POLITECNICO
MILANO 1863

Software Engineering 2: "PowerEnJoy"

Requirements Analysis and Specification Document

Version 1.0

Piccirillo Luca - 790380
Zampogna Gian Luca - 863097
Zini Edoardo - 875275

November 13th, 2016

Contents

1 Introduction

1.1 Purpose

This document is intended for an audience of both system developers and stakeholders. It contains the description of all the requirements the customer asked to be covered by the implementation, all the constraints the system has to deal with, and a variety of scenarios describing system interactions and features.

1.2 Scope

This document contains a description of what the digital management system has to implement in order to support an electric-only car-sharing service called PowerEnJoy, providing all basic car-sharing services plus some additional special pricing policies.

1.3 Definitions, Acronyms, Abbreviations:

1.3.1 Acronyms and Abbreviations

- GPS = Global Positioning System
- PEJ = PowerEnJoy
- PPP = Payment Processor Provider
- PU = PowerUser
- SBL = Service Business Logic

1.3.2 Definitions

- **Car:** every vehicle, which respects the requirements, that the system allows the users to use.
- **Current Fare:** amount of money that the user would pay if the ride ended in that moment. It does not include any discount or penalty for any other specific condition.
- **PowerEnJoy:** name of the service for which the management software to be developed is described in this document; we are going to use this name from now on referring to the software application itself instead of the service. Also referred to from now on as "system" or "platform".
- **PowerUser:** user who is already registered to the service and is currently logged in.
- **Safe Parking Area:** pre-defined areas (i.e. streets) where a user is allowed to park.

- **Service Business Logic:** software logic of the service the user do not directly interact with.
- **Special Parking Area:** pre-defined areas (i.e. streets) where a user is allowed to park and where the batteries of the Cars can be plugged into the power grid.
- **Total Base Fare:** amount of money that the user pays for the ride duration only. It does not include any discount or penalty for any other specific condition.
- **Total Ride Fare:** amount of money that the user pays. It includes any discount or penalty for any other specific condition satisfied during the ride.
- **Visitor:** user who is not logged in the system.

1.4 Reference

- Project's Assignment document: AA 2016-2017 Software Engineering 2 - Project goal, schedule, and rules.
- IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications.
- UML Use case diagram PDF from the course "Sistemi Informativi" of Politecnico of Milan.

1.5 Overview

This document is split in 4 part:

- **Overall Description:** contains general information about the features of the system, constraints, assumptions.
- **Scenarios:** contains an informal description of relevant or typical system usage scenarios.
- **Specific Requirements:** contains the detailed descriptions of all the interfaces and requirements to guarantee goals coverage, including relative domain properties.
- **Appendix:** contains the UML class diagram of the system and Alloy models of relevant world entities.

2 Overall Description

2.1 Product perspective

The product to be developed in order to support PowerEnJoy service consists of both a mobile application and an onboard ride assistant. The first, independently whether it will be implemented as standalone application or as web-based application, lets the customer perform all required interactions like registering to the service, searching for a Car, and reserving it. The second is intended to assist the customer up to the end of his ride. Customers will only need to interact with those two system components to access and make use of the service.

2.1.1 Quality of Service

- **Availability:** the system does not have any availability requirement.
- **Exceptions Handling:** the system must be able to handle exceptions in case of unsatisfying domain properties e.g. Credit card without enough money,...
- **Reliability:** the system does not have any reliability requirement.
- **Scalability:** the system must be able to properly manage an increasing number of users, or an unexpected peak of the number of request.
- **Security:** the system must protect user data and ensure that online secure connections take place among the service, the Cars, the user and the payment processor provider the system relies on.
- **Usability:** the interface will be user-friendly (i.e. it will always shows the user only the actions he can actually perform).

2.2 Product functions

For better orienteering through this document, here is a main functionality summary.

- Available Cars map explorer
- Personal account management
- Car reservation assistant
- Onboard ride assistant
- Automatic payment processing
- Remote vehicle interfacing

2.3 User characteristics

All kinds of user interaction with the system start from the mobile application or from the onboard ride assistant. A Visitor can only have a preview of the Cars availability just by opening up the application. All other functions require the user to sign in the service as PowerUser.

2.4 Constraints

2.4.1 Regulatory policies

The system must satisfy existing privacy regulations about users' sensible data.

2.4.2 Interfaces to other applications

The system must be compatible with payment API of the existing payment processor provider.

2.4.3 Safety and security considerations

The system must not disclose customers private service-related information without the appropriate permissions.

2.5 Assumptions and dependencies

- Cars belonging to PowerEnJoy service have a clearly recognizable logo.
- Cars are already equipped with onboard infotainment device which is able to provide basic offline navigation services in case of missing Internet connectivity.
- All Cars have an interface that is able to provide any kind of data from all sensors available in the car itself.
- Availability of the following sensors (or functional equivalents) is assumed: GPS receiver, battery status, core vehicle diagnostic.
- Onboard navigation software will not be developed, an existing solution will be integrated.
- Predefined Safe Parking Areas are assumed to be under mobile data connectivity service coverage.
- Special Parking Areas and exclusive power sockets allocation are pre-determined in such a way that spreading Cars among all the available sockets and filling them all corresponds to the best possible distribution of vehicles.
- User's device used to access the service via mobile app is able to send geolocation data while nearby the reserved Car (i.e. both data connectivity and GPS sensors are functional).

2.6 Goal Summary

[G1] Allow any kind of user to view the map of the available nearby Cars.

[G2] Allow Visitor user to register to the service.

[G3] Allow Visitor user to log-in and out as a PowerUser.

[G4] Allow PowerUser to check the status of the Car.

[G5] Allow PowerUser to reserve a Car.

[G6] Allow PowerUser to cancel a reservation.

[G7] Allow PowerUser to check the position of the reserved car.

[G8] Allow PowerUser to unlock and enter the Car when inside the specific range.

[G9] Allow PowerUser to get driving directions to his destination.

[G10] Bill the PowerUser for the amount of time spent riding a Car.

[G11] Allow PowerUser to see a list of the closest Special Parking Areas to his destination.

[G12] Allow PowerUser to keep track of the current charged fare.

[G13] Allow PowerUser to check whether he can be eligible for any discount or penalty.

[G14] Allow PowerUser to get a money saving alternative destination.

[G15] Allow the system to lock the Car in a Safe Parking Area at the end of the ride.

[G16] Allow the system to apply penalty or discount according to the given criteria.

[G17] Let the system bill the PowerUser for the total ride fare and issue a payment request for that amount at the end of the ride.

3 Scenarios

3.1 Scenario 1: Typical usage situation

Al is a student, he has to attend a lecture at university, however public transport is on strike, so he decides to rent a Car. He installs the PowerEnJoy app on his mobile phone, he opens it and looks for a nearby Car on the map. He find one at 200 meters from his home. At this point he decides to register to the PEJ service fulfilling all camps with his identity, email, contact info, driving license number and expiration, privacy agreement confirmation, credit card number and expiration, billing identity. He receives the mail with the password, he logs in as PowerUser, and he reserves the previously seen Car. In 20 minutes he is near the car. He opens the app to confirm his proximity to the Car, the system let him ask the unlocking of the Car so that Al can enter, insert his destination on the screen of the Car, start the engine and drive towards his destination. During the ride he can keep the battery level and the current fare under control watching them on Car screen. Once arrived he stops, parks and since he read on the Car screen that he is inside a Safe Parking Area he exits and leaves the Car. The Car is locked automatically and the total charge will be deducted from his payment method. Al receives a mail containing the details about the duration of his ride and the corresponding Total Ride Fare.

3.2 Scenario 2: No Car available

John has just left a party, his only wish is to get home as soon as possible but unfortunately he finds out that his car does not start. Looking around he sees a Car with the PowerEnJoy logo on the side. He googles it and find the PowerEnJoy app. After opening it, he starts looking for that Car in order to reserve it, however he discovers that it is not shown on the map, so it can’t be rented. At that point he gives up and falls back to the night service of the local public transport.

3.3 Scenario 3: Car pooling discount

Jack is an architect, and he is expected to attend a meeting located in the other side of the city. The day before he finds out that due to the high pollution level his old car is not authorized to be used. Since he needs to carry a lot of big papers with him, he does not want to use the public transport. He decides to rent a car using the PowerEnJoy service. Furthermore he proposes to a neighbour to share the ride, so since both Jack, his wife and his neighbour need to reach the same destination, they can get a discount. The following day Jack opens the PowerEnJoy app, selects the closest Car and, with both his wife and his neighbour, reaches his destination and parks. They leave the Car, and since there were at least three people from the beginning of the ride up to the end, the system applies a discount before deducting the charge. Jack receives a mail containing the details about duration of the ride, discount, and the Total Ride Fare.

3.4 Scenario 4: Charged ($\geq$50%) battery discount

Elizabeth is attending the local gym, she doesn't live too far from it, but she is not comfortable walking alone in the evening. So she decides to take advantage of the PowerEnJoy service, since there is a Special Parking Area few meters from her home. Once arrived to the gym and stopped the Car, she finds out that she has consumed far less than half of the battery. Since she is eligible for a discount the system applied it before deducting the Total Ride Fare.

3.5 Scenario 5: Money saving option

Victor is an hard worker, he has just moved from another city and he hasn't bought a car yet since he need to save as much money as possible. He has just discovered the existence of the PowerEnJoy service and its money saving option. He opens the app, completes the registration and reserves a near Car. Victor gets in the Car before his reservation expires and inserts his destination on the screen of the Car. He enables the money saving option and the navigator shows the Special Parking Areas where he is expected to park in order to get a discount. Victor considers that solution suitable for his needs, so he starts his ride. The navigator drives him to the Special Parking Area, Victor parks the Car and exits it. Within the prescribed time limit he plugs the Car into the power grid, this way the discount is applied.

3.6 Scenario 6: Reservation cancelled

Elena is working as babysitter looking after the son of her friends. She left her friends' house only after the child's parents are back, and since they are very punctual people she has already reserved a Car using the PoweEnJoy app. Unfortunately, due to a traffic jam, they are late; after they informed her, she opens the PowerEnjoy app and cancels her reservation.

3.7 Scenario 7: 1€ fee

Nathan and Samuel are two brothers. They reserved a Car in order to go to the cinema. They are leaving home when a water pipe breaks and starts pouring water everywhere in the kitchen. They immediately close the water, but since there is water everywhere they decide to stay home and clean up the wet floor. They forget their reservation, so after the established hour the system deducts the 1€ penalty and sends an email to Nathan, who is the PowerUser, noticing him about it.

4 Specific Requirements

4.1 External interface requirements

4.1.1 User interfaces

There are two user interfaces in our system: one for the smartphone used to access the service; one for the onboard infotainment device of the Car.

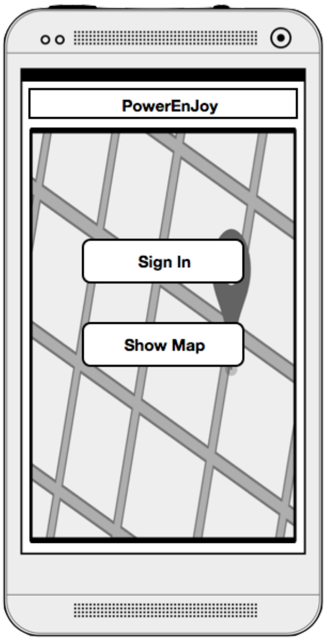
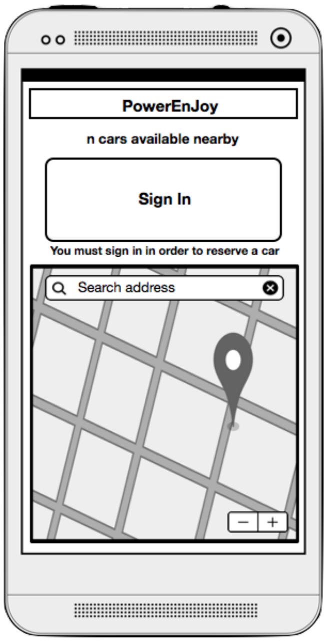


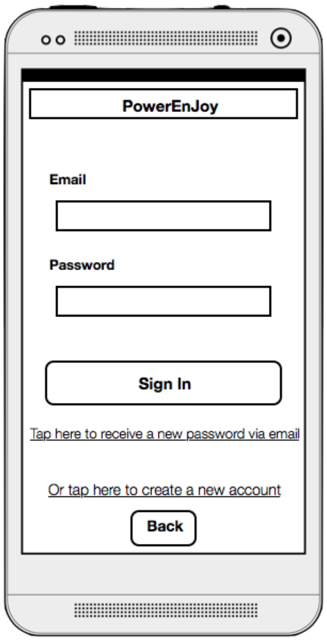
Figure 1: From the first screen a user can either sign in, and if necessary create a new account, or consult the map of available Cars.



n cars available nearby

You must sign in in order to reserve a car

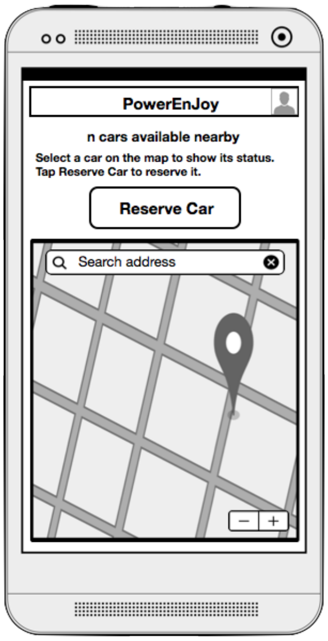
Figure 2: If a user chooses the map, he will see a map with the positions of all the available Cars; he is asked to sign in in order to proceed with the reservation.



