigated deeply and for every goal a list of functional requirements will be presented. The section will conclude with a list of non-functional requirements for the S2B.
- **Section 4: Appendix**, in this section Alloy will be used to model some critical parts of the system.

2 Overall Description

2.1 Product Perspective

The proposed solution for the system of PowerEnjoy won't have to rely or to be interfaced with a previous car-sharing system of the same company. The system core functionalities will be developed from scratch, however part of the offered functionalities will rely on external services for these main reasons:

1. Some functionalities cannot be developed properly without interfacing with specific databases or informative systems.
2. Reduce development costs by relying on well known APIs already present in the market.
3. Guarantee the portability of the system on multiple platforms.

The system will be developed having in mind: a web app for mobile devices (Android, iOS) for users and a native Android app for the touchscreen Android device on the cars (linked with the car's hardware).

2.1.1 User Interfaces

The interface of the web app will be developed to support screen devices ranging from 3 inches diagonal and up, to support both devices with small screen sizes

and phablets. The interface should also support a correct visualization even when the smartphone screen is rotated (orientation change).

After the user is logged in, the user won't be redirected immediately on a map activity to find and reserve available cars; he will be redirected on a menu activity. This decision will help in the future if our customer will ask us to add new functionalities like: support the user to change his personal informations, support the user to pay PowerEnjoy when there are some debts with the company (since one or more payments on the credit card of the user have failed).

The interface of the native Android app of the car will be developed to be simple and immediate.

2.1.2 Hardware Interfaces

The S2B won't interact directly with any hardware interface, thanks to a device pre-installed by the cars' provider on every car (see Domain Assumptions).

2.1.3 Software Interfaces

The system will use the following external services to provide part of its functionalities:

- **Car Interfacing**
  - **Name:** VehicleInterface
  - **Mnemonic:** Car
  - **Purpose:** the VehicleInterface will be used to get informations from a car (like its battery level and the number of passengers) and to lock/unlock that car when needed.
- **Maps Service Interfacing**
  - **Name:** Google Maps
  - **Mnemonic:** MapsService
  - **Purpose:** Google Maps will be used by the car's app, and in the user's web app to let a user find available cars on a map of the city area.
- **Billing Service Interfacing**
  - **Name:** Virtual POS
  - **Mnemonic:** BillingService
  - **Purpose:** the Virtual POS will be used to handle users' payments.

2.2 Product Functions

The main functionalities that will be provided by the S2B, considering also the ones provided in part through external interfacings, are the following:

- Guest registration.
- User login.
- Visualization of available cars on a map of the city.
- Find available cars inside the city area.
- Reserve available cars inside the city area.
- Unlock a reserved car through a smartphone mobile web application.
- Computation of the best power safe area as a destination to save money for the user.
- Cost and discount computation for every car reservation.
- Car reservation payment handling.

2.3 User Characteristics

The usage of the S2B will not require users with a particular educational level, experience or technical expertise. Every user who knows how to use applications on a smartphone device, how to enable an internet connection, how to enable GPS and who is familiar in providing payment informations on e-commerce websites, has enough expertise to interact with the S2B easily. In order to guarantee more usability of the software, a design based on the design guidelines for Android and iOS applications will be adopted where possible.

2.4 Constraints

2.4.1 Regulatory Policies

The S2B will need permissions to acquire the GPS position of a user.

2.4.2 Hardware Limitations

The hardware limitations for both the car app and the mobile web app are:

- 3G technology.
- GPS technology.
- Memory to download the application package.
- Memory to install the application package.

2.4.3 Parallel Operations

The S2B shall support parallel requests by multiple users for all the functionalities that can be provided to final users like: login, research of available cars, car reservation and so on.

2.5 Assumptions and Dependencies

We assume that these properties hold in the analyzed world:

2.5.1 Text Assumptions

- **[T1]** There is not a previous system, the system will be developed from scratch.
- **[T2]** The main device used by users and operators to access the functionalities of the S2B is a mobile device, however part of the interaction between the user and the S2B happens also through the screen on board of every car.
- **[T3]** The PowerEnjoy company has already bought all the cars that will be used by our future users. We have to develop an Android app running in the car which interacts with a software interface abstracting the car electronics, and with the user through a screen on board.
- **[T4]** The user cannot cancel a reservation of a car. We assume that our customers wants to encourage the usage of the electric cars in the short time. In fact a user can only reserve an available car at most 1 hour before the pick-up time, so we think that there is an high probability that a user performing a reservation of a car will actually be very interested in exploiting that reservation. Furthermore our customers are asking us to apply a fee (1 EUR) to the user if a reserved car is not picked-up, which we think is a small fee considering the money lost by the missed usage of that car.
- **[T5]** We assume that the battery cited in the text is considered as the electric car tank. In fact in the real world every electric car has two main batteries: the battery array which serves as a tank and a service battery which provides the energy necessary for the functioning of all the electrical equipments in the car. See the "Maintenance Manual For Electric Cars" in the Reference Documents for more informations.
- **[T6]** We assume from the text that every car of the company has at least 3 seats.

2.5.2 Domain Assumptions

- **[D1]** Every car has a GPS module that is always on.

- **[D2]** Every car has a 3G module that is always on.
- **[D3]** The operating system running on board of a car is always on even when the screen on board is turned off to save service battery energy.
- **[D4]** Inside every car there is a pre-installed device made of: a low level embedded board, which is connected to the CAN-bus of the car; an Android board on top of which is running Android OS and connected to the embedded board. The cars' provider is also providing us a software interface that can be used by our S2B to get informations from the car and to control its actuators.
- **[D5]** All the sensors in the cars work correctly.
- **[D6]** Every car is always localizable using GPS.
- **[D7]** The GPS data are always correct.
- **[D8]** A car occurring in a car failure will always be able to communicate to the system.
- **[D9]** A user occurring in a car failure during the usage of his reserved car will stop using that car as soon as possible.
- **[D10]** The devices used by guests, users and operators, when interacting with the S2B, are always connected with GPS technology and Internet connection (3G connectivity technology or WiFi).
- **[D11]** The set of safe areas is initialized in the S2B at startup-time.
- **[D12]** Every safe-area has a predefined number of parking slots.
- **[D13]** The safe areas are far enough from each other such that each safe-area can be identified by a unique GPS coordinate and a radius of variable dimension.
- **[D14]** Every parking slot in a power safe area has a dedicated and working power plug.
- **[D15]** When a car is plugged to a power plug inside a power safe area, the battery array recharges and the service battery recharges as well, so we can consider that the service battery is never empty. See the "Maintenance Manual For Electric Cars" in the Reference Documents.
- **[D16]** The number of available cars of the company is less than or equal to the sum of all the parking slots in all the safe areas, considering both normal and power safe areas.
- **[D17]** The safe areas are distributed uniformly inside the city area: we make this assumption, otherwise the cars cannot be distributed uniformly in the city.

- **[D18]** A vehicle not owned by PowerEnjoy cannot be left in a parking slot inside a safe area.
- **[D19]** Every operator is already assigned to a predefined email and password by the system, they don't need to register to the system.
- **[D20]** Every operator checks periodically if there are unavailable cars. We assume this since they're paid to do the maintenance.
- **[D21]** An operator that has set a car as under maintenance will actually manage to fix it and then set it back to available.
- **[D22]** Every car is identified by a unique license plate.
- **[D23]** A user won't drive a car of the company without a valid driving license.
- **[D24]** The unique PIN received back by a user at the end of his registration procedure will always be known by that user.
- **[D25]** A user cannot let another user use the car he has reserved by providing him his credentials and personal PIN.

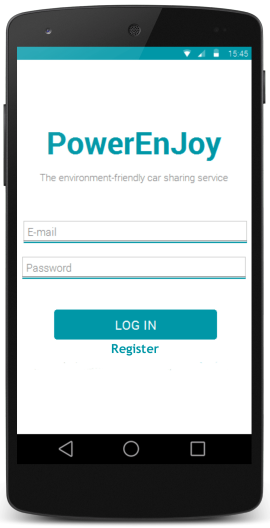
3 Specific Requirements

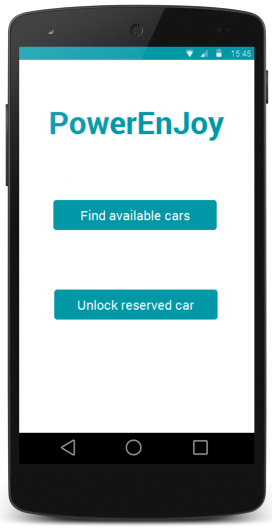
3.1 External Interfaces Requirements

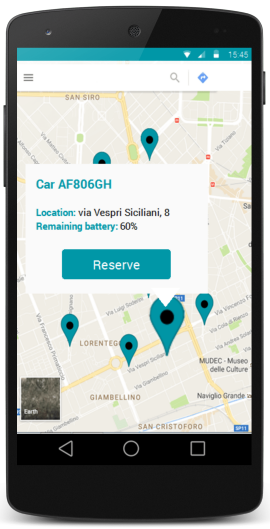
3.1.1 User Interfaces

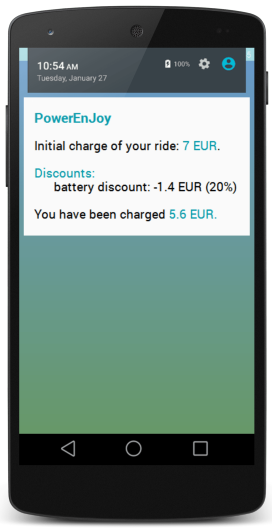
The following mockups shall be intended only as a way to understand better some requirements, not as design decisions.

- **Mobile Application: User Interfaces**

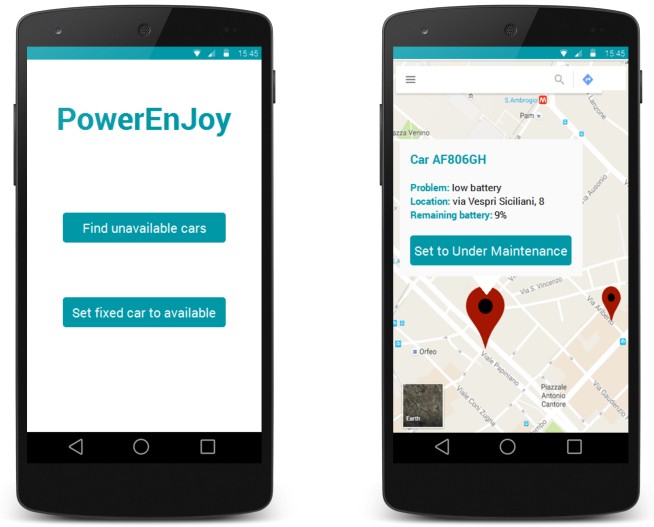




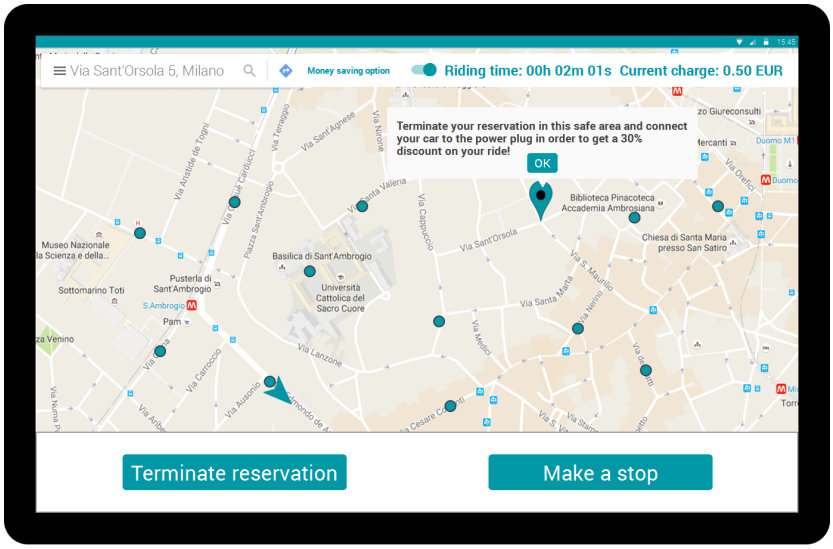




- **Mobile Application: Operator Interfaces**



- **Car Application: Screen Interface**



3.1.2 Hardware Interfaces

The company providing the cars to PowerEnjoy has put inside every car a device, alimented by the service battery and made internally of 2 boards (see Domain

Assumptions), on top of which runs Android OS. Through a software interface (VehicleInterface) provided by the cars' provider to us, we can abstract the interaction with the car without having to deal directly with hardware interfaces.

3.1.3 Software Interfaces

The system will use the following external services to provide part of its functionalities:

- **Car Interfacing**
  - **Name**: VehicleInterface
  - **Mnemonic:** Car
  - **Source:** Papers on the Green Move project.
  - **Purpose:** this external service will be used to retrieve useful data from the car's sensors, such as possible mechanical failures, the battery's state of charge, if the battery is currently recharging, the opening and the closing of the doors and the trunk, the locking and the unlocking of the car, the speed of the car, the ignition of the engine and the number of passengers in the car. This interface will also be used to carry out some important actions directly on the car, such as the lock/unlock of the doors. Further details about this software interface, however, are beyond the purposes of this document.
- **Maps Service Interfacing**
  - **Name:** Google Maps
  - **Mnemonic:** MapsService
  - **Source:** https://developers.google.com/maps/
  - **Purpose:** this external service will be used to get a detailed map view of the city area, to highlight the cars' and safe areas' GPS positions on the map view, to use utilities to let the user type only addresses inside the city area and to show a path from the car actual GPS position to the GPS position of the destination safe area.
- **Billing Service Interfacing**
  - **Name:** Virtual POS
  - **Mnemonic:** BillingService
  - **Source:** https://stripe.com/docs/api
  - **Purpose:** this external service is a Virtual POS that will be used to manage easily and safely the variety of payment informations that can be provided by the users. This service will be used to account and receive a response about the status of a payment accounted to a user. It will also be used to verify if the payment informations provided by a user are valid.

3.1.4 Memory

For the mobile web app and the car app, it is required at least the necessary space on secondary memory for installing the app's package.

3.2 Functional Requirements

3.2.1 [G1] Allow users to find available cars and reserve them.

- **[G1.1] Allow users to access the system.**
  - **[R1.1.1]** The system shall allow a user to login if and only if his provided credentials are correct.
  - **[R1.1.2]** The system shall guarantee that every user is associated to a unique PIN.
  - **[R1.1.3]** The system shall send a unique PIN to a user at the end of his registration procedure.
  - **[R1.1.4]** The system shall not allow a guest to complete the registration if the email provided has already been used by another user.
  - **[R1.1.5]** The system shall not allow a guest to complete the registration if he is not at least 18 years old.
  - **[R1.1.6]** The system shall not allow a guest to complete the registration if he has provided an invalid payment method.
- **[G1.2] Allow users to find available cars.**
  - **[R1.2.1]** The system shall allow a user to find available cars starting from his GPS location.
  - **[R1.2.2]** The system shall allow a user to find available cars starting from a specified address.
  - **[R1.2.3]** The system shall be able to locate every car of the company.
- **[G1.3] Allow users to reserve available cars.**
  - **[R1.3.1]** The system shall allow a user to reserve a car if and only if that car is available.
  - **[R1.3.2]** The system shall not allow a user to make a car reservation if he has a car reservation not terminated yet.
  - **[R1.3.3]** The system shall guarantee that a car cannot be reserved at the same time by more than one user.

3.2.2 [G2] Allow users to use the reserved cars inside the city area.

