POLITECNICO MILANO 1863

Politecnico di Milano

Software Engineering 2 project

PowerEnJoy

Requirement Analysis and Specification Document

Matteo Penco

mat. 875740

Riccardo Pressiani

mat. 874948

Version 1.1 approved

December 11, 2016

Revision History

Revision	Date	Author(s)	Description
0.1	26.10.16	RP	Document created
1.0	13.11.16	RP and MP	Document approved
1.0.1	14.11.16	RP and MP	Fix bug in Alloy model
1.1	11.12.16	RP and MP	Several fixes – Fix front page – Fix Use Case Model – Fix assumption A1

Hours of work

Matteo Penco	35h
Riccardo Pressiani	35h

Contents

Chapter 1

Introduction

This section introduces the Requirements Analysis and Specification Document for the PowerEnJoy platform to its readers.

1.1 Purpose

This document describes the software functional and non functional requirements for the PowerEnJoy platform. The information contained in this document is intended to be used by project managers and the developer team that will implement and verify the correct functioning of the system.

1.2 Scope

PowerEnJoy is inteded to be a management system for a car-sharing system that exclusively employs electrical powered cars. The system allows users to find all the available cars near a given location which can be their current position or a specific address typed in. The user can book one of the cars available for a limited amount of time. Once the booked car is reached by the user, it can be unlocked from one of the smart devices of the user. After entering a PIN, decided by the user during the registration phase, on the onboard computer, the engine can be started and the rental begins. The cost of the service is calculated on the total duration of the rental multiplied by a fixed rate per minute. The user is continuously informed about the cost of the ongoing rental by the onboard computer. The user can end the rental by parking the car and stopping the engine.

The PowerEnJoy platform is intended to be available on the major mobile devices, such as smartphones and tablets running iOS [1] or Android [2].

1.3 Definitions, acronyms and abbreviations

1.3.1 Definitions

Onboard computer An onboard computer is intended to be the embedded device integrated in the car system that provide two main functions: on one hand it shows the user basic information about the ongoing rental, on the other hand it is in charge of all the communications related to the state of the car between the car and the PowerEnJoy management system.

Safe area A safe area is an area, within predefined edges, in which the users are allowed to park the rented cars. The users is not allowed to park, and therefore end a rental, if he/she is not inside a safe area.

Special parking area A special parking area is a safe area where an electric car charging system is installed.

Car states One and only one of the four following states is assigned to each car:

> **AVAILABLE** A car is AVAILABLE when is parked in a safe area and can be booked by a user. It changes to RESERVED when a user books it or to UNAVAILABLE if it gets damaged.

> **RESERVED** A car is RESERVED when a user has booked it. This means that it cannot be booked by any other user until it becomes AVAILABLE again. A RESERVED car changes to AVAILABLE if the booking is canceled, to IN USE if the user that has booked it begins the rental, or to UNAVAILABLE if it gets damaged.

> **IN USE** A car is IN USE when a user is driving it. A car that is IN USE changes to AVAILABLE when the rental ends or to UNAVAILABLE if one of the conditions written below subsists.

> **UNAVAILABLE** A car is UNAVAILABLE if it's parked in a safe area with low battery ($< 20\%$), if the battery is completely exhausted or if the car is damaged. An UNAVAILABLE car changes to AVAILABLE only after being recovered by an external company in order to fix, depending on the situation, all problems written above.

Over-The-Air updates An Over-The-Air update is the wireless delivery of new software or data to mobile phones and tablets.

Sign up Sign up is intended to be the registration phase. During this process, the user provides the system the credentials required along with the information about the documents required.

1.3.2 Acronyms

API Application Programming Interface

GPS Global Positioning System

OTA Over-The-Air

PIN Personal Identification Number

UML Unified Modeling Language

1.3.3 Abbreviations

24/7 24 hours a day, 7 days a week.

ID Identification

1.4 References

- Assigments document for the first semester project of the Software Engineering 2 course held at Politecnico di Milano by Mottola Luca and Di Nitto Elisabetta [3].

- IEEE Recommended Practice for Software Requirements Specifications [4].

1.5 Overview

A general introduction about the PowerEnJoy project has been given in this chapter. In the following chapters the reader will be provided with a more detailed description of the system.

In Chapter 2, an overall description of the aspects that will affects the the product and its requirements will be provided.

In Chapter 3, all goals and software requirements of the PowerEnJoy system will be pointed out.

In Appendix A, some possible scenarios of PowerEnJoy use are provided along with some graphic description of the most complex aspects of the system.

In Appendix B, a model of the PowerEnJoy system written in Alloy [5] is provided.

Chapter 2

Overall Description

In this chapter all the aspects that affect the product and his requirements are presented. A general background of the enviroment in which the product works is provided. This includes the main functions of the product along with the constraints and the domain assumptions considered in the requirements described in the next chapter.

2.1   Product perspective

In this section some graphic concepts about PowerEnJoy application and website are presented, in order to give a general idea about how would be the interaction with the system.
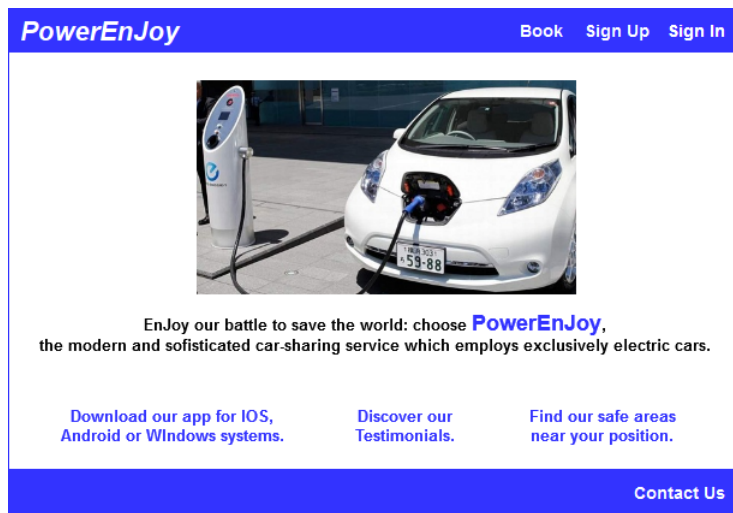

Figure 2.1: This image represents the possible homepage of PowerEnJoy website.
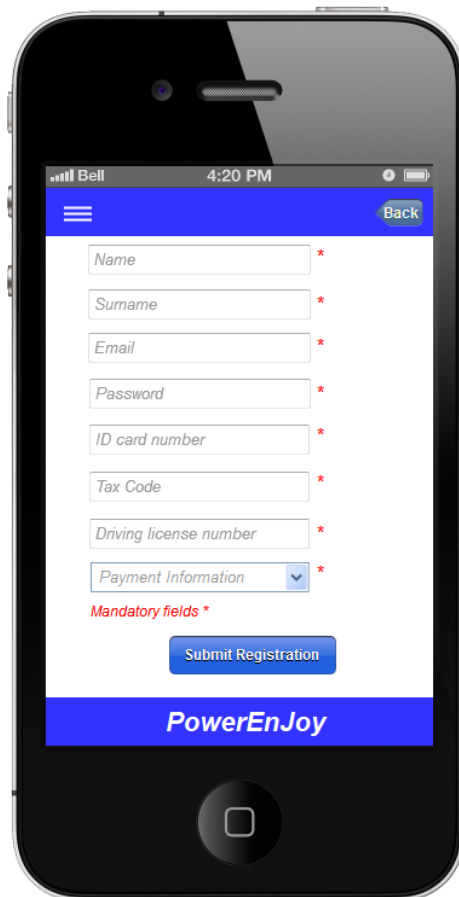
Figure 2.2: This image represents the possible interface of PowerEnJoy mobile phone application visible from the user in order to register to the system.
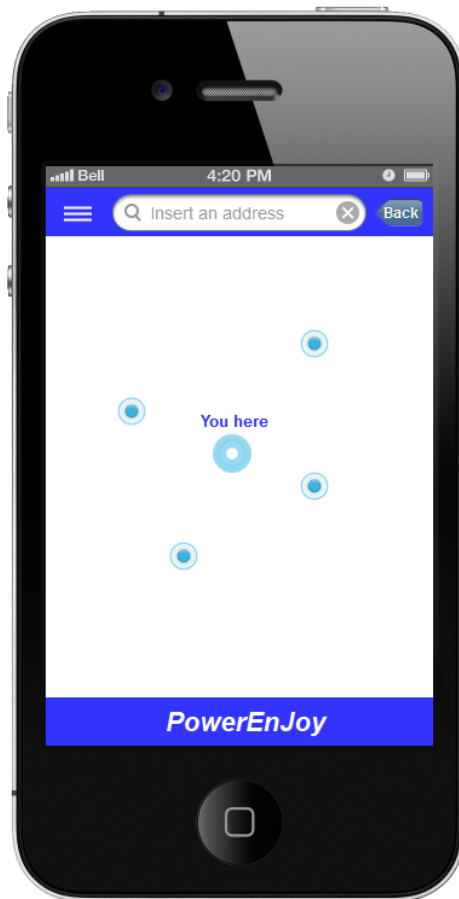
Figure 2.3: This image represents the possible interface of PowerEnJoy mobile phone application visible from the user in order to book an available.
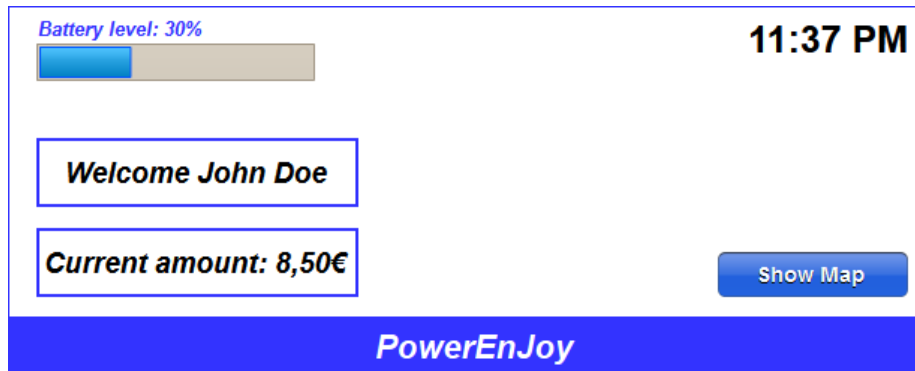
Figure 2.4: This image represents the possible interface of PowerEnJoy onboard computer present on the car.