Tap here to receive a new password via email

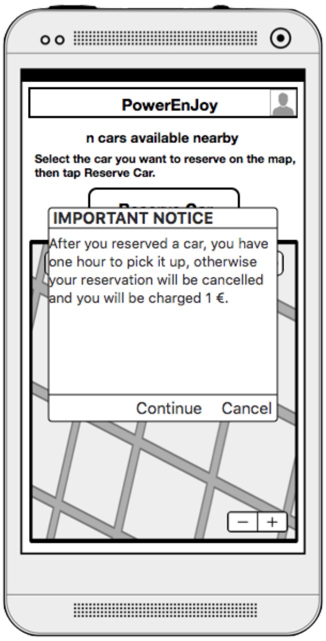
Or tap here to create a new account

Figure 3: Whether a user started from the map or from the first screen, he will have to either enter his credentials or to register. In case of forgotten credentials he is given the possibility to ask for a new password.



Select a car on the map to show its status.
Tap Reserve Car to reserve it.

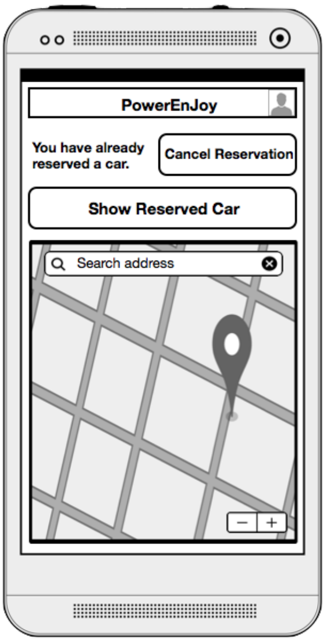
Figure 4: A PowerUser can look for a suitable Car nearby his position or entering an address in the appropriate bar above the map. After finding a Car the PowerUser can select it to know the residual battery and to reserve it.



Select the car you want to reserve on the map, then tap Reserve Car.

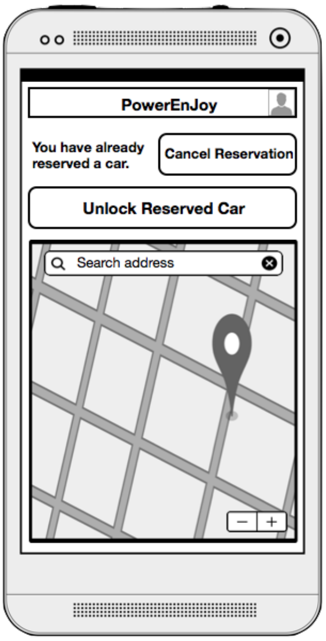
After you reserved a car, you have one hour to pick it up, otherwise your reservation will be cancelled and you will be charged 1 €.

Figure 5: Before actually reserving a Car, the system reminds the PowerUser about the one hour expiration of his reservation and the relative penalty fee.



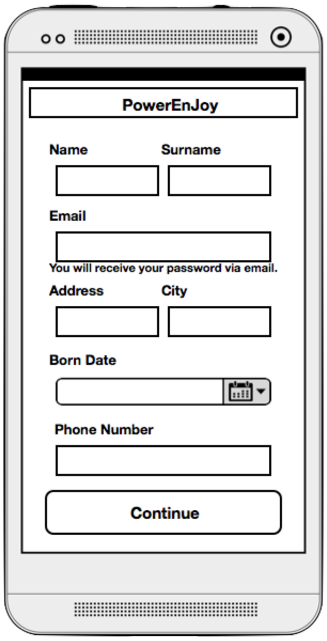
You have already reserved a car.

Figure 6: After the selected Car has been reserved, a PowerUser can look for his Car thanks to the provided map or cancel his reservation. In this case a confirmation box pops up before actually cancelling it.



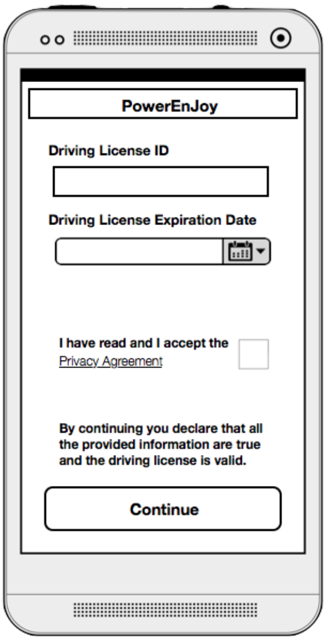
You have already reserved a car.

Figure 7: When a PowerUser is near his reserved Car, he is given the possibility to ask the system to unlock it. If the PowerUser changes his mind he can still cancel the reservation, this possibility will remain valid until the PowerUser ignites the engine.



You will receive your password via email.

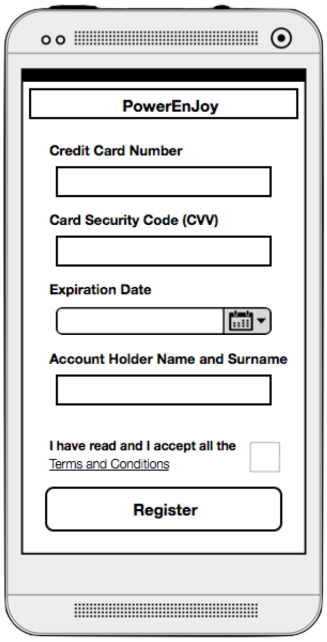
Figure 8: This is the first page of the registration procedure; it requires a new user to enter his personal data.



I have read and I accept the Privacy Agreement □

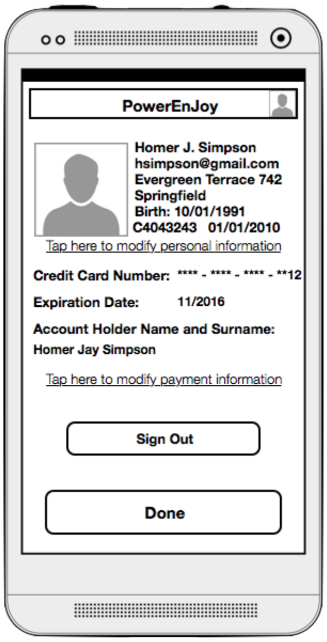
By continuing you declare that all the provided information are true and the driving license is valid.

Figure 9: After entering persona data the user is required to provide his driving license details and to accept the Privacy Agreement.



I have read and I accept all the Terms and Conditions □

Figure 10: Last page of the registration procedure, the user is asked for his payment information. Furthermore he has to accept the Terms and Conditions of the service in order to complete the registration. These Terms and Conditions contain all the legal constraints required by law, and ensure that any misbehaviour perpetrated by a user is his own solely responsibility and cannot be ascribed to the company running the PowerEnJoy sharing service.



Homer J. Simpson
hsimpson@gmail.com
Evergreen Terrace 742
Springfield
Birth: 10/01/1991
C4043243 01/01/2010

Tap here to modify personal information

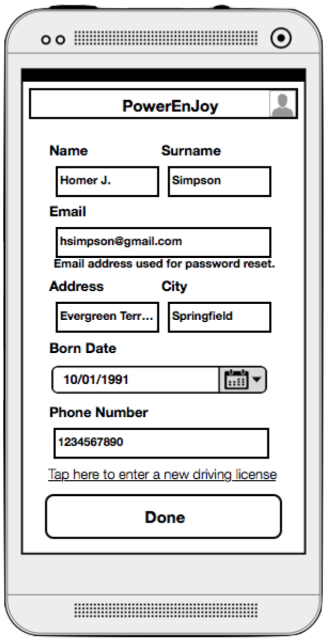
Credit Card Number: **** - **** - **** - **12

Expiration Date: 11/2016

Account Holder Name and Surname:
Homer Jay Simpson

Tap here to modify payment information

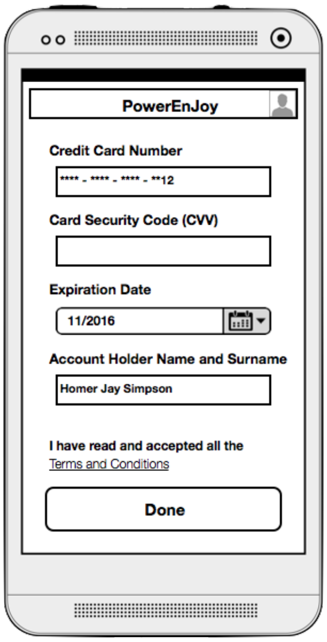
Figure 11: A PowerUser can access his personal account page by tapping on the avatar in the top-right corner. By doing so, this is what he will be shown. From this page it is also possible to modify both personal and payment information, or to sign out.



Name: Homer J.
Surname: Simpson
Email: hsimpson@gmail.com
Email address used for password reset.
Address: Evergreen Terr...
City: Springfield
Born Date: 10/01/1991
Phone Number: 1234567890

Tap here to enter a new driving license

Figure 12: This is the page to be used by a PowerUser in order to modify his personal information.


PowerEnJoy

Credit Card Number

**** - **** - **** - **12

Card Security Code (CVV)

Expiration Date

11/2016

Account Holder Name and Surname

Homer Jay Simpson

I have read and accepted all the
Terms and Conditions

Figure 13: This is the page to be used by a PowerUser in order to modify his payment information.

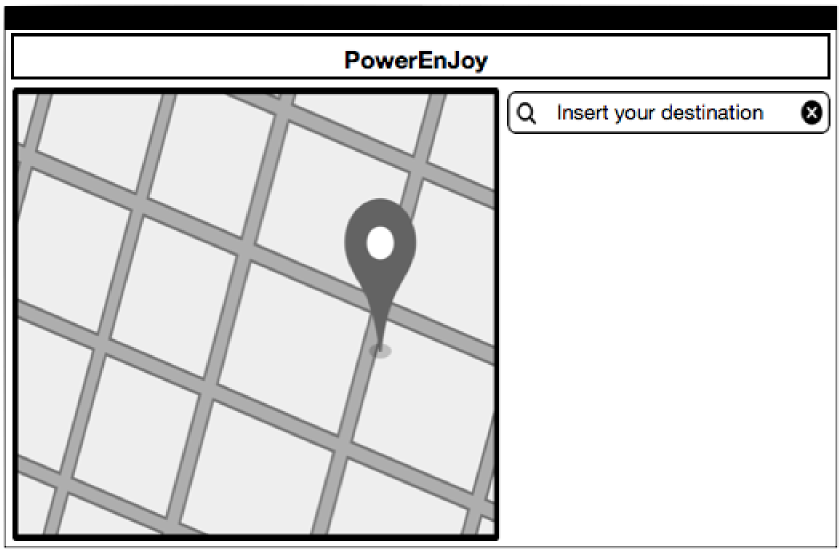


Figure 14: This is the first screen a PowerUser will see entering in a Car. The map on the left is a placeholder for the third party navigation software.