- **[R2.1]** The system shall be able to check and set the status of every car of the company.

- **[R2.2]** The system shall be able to query the sensors of a car to get informations.
- **[G2.1] Manage the user's reservation.**
  - **[R2.1.1]** The system shall guarantee that a reserved car can be unlocked only by the user who made its last reservation, after he has inserted correctly his PIN.
  - **[R2.1.2]** The system shall guarantee that a reserved car can be unlocked only if the unlock is requested within 10 meters from the current GPS position of that car. See [RR1]
  - **[R2.1.3]** The system shall start charging money to a user for his car reservation when the engine of the reserved car ignites for the first time during that reservation. See [RR2]
  - **[R2.1.4]** The system shall charge money to a user for his car reservation with a predefined amount of money per minute.
  - **[R2.1.5]** The system shall display to the user the current charges of his car reservation on the screen inside the car he reserved.
  - **[R2.1.6]** The system shall detect when a car leaves or enters a safe area. See [RR3]
  - **[R2.1.7]** The system shall allow a user to communicate his intention to make a stop or to terminate his reservation through the screen on the reserved car.
  - **[R2.1.8]** The system shall allow a user to terminate his reservation only in a safe area.
  - **[R2.1.9]** The system shall evaluate if a reservation can terminate in the safe area chosen by the user, considering the number of available parking slots in that safe area.
  - **[R2.1.10]** The system shall terminate the user reservation when the user exits properly from the reserved car, after having parked that car in an approved safe area. See [RR4]
  - **[R2.1.11]** The system shall stop charging money to a user when his reservation is terminated.
  - **[R2.1.12]** The system shall guarantee that if a user doesn't unlock for the first time the car he reserved within 1 hour from his reservation, that reservation terminates.
  - **[R2.1.13]** The system shall guarantee that if a user doesn't unlock for the first time the car he reserved within 1 hour from his reservation, that car is set as available.
  - **[R2.1.14]** The system shall lock the car once the user exits properly from that car and his reservation is terminated in a safe area.
  - **[R2.1.15]** The system shall lock the car once the user exits properly from that car after parking for a stop.

- **[R2.1.16]** The system shall set the car as available once the reservation on its last ride is terminated and the car is locked.
- **[R2.1.17]** The system shall notify the user about the bill of his last ride, considering also the discounts and penalties received.

- **[G2.2] Handle unexpected car situations and user behaviour.**
  - **[R2.2.1]** The system shall be able to detect a failure in a car.
  - **[R2.2.2]** The system shall set a reserved or available car as unavailable if a failure in that car occurs.
  - **[R2.2.3]** The system shall terminate the last eventual reservation of a car if that car is set as unavailable. See [RR5]
  - **[R2.2.4]** The system shall not allow a user to make a car stop inside a safe area or outside the city area. See [RR6]
  - **[R2.2.5]** The system shall notify a user on the screen of the car he reserved when he is leaving the city area with that car. See [RR7]
  - **[R2.2.6]** The system shall guarantee that an operator can find the unavailable cars.
  - **[R2.2.7]** The system shall allow an operator to set as under maintenance at most one unavailable car at a time.
  - **[R2.2.8]** The system shall set a reserved car as unavailable if its engine is not turned on within 20 minutes from the last unlocking of that car. See [RR8]
  - **[R2.2.9]** The system shall set a reserved car as unavailable if the reservation terminates in a safe area with more than 80% of its battery empty. See [RR10]
  - **[R2.2.10]** The system should not allow anymore the access to a user to which a payment was not successfully accounted.

3.2.3 [G3] Guarantee a uniform distribution of the cars inside the city area.

- **[R3.1]** The system shall apply a discount of 10% on the user's last ride if he has picked up at least 2 other passengers during the last ride and those passengers have stayed on the car for at least 50% of the total time of that ride. See [RR9]
- **[R3.2]** The system shall be able to get informations about the presence and number of passengers in a car.
- **[R3.3]** The system shall implement a money saving option through which the user inserts his destination on the screen of the reserved car and the system shows on that screen the power safe area where to leave that car in order to get a discount.

3.2.4 [G4] Incentivize users to use properly the reserved cars.

- **[R4.1]** The system shall apply a discount or a penalty on the last ride to a user only if his reservation is terminated correctly (see Definitions).
- **[G4.1] Reward the users that ease the experience of other users.**
  - **[R4.1.1]** The system shall apply a discount of 20% on the last ride to a user leaving after the ride more than 50% of the battery in the reserved car.
  - **[R4.1.2]** The system shall apply a discount of 30% on the last ride to a user who leaves the reserved car in a power safe area and plugs it.
- [**G4.2] Penalize the users that ruin the experience of other users.**
  - **[R4.2.1]** The system shall charge 30% more on the last ride to a user who leaves the car more than 3 km from the nearest power safe area or with more than 80% of the battery empty.
  - **[R4.2.2]** The system shall guarantee that if a user doesn't unlock for the first time the car he reserved within 1 hour from his reservation, a fee of 1 EUR is accounted to that user.

3.3 Requirements Rationales

3.3.1 [RR1]

- We think that 10 meters is a good distance, otherwise a user can potentially unlock the car he reserved far away from the car position and someone else may enter in that car in the meanwhile.

3.3.2 [RR2]

- We specified "for the first time" since it can happen that the engine of the car turns off during the car ride and the user may have to turn it on manually again. We want to start charging the user since the first intentional ignition of the engine during his reservation.

3.3.3 [RR3]

- This requirement is necessary in order to keep updated the number of available parking slots in a safe area and to avoid the termination of a reservation in a safe area which it's already fully occupied by other cars.

3.3.4 [RR4]

- The incorrect exit of the user from the reserved car is included in the management of a car failure (see [G2] Allow users to use the reserved cars inside the city area).

3.3.5 [RR5]

- "Eventual" reservation because a car can have failures also when it is still available inside a safe area or when it has been reserved but not picked up yet from the safe area.

3.3.6 [RR6]

- We think that it's reasonable not to allow a car stop inside a safe area, otherwise a user can occupy with his reserved car a free parking slot in a safe area and this action may prevent another user to use that parking slot to terminate his reservation.
- We don't want to allow a car stop outside the city area since we don't want to provide service outside the city area.

3.3.7 [RR7]

- We don't want to support the service outside the city area, so we think that the user should be notified when he leaves the city area.

3.3.8 [RR8]

- After the unlock, the user may decide to not enter in the car or to enter in the car without turning on (igniting) the engine. Without this requirement, in the above cases the user can potentially keep the car in the reserved status for days without paying anything. We think that 20 minutes is a reasonable time, in fact we don't want to damage users which may take more time to turn on the engine after the unlock.

3.3.9 [RR9]

- This requirement is listed under this goal since we think that, if multiple people who know each other want to go to together to the same place using the cars of the company, it is better if they share the same car as much as possible instead of going in the same place with multiple cars. Otherwise our company will find multiple cars arriving in very near safe areas and this situation may be harmful since it may go against the goal [G3] Guarantee a uniform distribution of the cars inside the city area.
- With the 50% time constraint, we want to avoid that the user fools down the system to get a discount by picking up two passengers and making them exit the car immediately.

3.3.10 [RR10]

- We think that this is reasonable since a user won't trust an available car with a battery level $<20\%$, and if the user doesn't trust the car, he simply

won't choose it, which goes against the goal [G2] Allow users to use the reserved cars inside the city area.

3.4 Scenarios

The following scenarios are a description of imagined sequences of events that include the interaction of the S2B with its environment and users, as well as the interaction among its components. Scenarios are useful to derive abstract use cases, which can then help to identify requirements that may have not been formally specified by any of the stakeholders.

3.4.1 Scenario 1

Alonso doesn't like the public transportation, he thinks that those vehicles are too slow to move inside the city area. A friend told him that a new car-sharing service called PowerEnjoy is available in the city, so Alonso decides to go on the app store and download the PowerEnjoy mobile app. Alonso opens the PowerEnjoy app. Alonso is in the main activity of the app. He presses the button "Register" and the registration procedure starts. Alonso fills correctly the registration form with his personal data, he provides a valid credit card as payment information, he accepts the "License agreement contract for PowerEnjoy" and presses the button "Complete" to end the registration procedure. After some seconds, the registration procedure completes correctly and Alonso receives an email with his unique PIN. The PIN can be used by Alonso to unlock the cars he will reserve with PowerEnjoy.

3.4.2 Scenario 2

Alicia lives inside the city area, near the city boundaries. She has just known from a close friend that her favourite dress shop "Dressando" in the center of the city is offering special discounts for that day only. Alicia decides that it's the time to try the PowerEnjoy car sharing service, to exploit the Dressando special discount. Alicia has completed the registration procedure to PowerEnjoy two days ago. Alicia opens the PowerEnjoy app on her smartphone. She is in the main activity. She inserts correctly her email and password and presses the button "Login". After the successful login Alicia is redirected to a menu activity. Alicia presses the button "Find available cars", she is redirected to a map activity showing the available PowerEnjoy cars in the city area. Alicia sees her position on the map and locates an available car near her position. She selects that car on the map. A dialog pops up showing the license plate of that car, its battery level, its location and a button "Reserve". She presses the button "Reserve". She goes out and when she is 1 meter near the reserved car, she presses the button "Unlock reserved car". She inserts her personal PIN and presses "Confirm". After some seconds, the reserved car is unlocked and Alicia enters in that car.

3.4.3 Scenario 3

Aldo, Giovanni and Giacomo are friends. They are all vegetarians. They have booked at the vegetarian restaurant "Gnam Veggie", which is inside the city area, for that evening. Aldo uses PowerEnjoy regularly. Aldo knows that if he picks up with him during a ride at least 2 other passengers for at least 50% of the time of the ride, he will receive a discount of 10% on the last ride. Aldo proposes to Giovanni and Giacomo to reserve a PowerEnjoy car to go to the restaurant, and they agree. Aldo manages to reserve a car near his house inside the city area. He went out, he goes very near (<10m) to that car and manages to unlock it, after having inserted correctly his PIN. Giovanni and Giacomo are both at Giovanni's house, which is inside the city area. Aldo goes to Giovanni's house, he parks the car, turns off the engine, presses on the screen of the car the option "Make a stop" and exits correctly the car. After more or less 30 seconds, the car is locked. Aldo goes to advice Giovanni and Giacomo and then they return to the reserved car. Aldo unlocks the car again and they enter. Aldo ignites the engine and starts driving to "Gnam Veggie". Aldo finds a safe area near the restaurant, he parks the car, turns off the engine, presses the button "Terminate reservation". The three friends exit correctly from the car. After some time the car gets locked. Aldo receives after a while a notification specifying the bill of the ride performed and the discount applied: 10% for having shared the car with 2 other passengers for at least 50% of the time of the ride.

3.4.4 Scenario 4

Will, that is already registered to PowerEnJoy, needs to cover a short distance by car. He logs to the PowerEnJoy application on his smartphone and selects the "Find available cars" button; he then reserves the available car that is closest to him. Within one hour from the reservation, Will goes on the chosen location and through the application on his smartphone informs the system that he is nearby by clicking on the "Unlock reserved car" button and inserting his personal PIN code. The car unlocks itself within 30 seconds and he enters. He immediately turns on the engine and the system starts charging him for a given amount of money. After covering a short distance, Will finds a safe area for parking and exits properly from the car, leaving it with 20% battery empty. After about 30 seconds, the system automatically locks the car. After a short time, the user receives a notification from the PowerEnJoy application with his charge for the last ride, where he is also informed that he received a 20% discount on his last ride, because he left his car with less than 50% of the battery empty.

3.4.5 Scenario 5

Mike has to go to a concert in the city and decides to use PowerEnJoy to reach his destination; he is registered to the system. About an hour before the time he wants to leave, he logs to PowerEnJoy and reserves a car that is close to where he lives. When the time comes, he reaches the location, clicks on the "Unlock reserved car" button, inserts his PIN code correctly and enters the car.

He doesn't use the money saving option, as he doesn't like to travel by feet, so he always parks the car as close to his destination as possible. Mike doesn't like to travel with other passengers, so he doesn't pick up anyone onto the car during his ride. The concert holds far away from where Mike lives, so when he reaches his destination the car has more than 80% of the battery empty. There are no special parking areas with power grid stations nearby, so he parks the electric car in the safe area that is closest to the concert and presses the button "Terminate reservation". He exits properly from the car and receives a notification with his charge for the last ride, where he is also informed that he has been charged 30% more than the specified price because he left the car with more than 80% of the battery empty, and he did not get any discount because he did not act in a virtuous way (by plugging the car to a power grid, sharing it with at least two other passengers or using the money saving option).

3.4.6 Scenario 6

John is a regular employee in a business company and after a long day of work he is finally close to going back home. While checking the last reports, his friend Paul reminds him about the national strike of the public transport occurring that day: John, who has already tried the PowerEnjoy services a couple of times, decides to login using the PowerEnjoy application, he presses the "Find available cars" button and when the map activity is completely loaded, he locates an available car near his office address and reserves it by pressing the button "Reserve". Fifteen minutes later, John's colleague Mary offers him a ride back home: John, who has always had a crush on Mary, cannot refuse and accepts enthusiastically. On the way back home with Mary, John suddenly remembers of his missed reservation: he checks his smarthphone and he sees that the PowerEnjoy app has notified him about a fee of 1 EUR for the missed reservation. A trivial cost for being able to talk with Mary on the way home.

3.4.7 Scenario 7

Lorenzo is a student of Politecnico of Milan but, unfortunately, he has having quite a struggle in succeeding in the exams. Since his parents are becoming tired of economically supporting him, Lorenzo is trying to sparing some of his monthly budget. After finding out about the Money Saving Option offered by PowerEnjoy, Lorenzo, already a registered user, decides to give it a try. Lorenzo starts up the PowerEnjoy app on his smartphone; from the menu of the main screen of the app he selects the button for the login and inserts his credentials; once logged in, he uses the function for finding an available car from his current position and he reserves it. Ten minutes later, Lorenzo is close to the car: using the PowerEnjoy app, Lorenzo unlocks the car by selecting the corresponding "Unlock reserved car " option on the menu and inserting his personal PIN; once in the vehicle, Lorenzo checks the "Saving money option" showed on the screen: he then enters his destination, the home of his physics tutor, and the closest safe area to the inserted destination is shown to him; Lorenzo accepts it by pressing

the corresponding button on the screen. After his journey, Lorenzo parks in the defined safe area; he then press the option "Terminate reservation" on the car's screen and exits properly from the car. A few minutes later Lorenzo is notified via app about the charged price and the applied discount. Lorenzo can now travel through the city without the worry of spending too much money, thus focusing more on his studies.

3.4.8 Scenario 8

Thomas loves going to the cinema. Tonight, Thomas wants to watch an interesting action movie that he heard about from a friend: as a registered user of PowerEnjoy, he logs into the system inserting his credentials, he locates a car near his home address and he reserves it. After thirty minutes from his reservation, Thomas arrives at 5 meters from the reserved car: he requests the car unlock by inserting his personal PIN in the app and after about 30 seconds the system unlocks the car. After his journey, Thomas parks the car in one of the power safe areas in the city area. Thomas remembers that he has not chosen the "Money Saving Option" and notices that the car has 40% of battery left: after selecting the "Terminate reservation" option on the screen of the car and exiting properly from the car, he decides to plug the car into one power plug of the power grid of that power safe area. While entering the cinema, Thomas gets a notification from the PowerEnjoy app: he obtained a 30% discount on the last ride. Happy for his reward, now Thomas can now enjoy better the movie.