2.2 Product functions

Registration The system allows and requires the user to register before accessing the services of the app. In this phase the user uploads his/her ID document and driver licence and provides the credential for his/her preferred payment method.

Map exploration The system allows the user to explore a map and to look for available cars near either his/her location or a given address.

Car reservation The system allows the user to reserve a car for a limited amount of time. During this interval the car is parked but cannot be reserved or rented by any other user of the PowerEnJoy platform.

Car rental The system allows the user to rent a car that he/she has previously reserved. The rental starts once the user ignites the engine of the car. The cost of the service is determined by the duration of the rental.

2.3 User characteristics

The intended user of PowerEnJoy is not required to have any particular technical expertise to access the services provided by the application. This is achieved by providing an application with only a few interfaces. Moreover, the user is guided through a simple flow that will allow him/her to rent a car.

For what concerns the user's devices, he/she is not required to have any particular or high expensive device. In fact, once a device has a bluetooth module and an active data connection available, the owner will be allowed to access all the services provided by the PowerEnJoy platform.

2.4 Constraints

2.4.1 Regulatory polices

During the registration phase the user is required to allow PowerEnJoy to manage the sensible data he/she is providing. Moreover, the user needs to allow the system to access his current location in order to use the services provided by the platform. All the issues related to the user privacy must follow the state regulations.

2.4.2 Hardware limitations

- The user must have access to an Internet connection in order to reserve a car from the web app or the mobile app.

- The user must activate the GPS module and have access to a data connection on one of its devices in order to unlock the car once he/she has reached it.

- The GPS receiver equipped in the cars must be always active in order to communicate the car location to the system.

- All the cars must have an active data connection in order to communicate and receive the necessary information from the system.

2.4.3 Interfaces to other applications

Payment management service

The system needs to integrate an external service for the management of the payment methods. All the major payment methods must be supported by the application, including the most popular credit card circuits (including Visa, MasterCard and American Express) and PayPal.

Car recovery service

The system needs to integrate the APIs provided by an external company that is in charge of arranging the cars recovery. In case of particular situation in which the car needs extraordinary assistance, the system alerts the external company.

2.5 Assumptions and dependencies

[A1] An external service is integrated in the system in order to verify the correctness of the documents uploaded by the user during the registration phase.

[A2] The system submits in the database the user registered matched with the provided PIN.

[A3] Every car is equipped with a sensor that communicates to the system if the car's engine is running or not. The value of this sensor is ON for the former case, OFF for the latter.

[A4] Every car is equipped with an onboard computer.

[A5] Every car is equipped with a GPS receiver that communicates the car location to the system. The data transmitted to the system corresponds to the geographic coordinates of the car location.

[A6] The system can automatically lock and unlock the car doors.

[A7] The car locks the engine until the correct user PIN is entered in the onboard computer or when a rental ends.

[A8] The car's seats are equipped with a sensor able to detect the presence of a person seated in the car. The signal transmitted to the system is equal to 1 if a person is detected, 0 (zero) otherwise.

[A9] The car can inform the system whether the doors are closed or not. If at least one of the door is open, the value transimitted is equal to 0 (zero), 1 otherwise.

[A10] The set of safe areas is defined by default in the system.

[A11] The set of safe areas provided with power grid stations is defined by default in the system.

[A12] The rental cost per minute is defined by default in the system.

[A13] The car informs the system about the battery level. The value transmitted is equal to the percentage of the battery charge available. Moreover, the car communicates to the system if the battery is charging or not.

[A14] The car informs the system about its component state (e.g. A message is sent in case of engine failure).

[A15] An external service is integrated in the system in order to manage billing operations.

[A16] An external company is in charge of arranging the cars recovery operations.

Chapter 3

Specific Requirements

This chapter provides a high detaild description of both functional and non functional software requirements.

3.1 Functional requirements

3.1.1 Goal 1

User is able to register in the system.

3.1.1.1 Functional requirement 1.1

If credentials and payment information given by the user are valid and he is not present in the database as an already registered user, then the new user is registered and submitted in the database.

Domain assumptions required: [A1]

3.1.2 Goal 2

User registered receives a PIN to insert in the reserved car's onboard computer in order to drive it.

3.1.2.1 Functional requirement 2.1

If user's registration succeeds, then the system generates a new PIN for the registered user and communicates it via mail.

Domain assumptions required: [A1, A2]

3.1.3 Goal 3

User is able to log in the system.

3.1.3.1 Functional requirement 3.1

If the user inserts correct credentials and password, then the system allows the user to log in.

Domain assumptions required: [A1]

3.1.4 Goal 4

The user is able to find the location of all the available cars near is current position or near a specified address.

3.1.4.1 Functional requirement 4.1

If the user has his GPS activated and there are available cars near his position or near the specified address, then the system shows onto the web app the position of these cars.

Domain assumptions required: [A5]

3.1.5 Goal 5

User is able to book a car for up to an hour before picking it up.

3.1.5.1 Functional requirement 5.1

If the user is logged, has no ongoing booking or rental and the car selected is AVAILABLE, then the user can book the car. The system creates a booking related to that user and that car.

3.1.5.2 Functional requirement 5.2

The system tags the car as RESERVED.

3.1.6 Goal 6

User is able to check the battery level of a car before booking it.

3.1.6.1 Functional requirement 6.1

If the user chooses a car to book then, before confirming the booking, the system shows the battery level of the car chosen.

Domain assumptions required: [A13]

3.1.7 Goal 7

User is able to cancel a booking of a car.

3.1.7.1 Functional requirement 7.1

If the user has booked a car, then the user can cancel the booking and the system tags the car as AVAILABLE.

3.1.8 Goal 8

If a car is not picked up within one hour from the booking the reservation expires and the user pays a fee of 1 EUR.

3.1.8.1 Functional requirement 8.1

When the user books a car, a stopwatch starts: if one hour has passed and no rental, related to that booking, has been created, then the booking expires.

3.1.8.2 Functional requirement 8.2

If the booking has expired, the system tags the car as AVAILABLE.

3.1.8.3 Functional requirement 8.3

If the booking has expired, the system charges the user a fee of 1 EUR.

Domain assumptions required: [A15]

3.1.9 Goal 9

User is able to open the reserved car.

3.1.9.1 Functional requirement 9.1

If the comparison, through GPS systems, between positions of the user and the reserved car indicates that the user reached the "opening distance", then the user can request the system to open the reserved car.

Domain assumptions required: [A5, A6]

3.1.10 Goal 10

The user can start a rental by inserting his/her PIN and then by igniting the engine.

3.1.10.1 Functional requirement 10.1

Once the user has entered his/her PIN, if the system confirms that it is correct, the user is allowed to ignite the engine.

Domain assumptions required: [A7]

3.1.10.2 Functional requirement 10.2

If the engine is set to ON, the system creates a rental related to the user's ongoing booking and a stopwatch starts in order to measure the time elapsed from the begin of the rental.

Domain assumptions required: [A3]

3.1.10.3 Functional requirement 10.3

The system calculates the current cost of the rental by multiplying the minutes elapsed by a fixed rate per minute.

Domain assumptions required: [A12]

3.1.10.4 Functional requirement 10.4

When the rental begins, the system tags the car as IN USE.

3.1.11 Goal 11

When a user is driving a car, the information about the ongoing rental is displayed by the onboard computer. The displayed information includes:

- The battery level of the car.

- The actual cost of the ongoing rental.

- A map that can be used as a GPS navigation device or to find both safe areas and special parking areas.