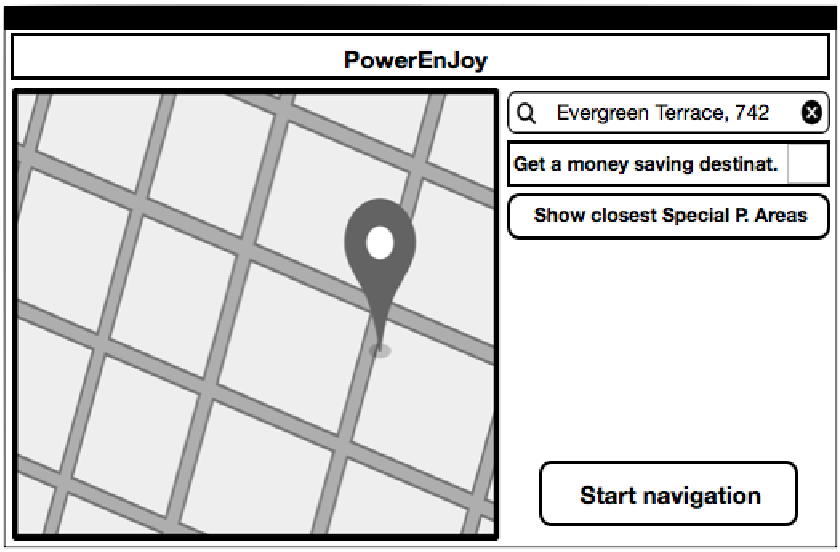
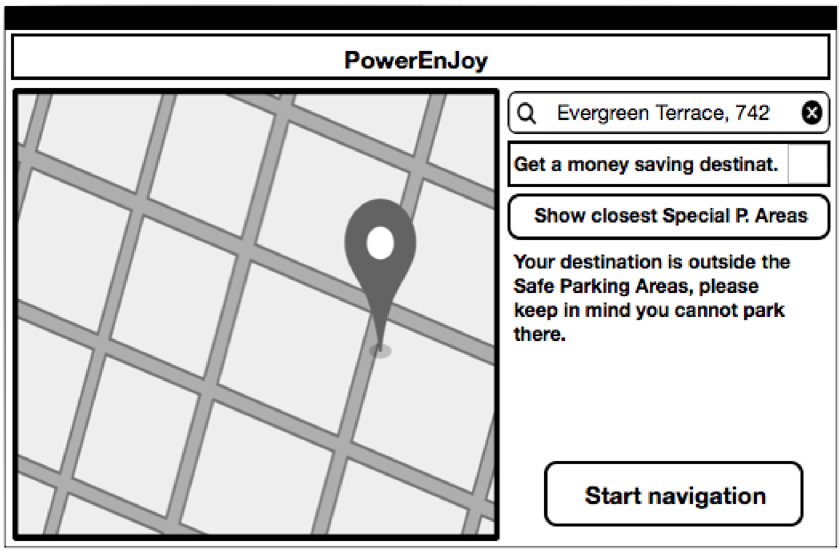
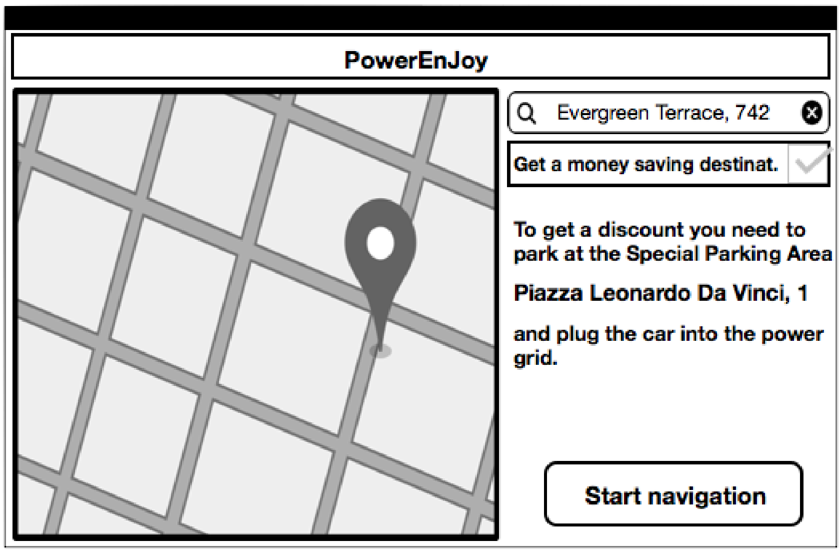


Figure 15: A PowerUser searches for his destination and if it is available in the navigation software he is allowed to start the navigation. Furthermore, he can enable the money saving option using "Get a money saving destination", or ask for the list of the closest Special Parking Areas to his destination.



Your destination is outside the Safe Parking Areas, please keep in mind you cannot park there.

Figure 16: A PowerUser searches for his destination and if it is not inside the set of Safe Parking Areas he is reminded he cannot park in that position. If he wants he can enable the money saving option using "Get a money saving destination", or ask for the list of the closest Special Parking Areas to his destination.


PowerEnJoy

Evergreen Terrace, 742

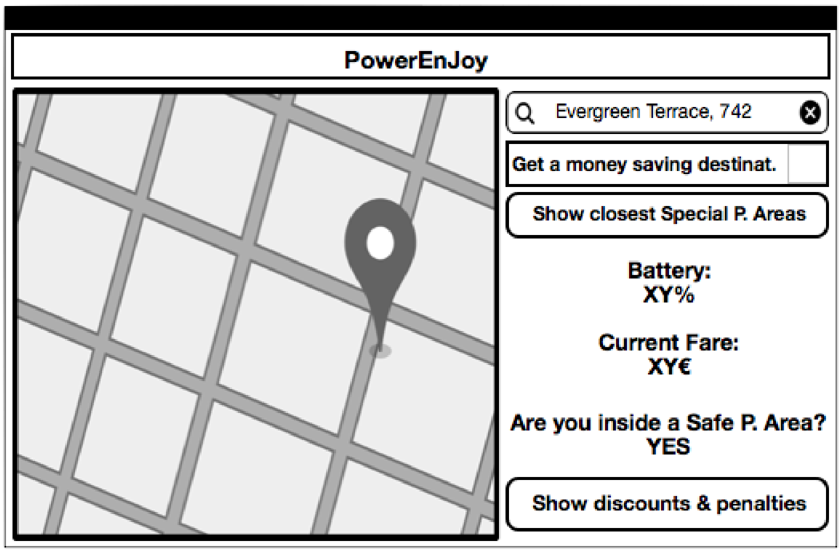
Get a money saving destinat. ☑

To get a discount you need to park at the Special Parking Area

Piazza Leonardo Da Vinci, 1

and plug the car into the power grid.

Figure 17: When a PowerUser enables the money saving option he is provided with the address of the Special Parking Area where he is supposed to park and plug the car into the power grid in order to get the discount.


PowerEnJoy

Evergreen Terrace, 742

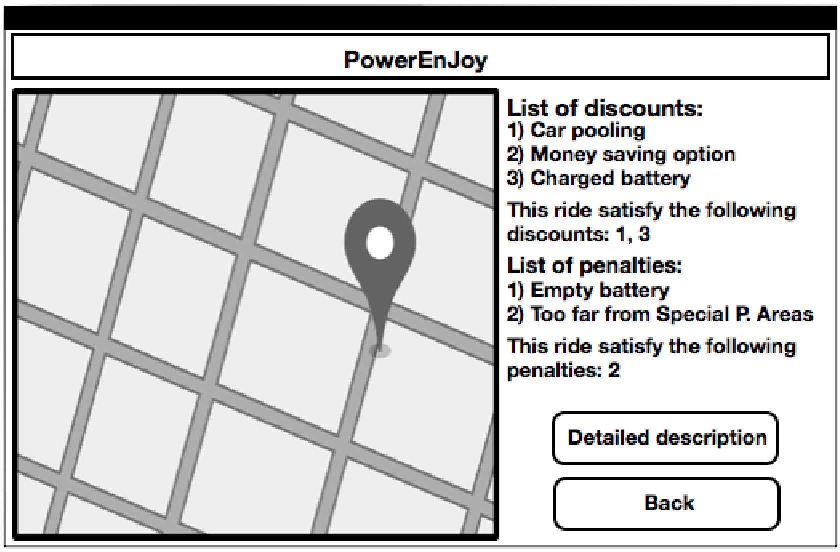
Get a money saving destinat. □

Battery:
XY%

Current Fare:
XY€

Are you inside a Safe P. Area?
YES

Figure 18: This is the screen the PowerUser sees while he is driving. If he needs to change the destination he is free to do that, if he wants to know more about discounts and penalties he can select the apposite button, if he decides to plug the car into the power grid he can ask a money saving destination or the list of the closest Special Parking Areas to his destination.



List of discounts:
1) Car pooling
2) Money saving option
3) Charged battery

This ride satisfy the following discounts: 1, 3

List of penalties:
1) Empty battery
2) Too far from Special P. Areas

This ride satisfy the following penalties: 2

Figure 19: This is the screen that sums up all discounts and penalties, and it also declares explicitly which one is valid at the moment. If a PowerUser needs a more detailed description he just have to select the appropriate button to show a window containing all the details and the explanations of each discount and penalty.

4.1.2 Hardware interfaces

The system interfaces directly with sensors and actuators of the Cars, like GPS, door locking and unlocking mechanism, battery sensor, power socket detector.

4.1.3 Communication interfaces

The system requires Internet connectivity across all involved devices.

4.1.4 Software interfaces

The system relies on a third party payment processor provider to issue payment requests, and a third party navigation software for the onboard Car equipment which provides driving directions.

4.2 Functional requirements

4.2.1 Goal 1

[G1] Allow any kind of user to view the map of the available nearby Cars.

- Requirements

  [R1] Users must have access to a map indicating the user current location.

  [R2] Users must be able to pan and scroll a map in any direction.

[R3] Every available Car must be shown on a map.

[R4] Users shall have an input for inserting an address in which center the map.

[R5] Users shall have an input to center the map view on their position.

- Domain properties

[P1] B% and C% are remaining battery power percentages defined during setup.

[P2] No Car with less than B% battery charge is taken into account as available for booking.

[P3] No Car that is currently under charging and has less than C% battery charge is taken into account as available for booking.

4.2.2 Goal 2

[G2] Allow Visitor user to register to the service.

- Requirements

[R6] Let a Visitor user start the registration wizard while he's still not logged in.

[R7] The registration form must contain input fields for user identity, email, contact info, driving license number and expiration, privacy agreement confirmation, credit card number and expiration, billing identity.

[R8] The chosen email address must not be already used by another PowerUser.

[R9] The credit card must be verified not to be blocked or expired upon registration.

[R10] The user must receive a system generated password to the registered email address.

- Domain properties
  - None

4.2.3 Goal 3

[G3] Allow Visitor user to log-in and out as a PowerUser.

- Requirements

[R11] A Visitor user must always see an input to access log-in form as long as he's still not logged in.

[R12] An input to perform log-out must always be available to the PowerUser if he's currently logged in.

- Domain properties
  - None

4.2.4 Goal 4

[G4] Allow PowerUser to check the status of the Car.

- Requirements
  - [R13] For each available Car the PowerUser must be able to view its remaining battery charge.
  - [R14] For each available Car the PowerUser must be able to view its current position.
- Domain properties
  - None

4.2.5 Goal 5

[G5] Allow PowerUser to reserve a Car.

- Requirements
  - [R15] The PowerUser must have the ability to start the reservation wizard for all and only available Cars.
  - [R16] The PowerUser must see a reminder about unfulfilled reservation penalty before confirmation.
  - [R17] Show an input to allow the PowerUser to confirm and finalize the reservation.
  - [R18] The system shall prevent the PowerUser to reserve more than a Car from the same geographical region at a time.
- Domain properties
  - [P4] Each geographical region has its own set of assigned Cars.

4.2.6 Goal 6

[G6] Allow PowerUser to cancel a reservation.

- Requirements
  - [R19] If a reservation exists for the PowerUser, show him an input to request cancellation.
  - [R20] The system shall prompt the PowerUser for cancellation confirmation.
  - [R21] A reservation must be automatically cancelled by the system after 1 hour from its creation.
  - [R22] If a reservation is cancelled because of timeout, notify the PowerUser about that occurrence.
- Domain properties
  - None

4.2.7 Goal 7

[G7] Allow PowerUser to check the position of the reserved car.

- Requirements

  [R23] As long as a reservation exists for the PowerUser he must always be able to get the position of the reserved Car on the map.

- Domain properties

  - None

4.2.8 Goal 8

[G8] Allow PowerUser to unlock and enter the Car when inside the specific range.

- Requirements

  [R24] The system must be able to remotely unlock the Car.

  [R25] The system must be able to compute the distance between the user location and his reserved Car.

  [R26] The PowerUser must have an input allowing him to send an unlock request.

  [R27] The system must accept the unlock request issued by the PowerUser if and only if the PowerUser is in the unlock allowance area.

  [R28] If the unlock request is accepted, the PowerUser must be able to enter the Car.

- Domain properties

  [P5] The PowerUser position is always the same as of its mobile device that runs the PowerUser application.

  [P6] The PowerUser is considered to be in the unlocking allowance area if the distance to the Car is at most equal to 5 meters.

4.2.9 Goal 9

[G9] Allow PowerUser to get driving directions to his destination.

- Requirements

  [R29] The user must be allowed to select a custom destination and start navigating to that location.

- Domain properties

  [P7] Navigation software always provides effective directions to the user if the destination is included in the Safe Parking Areas.

4.2.10 Goal 10

[G10] Bill the PowerUser for the amount of time spent riding a Car.

- Requirements

  [R30] Start counting the billing time from the first engine ignition.

  [R31] Stop the billing time counter exactly 1 second after the Car locking.

- Domain properties

  - None

4.2.11 Goal 11

[G11] Allow PowerUser to see a list of the closest Special Parking Areas to his destination.

- Requirements

  [R32] The system must be capable of providing a list of Special Parking Areas sorted by distance from an input location.