3.5 UML Models

3.5.1 Use Case

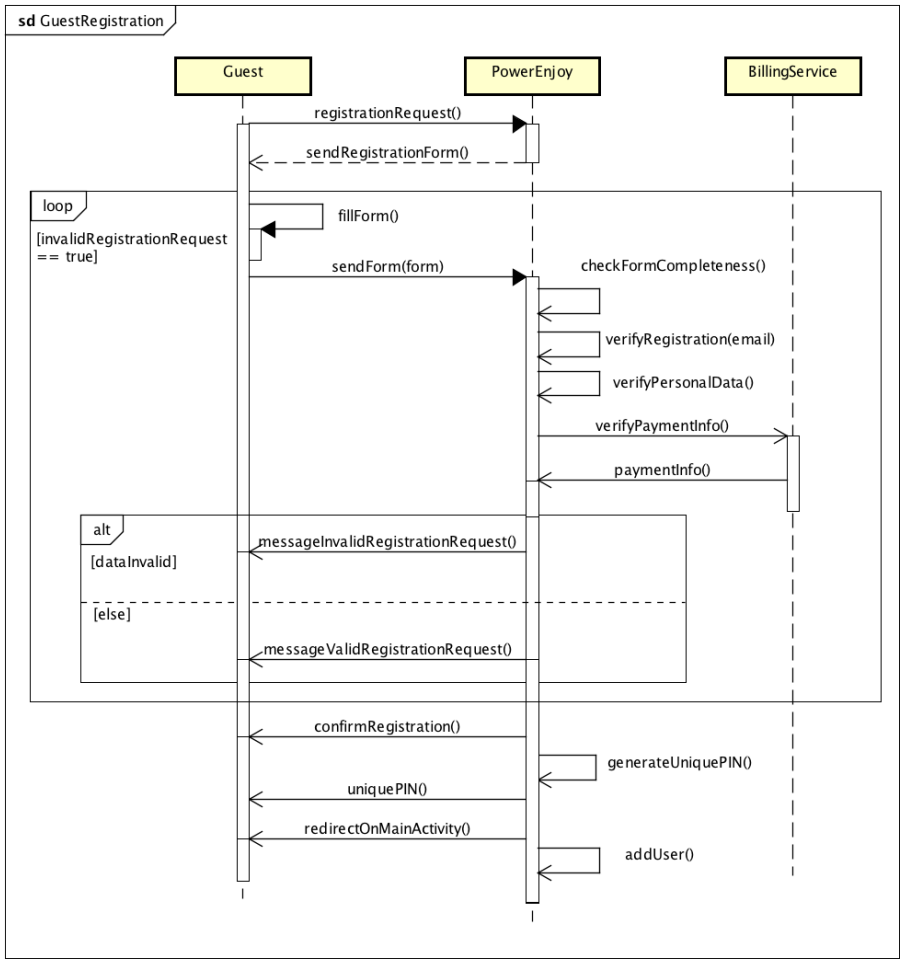
The following are relevant use cases for our S2B. Depending on the interest we have found in highlighting the interactions in a use case, a sequence diagram and/or a use case diagram may be provided.

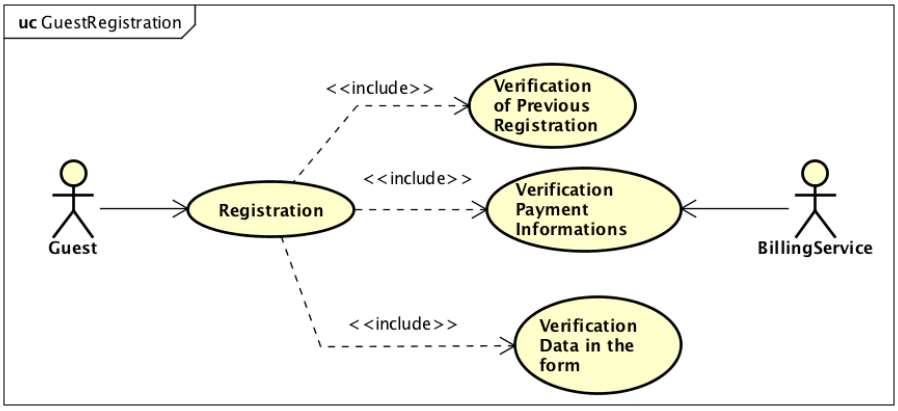
Use case [UC1] GuestRegistration

Goal	[G1] Allow users to find available cars and reserve them
Actors	Guest, BillingService
Entry Condition	none

Flow of events	• The Guest opens the main activity of the PowerEnjoy mobile web application. • The Guest touches the "Register" button to start the registration procedure. • The system replies by sending the registration form to the Guest. • The Guest fills the form with all his credentials, personal data, payment informations. He reads the "License agreement contract for PowerEnjoy" and highlights the option to accept the license agreement. Then he touches the "Confirm" button. All the data are sent to the system. • The system checks the form completeness. • The system has verified the form completeness and verifies the Guest credentials and personal data. • The system has verified the correctness of user credentials and asks payment validation to the BillingService. • The BillingService answers to the system that the payment informations are valid. • The system confirms the registration by sending a confirmation message to the Guest. • The system sends an email with the unique PIN generated for the new User.
Exit conditions	• The Guest is now a User. The User is redirected by the system again on the main activity. • The system adds an entry in the user's database for the new User.

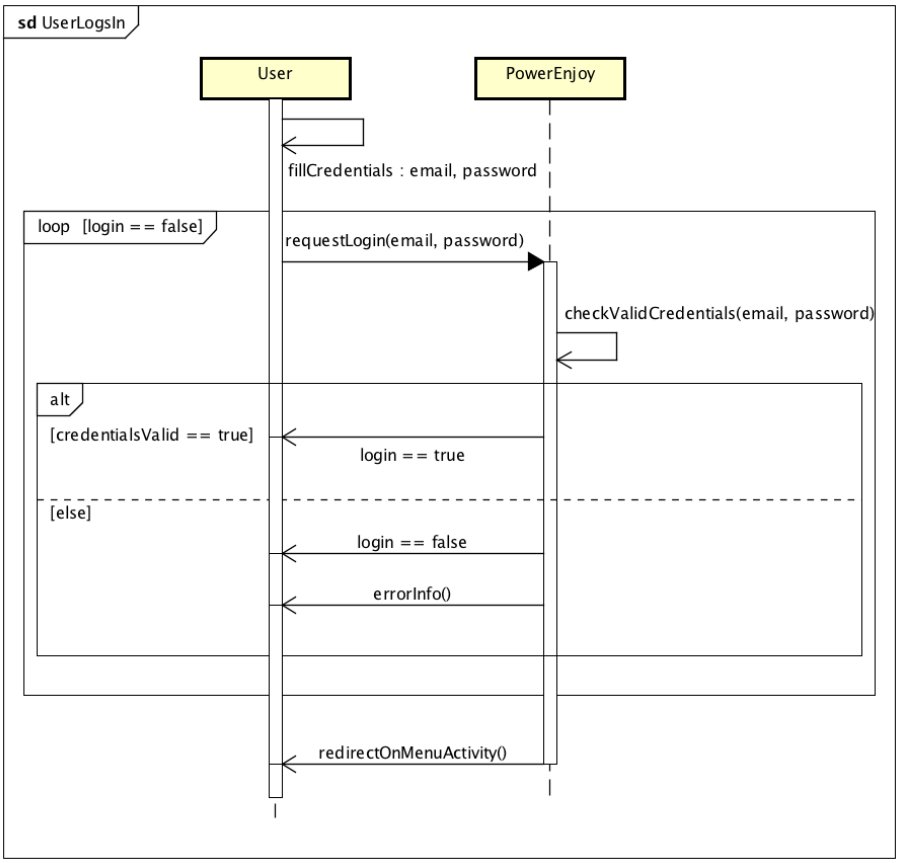
Exceptions	• The Guest has provided an email already used by another User : the system notifies the Guest to use another email. • The Guest is not at least 18 years old: the system notifies the Guest about this constraint on the use of the service. • The Guest has not filled all the data requested in the form: the system notifies to the Guest what is missing. • The Guest has provided invalid payment informations: the system notifies the Guest to insert different payment informations.

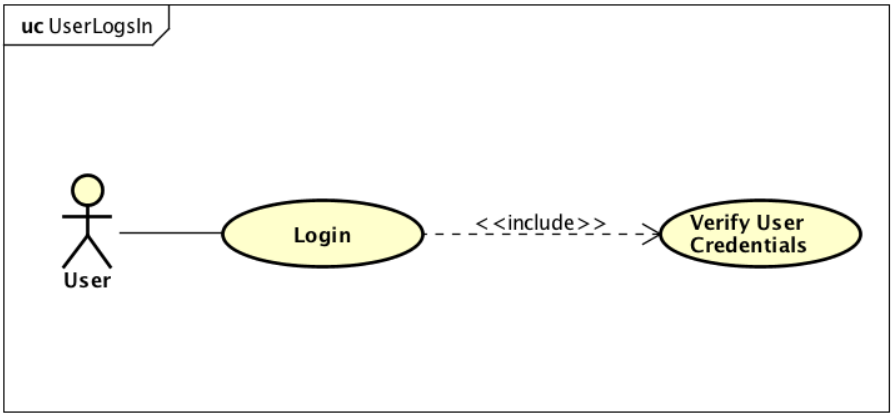




Use case [UC2] UserLogsIn

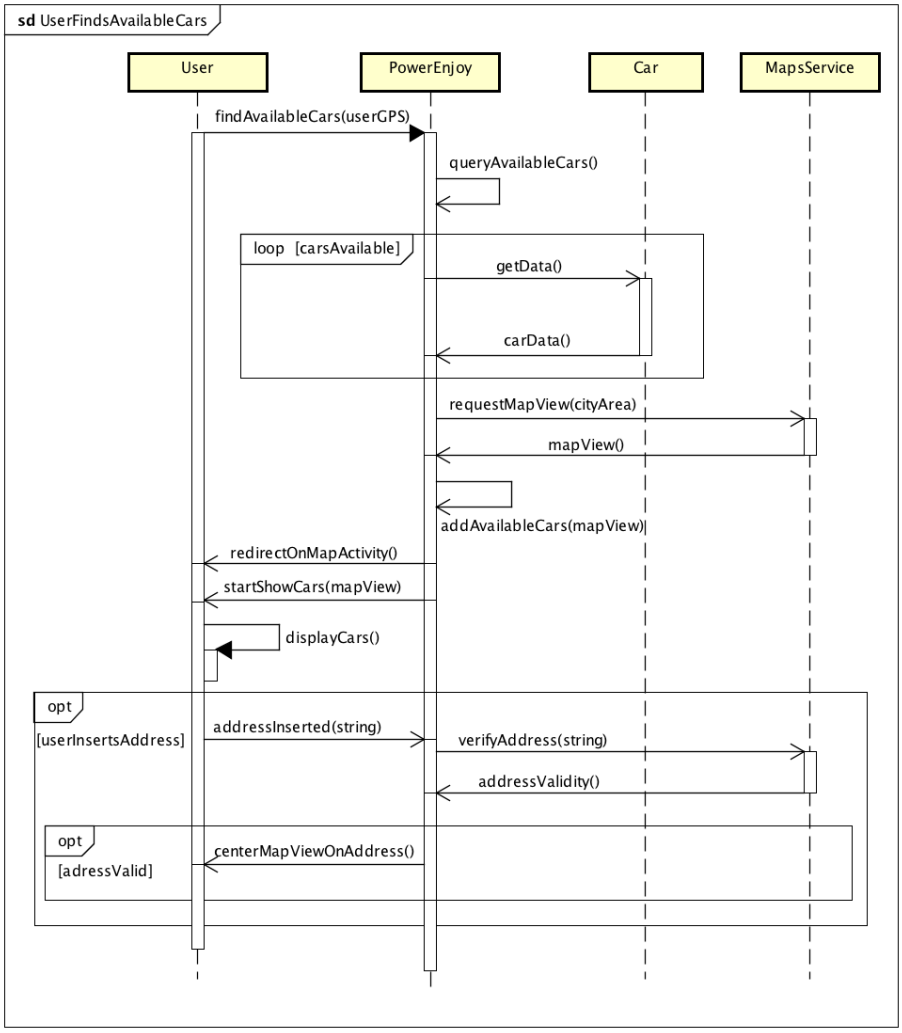
Goal	[G1] Allow users to find available cars and reserve them
Actors	User
Entry condition	User is registered to PowerEnjoy
Flow of events	• The User opens the main activity of the PowerEnjoy mobile web application. • The User inserts his email and password and touches the button "Login". When the button "Login" is pressed, the informations inserted are sent to the system. • The system verifies the correctness of the email and password provided by the User. • The system has verified that the credentials of the User are correct and sends a confirmation message to the User.
Exit conditions	The User is logged into the PowerEnJoy system and he is redirected by the system to the menu activity.
Exceptions	The email and/or password of the User are wrong, the system sends a message to the User about the error.

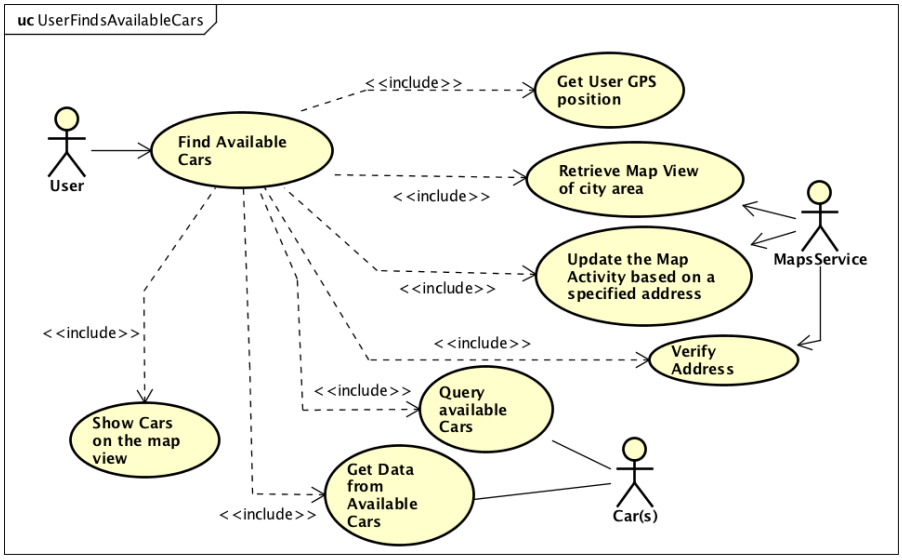

sd UserLogsIn
User
PowerEnjoy
fillCredentials : email, password
loop [login == false]
requestLogin(email, password)
checkValidCredentials(email, password)
alt
[credentialsValid == true]
login == true
[else]
login == false
errorInfo()
redirectOnMenuActivity()