3.1.11.1 Functional requirement 11.1

When the rental begins, the onboard computer must show the information required.

Domain assumptions required: [A4, A12, A13, A10, A11]

3.1.11.2 Functional requirement 11.2

The system must constantly update the onboard computer with the most recent data available about the location of safe areas and special parking areas.

Domain assumptions required: [A10, A11]

3.1.12 Goal 12

The rental ends when the user parks in a safe area and exits the car.

3.1.12.1 Functional requirement 12.1

If the engine is set to OFF, the GPS location of the car is whitin one of the safe areas, the seat sensors are all set to 0 (zero) and all the car's doors are closed the rental ends.

Domain assumptions required: [A3, A5, A8, A9, A10]

3.1.12.2 Functional requirement 12.2

If the rental is ended the related watchstop stops and the final cost of the fare is calculated by the system.

3.1.12.3 Functional requirement 12.3

If the user is no eligible for additional discount or penalities on the final cost of the rental, the user is billed according to the payment method that he/she has provided.

Domain assumptions required: [A15]

3.1.12.4 Functional requirement 12.4

The car related to an ended rental is tagged as AVAILABLE by the system.

3.1.13 Goal 13

If the user shares the rented car with at least two passengers during the whole duration of the rental, he/she is eligible for a 10% discount on the total cost of that rental.

3.1.13.1 Functional requirement 13.1

If at least three seat sensors are set to 1 from the moment the engine is set to ON to the moment it is set to OFF, the system applies a 10% discount on the total cost of the last rental.

Domain assumptions required: [A3, A8]

3.1.14 Goal 14

If the user leaves a car with the battery level higher than 50%, he/she is eligible for a 20% discount on the last ride.

3.1.14.1 Functional requirement 14.1

If the user ends a rental leaving the car with the battery level higher than 50%, the system applies a 20% discount on the total cost of the last rental.

Domain assumptions required: [A13]

3.1.15 Goal 15

If the car is left within one of the special parking areas and the user takes care of plugging the car into the power grid, he/she is eligible for a 30% discount on the last ride.

3.1.15.1 Functional requirement 15.1

If the rental is ended, the GPS location of the car is whitin one of the special parking areas and the battery is charging, the system applies a 30% discount on the total cost of the last rental.

Domain assumptions required: [A5, A11, A13]

3.1.15.2 Functional requirement 15.2

In order to apply the discount, the car has to be plugged to the power grid within a minute after the rental is ended.

Domain assumptions required: [A13]

3.1.16 Goal 16

If the car is left at more than 3KM from the nearest special parking area or with the battery charge lower than 20%, the user will pay an extra fee that will correspond to the 30% of the cost of the last ride.

3.1.16.1 Functional requirement 16.1

The system charges the user 30% more on the total cost of the last ride if at least one of the following condition is verified:

- The rental is ended and the distance between the GPS coordinates of the car and the ones of the nearest special parking area is more than 3KM.

- The rental is ended and the battery level is lower than 20%.

Domain assumptions required: [A5, A11, A13]

3.1.17 Goal 17

In case of low battery of a parked car, damaged car or empty battery, the car becomes unavailable until is recovered by an external company in order to fix all problems, depending on the situation, written above.

3.1.17.1 Functional requirement 17.1

The system sends an alert, containing the car's location, to an external company that will arrange the recovery of the car if at least one of the following conditions is true.

- A car is parked in a safe area and its battery level is lower than 20%.

- A car's battery level is equal to 0 (zero).

- One of the car's components is in failure.

3.1.17.2 Functional requirement 17.2

The car is tagged as UNAVAILABLE until all these conditions are satisfied.

- The car is parked in a safe area.

- The car's battery level is 100%.

- None of the car's components is in failure.

Domain assumptions required: [A5, A13, A14, A16]

3.1.17.3 Functional requirement 17.3

If there is a car related to an ongoing rental whose battery reach the 0% of charge, the user who is driving that car is charged with a 30 EUR fine.

3.1.17.4 Functional requirement 17.4

Once all the conditions above are satisfied, the system tags the car as AVAILABLE.

3.2 Software system attributes

3.2.1 Availability

The services provided by PowerEnJoy are available 24/7. A 99.9% availability is guaranteed.

3.2.2　Security

All the sensitive data are transferred using the latest security protocols. No plaintext data transfers are admitted.

3.2.3　Maintainability

Since PowerEnJoy provides a 24/7 service, the software to be developed will support online updates without service interruptions. Moreover, the onboard computers software will support OTA updates. If it will be considered necessary to turn the system offline, the users must be informed with due advance.

3.2.4　Portability

A mobile application is available for the main mobile operating systems, such as iOS [1] and Andriod [2]. Moreover, the company website allows the user to registrate and access their account. Only the booking phase can be managed from the website.

Appendix A

Scenarios and UML Models

In this appendix are included some scenarios and UML models in order to provide graphical representations of some of the most complex processes of the systems.

A.1 Scenarios

Here some possible scenarios of usage of the system are presented.

A.1.1 Scenario 1

Sean has finished university at 18:00, he has to go back home by train but that day Trenord dediced to stage a walkout. Luckily Sean is registered to the car-sharing service PowerEnJoy and can easily go back home even without train. He logs in the app on his mobile phone and looks for a car near his current position: he finds one really close but with a low battery level, so he decides to book another car a little more far but with almost full battery level. In the while two Sean's friends, who live near Sean, ask him if he could give them a lift: Sean gladly accepts because he's a good friend and in this way he can have a discount of 10% on the ride. They all reach the booked car within an hour and go back home. Sean's friends, who are not registered to PowerEnJoy, found the ride really pleasant and comfortable and think to register in the near future to PowerEnJoy.

A.1.2 Scenario 2

Sean just came back home by train after university, he's at train station but it's a really cold day and so he doesn't want to walk home. He decides to book a PowerEnJoy car to get home in a fast and comfortable way. While he walks to reach the booked car, Sean's mom calls him to inform that she's coming back from work and she can pick him up. Then Sean easily cancels his booking without paying any fee.

A.1.3 Scenario 3

Sean has to go to say hello to his friend who works in a pub a bit distant from the city where Sean lives. He decides to take advantage of PowerEnJoy service and so looks for a car to book. The only available car is quite far, but it's the only possibility for Sean cause he lent his car to his sister who needed it for work. He leaves home and walks to reach the booked car.. but there is a problem: Sean is not very good at orienting himself. He has a clear and simple map showed by PowerEnJoy's app on his mobile phone, but he loses himself anyway. Sean walks for more than one hour to reach the car but he can't find it, so PowerEnJoy charges him 1 EUR and his booking is canceled. Luckily, in the while another PowerEnJoy car becomes available near Sean's position, so he books it and can finally reach his friend.

A.1.4 Scenario 4

Sean is driving his PowerEnjoy car to get home but forgot about checking the battery level before starting the car ride. Suddenly he realizes that car's battery is almost exhausted, so he looks for a safe area equipped with power grid station on the onboard computer, but the nearest area is too far to be reached in time. He's forced to park the car with battery exhausted on side of the street: he will have to pay the amount of money indicated on the onboard computer of the car plus an additional fee for having exhausted the car's battery.
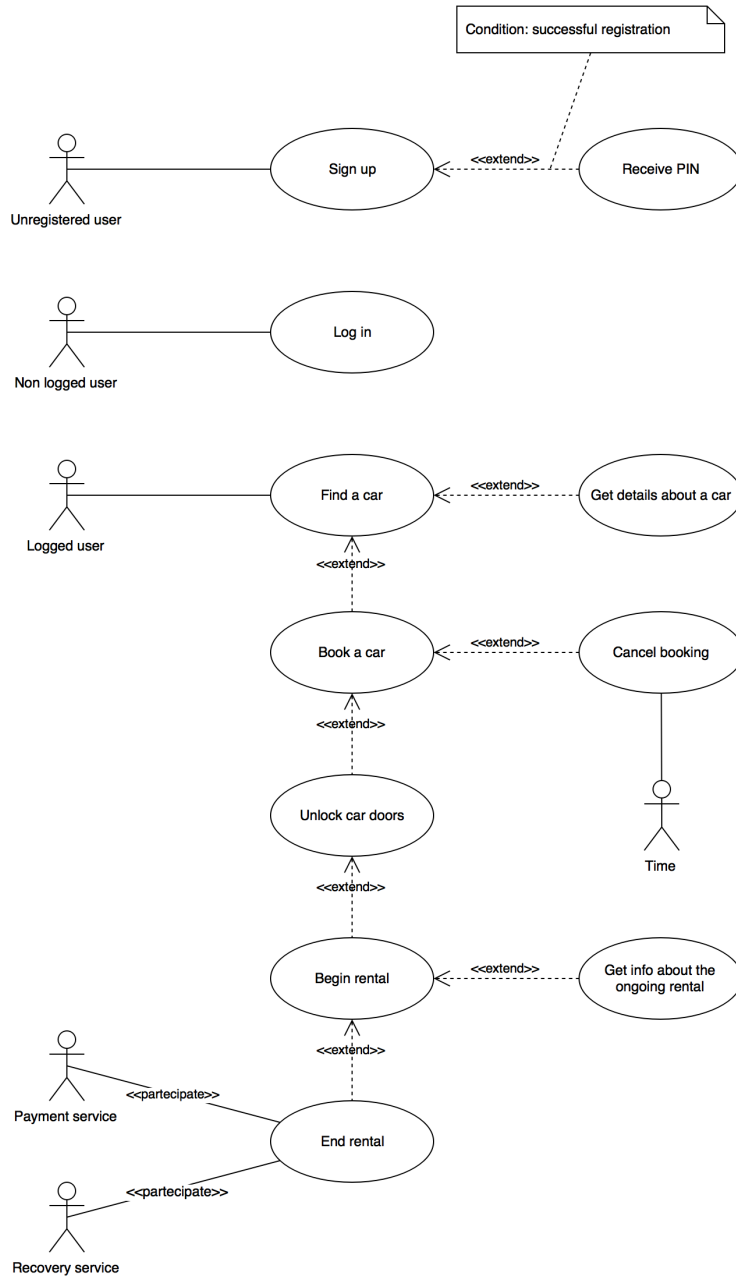
A.2   Use case model