  [R33] PowerUser must be allowed anytime during the navigation to input a custom location and be acknowledged about all nearest Special Parking Areas from the selected location.

- Domain properties

  - None

4.2.12 Goal 12

[G12] Allow PowerUser to keep track of the Current Fare.

- Requirements

  [R34] Show on the Car screen a live updated counter indicating the Current Fare amount that the user would actually pay if the ride ended in that same moment, as long as the ride is being charged.

- Domain properties

  [P8] The Current Fare starts being counted when the engine ignites for the first time and stops exactly 1 second after the Car locking.

4.2.13 Goal 13

[G13] Allow PowerUser to check whether he can be eligible for any discount or penalty.

- Requirements

  [R35] Provide through Car screen an input to access an overview of all discounts and penalties.

[R36] For each shown discount or penalty allow the PowerUser to get a brief informal description of corresponding criteria.

[R37] For each shown discount or penalty allow the PowerUser to know if the current ride satisfies all needed criteria at the moment.

- Domain properties
  - None

4.2.14 Goal 14

[G14] Allow PowerUser to get a money saving alternative destination.

- Requirements

[R38] Show the PowerUser the option to get a money saving destination alternative after entering desired destination address.

[R39] Always show to PowerUser an input to get money saving proposal even if there already exists a selected destination.

[R40] The destination proposal must correspond to the nearest (w.r.t. PowerUser selected destination) Special Parking Area where there are less than N# Cars attached to the power source.

[R41] If the PowerUser accepts the money saving destination proposal, the current selected destination must be updated accordingly.

- Domain properties

[P9] N# is an integer number defined during setup.

[P10] Special Parking Areas provide at least N# power sockets as exclusively available to PowerEnJoy customers.

4.2.15 Goal 15

[G15] Allow the system to lock the Car in a Safe Parking Area at the end of the ride.

- Requirements

[R42] The system must lock the Car if its position belong to the set of Safe Parking Areas, engine is stopped, all doors are closed and S# seconds passed from the last door closure.

- Domain properties

[P11] S# is a time-span defined during setup.

[P12] At least a free parking slot is available for the user among all Safe Parking Areas.

4.2.16 Goal 16

[G16] Allow the system to apply penalty or discount according to the given criteria.

- Requirements

  [R43] The system shall apply a discount of 10% on the last ride Total Base Fare if the number of passengers at the end of the ride is greater or equal to the number of passengers at the start of the ride and the number of passengers at the start of the ride was at least 3, driver included.

  [R44] The system shall apply a discount of 20% on the last ride Total Base Fare if the remaining battery power at the end of the ride is greater or equal then 50%.

  [R45] The system shall apply a discount of 30% on the last ride Total Base Fare if the Car position at the end of the ride is within a Special Parking Areas and the power socket is detected as connected two minutes from the Car locking.

  [R46] The system shall apply a penalty of 30% on the last ride Total Base Fare if the position of the Car at the time of locking is more than 3km far from the nearest power grid station, or the remaining battery power is less than 20% and the Car is not detected as attached to power grid within two minutes from locking.

- Domain properties
  - None

4.2.17 Goal 17

[G17] Let the system bill the PowerUser for the Total Ride Fare and issue a payment request for that amount at the end of the ride.

- Requirements

  [R47] The system must charge the PowerUser the Total Ride Fare after two minutes and an half from the Car locking.

  [R48] The system must issue the payment request to the Payment Processor Provider.

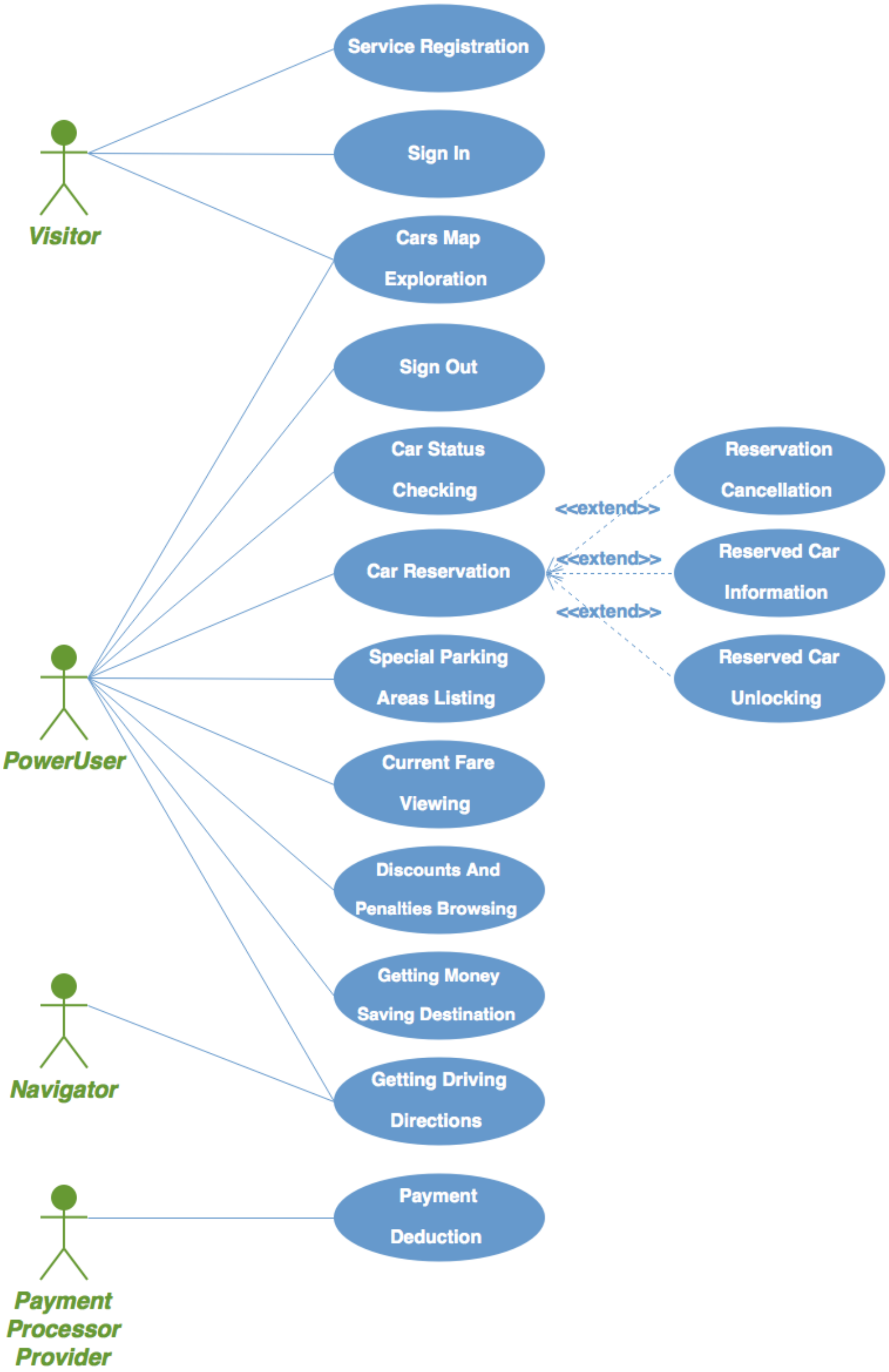
  [R49] The system must bill the PowerUser a penalty of 1€ as soon as a reservation he made expires by timeout.

  [R50] The PowerUser shall be notified of any money charge by email.

  [R51] The PowerUser shall be notified of the payment transaction result.

- Domain properties
  - None

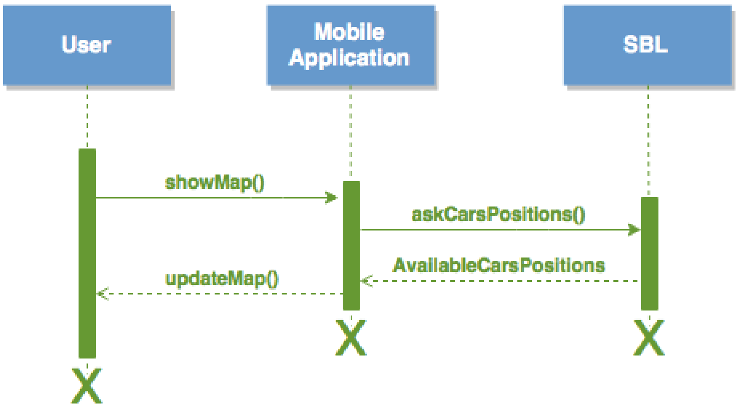
4.3 Use cases



A use case *extends* another use case when its use is optional and the use case that is extended is already reasonably complete.

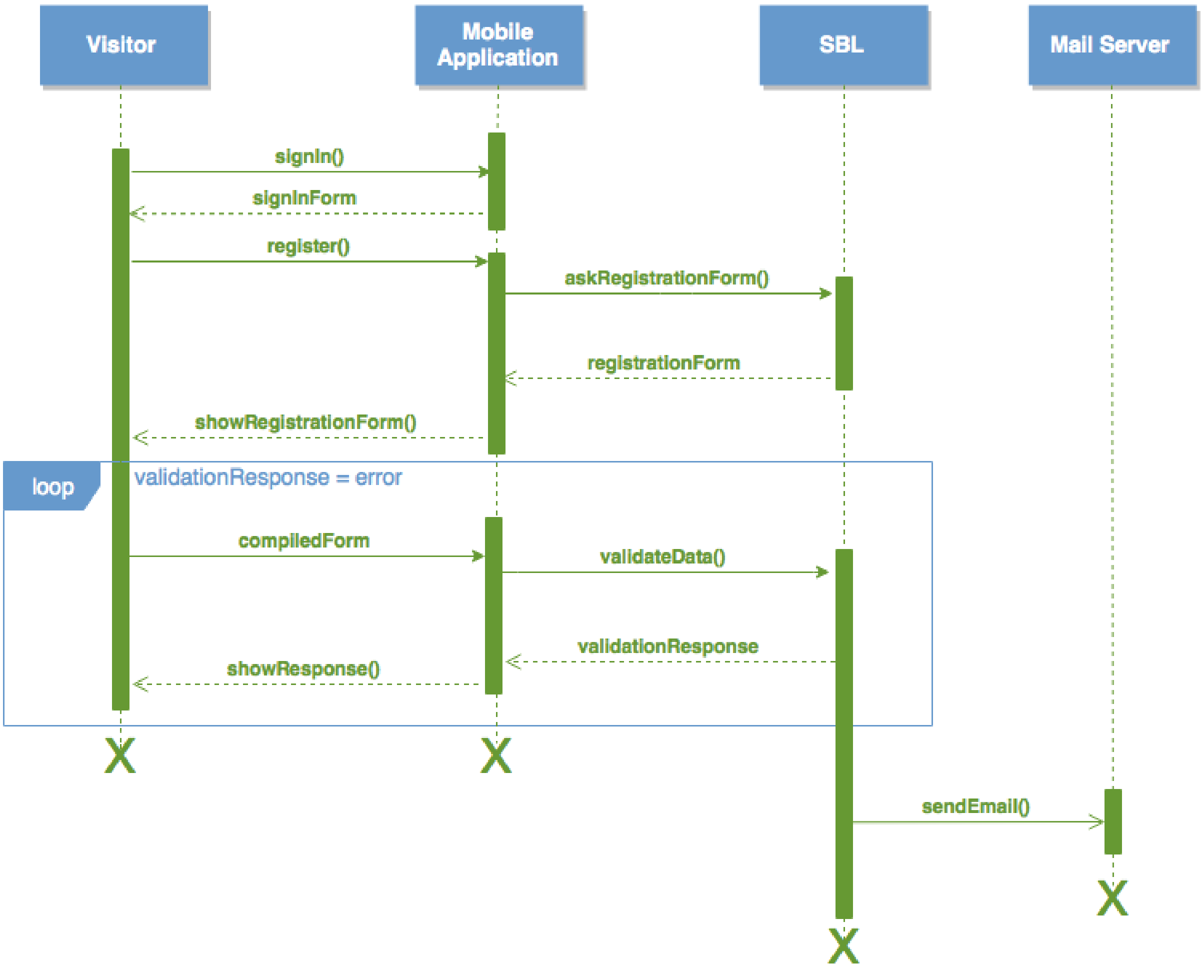
4.3.1 Cars Map Exploration

Actor	Visitor, PowerUser
Goal	[G1]
Entry Condition	NULL
Event Flow	1. The user launches the application. 2. He taps on "Show Map". 3. He sees a map with the available Cars. 4. He sees a pointer corresponding to his current position. 5. He scrolls the map in any direction.
Output Condition	NULL
Exceptions	• Cannot obtain current position. The exception is handled notifying the user about the exception.