uc UserLogsIn
User
Login
<<include>>
Verify User Credentials

Use case [UC3] UserFindsAvailableCars

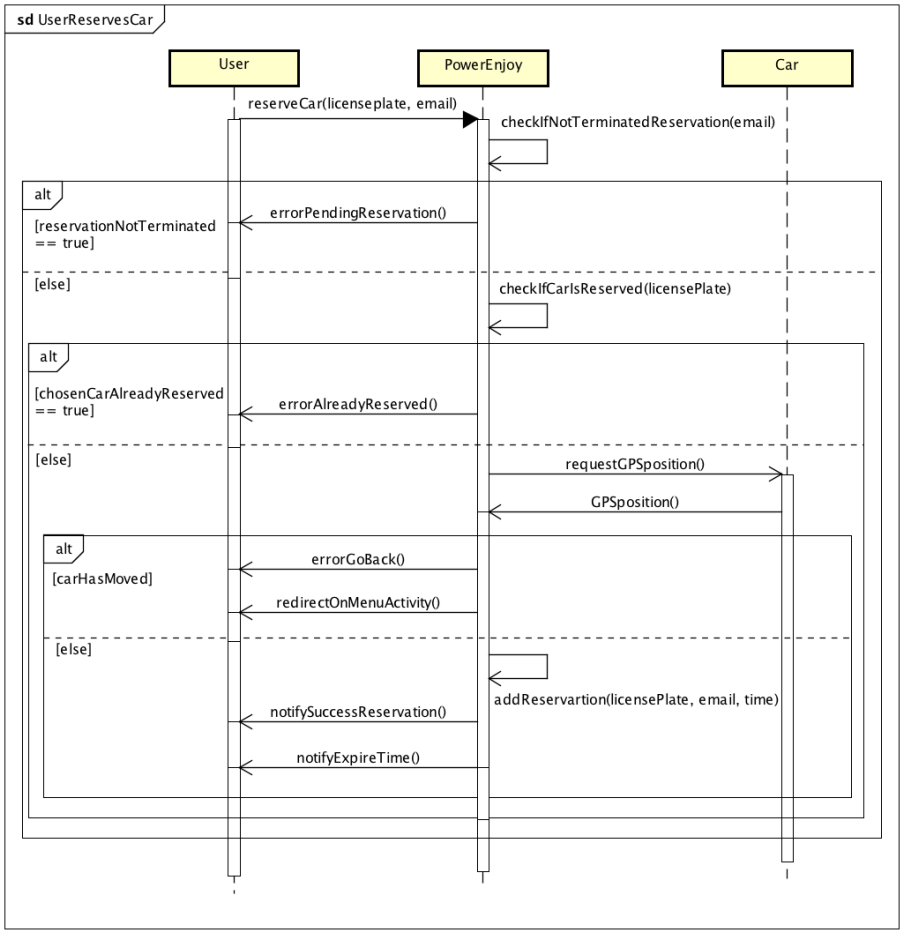
Goal	[G1] Allow users to find available cars and reserve them
Actors	User, Car(s), MapsService
Entry condition	"UserLogsIn"
Flow of events	• The User, using his mobile device, presses the button "Find Available Cars" in the menu activity of the PowerEnjoy application. When the button is pressed, the GPS position of the User is sent to the system. • The system queries every available Car for its data (battery level in particular). • The system requests a map view of the city area to the MapsService. • The system adds the available Cars data to the map view. • The system redirects the User to a map activity where the map view is displayed. • Every available Car is shown on a map view on the screen of the User. • The User may insert a specified address to highlight on his device screen a portion of the map view. The address can be inserted through a top bar inside the map activity. • The system may ask the MapsService about the validity of the address inserted by the User. • The system may update the map activity based on the address inserted by the User.
Exit conditions	The User can now touch the image of a Car inside the map view and if this happens the use case "UserReservesCar" is invoked.

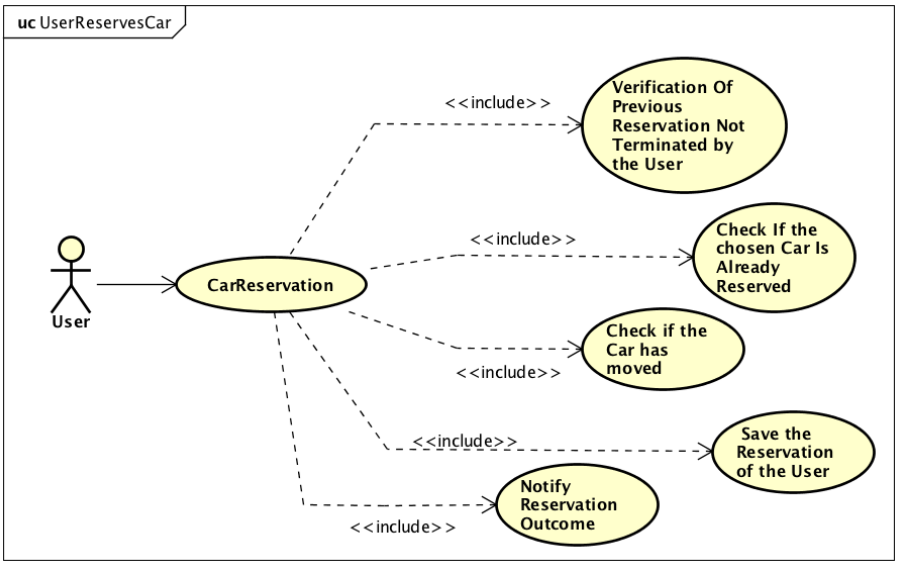

sd UserFindsAvailableCars
User
PowerEnjoy
Car
MapsService
findAvailableCars(userGPS)
queryAvailableCars()
loop [carsAvailable]
getData()
carData()
requestMapView(cityArea)
mapView()
addAvailableCars(mapView)
redirectOnMapActivity()
startShowCars(mapView)
displayCars()
opt
[userInsertsAddress]
addressInserted(string)
verifyAddress(string)
addressValidity()
opt
[adressValid]
centerMapViewOnAddress()


uc UserFindsAvailableCars
User
Find Available Cars
<<include>>
Get User GPS position
Retrieve Map View of city area
<<include>>
MapsService
Update the Map Activity based on a specified address
<<include>>
<<include>>
Verify Address
<<include>>
<<include>>
Query available Cars
Show Cars on the map view
Get Data from Available Cars
<<include>>
Car(s)

Use case [UC4] UserReservesCar

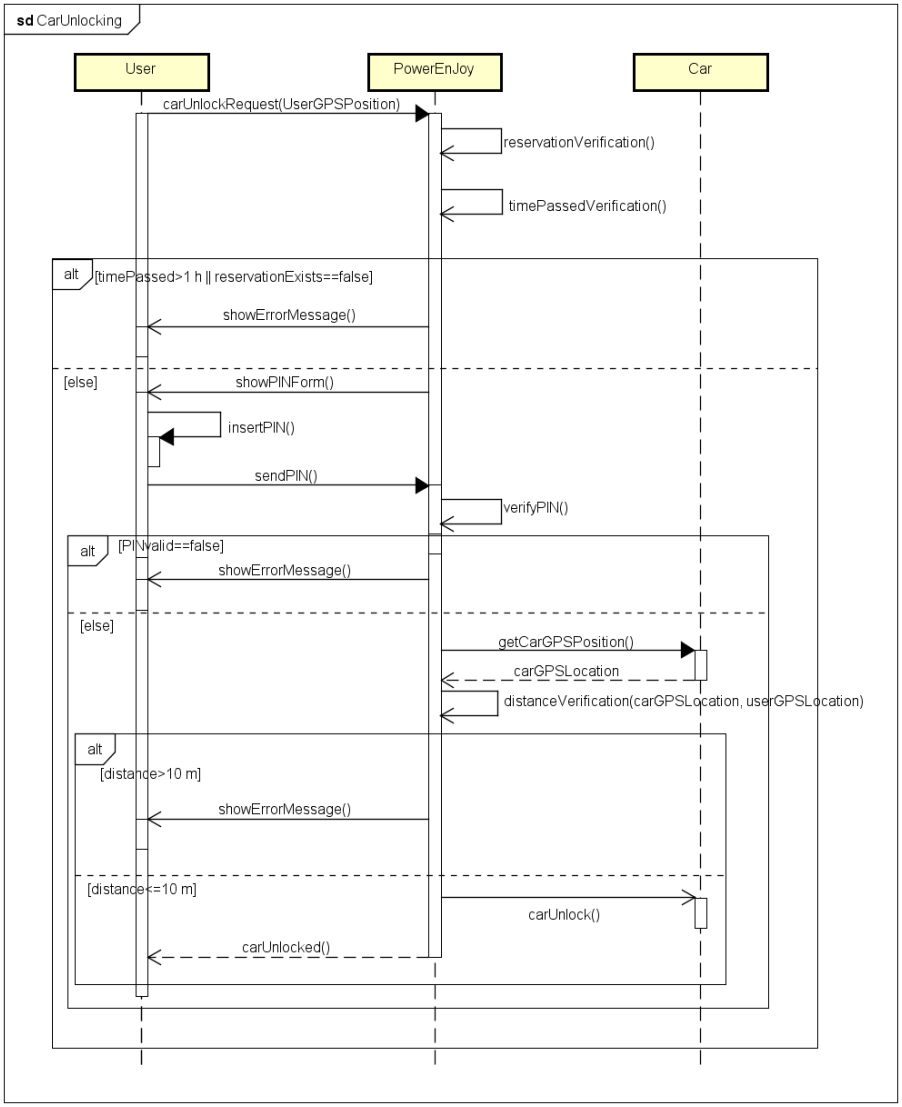
Goal	[G1] Allow users to find available cars and reserve them
Actors	User, Car
Entry conditions	"UserFindsAvailableCars"
Flow of events	• The User is inside the map activity and presses on a Car icon. • A popup dialog is shown to the User showing the address of the city in which the Car is, the Car's battery level and its license plate. All the informations are based on what the User has received when the map activity was started. The dialog shows also a button "Reserve". • The User presses the button "Reserve". This information is sent to the system. • The system checks if the User has already done a reservation of a Car which is not terminated yet. • The system checks if the chosen Car is reserved or not. • The system queries the chosen Car to see if it is really available at its specified position. • The system saves the reservation of the User for the chosen Car starting from the reservation request time.
Exit conditions	The system notifies the User about the success of the reservation procedure indicating also the time when the reservation will expire if the reserved Car is not be picked-up.
Exceptions	• The User has already done a reservation of a car which is not terminated yet: the system notifies the User that he can reserve another car only when the other reservation is terminated. • The User cannot complete the reservation since during the reservation procedure, another User has managed to reserve the chosen car: the system notifies the User of being too late and to find another available car. • The User waits too much time and when he presses on the car icon, the system notifies the User that the chosen car is no more there. The system redirects the User to the menu activity so he can invoke again the "UserFindsAvailableCars" use case.

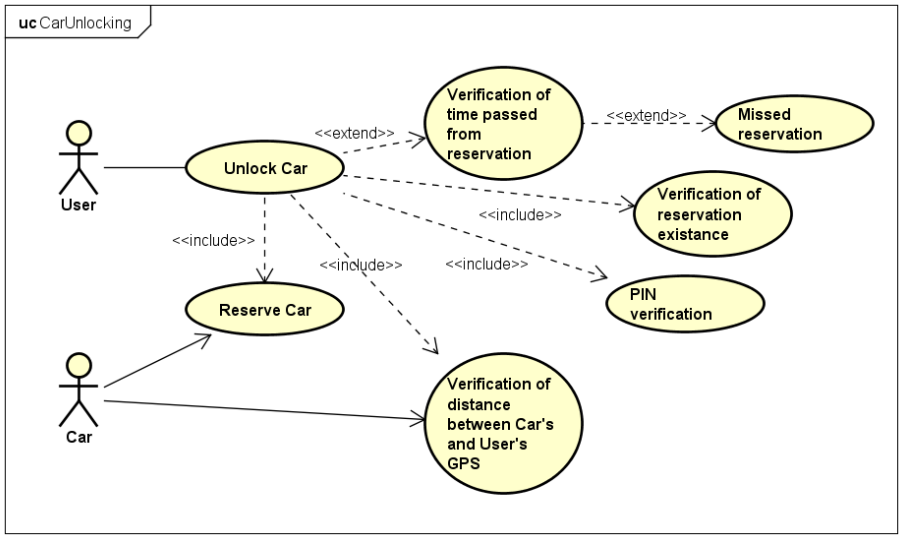

sd UserReservesCar
User
PowerEnjoy
Car
reserveCar(licenseplate, email)
checkIfNotTerminatedReservation(email)
alt
errorPendingReservation()
[reservationNotTerminated == true]
[else]
checkIfCarIsReserved(licensePlate)
alt
errorAlreadyReserved()
[chosenCarAlreadyReserved == true]
[else]
requestGPSposition()
GPSposition()
alt
errorGoBack()
[carHasMoved]
redirectOnMenuActivity()
[else]
addReservartion(licensePlate, email, time)
notifySuccessReservation()
notifyExpireTime()


uc UserReservesCar
User
CarReservation
<<include>>
Verification Of Previous Reservation Not Terminated by the User
<<include>>
Check If the chosen Car Is Already Reserved
Check if the Car has moved
<<include>>
<<include>>
Save the Reservation of the User
Notify Reservation Outcome
<<include>>

Use case [UC5] CarUnlocking

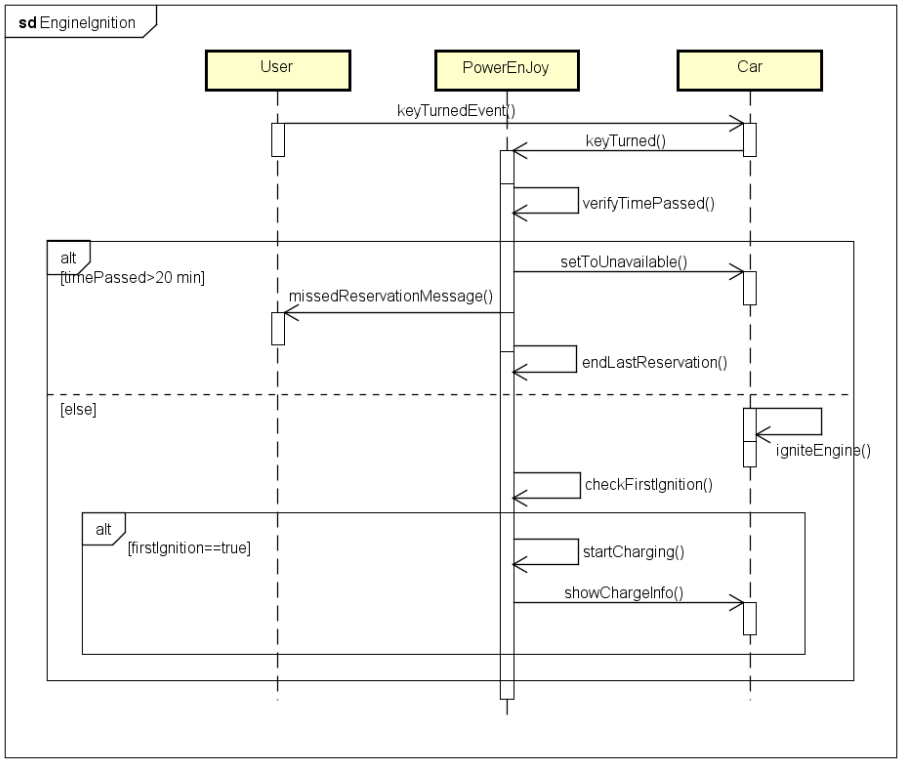
Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	User, Car
Entry conditions	"UserReservesCar" use case
Flow of events	• The User selects the "Unlock reserved car" button on the menu activity of the PowerEnJoy mobile application • The system verifies that a reservation actually exists and, if this is the first unlocking, that at most 1 hour has passed from the User's reservation • The system verification is successful • The User inserts his personal PIN code on the form that appears on the mobile application and confirms it by pressing "Ok" • The system verifies the correct correspondence between the PIN code inserted and the user that reserved the car • The system takes the User's GPS location from the User's mobile phone • The system requests the reserved car's GPS location to the Car • The Car provides its current GPS location • The system verifies that the User's GPS location is at most 10 m distant to the reserved car's GPS location
Exit conditions	The distance verification is successful and the car unlocks automatically within 30 seconds.
Exceptions	• The User has done no reservation; an error message is displayed on screen of the mobile phone of the User • More than 1 hour has passed from the reservation to the first unlocking request; the "MissedReservation" use case is invoked • The PIN code inserted by the User is incorrect; the system shows an error message on the mobile application and asks the User to try again • The control of the distance between the car and the User fails; the failure is communicated via notification on the PowerEnJoy app and the User can try again