Figure A.1: The diagram above represents the use case model of the PowerEnJoy project. All the use cases are included along with the actors involved in their execution.

A.2.1 Use case description 1 - Sign up

Name   Sign up

Actors   User

Entry conditions There are no entry conditions.

Flow of events

- The user accesses the homepage of PowerEnJoy or opens the app.
- The user inserts his/her credentials in the registration form.
- The user clicks on the sign up button.
- The system processes the registration of the user.

Exit conditions The user is successfully registered to the system.

Exceptions

- Credentials provided by the user are not correct. In this case the system notifies the user of the error and let him/her to input again his/her credentials.
- User is already registered. In this case the system notifies the user of the impossibility to register.

A.2.2 Use case description 2 - Receive the PIN

Name   Receive the PIN

Actors   User

Entry conditions The user must have completed the registration.

Flow of events

- The system generates the new PIN for the registered user
- The system sends to the registered user his/her PIN via mail

Exit conditions The user successfully receives the PIN to use PowerEnJoy cars.

Exceptions

- The user doesn't receive the PIN via mail. In this case the user can ask to the system to send again the mail with the PIN.

A.2.3 Use case description 3 - Log in

Name Log in

Actors User

Entry conditions The user has to be already registered.

Flow of events

- The user accesses the homepage of PowerEnJoy or opens the app
- The user inputs his email and password
- The user clicks on the log in button
- The system redirects the user on his personal page

Exit conditions The user is successfully redirected to his personal page.

Exceptions

- Email or password provided by the user are not correct. In this case the system notifies the user of the error and let him/her to input again his/her credentials.
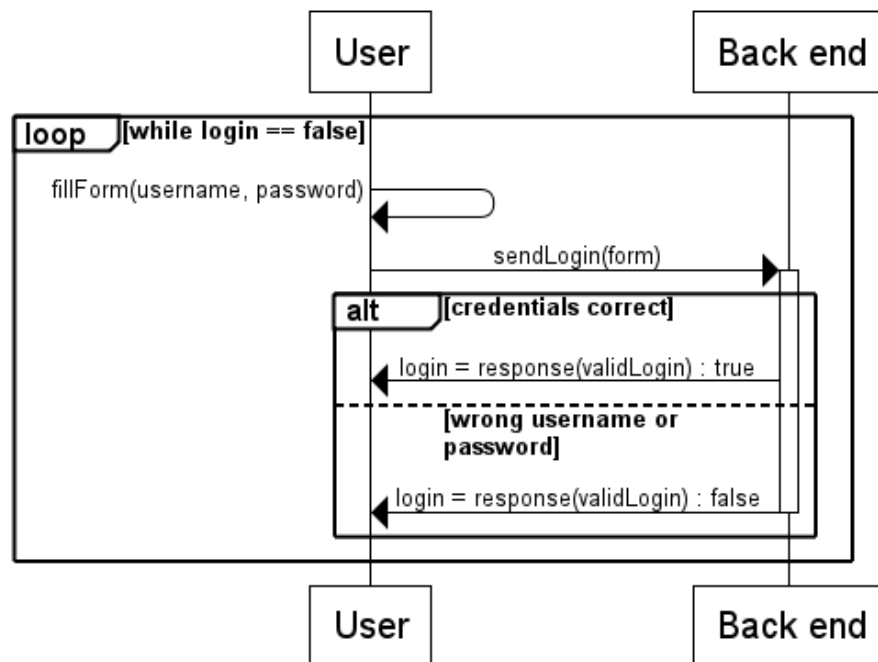
Figure A.2: The sequence diagram above describes the log in procedure.

A.2.4   Use case description 4 - Find a car

Name   Find a car

Actors   User

Entry conditions The user has to be logged in.

Flow of events

- The user inserts a specified address or chooses his current position
- The system loads on the map all available cars and all safe areas near the position provided by the user

Exit conditions The user can see on the map all available cars and all safe areas near the position provided.

Exceptions

- The user inserts an invalid address. In this case the system notifies the user of the error and let him/her to insert a new address.
- User's GPS doesn't work. In this case the system notifies the user of the impossibility to process the request.

A.2.5   Use case description 5 - Get details about a car

Name   Get details about a car

Actors   User

Entry conditions The user must have looked for available cars on the map.

Flow of events

- The user clicks on an available car shown on the map
- The system shows the details about the car chosen by the user

Exit conditions The user can see the details of the chosen available car.

Exceptions

- Given the assumptions written below, there are no exceptions in this flow of events.

A.2.6 Use case description 6 - Book a car

Name   Book a car

Actors   User

Entry conditions The user must have looked for available cars on the map.

Flow of events

- The user clicks on an available car shown on the map
- The system asks the user if he/she wants to confirm the action
- The user clicks on the confirmation button
- The system confirms the booking of the chosen car and gives the opportunity to cancel the booking

Exit conditions The user successfully books a car.

Exceptions

- Another user books a car before the user confirms the booking. In this case the system redirects the user on the map updated with the new situation of available cars.
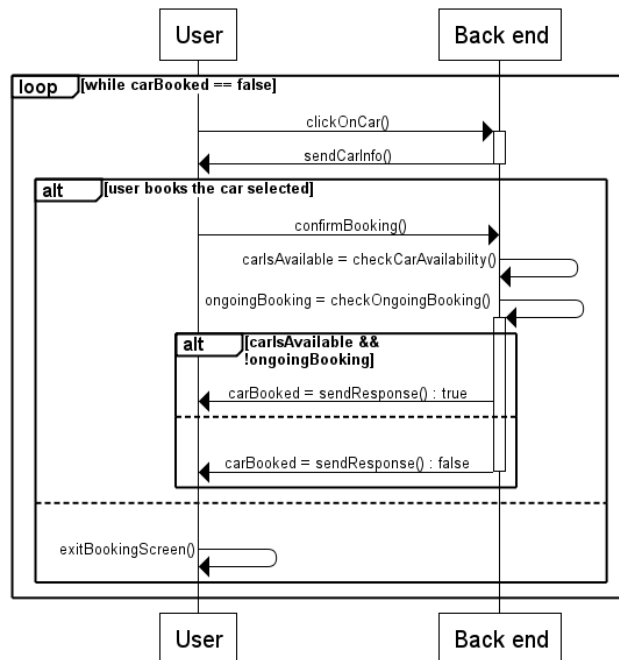
Figure A.3: The sequence diagram above describes the car booking procedure.

A.2.7 Use case description 7 - Cancel booking

Name   Cancel booking

Actors   User

Entry conditions The user must have booked a car.

Flow of events

- The user clicks on the button to cancel booking
- The system asks the user if he/she wants to confirm the action
- The system confirms the cancellation of booking

Exit conditions user successfully cancels booking of the car.

Exceptions

- It passed an hour since the booking confirmation. In this case the system has already cancelled automatically booking.
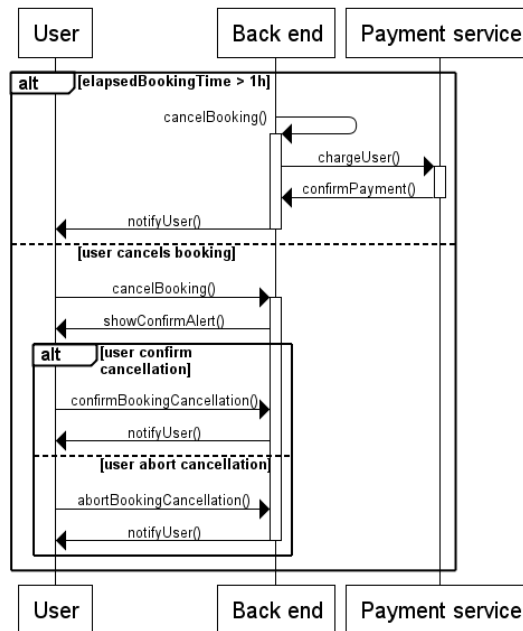
Figure A.4: The sequence diagram above describes the booking cancellation procedure.