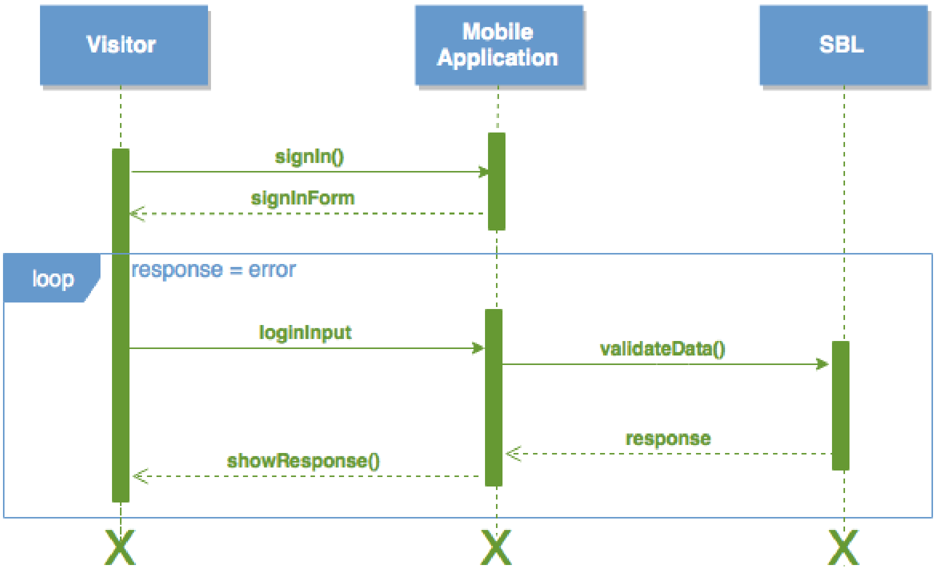
4.3.2 Service Registration

Actor	Visitor
Goal	[G2]
Entry Condition	NULL
Event Flow	1. Visitor launches the application. 2. He accesses the Sign In view. 3. He taps on the apposite string to create a new account. 3. He compiles all required fields. 4. The system sends the password to the specified email.
Output Condition	A new PoweUser account exists.
Exceptions	• Input fields validation failure. • Cannot send email to selected address. These exceptions are handled notifying the user about the exception, asking him to give new valid/working inputs.



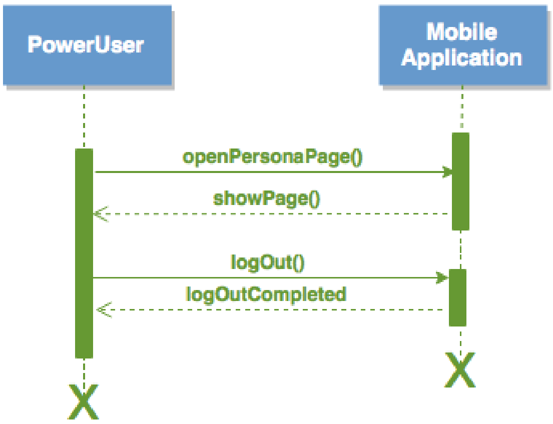
4.3.3 Sign In

Actor	Visitor
Goal	[G3]
Entry Condition	NULL
Event Flow	1. Visitor reaches the Sign In page. 2. He inputs his PowerUser's email address and his PowerUser's password. 3. He confirms tapping the "Sign In" button.
Output Condition	Visitor is now authenticated as a PowerUser.
Exceptions	• Wrong credentials. The exception is handled notifying the user about the exception, asking him to check for errors in given credentials.



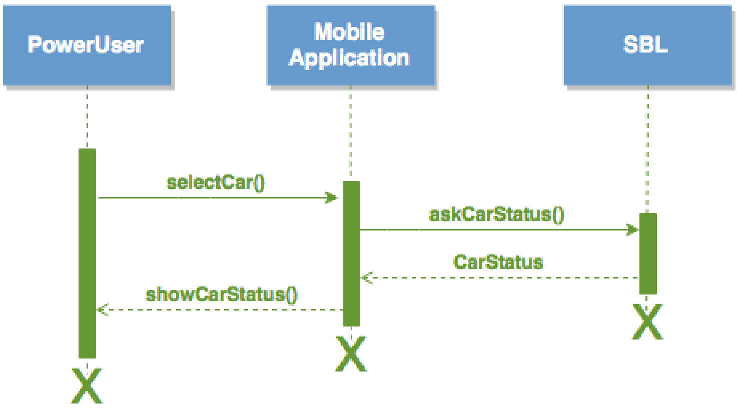
4.3.4 Sign Out

Actor	PowerUser
Goal	[G3]
Entry Condition	Existent PowerUser authenticated session.
Event Flow	1. PowerUser reaches his personal page. 2. He taps on the "Sign Out" button to deactivate the current signed-in session.
Output Condition	PowerUser is no more recognized as such, until he signs in again.
Exceptions	NONE



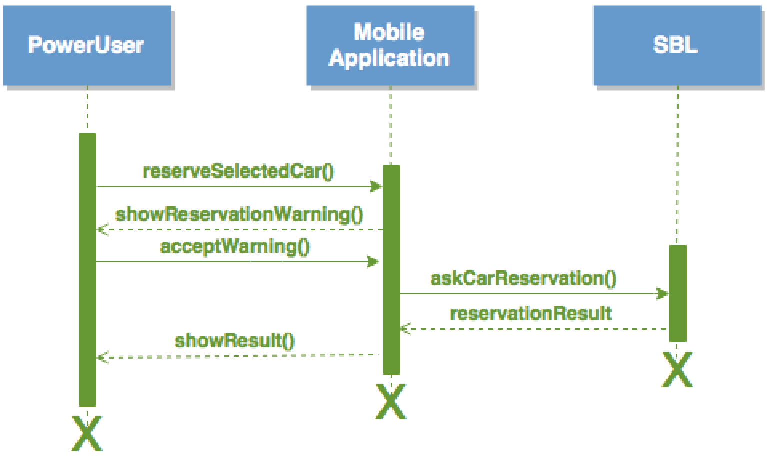
4.3.5 Car Status Checking

Actor	PowerUser
Goal	[G4]
Entry Condition	A Car is shown on the map.
Event Flow	1. The PowerUser selects an available Car on the map. 2. He gets details about the remaining battery charge.
Output Condition	PowerUser sees an input to proceed reserving that Car.
Exceptions	NONE



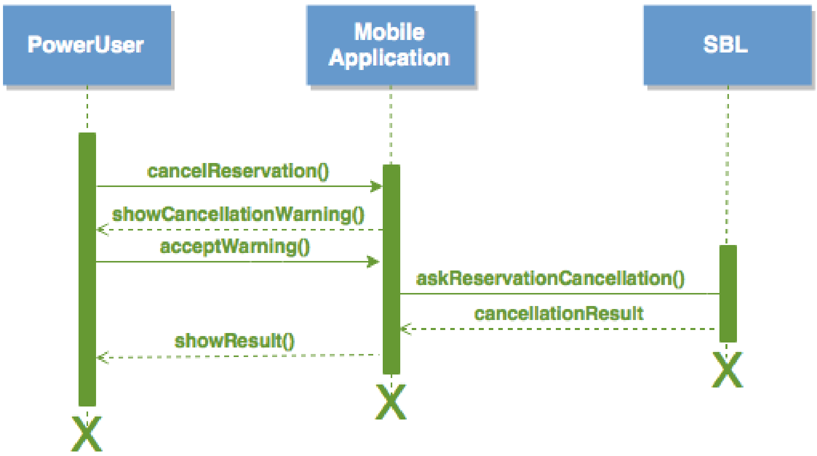
4.3.6 Car Reservation

Actor	PowerUser
Goal	[G5]
Entry Condition	PowerUser has no other active reservation.
Event Flow	1. PowerUser starts the reservation wizard. 2. He sees a warning about reservation cancellation timeout. 3. He accepts to continue reserving the Car. 4. He sees a timer to the time of reservation expiration.
Output Condition	PowerUser has a pending reservation for the selected Car.
Exceptions	• Car no longer available. The exception is handled notifying the user about the exception, asking him to choose another Car.



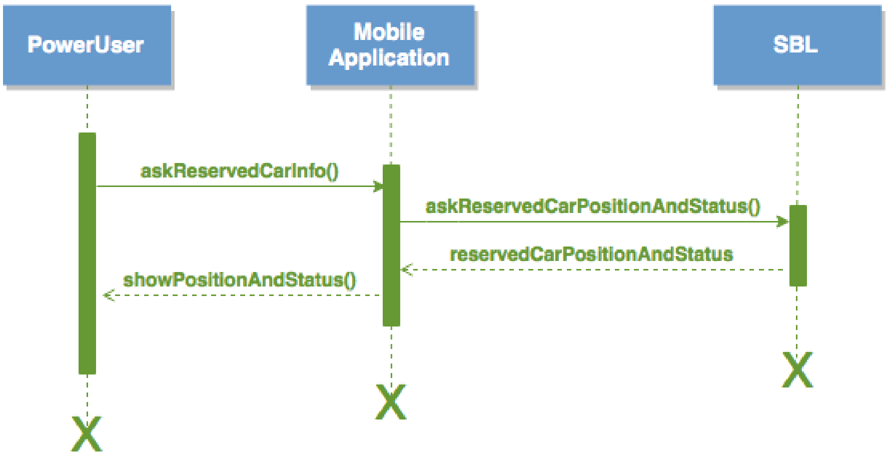
4.3.7 Reservation Cancellation

Actor	PowerUser
Goal	[G6]
Entry Condition	Existent pending Car reservation for PowerUser.
Event Flow	1. PowerUser asks to cancel the current pending reservation. 2. He sees a confirmation message. 3. He accepts to continue cancelling the reservation.
Output Condition	PowerUser has no pending Car reservation.
Exceptions	• Reservation expired meanwhile. The exception is handled aborting the cancellation wizard.



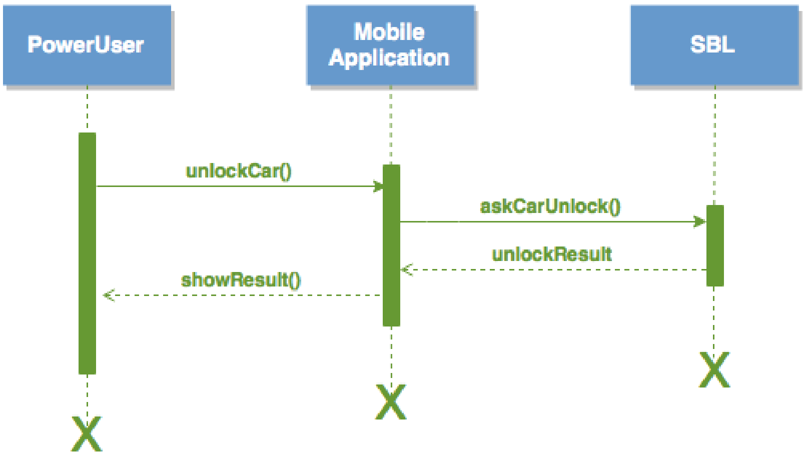
4.3.8 Reserved Car Information

Actor	PowerUser
Goal	[G7]
Entry Condition	Existent pending Car reservation for PowerUser.
Event Flow	1. PowerUser asks to show the reserved Car on the map. 2. He sees the Car on the map with its remaining battery charge.
Output Condition	NULL
Exceptions	NONE



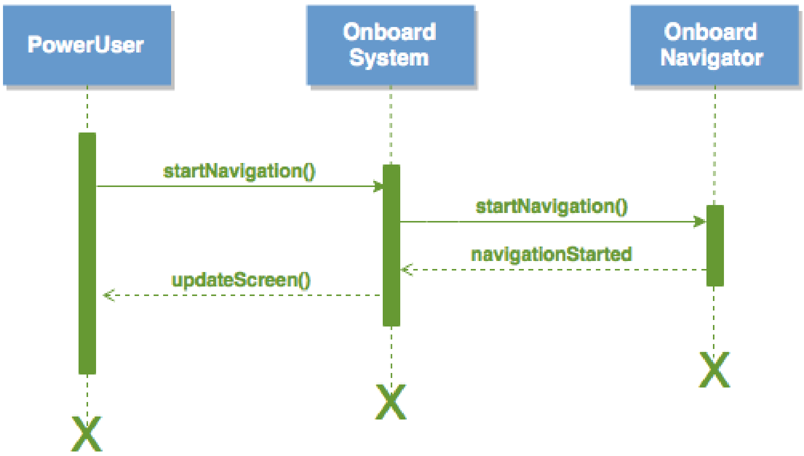
4.3.9 Reserved Car Unlocking

Actor	PowerUser
Goal	[G8]
Entry Condition	Existent pending Car reservation for PowerUser and PowerUser is inside the allowed unlocking area.
Event Flow	1. PowerUser sees an option to unlock the reserved Car. 2. He asks the Car to be unlocked. 3. The system unlocks the Car. 4. PowerUser opens the door and enters the Car.
Output Condition	Car doors are unlocked.
Exceptions	• Unable to unlock the Car. The exception is handled asking the PowerUser to retry in a few seconds.