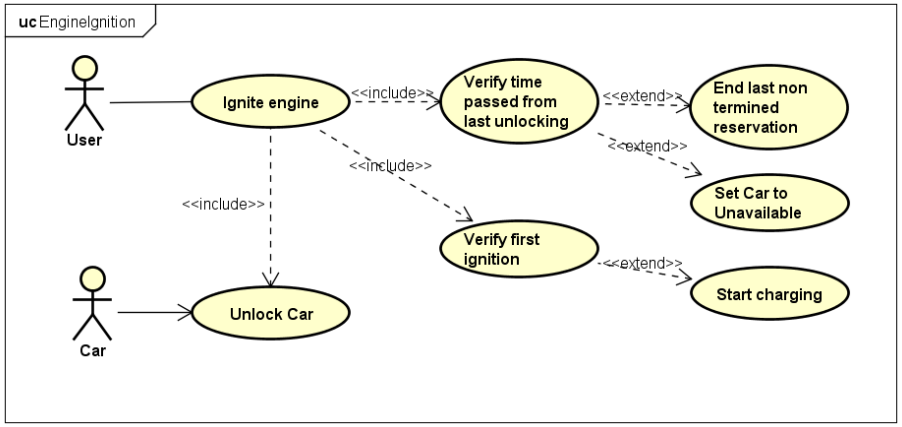

sd CarUnlocking
User
PowerEnJoy
Car
carUnlockRequest(UserGPSPosition)
reservationVerification()
timePassedVerification()
alt
[timePassed>1 h || reservationExists==false]
showErrorMessage()
[else]
showPINForm()
insertPIN()
sendPIN()
verifyPIN()
alt
[PINvalid==false]
showErrorMessage()
[else]
getCarGPSPosition()
carGPSLocation
distanceVerification(carGPSLocation, userGPSLocation)
alt
[distance>10 m]
showErrorMessage()
[distance<=10 m]
carUnlock()
carUnlocked()



Use case [UC6] EngineIgnition

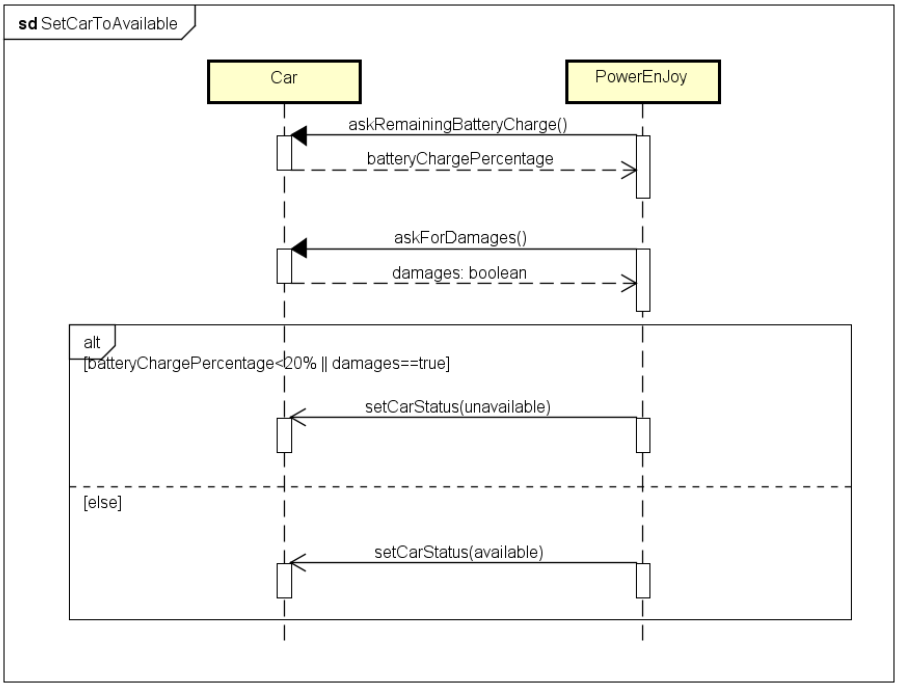
Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	Car
Entry conditions	"CarUnlocking" use case
Flow of events	• The User enters his reserved and unlocked car • The User turns the car's key • The system verifies that at most 20 minutes have passed from the last unlocking of the car • The verification is successful • The system verifies if the event of the engine ignition is the first one during this reservation
Exit conditions	The engine ignites; if the event of the engine ignition is the first one during this reservation, the system starts charging the user for the established amount of money.
Exceptions	More than 20 minutes have passed from the last unlocking of the car; the car is set to unavailable and the last non terminated reservation is ended automatically.

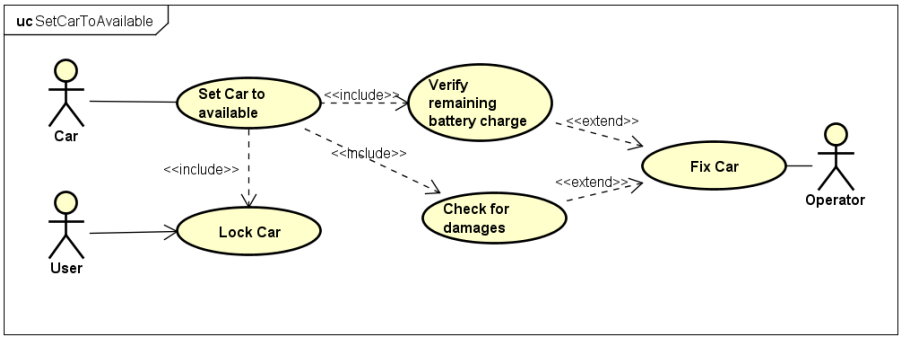

sd EngineIgnition
User
PowerEnJoy
Car
keyTurnedEvent()
keyTurned()
verifyTimePassed()
alt
[timePassed>20 min]
setToUnavailable()
missedReservationMessage()
endLastReservation()
[else]
igniteEngine()
checkFirstIgnition()
alt
[firstIgnition==true]
startCharging()
showChargeInfo()


uc EngineIgnition
User
Ignite engine
<<include>>
Verify time passed from last unlocking
<<extend>>
End last non termined reservation
<<extend>>
<<include>>
Set Car to Unavailable
<<include>>
Verify first ignition
<<extend>>
Start charging
Car
Unlock Car

Use case [UC7] SetCarToAvailable

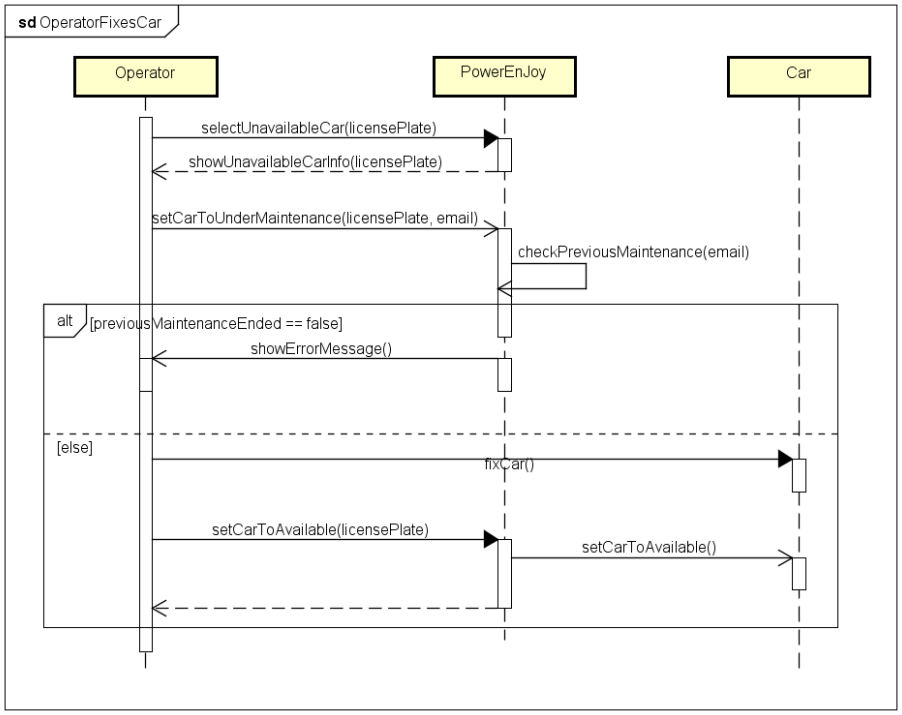
Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	Car
Entry conditions	"LockTheCarAutomatically" use case
Flow of events	• The system, using the vehicle software interface, queries the battery sensor of the Car about the remaining charge in the battery array of the Car. • The Car provides the remaining battery percentage to the system. • The system verifies that the Car's battery is at most 80% empty. • The system asks the Car's damage sensor if the Car has some damages. • The Car's damage sensor tells the system that the Car has no damage.
Exit conditions	The Car is set as available by the system, so that another User can reserve it.
Exceptions	The Car's battery is more than 80% empty or some other Car failure has occurred; the system sets the Car to unavailable.

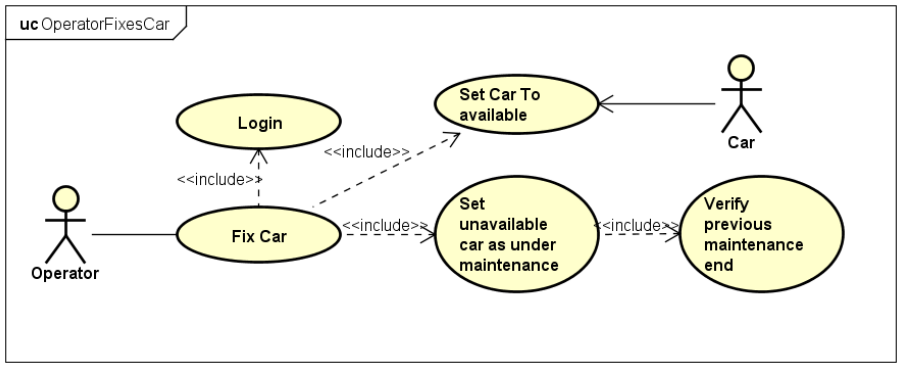




Use case [UC8] OperatorFixesCar

Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	Operator, Car
Entry conditions	The Car status has been set as unavailable (some Car failure has occurred or the battery is more than 80% empty); the Operator is already logged to the system using the PowerEnJoy mobile web application
Flow of events	• The Operator visualizes all the unavailable cars on the city map. • The Operator clicks on an unavailable car and visualizes its problem (kind of failure that has occurred) and the car's info. • The Operator sets the unavailable car as under maintenance by clicking on the dedicated button. • The system verifies that the Operator has no other cars under his maintenance at the moment. • The verification is successful and the system confirms the Operator's choice. • The Operator goes on-site and fixes the Car (by charging it through a portable electric car charger or by fixing the failure and by moving the Car in a safe area if necessary)
Exit conditions	When the Car is fixed, the Operator sets the Car back to available through the mobile application
Exceptions	The Operator had another car under his maintenance when making the maintenance request; the system shows an error message and leaves the selected Car as unavailable.


sd OperatorFixesCar
Operator
PowerEnJoy
Car
selectUnavailableCar(licensePlate)
showUnavailableCarInfo(licensePlate)
setCarToUnderMaintenance(licensePlate, email)
checkPreviousMaintenance(email)
alt
[previousMaintenanceEnded == false]
showErrorMessage()
[else]
fixCar()
setCarToAvailable(licensePlate)
setCarToAvailable()

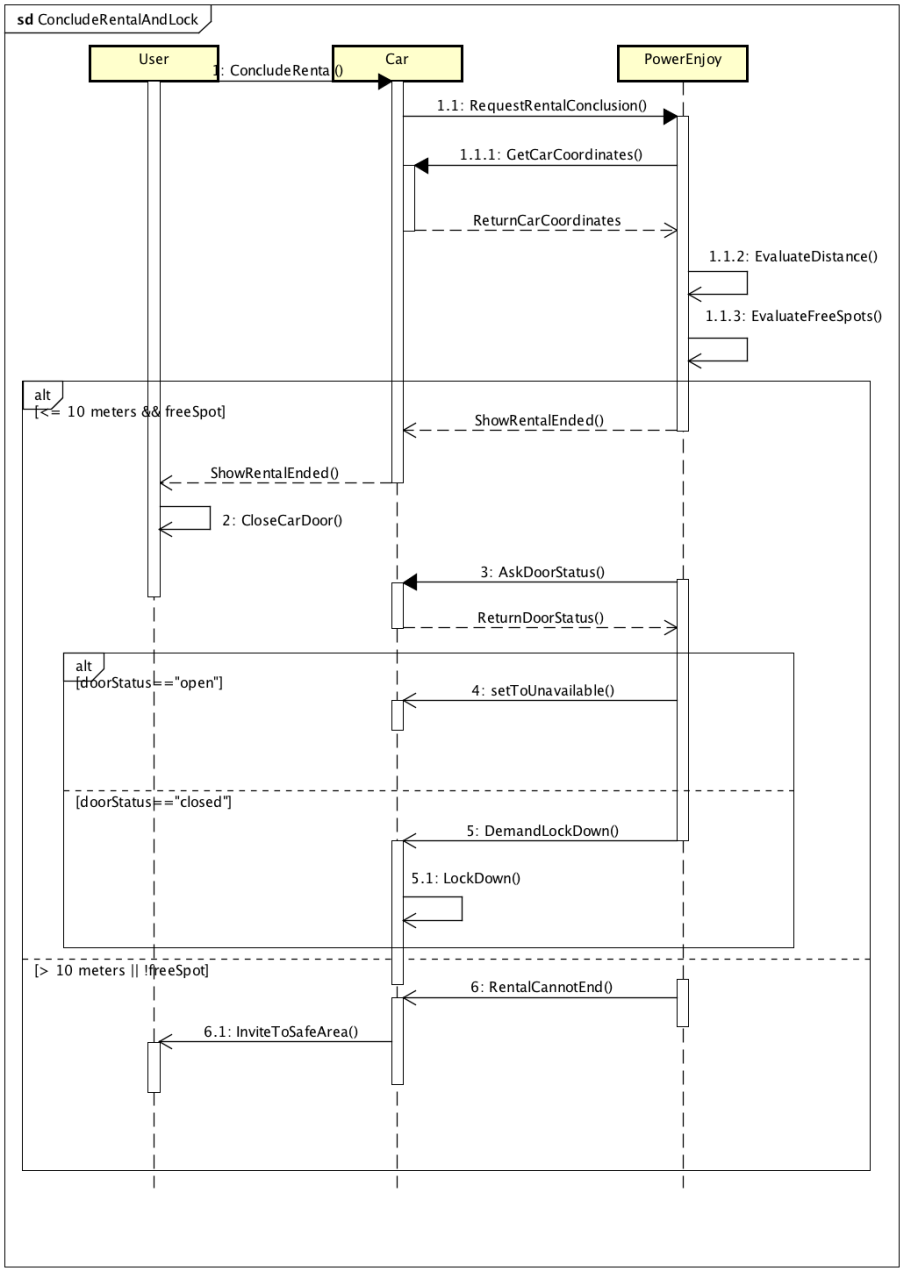

uc OperatorFixesCar
Login
Set Car To available
Car
<<include>>
<<include>>
Operator
Fix Car
<<include>>
Set unavailable car as under maintenance
<<include>>
Verify previous maintenance end