A.2.8 Use case description 8 - Unlock car doors

Name   Unlock car doors

Actors   User

Entry conditions The user must have booked a car.

Flow of events

- The user is notified by the system that he's near enough to the booked car to unlock its doors
- The user clicks on the button to unlock the booked car
- The system unlocks car doors

Exit conditions The user successfully unlocks car doors.

Exceptions

- It passed an hour since the booking confirmation after the system gave the possibility to the user to unlock car doors. In this case the user can't unlock car doors because the system has already cancelled automatically booking.
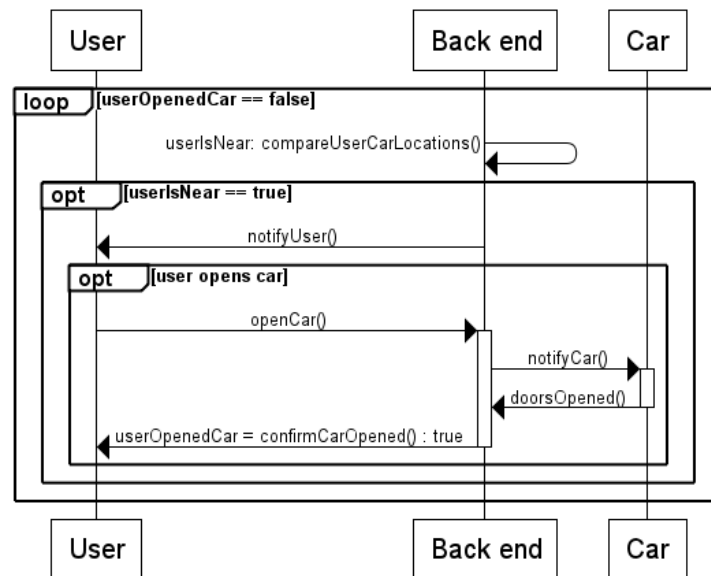
Figure A.5: The sequence diagram above describes the car doors unlocking procedure.

A.2.9 Use case description 9 - Begin rental

Name   Begin rental

Actors   User

Entry conditions The user must have unlocked car doors.

Flow of events

- The user inserts the PIN provided at the moment of registration on the onboard computer of the car.
- The user ignites the engine.

Exit conditions The user starts driving the car.

Exceptions

- The PIN inserted by the user is not correct. In this case the system doesn't allow the usere to ignite the engine until he inserts the correct PIN.
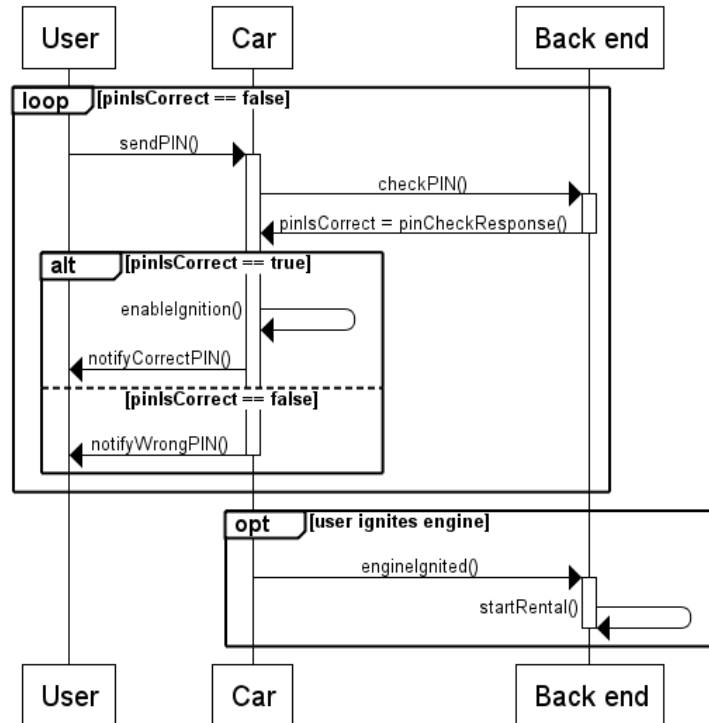
Figure A.6: The sequence diagram above describes the rental beginning procedure.

A.2.10 Use case description 10 - Get info about the ongoing rental

Name Get info about the ongoing rental

Actors User

Entry conditions The user must have started a rental.

Flow of events

- The user interacts with the onboard computer interface in order to get information about the ongoing rental.
- The systems shows on the onboard computer the desired information.

Exit conditions The user successfully gets desired information about the ongoing rental.

Exceptions

- The car's battery runs out. In this case the onboard computer can't give any further information about the ongoing rental except the total cost of the rental reached before the battery exhausted.

A.2.11 Use case description 11 - End rental

Name End rental

Actors User

Entry conditions The user must have started a rental.

Flow of events

- The user parks the car in a safe area.
- The user turns off the engine.
- The user and, eventually, passengers exit the car and close car doors.

Exit conditions The user successfully ends the rental.

Exceptions

- The user parks the car non in a safe area. In this case the system doesn't allow the user to end the rental.
- The user parks the car in a safe area but doesn't turn off the engine. In this case the system doesn't allow the user to end the rental.

- The user parks the car in a safe area, turns off the engine but he/her and eventually passengers don't exit the car or close car doors. In this case the system doesn't allow the user to end the rental.
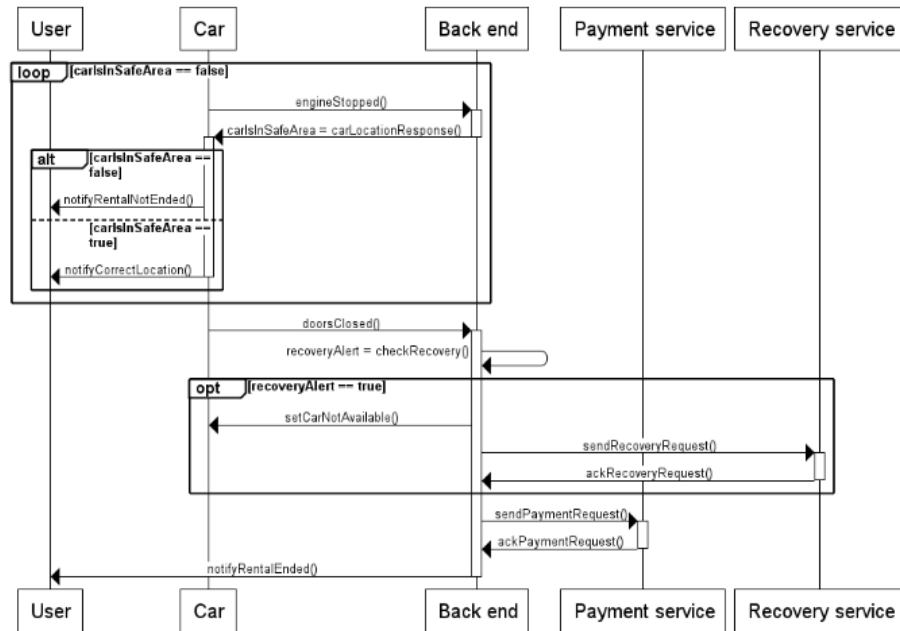

Figure A.7: The sequence diagram above describes the rental ending procedure.
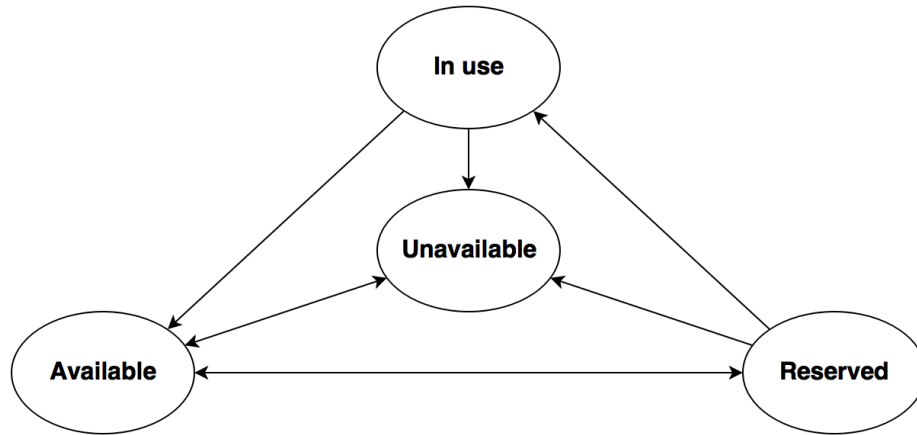
A.3 State diagrams