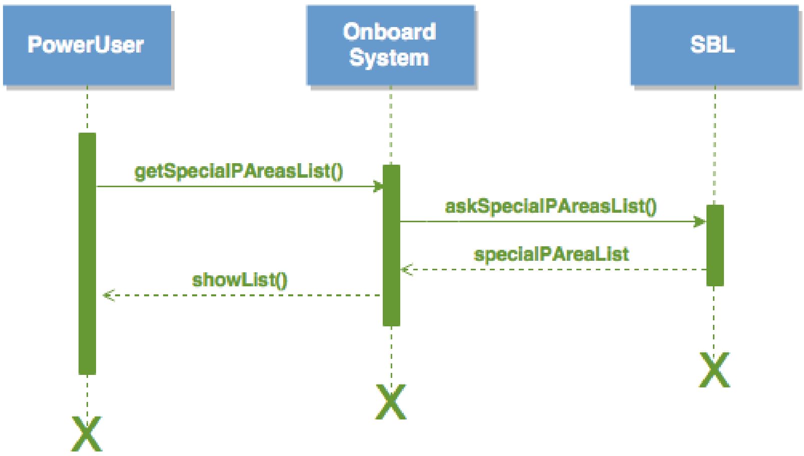
4.3.10 Getting Driving Directions

Actor	PowerUser, Navigator
Goal	[G9]
Entry Condition	There exists a selected destination.
Event Flow	1. PowerUser chooses to start navigation in order to be given directions by the onboard navigator for the previously selected destination. 2. He follows the directions.
Output Condition	The destination is reached.
Exceptions	• Cannot find driving direction for selected destination. The exception is handled asking the PowerUser to select another destination to get driving directions to which is surely reachable by car.



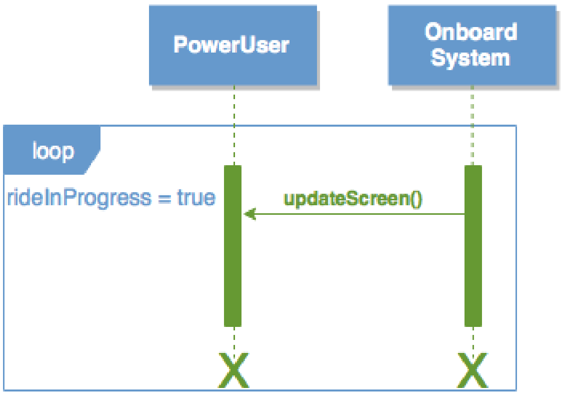
4.3.11 Special Parking Areas Listing

Actor	PowerUser
Goal	[G11]
Entry Condition	There exists a selected destination.
Event Flow	1. PowerUser asks for a list of Special Parking Areas. 2. The system lets him scroll a list of Special parking Areas sorted by distance to his selected destination. 3. PowerUser optionally selects one of those as new destination.
Output Condition	NULL or current selected destination changed.
Exceptions	NONE



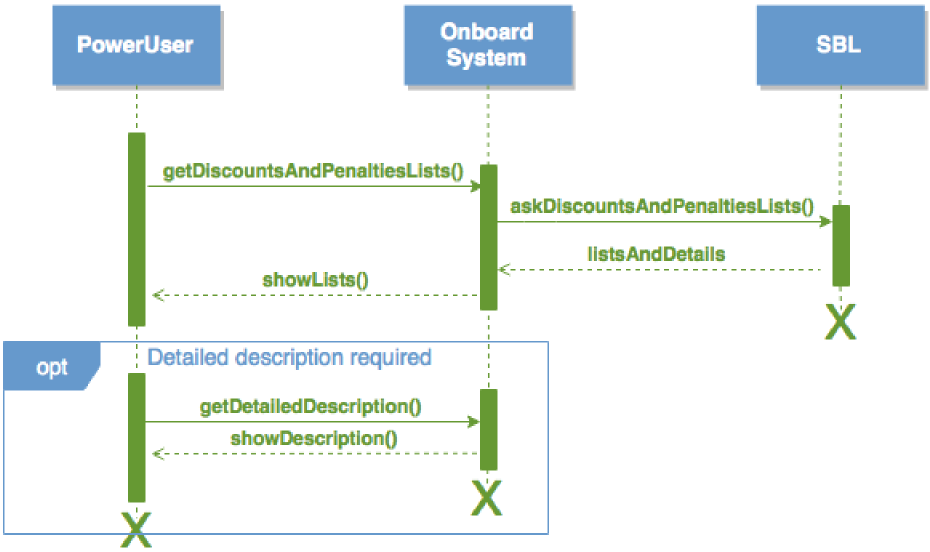
4.3.12 Current Fare Viewing

Actor	PowerUser
Goal	[G12]
Entry Condition	Ride is being currently charged.
Event Flow	1. PowerUser sees the updated Current Fare amount on the onboard screen of the Car.
Output Condition	NULL
Exceptions	NONE



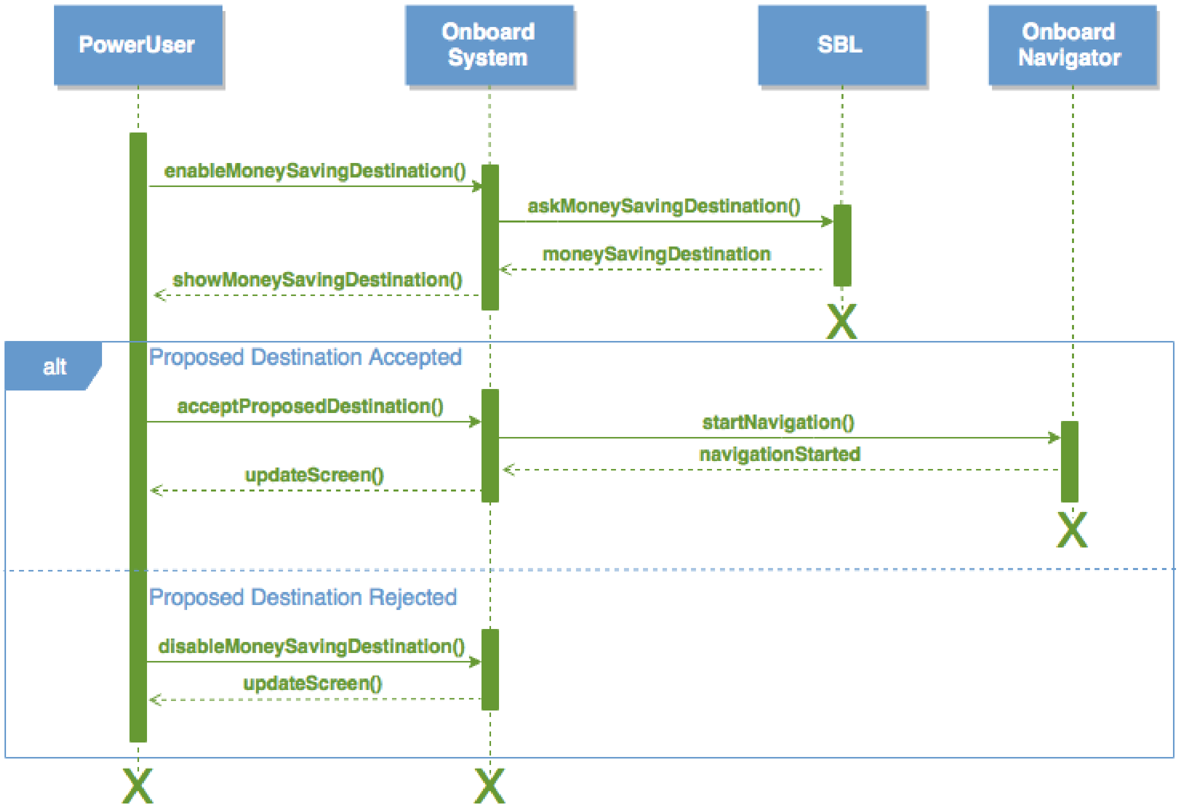
4.3.13 Discounts And Penalties Browsing

Actor	PowerUser
Goal	[G13]
Entry Condition	NULL
Event Flow	1. PowerUser opens the discounts and penalties lists. 2. He gets the lists of all existing available discounts and penalties policies. 3. He gets a visible feedback stating if those criteria yield for the current ride. 4. At the end of the lists he can ask for a detailed description.
Output Condition	NULL
Exceptions	NONE



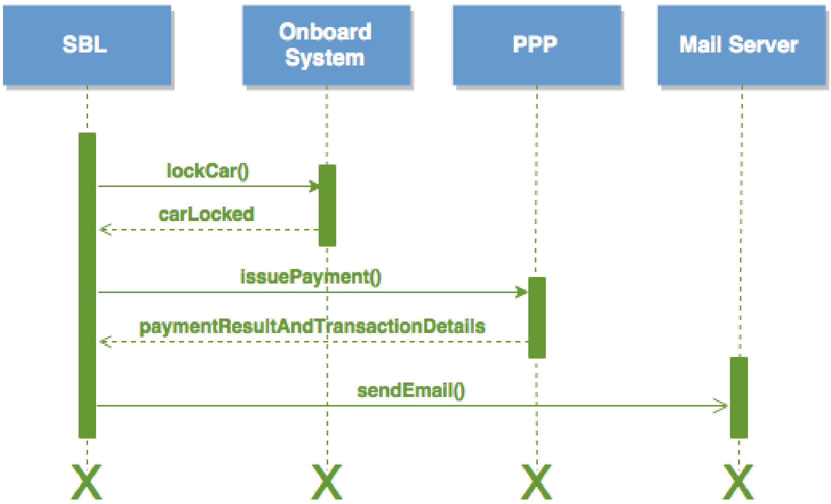
4.3.14 Getting Money Saving Destination

Actor	PowerUser
Goal	[G14]
Entry Condition	A destination is being selected or there already exists one.
Event Flow	1. PowerUsers enables the Money Saving destination. 2. The system provides him a single Special Parking Area and its location on a map. 3. PowerUser either accepts the proposal by starting the navigation or rejects it by disabling the Money Saving destination. 4. If accepted, the selected destination is updated accordingly.
Output Condition	Suggested proposal rejected or accepted.
Exceptions	NONE


PowerUser
Onboard System
SBL
Onboard Navigator
enableMoneySavingDestination()
askMoneySavingDestination()
moneySavingDestination
showMoneySavingDestination()
alt
Proposed Destination Accepted
acceptProposedDestination()
startNavigation()
navigationStarted
updateScreen()
Proposed Destination Rejected
disableMoneySavingDestination()
updateScreen()
X
X
X
X