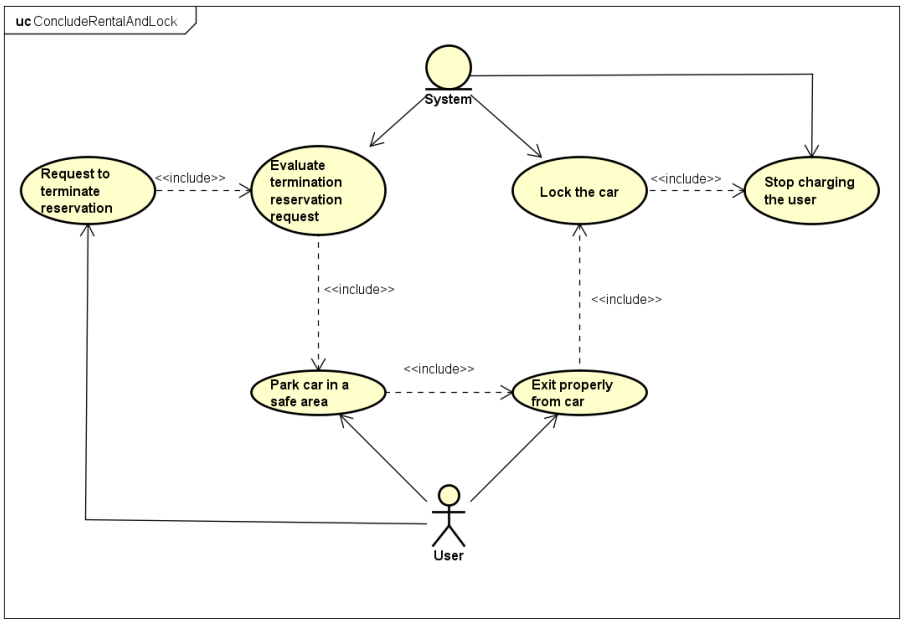
Use case [UC9] ConcludeRental

Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	User, Car
Entry conditions	The User has logged into the system, localized and reserved a car. He has then picked up the car within 1 hour from his reservation, and finally has concluded his journey by parking the car in a safe area or in a power safe area.
Flow of events	• The User presses on the car on-board screen the option "Terminate reservation". • The system, using the vehicle software interface, asks the Car's GPS the current location of the Car. • The Car's GPS provides the required information to the system. • The system confirms that the received coordinates corresponds to one of the known safe areas (with a reasonable approximation of 10 meters) and that there is at least one available parking slot in that safe area. • The system confirms the end of the rental on the Car on-board screen. The system registers that there is one less available parking slot in that safe area.
Exit conditions	After the proper exit of the User, the system proceeds to lock the Car, the use case "LockTheCarAutomatically" is invoked.
Exceptions	The system can't find a correspondence between the coordinates of the Car and the coordinates of any safe area or detects that there are no parking slots available in that safe area. The system notifies the User through the Car on-board screen that his rental cannot end until he reaches another safe area.

Use case [UC10] LockTheCarAutomatically

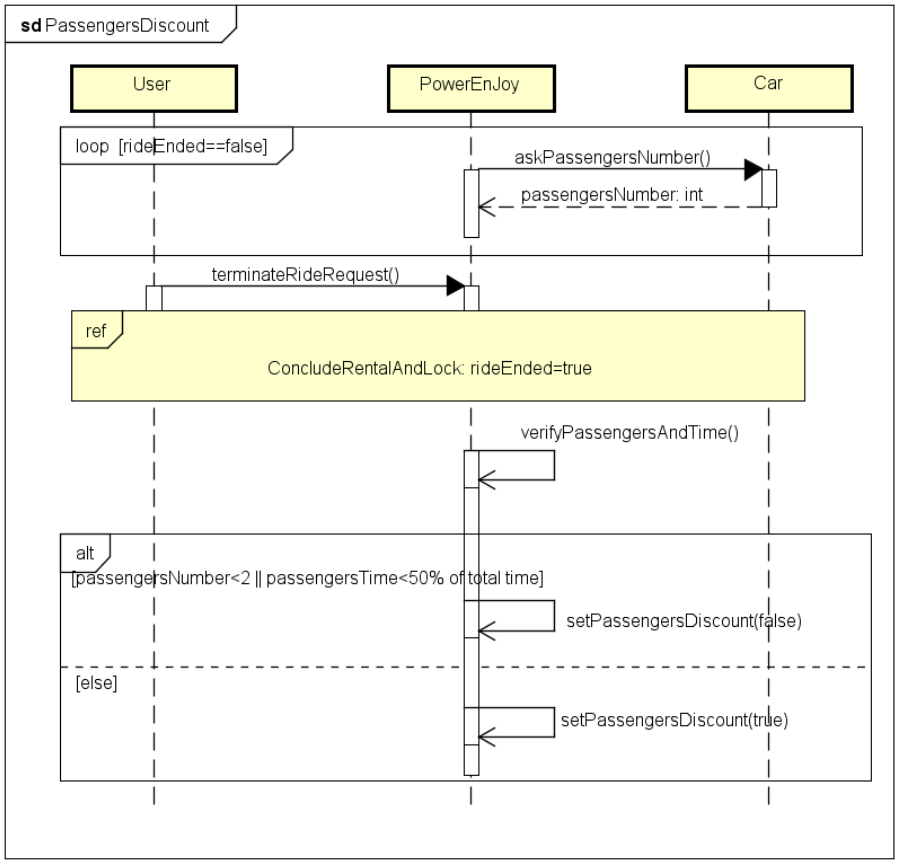
Goal	[G2] Allow users to use the reserved cars inside the city area
Actors	User, Car
Entry conditions	The User has decided to conclude his rental (use case "ConcludeRental") or to "Make a Stop". The User has exited the Car.
Flow of events	• The system, using the vehicle software interface, asks the Car's sensors if the car doors, the windows and the trunk are closed. • The Car's sensor tells the system that the doors, the windows and the trunk are closed. • The system requests the Car lock. • The Car locks down within 30 seconds from the request. • The system stops charging the User.
Exit conditions	The system proceeds with the evaluation of the User's bill, "PayLastRide"use case is invoked. Then the "SetCarToAvailable" use case is invoked.
Exceptions	The Car's door sensor tells the system that some car door, some windows or the trunk aren't closed; the Car status is set to unavailable and since that the user's reservation is ended by the system the system stop charging the user.


sd ConcludeRentalAndLock
User
Car
PowerEnjoy
1: ConcludeRental()
1.1: RequestRentalConclusion()
1.1.1: GetCarCoordinates()
ReturnCarCoordinates
1.1.2: EvaluateDistance()
1.1.3: EvaluateFreeSpots()
alt
[<= 10 meters && freeSpot]
ShowRentalEnded()
ShowRentalEnded()
2: CloseCarDoor()
3: AskDoorStatus()
ReturnDoorStatus()
alt
[doorStatus=="open"]
4: setToUnavailable()
[doorStatus=="closed"]
5: DemandLockDown()
5.1: LockDown()
[> 10 meters || !freeSpot]
6: RentalCannotEnd()
6.1: InviteToSafeArea()



Use case [UC11] PassengersDiscount

Goal	[G3] Guarantee a uniform distribution of the cars inside the city area
Actors	User, Car
Entry conditions	"EngineIgnition" use case
Flow of events	• During the ride, the system asks the Car's passenger sensors about the number of passengers in the Car • The Car's passenger sensors provide the number of passengers in the Car • The User decides to terminate his ride by pressing the dedicated button on the car's screen ("ConcludeRental" use case) • The system verifies that at least two other passengers were in the Car for at least 50% of the total time of the ride • The verification is successful
Exit conditions	A discount of 10% on the User's bill is registered.
Exceptions	The system establishes that there weren't at least two other passengers in the Car for at least 50% of the total time of the ride and the discount is not applied.

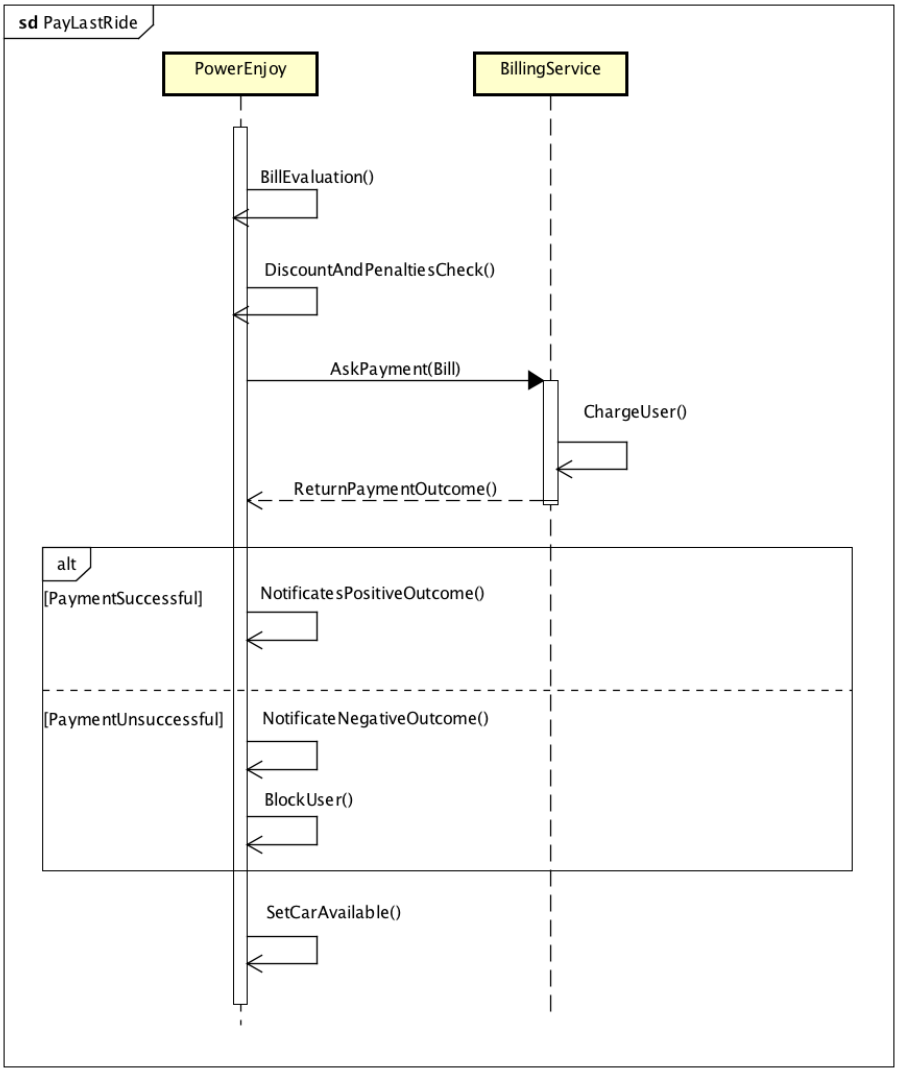

sd PassengersDiscount
User
PowerEnJoy
Car
loop [rideEnded==false]
askPassengersNumber()
passengersNumber: int
terminateRideRequest()
ref
ConcludeRentalAndLock: rideEnded=true
verifyPassengersAndTime()
alt
[passengersNumber<2 || passengersTime<50% of total time]
setPassengersDiscount(false)
[else]
setPassengersDiscount(true)

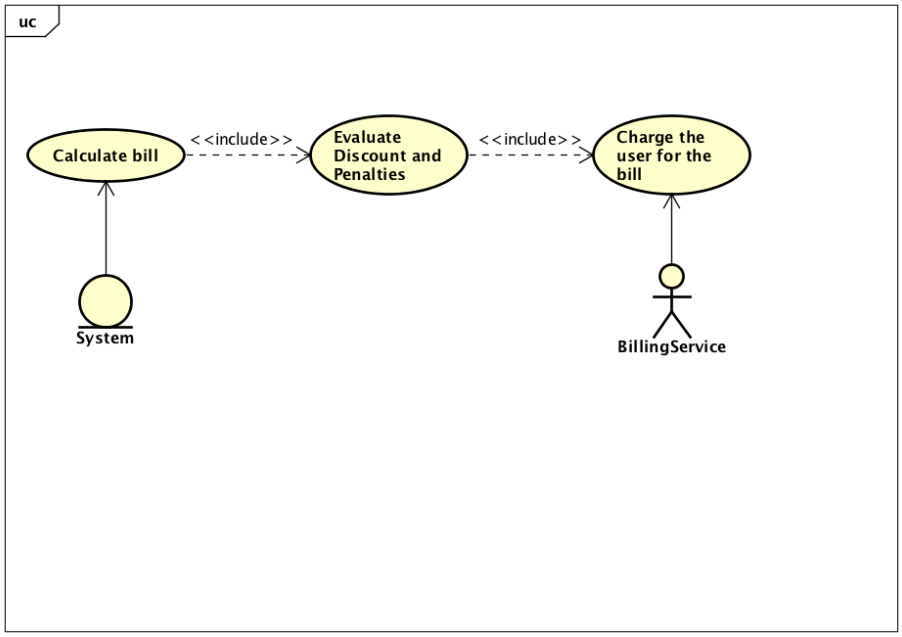
Use case [UC12] MoneySavingOption

Goal	[G3] Guarantee a uniform distribution of the cars inside the city area
Actors	User,Car
Entry conditions	"EngineIgnition" use case
Flow of events	• The User selects the "Money Saving Option" by activating a switch button on the Car's screen • The User inserts his destination address in the form that appears on the Car's screen • The system shows a map on the Car's screen with the location of the safe area or power safe area that is closest to the destination inserted, that allows the User to get a discount and that helps to reach a uniform distribution of the cars in the city area • During the ride, the system monitors the status of the shown safe area (number of parking slots that are available) and finds and shows a different safe area (repeating the previous point of the flow) on the Car's screen if there are no more parking slots left in the originally found safe area
Exit conditions	The User can park his car in the safe area or power safe area located by the system in order to get a discount at the end of the ride
Exceptions	The destination address inserted in not valid; the system shows a message on the Car screen saying to insert a valid address.

Use case [UC13] PayLastRide

Goal	[G4] Incentivize users to use properly the reserved cars
Actors	BillingService
Entry conditions	After the login ("UserLogsIn" use case) the User has localized ("UserFindsAvailableCars" use case) and reserved a car ("UserReservesCar" use cases). The User has unlocked the car ("CarUnlocking" use case) by inserting his personal PIN. The User has driven the reserved car and parked it in a safe area, requesting the end of the rental ("ConcludeRental" use case). The User has exited the car. The system has handled the car's lock ("LockTheCarAutomatically" use case).
Flow of events	• The system evaluates the total minutes of the User's last ride, starting from the first ignition of the engine. • The system, based on the previous evaluated minutes, calculates the User's bill value. • The system evaluates all possible discounts and penalties applicable to the user's bill: the system starts from the discount related to the remaining battery of the car ("GetBatteryDiscount" use case), proceeds to the discount related to the plugging of the car into a power grid ("PlugCarAndGetDiscount" use case), then evaluates the penalty related to parking away from a power safe area ("ParkFarFromGrid" use case) and, if the previous penalty was not applied, it evaluates the penalty related to the low status of the remaining battery ("LeaveLowBattery" use case). • If in "PassengersDiscount" use case the system established that the passengers discount has to be applied, a further discount of 10% on the User's bill is applied • The system interacts with the BillingService in order to account the payment to the User.
Exit conditions	The system notifies the User about the bill of the last ride, including eventual discounts and penalties applied.
Exceptions	The systems is notified that the payment was not successful. The system blocks the User.


sd PayLastRide
PowerEnjoy
BillingService
BillEvaluation()
DiscountAndPenaltiesCheck()
AskPayment(Bill)
ChargeUser()
ReturnPaymentOutcome()
alt
[PaymentSuccessful]
NotificatesPositiveOutcome()
[PaymentUnsuccessful]
NotificateNegativeOutcome()
BlockUser()
SetCarAvailable()



Use case [UC14] GetBatteryDiscount

Goal	[G4] Incentivize users to use properly the reserved cars
Actors	Car
Entry conditions	The system has evaluated the bill of the User's last ride ("PayLastRide" use case).
Flow of events	• The system, using the vehicle software interface, asks the Car's battery array sensor about the remaining charge. • The Car's battery array sensor provides the remaining battery array percentage to the system.
Exit conditions	The remaining battery in the battery array has more than 50% of the capacity and the system registers a 20% discount on the original calculated bill. The system evaluates a second possible discount ("PlugCarAndGetDiscount" use case).
Exceptions	The remaining battery in the battery array is less than 50% of the full capacity and so no discount is applied to the User.