Figure A.8: This state diagram represent the possible transitions among the car states.
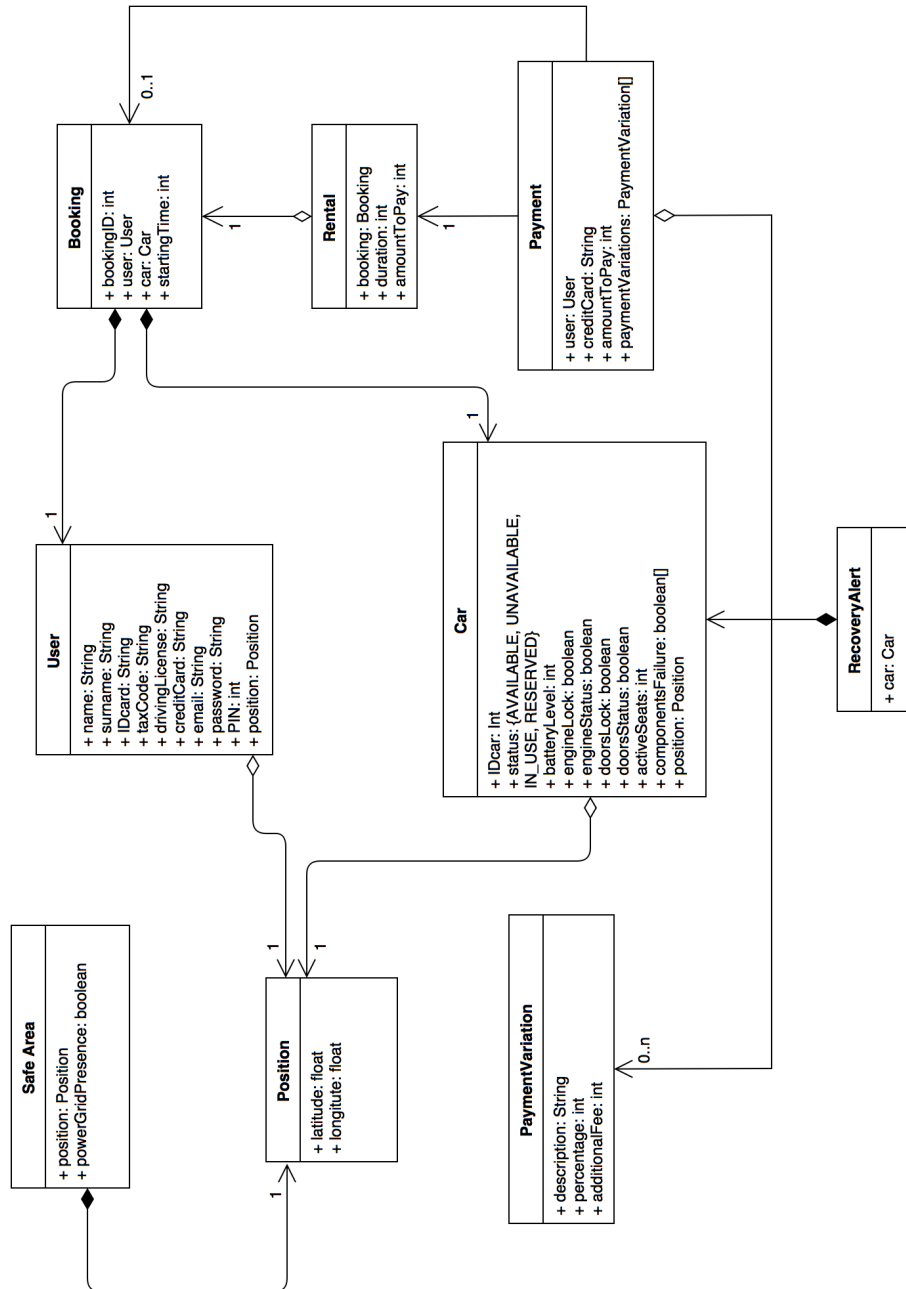
A.4 Class diagram

Figure A.9: This diagram represents the class structure of PowerEnJoy system.

Appendix B

Alloy Model

In this appendix a model of the PowerEnJoy system written in Alloy [5] is provided.

B.1   Signatures

```
open util/boolean

abstract sig CarStatus{}
one sig Available extends CarStatus{}
one sig InUse extends CarStatus{}
one sig Reserved extends CarStatus{}
one sig Unavailable extends CarStatus{}

abstract sig BatteryLevel{}
one sig BatteryLevelHigh extends BatteryLevel{}        // 100% -> 50%
one sig BatteryLevelMid extends BatteryLevel{}         // 49% -> 20%
one sig BatteryLevelLow extends BatteryLevel{}         // 19% -> 1%
one sig BatteryLevelEmpty extends BatteryLevel{}       // 0%

sig Car {
    IDCar : Int,
    activeSeats : Int,
    position : Position,
    status : CarStatus,
    batteryLevel : BatteryLevel,
    componentsFailure : Bool,
} {
    IDCar > 0
    activeSeats > 0 && activeSeats <= 5
}
```

```alloy
sig Position {
    latitude: Int,
    longitude: Int,
}

sig SafeArea {
    position: Position,
    powerGrid: Bool,
    nearToPowerGrid: Bool,   // boolean that indicates if there is at least
                             // one safe area with power grid in 3km range
                             // from the considered safe area
}

sig TaxCode {}

sig User {
    taxCode: TaxCode,
    position: Position,
}

sig Booking {
    bookingID: Int,
    user: User,
    car: Car,
    ended: Bool,
    elapsedTime: Bool,       // True if it passed more than hour
                             // since the user booked the car
} {
    bookingID > 0
}

sig Rental {
    booking: Booking,
    ended: Bool,
}

abstract sig Payment {}

sig PaymentRental extends Payment {
    rental: Rental,
}

sig PaymentFee extends Payment {
    booking: Booking,
}

sig RecoveryAlert {
    car: Car,
}
```

B.2 Facts

```
fact carsAreUnique {
    all c1, c2: Car | (c1 != c2) => c1.IDCar != c2.IDCar
}

fact positionsAreUnique {
    all p1, p2: Position |
        (p1 != p2) => (p1.latitude != p2.latitude) ||
        (p1.longitude != p2.longitude)
}

fact carsCannotBeInTheSamePlace {
    all c1, c2: Car | (c1 != c2) => (c1.position != c2.position)
}

fact safeAreaAreUnique {
    all s1, s2: SafeArea | (s1 != s2) => (s1.position != s2.position)
}

fact usersAreUnique {
    all u1, u2: User | (u1 != u2) => (u1.taxCode != u2.taxCode)
}

fact bookingsAreUnique {
    all b1, b2: Booking | (b1 != b2) => (b1.bookingID != b2.bookingID)
}

fact oneCarOneBooking {
    all b1, b2: Booking |
        (b1.ended = False && b2.ended = False && b1.car = b2.car) => (b1 = b2)
}

fact oneUserOneBooking {
    all b1, b2: Booking |
        (b1.ended = False && b2.ended = False && b1.user = b2.user) => (b1 = b2)
}

fact rentalIfNotElapsed {
    all r: Rental | r.booking.elapsedTime = False
}

fact oneBookingOneRental {
    all r1, r2: Rental | (r1 != r2) => r1.booking != r2.booking
}

fact feeIfElapsed {
    all p: PaymentFee | p.booking.elapsedTime = True
}
```

```
fact endBookingIfElapsed {
    all p: PaymentFee | p.booking.ended = True
}

fact paymentFeeAreUnique {
    all p1, p2: PaymentFee | (p1 != p2) => (p1.booking != p2.booking)
}

fact paymentRentalAreUnique {
    all p1, p2: PaymentRental | (p1 != p2) => (p1.rental != p2.rental)
}

fact payIfRentalEnded {
    all p: PaymentRental | p.rental.ended=True
}

fact endBookingIfEndRental {
    all r: Rental | (r.ended = True) => r.booking.ended = True
}

fact oneRentalOnePayment {
    all p1, p2: PaymentRental | (p1.rental = p2.rental) => p1=p2
}

fact endBookingIfUnavailable {
    all c: Car | some b: Booking |
        (b.car = c && c.status = Unavailable) => b.ended = True
}

fact endRentalIfUnavailable {
    all c: Car | some r: Rental | some b: Booking |
        (r.booking = b && b.car = c && c.status = Unavailable) => r.ended = True
}

fact alertIffUnavailable {
    all c: Car | (c.status = Unavailable) <=> some a: RecoveryAlert | (a.car = c)
}

fact alertsAreUnique {
    all a1, a2: RecoveryAlert | (a1!=a2) => (a1.car!=a2.car)
}

fact carIsReserved {
    all c: Car | some b: Booking |
        (b.car = c && b.ended = False) <=> (c.status = Reserved)
}
```

```
fact carInUse{
    all c: Car | some r: Rental | some b: Booking |
        (r.booking=b && r.booking.car=c && r.ended=False) <=> (c.status = InUse)
}

fact carIsUnavailable {
    all c: Car | some s: SafeArea |
        (c.batteryLevel = BatteryLevelEmpty ||
         c.componentsFailure = True ||
            (c.position = s.position &&
             s.powerGrid=False &&
             c.batteryLevel = BatteryLevelLow &&
             s.nearToPowerGrid = False)) <=> (c.status = Unavailable)
}
```