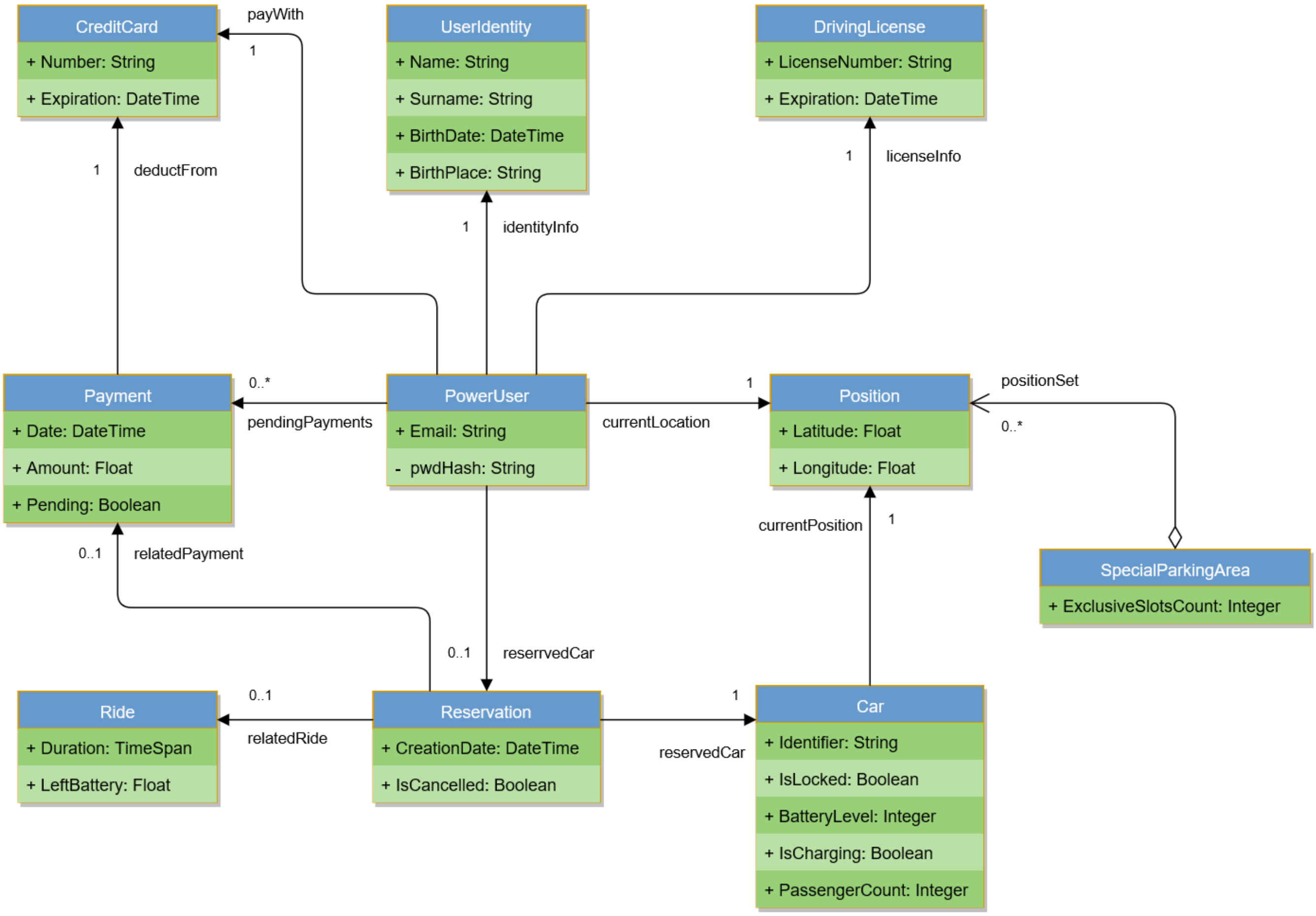
4.3.15 Payment Deduction

Actor	Payment Processor Provider
Goal	[G17]
Entry Condition	Locking grace period expired.
Event Flow	1. The system locks the Car and stops charging the PowerUser. 2. The system creates a new payment request for the ride just ended. 3. The system merges it with all pending payments for penalties or previous rides of the same account into a single comprehensive payment request. 2. The system issues the payment request to the PPP. 3. PPP responds back with operation result and transaction details. 4. The system sends an email to the PowerUser with payment result and details.
Output Condition	The amount of money has been transferred from PowerUser to PEJ.
Exceptions	• Not enough money. The exception is handled notifying the user of the issue and adding that payment request to the pending ones.



5 Appendix

5.1 Class diagram



5.2 Alloy

5.2.1 Purpose

The following are some Alloy models for system components that we believe require a formal checking apt to ensure specifications correctness with respect to the service-critical operations characterizing the PowerEnJoy platform.

5.2.2 Code

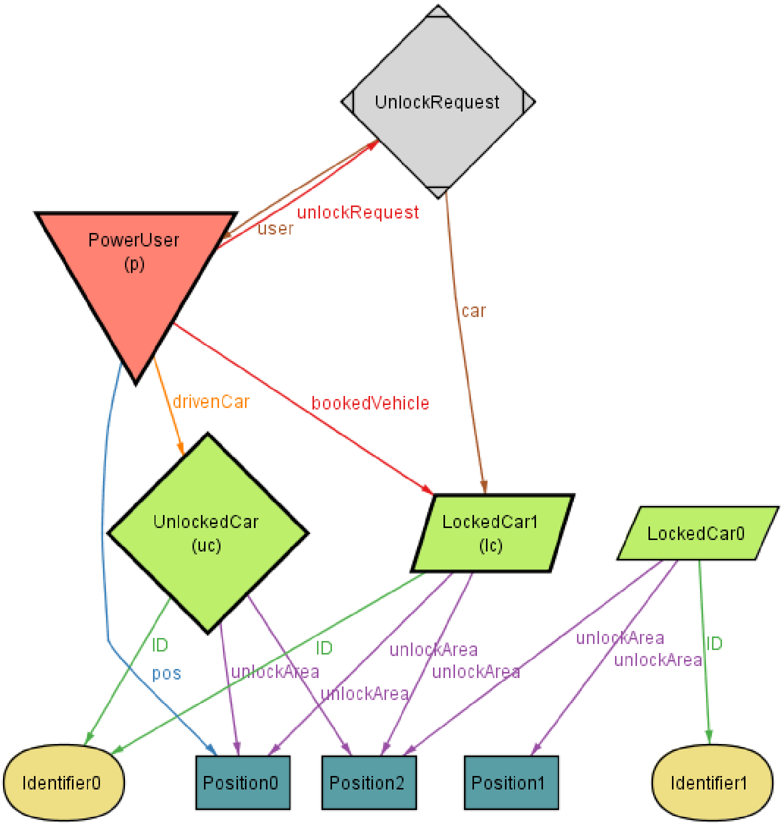
- Reserved Cars Unlocking

```
/**DESCRIBE A SITUATION WHERE CARS IS ALREADY BOOKED**/

/**SIGNATURES**/

abstract sig Person{}

sig Position{}
sig Identifier{}
sig PowerUser extends Person{
        pos: one Position,
        bookedVehicle:one LockedCar,
        unlockRequest: lone UnlockRequest,
        drivenCar: lone UnlockedCar
        }

sig UnlockRequest{
        user:one PowerUser,
        car:one LockedCar
        }{#UnlockRequest ≤ #PowerUser}


abstract sig Car{
        ID:one Identifier,
        unlockArea: some Position
        }

sig LockedCar extends Car{}

sig UnlockedCar extends Car{}

/**FACT**/

fact userBook1Car{ /*one user can book only 1 car*/
        no p,p’ :PowerUser | p.bookedVehicle=p’.bookedVehicle and p≠p’
        }

fact drivenCarEqualToBooked{/*user can drive only the booked car*/
         all p:PowerUser | p.drivenCar.ID=p.bookedVehicle.ID
        }

fact unlockedCarEqualLockedCar{/*number of unlocked car <=locked one*/
        #UnlockedCar≤#LockedCar
        }

fact noUserDriveSameCar{/*one user can drive only one car*/
        no p,p’:PowerUser | p.drivenCar= p’.drivenCar and p≠p’
        }
```

```
fact only2CarsWithSameID{
        all c,c':LockedCar | c.ID ≠ c'.ID implies c ≠ c'
        all c,c':LockedCar | c.ID = c'.ID implies c = c'
        all c,c':UnlockedCar | c.ID ≠ c'.ID implies c ≠ c'
        all c,c':UnlockedCar | c.ID = c'.ID implies c = c'
        }

fact noCarsInSamePosition{/*no different cars with same position*/
        all c,c':LockedCar | c.unlockArea=c'.unlockArea implies c=c'
        all c,c':LockedCar | c.unlockArea≠c'.unlockArea implies c≠c'
        all c,c':UnlockedCar | c.unlockArea=c'.unlockArea implies c=c'
        all c,c':UnlockedCar | c.unlockArea≠c'.unlockArea implies c≠c'
        }

fact drivedCarEqualBooked{/*users can only drive car they booked*/
        all p:PowerUser | p.drivenCar.unlockArea=p.bookedVehicle.unlockArea
        }

fact noUnlock4NonBookedCar{/*one user can unlock only the car he has booked
    ↪ */
        all p:PowerUser, c:LockedCar | p.unlockRequest.car=c implies p.
            ↪ bookedVehicle=c
        }

fact unlockCarRequireUser{/*no unlocked car without user*/
        #UnlockedCar>0 implies #PowerUser>0
        }

fact usersThatUnlockCarHaveAnUnlockRequest{
        all c:UnlockedCar,p:PowerUser | p.unlockRequest.car.ID=c.ID
        }

fact unlockOnlyInUnlockArea{/*user can unlock a car only if he is in car's
    ↪ unlockArea*/
        no p:PowerUser, c:UnlockedCar |
                (p.pos not in c.unlockArea) and p.unlockRequest.car.ID=c.ID
        }

fact sameCarHaveSameUnlockArea{ /* if 2 cars are the same, they have the
    ↪ same unlock area*/
        all c,c':Car | c.ID=c'.ID implies c.unlockArea=c'.unlockArea
        }

/**PREDICATES**/

pred unlockCar[uc:UnlockedCar, lc:LockedCar, p:PowerUser]{
        p.bookedVehicle=lc ∧

        uc.ID=lc.ID ∧
        uc.unlockArea=lc.unlockArea
        }

run unlockCar

/**ASSERTS**/

assert TwoCarAreSame{/*cars with same ID and unlockArea are the same car*/
        all c,c':Car | c.ID=c'.ID implies c.unlockArea=c'.unlockArea
        }

check TwoCarAreSame
```

```
assert NoUserUnlockBookedCar{ /*users can't have an unlockRequest for a car
    ↪ they haven't book*/
        all c:Car, u,u':UnlockRequest ,p,p':PowerUser | (p.bookedVehicle=c
            ↪ and p≠p' and u'≠u) implies
                        ((u.user=p and u.car=c) and not(u'.user=p' and u'.
                            ↪ car=c))
        }

check NoUserUnlockBookedCar

assert NoMultipleCarWithSameID{
        no c,c':LockedCar | c'.ID=c.ID and c'≠c
        no c,c':UnlockedCar | c'.ID=c.ID and c'≠c
        }

check  NoMultipleCarWithSameID

assert UserMakeRequest{/*users have to make request for unlock cars they
    ↪ have book*/
        all p:PowerUser,uc:UnlockedCar,lc:LockedCar,u:UnlockRequest |
                unlockCar[uc,lc,p] implies (p.pos in lc.unlockArea and
                        p=u.user and p.bookedVehicle=lc and u.car=p.
                            ↪ bookedVehicle and lc.ID=uc.ID)
        }

check UserMakeRequest
```

5 commands were executed. The results are:
-1: Instace found.unlockCar is consistent.
-2: No counterexample found. TwoCarAreSame may be valid.
-3: No counterexample found. NoUserUnlockBookedCar may be valid.
-4: No counterexample found. NoMultipleCarWithSameID may be valid.
-5: No counterexample found. UserMakeRequest may be valid.



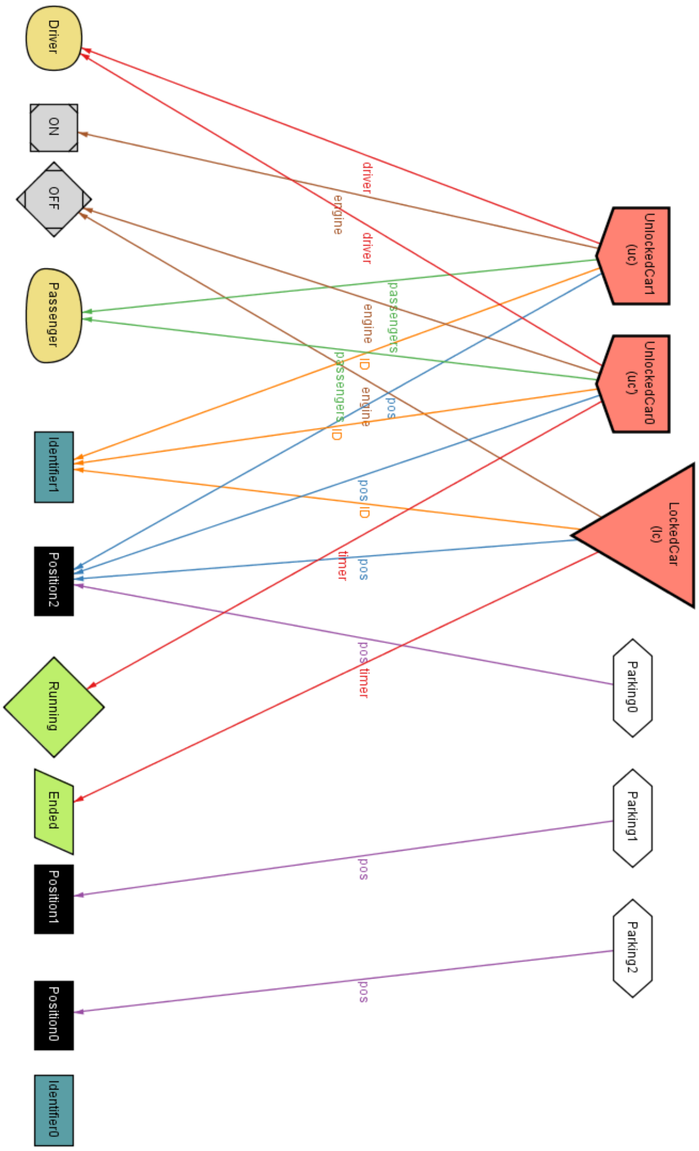
Unlocking car world.