Use case [UC15] PlugCarAndGetDiscount

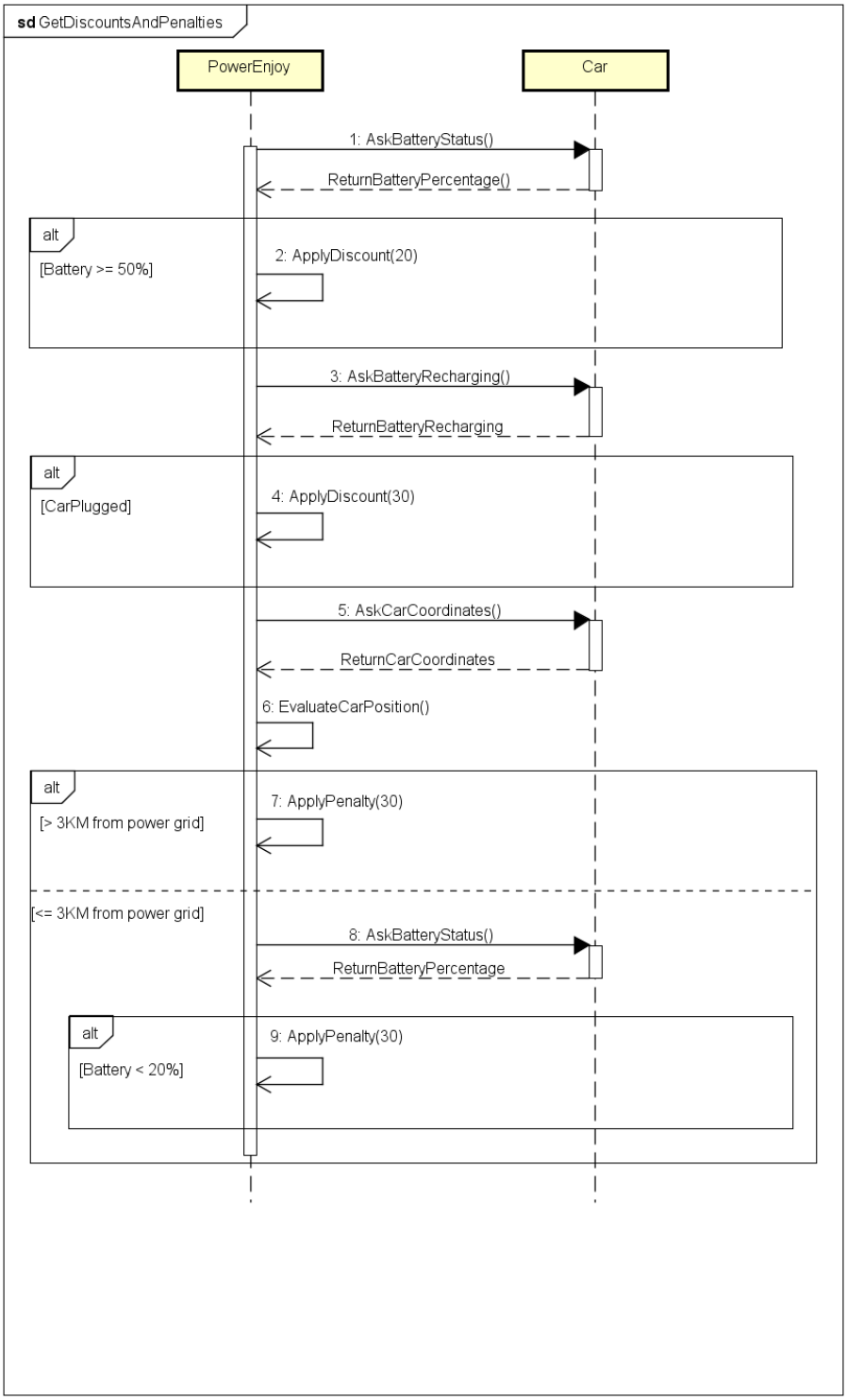
Goal	[G4] Incentivize users to use properly the reserved cars
Actors	User, Car
Entry conditions	The system has evaluated the remaining battery percentage in the battery array in order to apply the related discount ("GetBatteryDiscount" use case).
Flow of events	• The User plugs the Car using a power plug in a power safe area within 60 seconds after the lock of the Car. • The system, using the vehicle software interface, asks the Car's sensor if the Car is recharging. • The Car's sensor tells the system that the Car is recharging.
Exit conditions	The system calculates a 30% discount on the original calculated bill and adds any eventual previous calculated discounts ("GetBatteryDiscount" use case). The system proceeds to evaluate a possible penalty for the User ("ParkFarFromGrid" use case).
Exceptions	The User does not plug the Car into a power grid within 60 seconds after the lock of the car, the battery is not recharging and the system doesn't apply any discount.

UseCase [UC16] ParkFarFromGrid

Goal	[G4] Incentivize users to use properly the reserved cars
Actors	Car
Entry conditions	The system has evaluated if the Car was recharging in order to apply the related discount ("PlugCarAndGetDiscount" use case).
Flow of events	• The system asks through the vehicle software interface the Car's position, as detected by the Car's GPS. • The Car's GPS provides the GPS coordinate. • The system evaluates that the Car is more than 3 KM from the nearest power safe area.
Exit conditions	The system charges 30% more on the last ride, still considering eventual previous discount ("GetBatteryDiscount" use case, "PlugCarAndGetDiscount" use case) to be applied on the original bill.
Exceptions	The system evaluates that the Car is less than 3 KM from the nearest power safe area and no penalty is applied. The system checks for the battery status ("LeaveLowBattery" use case).

Use case [UC17] LeaveLowBattery

Goal	[G4] Incentivize users to use properly the reserved cars
Actors	Car
Entry conditions	The system has evaluated that the Car was far no more than 3 KM from the nearest power safe area from the car position ("ParkFarFromGrid" use case).
Flow of events	• The system, using the vehicle software interface, asks the remaining charge in the Car detected by the Car's battery array sensor. • The Car's battery array sensor provides the remaining battery percentage to the system through the software interface.
Exit conditions	The remaining battery in the battery array is less than 20% of the full capacity and the system registers a 30% charge on the last ride, that will be subtracted to any eventual previous discount ("GetBatteryDiscount" use case, "PlugCarAndGetDiscount" use case) on the original calculated bill.
Exceptions	The remaining battery in the battery array is more than 20% of the full capacity and no extra charge on the last ride is applied to the User

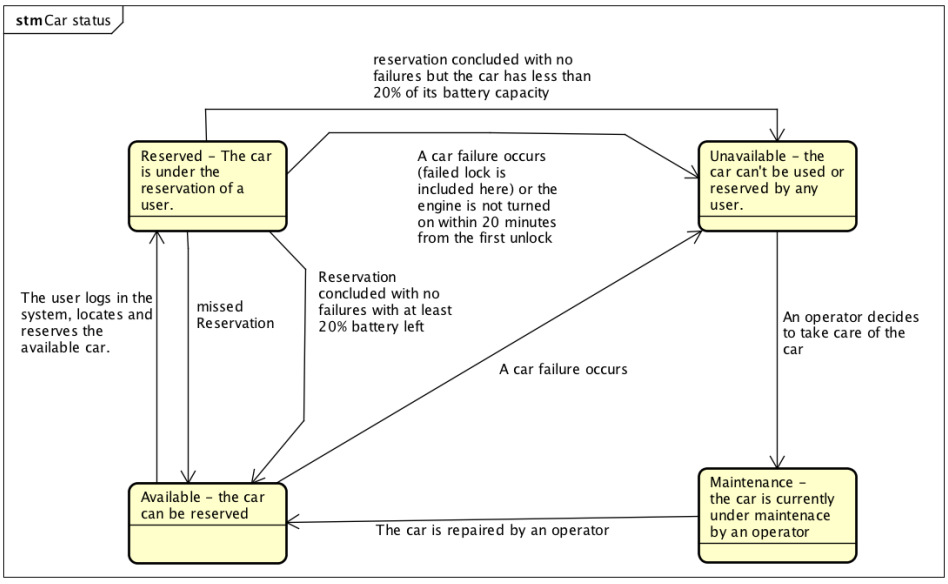

sd GetDiscountsAndPenalties
PowerEnjoy
Car
1: AskBatteryStatus()
ReturnBatteryPercentage()
alt
[Battery >= 50%]
2: ApplyDiscount(20)
3: AskBatteryRecharging()
ReturnBatteryRecharging
alt
[CarPlugged]
4: ApplyDiscount(30)
5: AskCarCoordinates()
ReturnCarCoordinates
6: EvaluateCarPosition()
alt
[> 3KM from power grid]
7: ApplyPenalty(30)
[<= 3KM from power grid]
8: AskBatteryStatus()
ReturnBatteryPercentage
alt
[Battery < 20%]
9: ApplyPenalty(30)

Use Case [UC18] MissedReservation

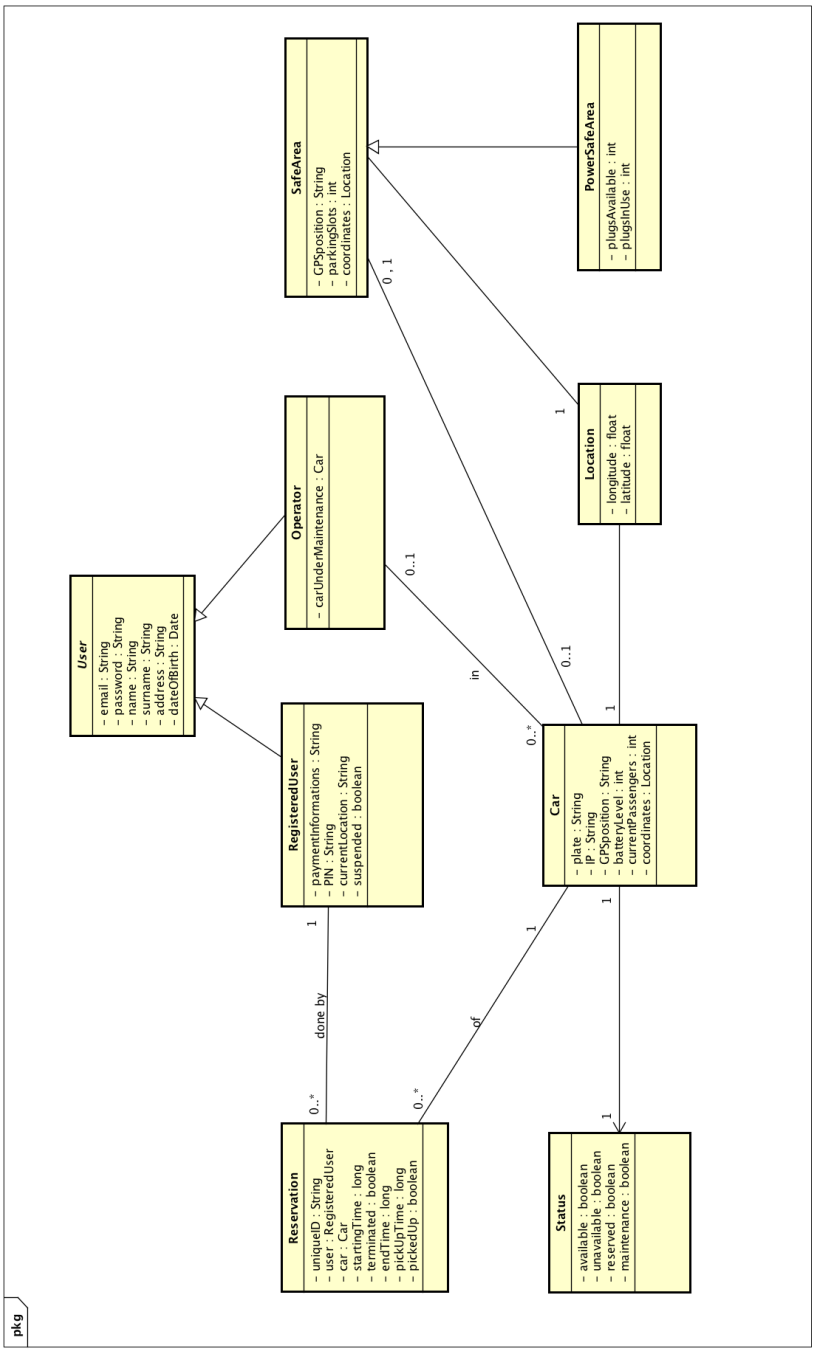
Goal	[G4] Incentivize users to use properly the reserved cars
Actors	User, Payment service
Entry conditions	The user has logged into the system ("UserLogsIn" use case), located and reserved a car ("UserFindsAvailableCars" use case, "UserReservesCar" use case).
Flow of events	• The user does not pick up the reserved car within 1 hour from his reservation. • The system interacts with an external payment service and charge the user a fee of 1 EUR. • The system notifies the user about the missed reservation and the consequent fee.
Exit conditions	The car that the user reserved, still fully functional, is now tagged as available, and other users can reserve it.
Exceptions	The users picks up the car in time and no fee is applied. The user does not have the requested money and the system, notified by the payment service, blocks his profile.

3.5.2 Statechart

The following is a statechart showing how we designed the evolution of the status of a car of the company.



3.5.3 Class Diagram



3.6 Non Functional Requirements

3.6.1 Performance Requirements

- [NF1] The architecture of the system should be designed in order to prioritize the interaction and communication with the cars when subject to high loads.
- [NF2] The number of terminals to be supported in parallel should be greater than 450 (which is 3*(#cars + #operators), where #cars = 100 and #operators = 50).
- [NF3] The unlock of a car should happen in less than 30 seconds starting from the initial request.

3.6.2 Design Constraints

- [NF4] The interaction between the cars and the system must be bidirectional.

3.6.3 Software System Attributes

3.6.3.1 Reliability

- [NF5] In order to obtain a constant and full control over the company's cars, failures of the system should not compromise the functionalities related to the management and the control of the cars.

3.6.3.2 Availability

- [NF6] The system shall guarantee 99% availability.

3.6.3.3 Security

- [NF7] The communication between a mobile application, when using Web Views to display PowerEnjoy web pages without leaving the app, and the PowerEnjoy servers should be build over the HTTPS protocol.
- [NF8] The relevant payment information provided by the users should not be stored in the database.

3.6.3.4 Maintainability The future application code must be documented as much as possible, to help the maintainers of the system and potential new developers that may become an active part in the development.

3.6.3.5 Portability The S2B should run on many different platforms: Android, iOS and in future maybe on a web browser. We will try to develop a web application that will not be specific to only one particular platform.