B.3   Predicates

```
pred show {
    #Car > 1
    #Booking > 1
    #PaymentFee > 1
    #RecoveryAlert > 1
}

pred batteryLowInSafeAreaWithNoPowerGridHasAlert {
    some c: Car, s: SafeArea, r: RecoveryAlert |
        (
            c.batteryLevel = BatteryLevelLow &&
            c.position = s.position && s.powerGrid = False &&
            r.car = c &&
            c.componentsFailure = False
        )
}

pred allRentalsHaveNonElapsedBooking {
    some r: Rental, b: Booking | (r.booking = b && b.elapsedTime = False)
}

pred bookingsElapsedHaveFeePayments {
    some b: Booking, p: PaymentFee |
        (b.elapsedTime = True && p.booking = b)
}
```

B.4 Assertions

```
assert noActiveBookingIfRentalEnded {
    all r: Rental | (r.ended=True) => no b: Booking |
        (r.booking=b && b.ended=False)
}

assert paymentIfRentalEnded {
    all p: PaymentRental | all r: Rental |
        (p.rental = r) =>  (r.ended = True)
}

assert noRentalActiveIfCarUnavailable {
    all c: Car | some r: Rental | some b: Booking |
        (c.status=Unavailable && r.booking=b && b.car=c) => (r.ended=True)
}

assert singleAlertIfCarUnavailable {
    all c: Car | (c.status=Unavailable) => one r: RecoveryAlert | (r.car=c)
}
```
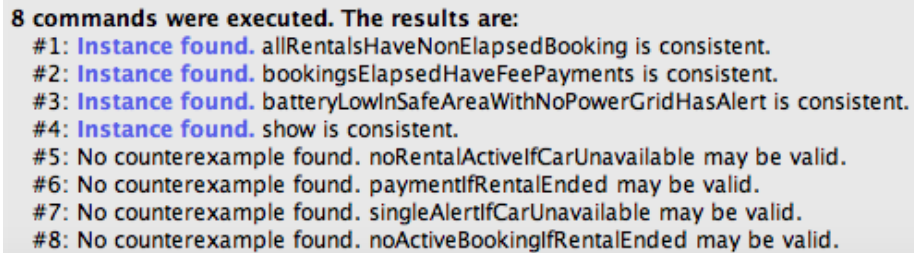
B.5 Commands and results

```
run show
run allRentalsHaveNonElapsedBooking
run bookingsElapsedHaveFeePayments
run batteryLowInSafeAreaWithNoPowerGridHasAlert


check noRentalActiveIfCarUnavailable
check paymentIfRentalEnded
check singleAlertIfCarUnavailable
check noActiveBookingIfRentalEnded
```

```
8 commands were executed. The results are:
  #1: Instance found. allRentalsHaveNonElapsedBooking is consistent.
  #2: Instance found. bookingsElapsedHaveFeePayments is consistent.
  #3: Instance found. batteryLowInSafeAreaWithNoPowerGridHasAlert is consistent.
  #4: Instance found. show is consistent.
  #5: No counterexample found. noRentalActiveIfCarUnavailable may be valid.
  #6: No counterexample found. paymentIfRentalEnded may be valid.
  #7: No counterexample found. singleAlertIfCarUnavailable may be valid.
  #8: No counterexample found. noActiveBookingIfRentalEnded may be valid.
```

Figure B.1: The image above is a screenshot of the Alloy Analyzer 4.2 console that successfully confirms all the commands executed.
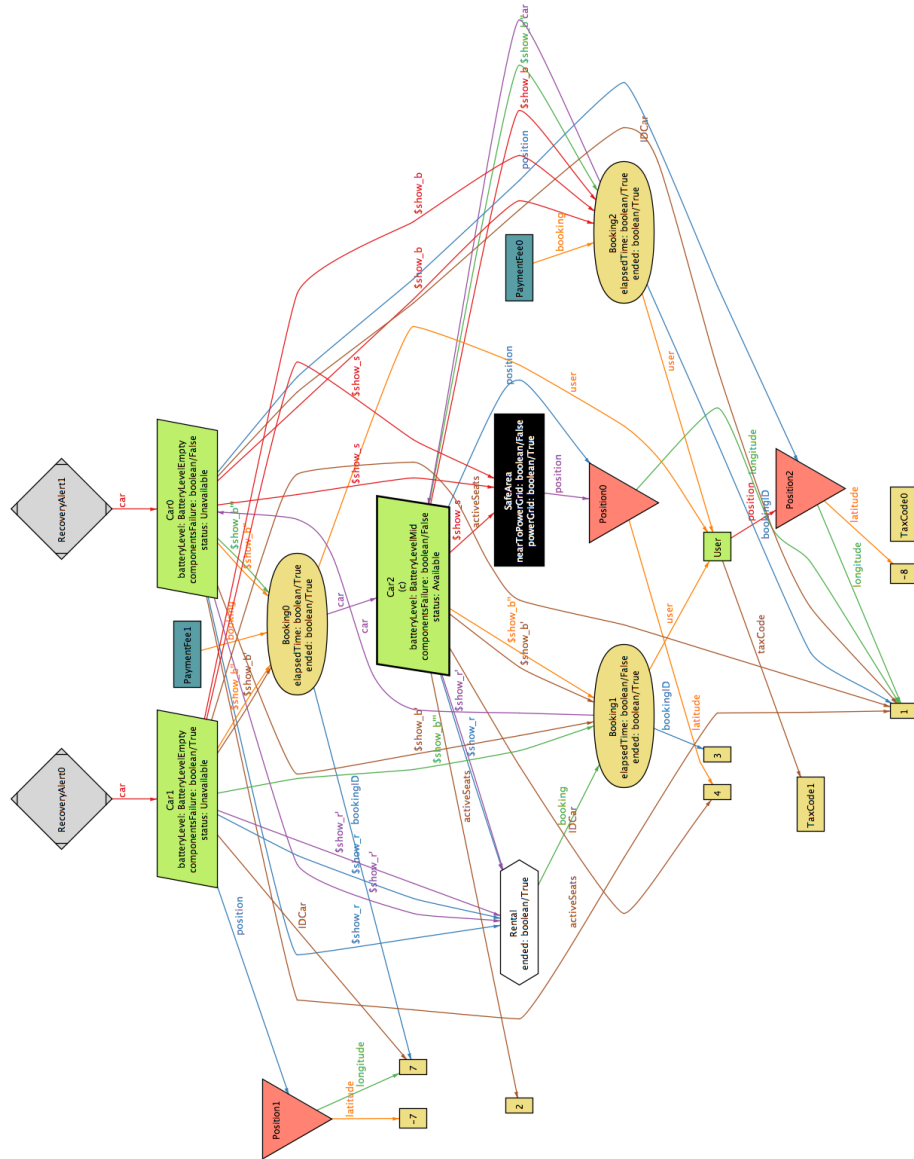
B.6 Instances generated