- Cars Locking

```
/**DESCRIBE A SITUATION WHERE CAR IS NOT MOVING**/

/**SIGNATURES**/

abstract sig Person{}

abstract sig Engine{}
sig ON extends Engine {}
sig OFF extends Engine{}

abstract sig LockingTime{}
sig Ended extends LockingTime{}
sig Running extends LockingTime{}

sig Position{}
sig Identifier{}

sig Parking{
        pos:one Position
        }

sig Passenger extends Person{}
sig Driver extends Person{}

abstract sig Car{
        ID:one Identifier,
        pos: one Position,
        passengers:set Passenger,
        driver: lone Driver,
        engine: one Engine,
        timer: lone LockingTime
        }{#passengers <4}

sig LockedCar extends Car{}
sig UnlockedCar extends Car{}{#driver=1}

/**FACT**/

fact only2CarsWithSameID{
        all c,c':LockedCar | c.ID ≠ c'.ID implies c ≠ c'
        all c,c':LockedCar | c.ID = c'.ID implies c = c'
        all c,c':UnlockedCar | c.ID ≠ c'.ID implies c ≠ c'

 /*no 3 cars with same id*/
        all c,c',c'':UnlockedCar | (c.ID = c'.ID and c.ID=c''.ID) implies
                ((c = c' and c.engine=c'.engine) or (c=c'' and c.engine=c''.
                    ↪ engine) or (c'=c'' and c'.engine=c''.engine))
        }

fact noDifferentCarsInSamePosition{
        all c,c':LockedCar | c.pos=c'.pos implies c=c'
        all c,c':LockedCar | c.pos≠c'.pos implies c≠c'
        all c,c':UnlockedCar | c.pos≠c'.pos implies c≠c'

 /*no 3 cars with same position*/
        all c,c',c'':UnlockedCar | (c.pos = c'.pos and c.pos=c''.pos)
            ↪ implies
                ((c = c' and c.engine=c'.engine) or (c=c'' and c.engine=c''.
                    ↪ engine) or (c'=c'' and c'.engine=c''.engine))
        }
```

```
fact onlyOneDriverPerCar{/*2 user can't drive the same car */
        all c,c':UnlockedCar | (c.engine=ON and c'.engine=OFF and c.ID=c'.ID
            ↪ ) implies c.driver=c'.driver
        all c,c':UnlockedCar | c.ID≠c'.ID implies c.driver≠c'.driver
        }

fact noUnlockTimeIfEngineOn{/*timer start only when engine is turned off*/
        all c:UnlockedCar | c.engine=ON implies #c.timer=0 and #c.driver=1
        }

fact timerIfEngineIsStopped{/*if engine is OFF there'is a timer + if engine
    ↪ OFF car is parked*/
        all c:UnlockedCar | c.engine=OFF implies c.timer=Running
        all c:UnlockedCar | c.engine=OFF implies ( c.pos in Parking.pos)
        }

fact oneParkingForPosition{
        all p,p':Parking | p.pos=p'.pos implies p=p'
        all p,p':Parking | p.pos≠p'.pos implies p≠p'
        }

fact noSamePassengersOnDifferentCars{
        all c,c':UnlockedCar | c.ID≠c'.ID implies ((c.passengers & c'.
            ↪ passengers)=none)
        }


/*locked cars fact*/

fact carsLockedOnlyWithEngineOff{
        all c:LockedCar | c.engine=OFF
        }

fact carsLockedIsInAParking{
        all c:LockedCar | c.pos in Parking.pos
        }

fact carsLockedHasNoPersonOnBoard{
        all c:LockedCar | #c.passengers=0 and #c.driver=0
        }

fact noCarsLockedIfShortTimeIsPasses{/*all lockedCars unlock after a given
    ↪ time*/
        all c:LockedCar | c.timer=Ended
        }

fact lockedCarRequiresOneUnlockedCar{
        all uc:UnlockedCar, lc:LockedCar | uc.ID=lc.ID implies uc.pos=lc.pos
        }

/**PREDICATES**/

pred turnOff[c,c':UnlockedCar]{
        c.engine=ON ∧

        c'.engine=OFF ∧
        c'.ID=c.ID ∧
        c'.pos=c.pos
        }

pred lockCar[lc:LockedCar, uc:UnlockedCar]{
        uc.engine=OFF ∧
        lc.ID=uc.ID ∧
        lc.pos=uc.pos
}

pred show [lc:LockedCar ,uc,uc':UnlockedCar]{
        turnOff[uc,uc']
        lockCar[lc,uc']
        }

run turnOff
run lockCar
run show

/**ASSERT**/

assert UnlockedCarIsTheLockedOne{/*check if locked and unlocked cars are the
    ↪   same*/
        all uc,uc':UnlockedCar ,lc:LockedCar |
                turnOff[uc,uc'] and lockCar[lc,uc'] implies
                        (uc.ID=lc.ID and uc.pos=lc.pos)
        }
check UnlockedCarIsTheLockedOne

assert LockedCarStatus{/*check if the status of LockedCar is correct*/
        all uc:UnlockedCar, lc:LockedCar | lockCar[lc,uc] implies lc.ID=uc.
            ↪ ID and lc.pos in Parking.pos and
                lc.engine=OFF and #lc.driver=0 and #lc.passengers=0 and lc.
                    ↪ timer=Ended and lc.pos=uc.pos
        }
check LockedCarStatus

assert UnlockedCarWithEngineOffStatus{/*check if the status of UnlockedCar
    ↪ with engine turned off is correct*/
        all uc,uc':UnlockedCar | turnOff[uc,uc'] implies (uc.driver=uc'.
            ↪ driver and
                uc.pos=uc'.pos and uc.engine≠uc'.engine and uc.ID=uc'.ID and
                    ↪  uc'.timer=Running and #uc.timer=0)
        }
check UnlockedCarWithEngineOffStatus
```

6 commands were executed. The results are:
-1: Instance found.turnOff is consistent.
-2: Instance found.lockCar is consistent.
-3: Instance found.show is consistent.
-4: No counterexample found. UnlockedCarIsTheLockedOne may be valid.
-5: No counterexample found. LockedCarStatus may be valid.
-6: No counterexample found. UnlockedCarWithEngineOffStatus may be valid.



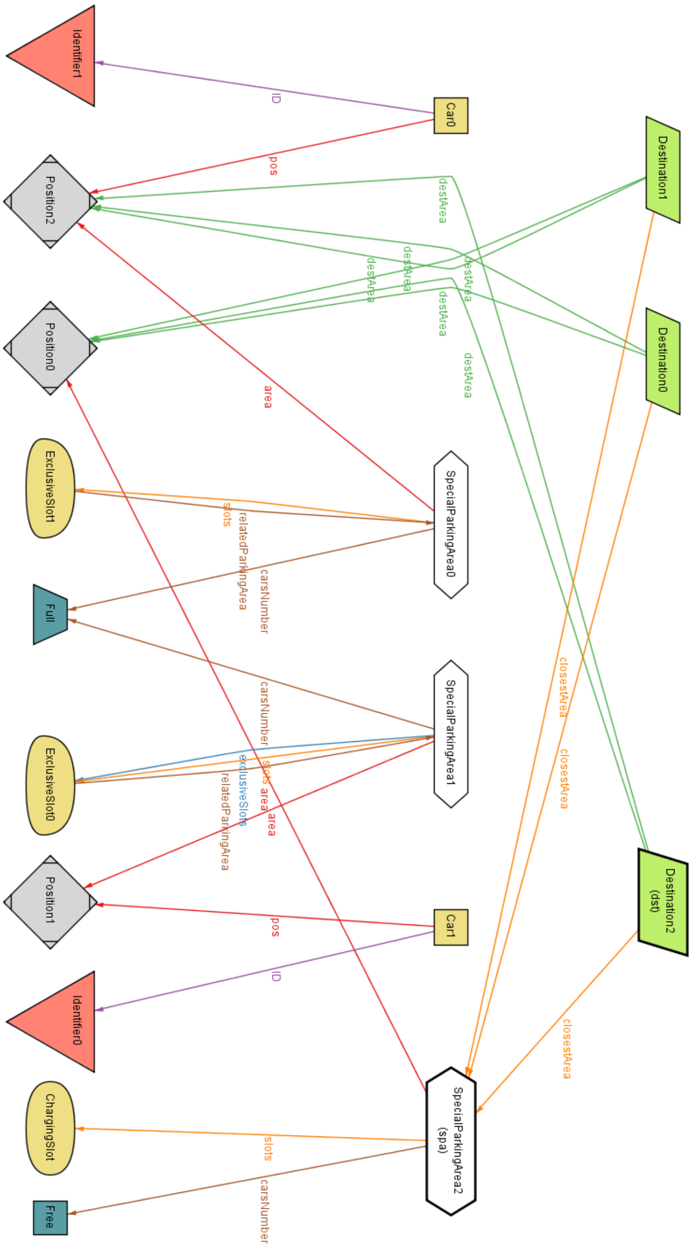
Locking car world.

- Money Saving Option

```
/**DESCRIBE A SITUATION WHERE THE BEST SPA IS ALREADY CHOSEN**/

/**SIGNATURES**/

sig Position{}
sig Identifier{}

sig Car{
        ID:one Identifier,
        pos: one Position
        }

abstract sig Status{}
sig Full extends Status{}
sig Free extends Status{}

sig ChargingSlot{}
sig ExclusiveSlot extends ChargingSlot{
  relatedParkingArea: one SpecialParkingArea
}

sig SpecialParkingArea{
        carsNumber: one Status, /*refers to the number of exclusiveSlots*/
        area: some Position,
        slots: some ChargingSlot,
        exclusiveSlots: set ExclusiveSlot,
        }{#slots ≥ #exclusiveSlots
                #area=#slots}

sig Destination{
        destArea: some Position, /*area within a specific range from
            ↪ original destination*/
        closestArea: one SpecialParkingArea
}

/**FACT**/

fact only2CarsWithSameID{
        all c,c':Car | c.ID ≠ c'.ID implies c ≠ c'
        all c,c':Car | c.ID = c'.ID implies c = c'
        }

fact noCarsInSamePosition{/*no different cars with same position*/
        all c,c':Car | c.pos=c'.pos implies c=c'
        all c,c':Car | c.pos≠c'.pos implies c≠c'
        }

fact exclusiveSlotsSubsetOfSlots{
  all spa: SpecialParkingArea, xs: ExclusiveSlot |
    (xs in spa.exclusiveSlots) implies (xs in spa.slots)
  }

fact noSlotInMultipleSpecialParkingArea{/*no 2 equal slot in different
    ↪ parkingArea*/
        all slot: ChargingSlot, spa1,spa2: SpecialParkingArea |
                spa1≠spa2 implies  (slot not in spa1.slots) or (slot not in
                    ↪ spa2.slots)
         all spa1,spa2: SpecialParkingArea, slot: ChargingSlot|
                (slot in spa1.slots) and (slot in spa2.slots) implies spa1=
                    ↪ spa2
  }
```

```
fact disjointParkingAreas{/*no 2 equal Special Parking Areas*/
  all spa1,spa2: SpecialParkingArea, p: Position |
        spa1≠spa2 implies  (p not in spa1.area) or (p not in spa2.area)
 all spa1,spa2: SpecialParkingArea, p: Position |
        (p in spa1.area) and (p in spa2.area) implies spa1=spa2
}

fact notTooDistantParkingAreas{
        all d:Destination | (d.closestArea.area & d.destArea) ≠none
        }

fact notFullSpecialParkingAreas{
        all d:Destination | d.closestArea.carsNumber=Free
        }

fact noExclusiveSlotOutsideSpecialParkingArea{
  all xs: ExclusiveSlot, spa: SpecialParkingArea |
    xs.relatedParkingArea = spa implies xs in spa.slots
  }

fact slotsCapacityConstraint{
  #(Car.pos & SpecialParkingArea.area)
    =< #SpecialParkingArea.slots
  }

/**PREDICATES**/

pred getAlternative[dst:Destination,spa:SpecialParkingArea]{
        dst.closestArea=spa
        }

run getAlternative

/**ASSERT**/

assert ClosestAreaIsReachableAndFree{
        all d:Destination, spa:SpecialParkingArea | getAlternative[d,spa]
            ↪ implies ((d.destArea & spa.area)≠none and
                spa.carsNumber=Free)
        }
check ClosestAreaIsReachableAndFree
```

2 commands were executed. The results are:
-1: Instance found.getAlternative is consistent.
-2: No counterexample found. ClosestAreaIsReachableAndFree may be valid.



Money saving option world.

5.3 Document revisions history

Version	**Date**	**Changes**
1.0-RC1	13/11/2016	First deadline release.
1.0-RC2	11/12/2016	Added: - *navigator* actor added in use case 4.3.10. - definition of *extends* added in the use case diagram. - original source document of the *extends* definition added in the reference section. - figure number added to each mockup. Fixed: - *th* added in the date on the 1st page. - title of subsection 5.4 updated to be consistent across all documents of the project. - all names are now preceded by their respective surnames in the section 5.4. - typo in Output Conditions of use case 4.3.7 fixed. - description of the figure 15 fixed. - typo in the description of the figure 16 fixed. - Service Back-end Logic renamed Service Business Logic.

5.4 Effort Spent

Teamwork	∼22h
Piccirillo Luca	∼25h
Zampogna Gian Luca	∼25h
Zini Edoardo	∼29h