3.7 Traceability Matrix

Row ID	Goal ID	Requirement ID	Use Case ID
r1	G1.1	R1.1.1-R1.1.6	UC1, UC2
r2	G1.2	R1.2.1-R1.2.3	UC3
r3	G1.3	R1.3.1-R1.3.3	UC4
r4	G2	R2.1-R2.2	no specific use case
r5	G2.1	R2.1.1-R2.1.17	UC5, UC6, UC7, UC9, UC10
r6	G2.2	R2.2.1-R2.2.10	UC7, UC10, UC8
r7	G3	R3.1-R3.3	UC11, UC12
r8	G4	R4.1-R4.2	UC13
r9	G4.1	R4.1.1-R4.1.2	UC14, UC15
r10	G4.2	R4.2.1-R4.2.2	UC16, UC17, UC18

4 Appendix

4.1 Alloy

Here's the complete code for our Alloy model. The .als file can be found in our GitHub repository. For clarity we separate the code in Signatures, Facts and Asserts and Predicates. We have also inserted two possible worlds generated by the code with a brief description.

4.1.1 Abstract Entity and Signature

```
open util/boolean


    sig Name, Surname, Addr{}
    sig Email, Password, PIN{}
    sig PaymentMethod{}

    abstract sig User {
            name: some Name,
            surname: some Surname,
            address: one Addr,
            email: one Email,
            password: one Password
    }




    sig RegisteredUser extends User{
            pin: one PIN,
            suspended: one Bool,
```

```
        paymentMethod: one PaymentMethod,
        reservations: set Reservation
}

sig Operator extends User {
        carUnderMaintenance : lone Car
}

//A reservation is considered active from its creation
//until its termination
sig Reservation {
        creator : one RegisteredUser,
        reservedCar: one Car,
        requestUnlock: some UnlockRequest, //user can make different
        //car's unlock requests during his reservation
        active: one Bool
}

sig UnlockRequest {
        correctUserPosition: one Bool,
        unlockOutcome: one Bool //the request can have a positive
       //or negative outcome
}

sig GPS{}

sig Car {
        position: one GPS,
       available: one Bool,
       inMaintenance: one Bool,
        reserved: one Bool,
        reservation : set Reservation,
        locked: one Bool,
        inCharge: one Bool
}

sig SafeArea {
        coordinates: one GPS,
        numberOfSpots: one Int,
        availableSpots: one Int,
        powerGrid: one Bool
}{
        numberOfSpots > 0
        availableSpots >= 0
}
```

4.1.2 Facts

```
//Two different users can't have the same email
    fact NoSameEmailForDifferentUsers {
            no disjoint u1, u2 : User |
            u1.email = u2.email
    }

    //Two different registered users can't have the same PIN
    fact NoSamePINForDifferentUsers {
            no disjoint u1, u2 : RegisteredUser |
            u1.pin = u2.pin
    }

    //If a car is under maintenance, one and at most one operator
    //is taking care of it
    fact carUnderMaintenanceByOperator {
            all c: Car | one o: Operator |
            (c.inMaintenance = True implies
            o.carUnderMaintenance = c)
    }

    fact carAndOperatorRelation {
            all c: Car | all o: Operator |
            o.carUnderMaintenance = c implies
            c.inMaintenance = True
    }

    //If a car is available or in maintenance no active reservations
    //are associated to it
    fact carAvailableOrInMaintenance{
            all c: Car |
            (c.available = True or c.inMaintenance = True)
            iff #getActiveCarReservation[c] = 0
    }

    //A Car can't be in maintenance and available
    fact carMaintenance {
            no c: Car |
            c.inMaintenance = True and c.available = True
    }

    //If a car is reserved, one active reservation is associated to it
    fact carReserved {
            all c: Car |
            c.reserved = True iff #getActiveCarReservation[c] = 1
```

```
}

//If a Reservation instance is related to a User entity, then it is
//also related to the corresponding Car entity
fact ReservationUserAndCar {
        all r: Reservation | all u: RegisteredUser | all c: Car |
        (r in u.reservations and c = r.reservedCar) implies
        (r in c.reservation)
}

//A Car entity is associated with only the Reservation entities on
//that car
fact CarReservation {
        all c: Car | all r: Reservation |
        r in c.reservation implies
        r.reservedCar = c
}

//A User entity is associated with only the Reservation entities
//created by that User
fact UserReservation {
       all r: Reservation | all u: RegisteredUser |
        r in u.reservations implies r.creator = u
}

//No contemporary reservations on the same car are allowed
fact NoMultipleActiveReservationOnSameCar {
        all c: Car |
        #getActiveCarReservation[c] <= 1
}

//No contemporary reservations of the same user are allowed
fact NoMultipleActiveReservationForSameUser{
        all u: RegisteredUser |
        #getActiveUserReservation[u] <= 1
}

//If a user is suspended, none of his reservation is active
fact userSuspended {
       all u: RegisteredUser | all r: Reservation |
        (u. suspended = True and r in u.reservations)
        implies r.active = False
}

//The unlock request is successful only when the user is close
//enough to the car and vice versa
```

```
fact unlockRequestOutcome {
        all u: UnlockRequest |
        u.unlockOutcome = True iff
        u.correctUserPosition = True
}

//Each Reservation has a unique set of UnlockRequest
fact unlockRequestAndReservation {
        all disjoint r1, r2 : Reservation |
        r1.requestUnlock & r2.requestUnlock = none
}

//The available spots of a safe area are calculated as
//the total number of spots minus the cars in that safe area
fact totalAvailableSpots {
        all a: SafeArea |
        a.availableSpots <= a.numberOfSpots and
        a.availableSpots = minus[a.numberOfSpots,
        #getTotalCarInSafeArea[a]]
}

//When a car is available, it is locked and situated in a
//safe area
fact availableCarPosition {
        all c : Car | one a: SafeArea |
        c.available = True implies
        (c.locked = True and c.position = a.coordinates)
}

//Safe areas are located at different locations
fact safeAreasDifferentLocation {
        all disjoint a1, a2 : SafeArea |
        a1.coordinates != a2.coordinates
}
```

4.1.3 Functions

```
//Return all the active reservations of a user
    fun getActiveUserReservation[u: RegisteredUser] : set Reservation {
            {r: Reservation | r.creator = u and r.active = True }
    }

    //Return all the active reservation associated to a car
    fun getActiveCarReservation[c: Car] : set Reservation {
            {r: Reservation | r.reservedCar = c and r.active = True }
    }

    //Return all the cars located in a safe area
    fun getTotalCarInSafeArea[a: SafeArea] : set Car {
            {c: Car | c.position = a.coordinates}
    }
```

4.1.4 Asserts and Predicates

```
//Check if the car can only be in one status at a time
    assert carStatusConsistency {
            no c: Car |
             (c.reserved = True and c.available = True) or
             (c.inMaintenance = True and c.available = True ) or
             (c.reserved = True and c.inMaintenance = True ) or
             (c.reserved = False and c.inMaintenance = False
            and c.available = False)
    }

    check carStatusConsistency

    //Check the interaction between cars and reservations
    assert ReservationConsistency {
            all c: Car | all r: Reservation |
            (#getActiveCarReservation[c] = 0 and r.reservedCar = c)
            implies r.active = False
    }

    check ReservationConsistency

    //Check if a Reservation entity is correctly related between User and Car
    assert ReservationOfAvailableCars {
            all u: RegisteredUser | all r: Reservation |
             (r.active = True and r in u.reservations) implies
             (r.reservedCar.reserved = True and
```

```
        r in r.reservedCar.reservation)
}

check ReservationOfAvailableCars

pred show() {
        #Name > 1
        #Addr > 1 //imposed for clarity porpouse,
        //but users with same name or surname are admitted
}

run show for 2
```

4.2 Alloy Response

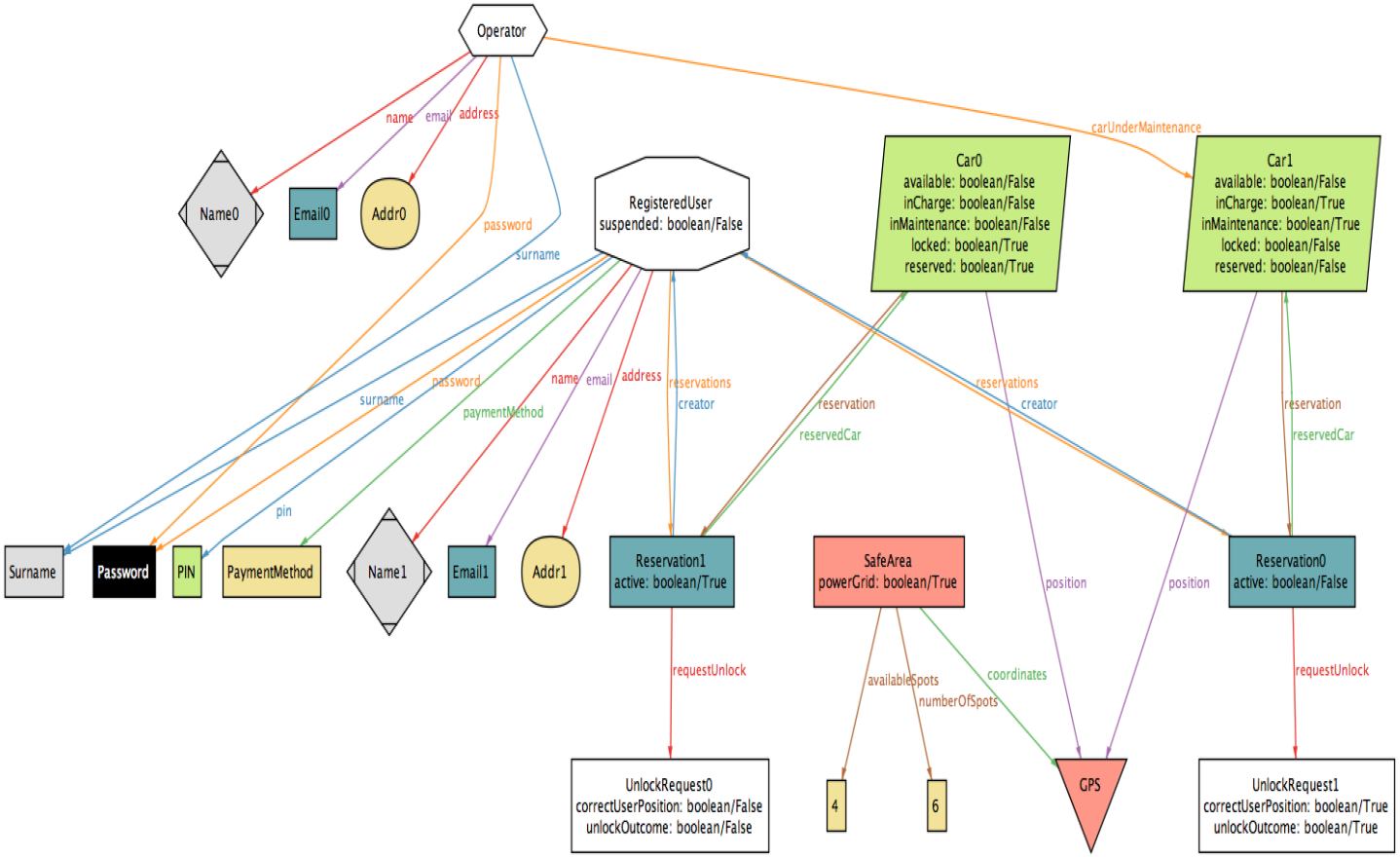
```
4 commands were executed. The results are:
  #1: No counterexample found. carStatusConsistency may be valid.
  #2: No counterexample found. ReservationConsistency may be valid.
  #3: No counterexample found. ReservationOfAvailableCars may be valid.
  #4: Instance found. show is consistent.
```

4.3 Alloy World

Here are presented two possible generated world using the Alloy verification software. Both diagrams are the predicate show() for 2 cases.

The first generated world represents the situation where a registered User has done two different reservations, with of course only one of them active at the moment. The active reservation is related to a locked and reserved Car situated in a Safe area: we notice that there is an unsuccessful unlock request related to this reservation, so at the moment of the generation of the world the user has tried to unlock his reserved car for the first time with a negative outcome. In the same safe area there is also another Car that is currently under maintenance: we see that the Safe Area has a consistent number of available spots (only 4, since two of them are occupied by the two Cars) and that the Car under Maintenance is associated with only one operator. By chanche, both the registered User and the Operator share the same password and surname.

The second generated world represents the registered user has done two different reservation on two different cars: notice that none of them is active at the moment of the generation of the world, thus creating a situation consistent with our requirements. We notice that Car0 is available: there isn't any reservation connected to it which is active and the Car is locked in the SafeArea0. Car1 is also available and locked, but it also recharging and it is situated in the SafeArea1. There is also an Operator in this world, but since there is no Car under maintenance he is ha not a Car associated to it.


Operator
name
email
address
carUnderMaintenance
Name0
Email0
Addr0
password
surname
RegisteredUser
suspended: boolean/False
Car0
available: boolean/False
inCharge: boolean/False
inMaintenance: boolean/False
locked: boolean/True
reserved: boolean/True
Car1
available: boolean/False
inCharge: boolean/True
inMaintenance: boolean/True
locked: boolean/False
reserved: boolean/False
surname
password
paymentMethod
name
email
address
reservations
creator
reservation
reservedCar
reservations
creator
reservation
reservedCar
pin
Surname
Password
PIN
PaymentMethod
Name1
Email1
Addr1
Reservation1
active: boolean/True
SafeArea
powerGrid: boolean/True
position
position
Reservation0
active: boolean/False
requestUnlock
availableSpots
numberOfSpots
coordinates
requestUnlock
UnlockRequest0
correctUserPosition: boolean/False
unlockOutcome: boolean/False
4
6
GPS
UnlockRequest1
correctUserPosition: boolean/True
unlockOutcome: boolean/True

Car1
available: boolean/True
inCharge: boolean/True
inMaintenance: boolean/False
locked: boolean/True
reserved: boolean/False
Car0
available: boolean/True
inCharge: boolean/False
inMaintenance: boolean/False
locked: boolean/True
reserved: boolean/False
RegisteredUser
suspended: boolean/True
Operator
SafeArea1
powerGrid: boolean/False
SafeArea0
powerGrid: boolean/True
Reservation1
active: boolean/False
Reservation0
active: boolean/False
UnlockRequest0
correctUserPosition: boolean/False
unlockOutcome: boolean/False
UnlockRequest1
correctUserPosition: boolean/True
unlockOutcome: boolean/True
GPS0
GPS1
2
3
5
6
Addr0
Addr1
Email0
Email1
Password
PaymentMethod
PIN
Name0
Name1
Surname
reservation
reservedCar
reservations
creator
position
coordinates
availableSpots
numberOfSpots
requestUnlock
password
address
email
name
surname
paymentMethod
pin

4.4 Software and tools used

- Git (https://github.com/) : for the version controlling of files shared between the team.
- Slack (https://slack.com/): used for team communication.
- GoogleDocs (https://www.google.it/intl/it/docs/about/): to write this document.
- Astah Professional (http://astah.net/editions/professional): to create all the UML diagrams.
- Alloy Analyzer (http://alloy.mit.edu/alloy/): to construct a model of part of our S2B and to prove its consistency.
- Lyx (http://www.lyx.org/): to format this document.
- JustInMind (https://www.justinmind.com/): to create the mockups.
- Android Device Art Generator (https://developer.android.com/distribute/tools/promote/device-art.html): to insert the mockups in an Android phone frame.

4.5 Hours of work

Total hours of work for the RASD creation:

- Stefano Brandoli: 38 hours
- Silvia Calcaterra: 34 hours
- Samuele Conti: 34 hours