Figure B.2: The image above is a the instance generated by the Alloy Analyzer 4.2 after running the predicate 'show'.
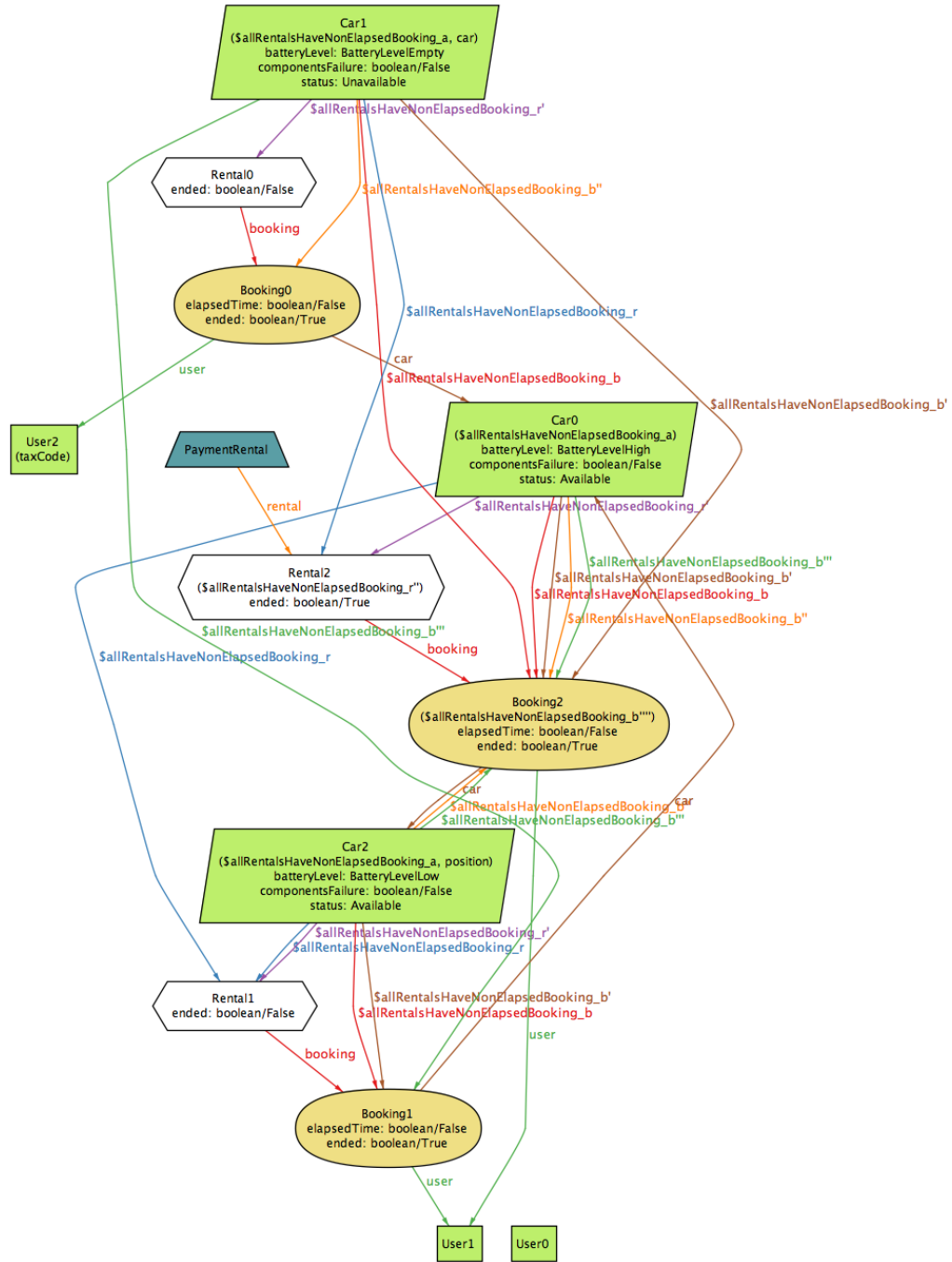
Figure B.3: The image above is a the instance generated by the Alloy Analyzer 4.2 after running the predicate 'allRentalsHaveNonElapsedBooking'.
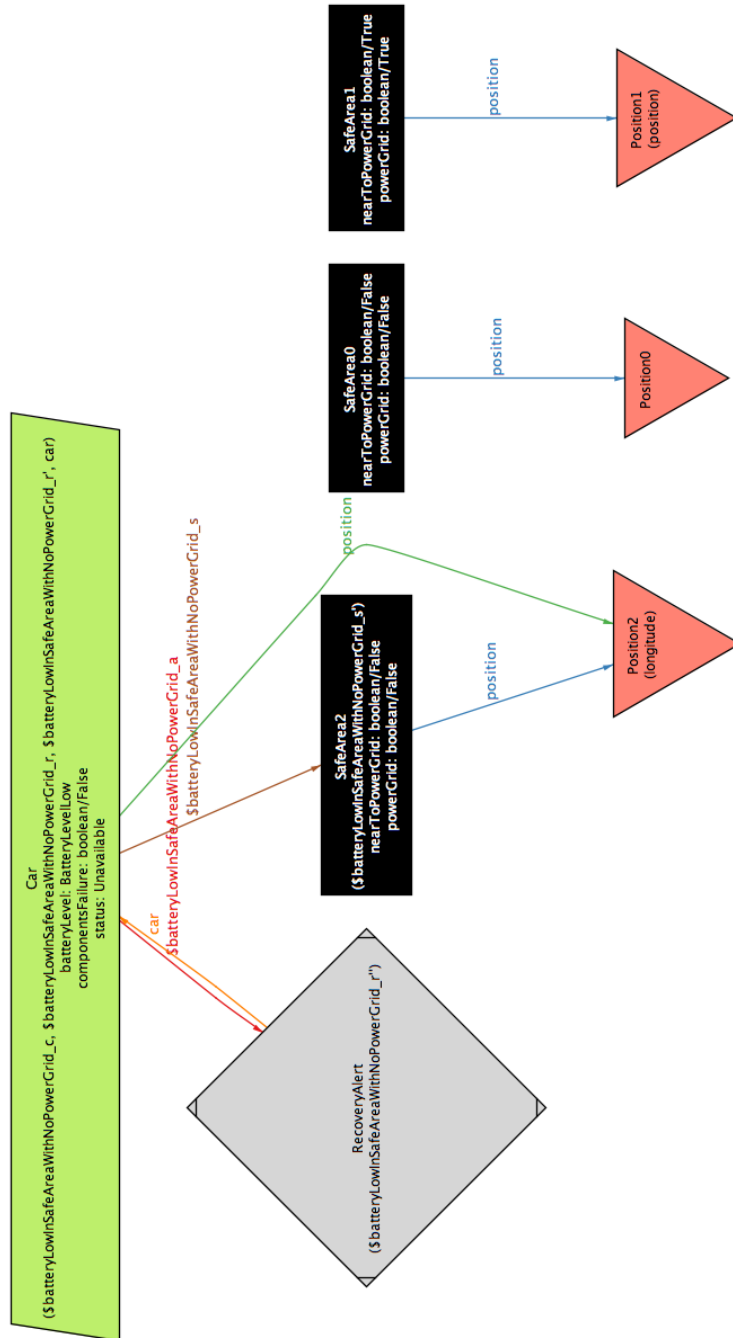
Figure B.4: The image above is a the instance generated by the Alloy Analyzer 4.2 after running the predicate 'batteryLowInSafeAreaWithNoPowerGrid-HasAlert'.
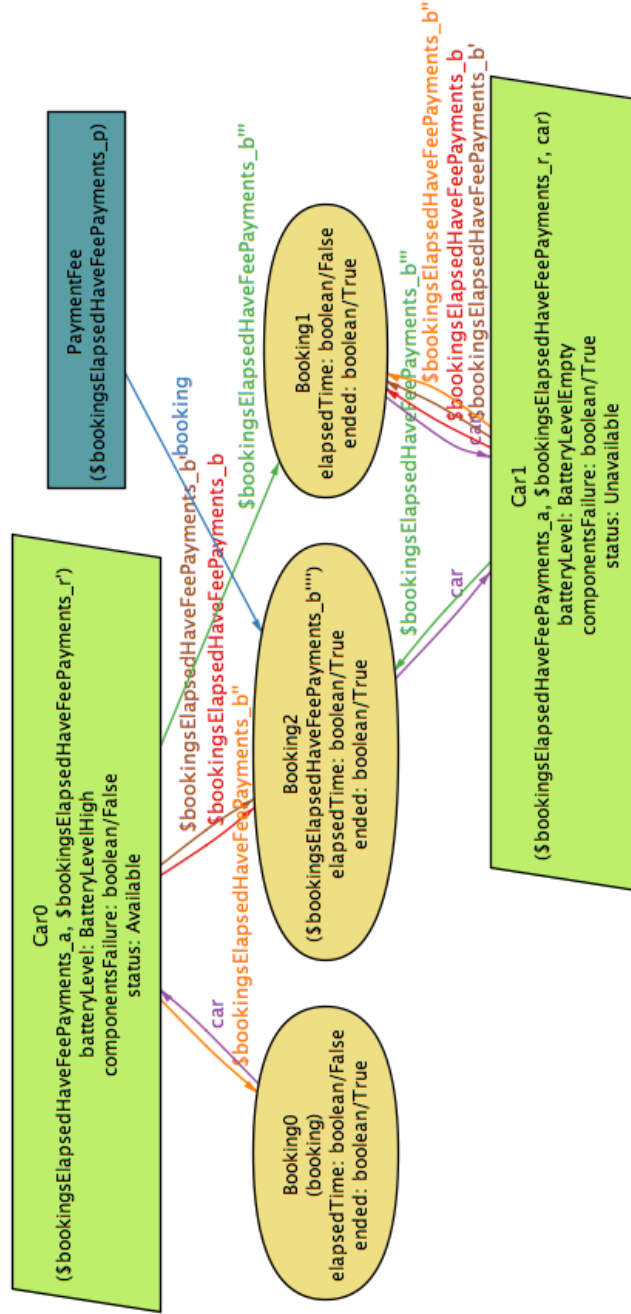
Figure B.5: The image above is a the instance generated by the Alloy Analyzer 4.2 after running the predicate 'bookingsElapsedHaveFeePayments'.

List of Figures

Bibliography

[1] Apple Inc. iOS. Available at `http://www.apple.com/ios/` (November 26, 2016).

[2] Google Inc. Android. Available at `https://www.android.com/` (November 26, 2016).

[3] L. Mottola and E. Di Nitto. *Assignments AA 2016-2017*. Politecnico di Milano.

[4] IEEE Recommended Practice for Software Requirements Specifications. *IEEE Std 830-1998*, pages 1–40, Oct 1998.

[5] Daniel Jackson. "Alloy: a language & tool for relational models". Available at `http://alloy.mit.edu/alloy/` (November 13, 2016).