Power EnJoy

Requirements Analysis and Specification Document

Version 1.2.0

Redaelli Marco 877622 Zanoli Francesco 877471

27/02/2017

Contents

Figure Contents

Chapter 1

Introduction

This document represents the Requirement Analysis and Specification Document (RASD). The main goal of this document is to completely describe the system in terms of functional and non-functional requirements, analyze the real need of the customer to modelling the system, show the constraints and the limits of the software and simulate the typical use cases that will occur after the development. This document is intended to all developers and programmers who have to implement the requirements, to system analyst who want to integrate other system with this one, and could be used as a contractual basis between the customer and the developer.

1.1 Scope

The software described in this document is a new digital management system for car-sharing service that exclusively uses electrical cars. Nothing like this software already exists so the development of it need to start from zero. The software's main goal is the simplification of the car sharing service and the management of the reservation and the usage of electric cars. It can be applied to different small and big city and even in a large urban area and it's composed by a mobile application who permit the user to interact with the system and a web platform that explain to the users all the procedure to access to the service. There are two type of users:

- Visitors: all the visitors have access to the login and registration page on the mobile application and will be also able to visit the information part of the website that include FAQ page and Home page
- Registered user: this user can, after the registration, reserve a car, drive it, park and charge in the predefined area and finally it can reports problems of the system.

After the login the user can look for nearest car and reserve it, he will than have 1 hour to get to the car open it and start the renting. At the end of his ride

the user have to park the car in a safe area and if it's necessary/possible plug it into recharge. Besides the specific user interfaces for users, the system offers also APIs to enable the development of additional services on top of the basic one.

1.2 Definitions, acronyms and abbreviations

1.2.0.0.1 Definitions

- User: Someone registered to the system
- Visitor: user that is not registered nor logged in
- System: the union of software and hardware to be developed and implemented
- Parking area: it is a reserved area, predefine by the system, where I can park the car but I cannot recharge it.
- Safe area: it is a reserved area, predefine by the system, where I can park the car and plug it into charge.
- Free car: The car is visible on the map and available for a reservation
- Reserved car: The car is not visible on the map and the user who reserved it didn't access yet.
- Internet: The Internet is the global system of interconnected computer networks that use the Internet protocol suite (TCP/IP) to link billions of devices worldwide.
- Service: The functionalities offered by myTaxiService via software applications to passengers and taxi drivers.
- System: The whole hardware and software parts that together deliver myTaxiService as a service to passengers and taxi drivers.
- Smartphone: A mobile phone with an advanced mobile operating systems, which combines features of a personal computer operating system with other features useful for mobile or handheld use.
- Application: A computer program (i.e. a piece of software) that per- forms a group of coordinated functions, tasks, or activities for the benefit of the user.
- World Wide Web: Also referred as web, is an information space where documents and other web resources are identified by URLs, interlinked by hypertext links, and can be accessed via the Internet.
- Web Page: A web document that is suitable for the World Wide Web and the web browser.

- Web Server: An information technology that processes requests via HTTP and stores, processes and deliver web pages to clients.

1.2.0.0.2 Acronyms

- RASD: requirements analysis and specification document
- AES: Advanced Encryption Standard
- FIFO: First In First Out
- ETA: estimated time of arrival
- API: application programming interface
- GPS: Global Positioning System IEEE: Institute of Electrical and Electronics Engineers.
- SRS: Software Requirements Specification, document based on IEEE 830.
- RASD: Requirements Analysis and Specification Document, also known as SRS.
- UML: Unified Modeling Language.
- Wi-Fi: Wireless Fidelity, technology for local area wireless computer networking technology based on IEEE 802.11 standard.
- WiMAX: Worldwide Interoperability for Microwave Access, technology based on IEEE 802.16 standard.
- API: Application Programming Interface, a set of public accessible functions, protocols and tools provided by a specific application for building software applications.
- GPS: Global Positioning System.
- DBMS: Database Management System.

1.3 References

- Software Engineering 2 Project AA 2016/2017: Assignments AA 2016-2017
- IEEE Standard 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- IEEE Standard 1016 tm -2009 Standard for Information Technology-System Design-Software Design Descriptions.
- RASD TaxiDriver example 2.pdf (past year's project)
- Wikipedia, the free encyclopedia

1.4 Overview

This document is essentially structured in four parts:

- Introduction: Overview of the RASD document. Specifically provides both a description of the software, and a set of information about the organization of the document.
- Overall Description: Provides a high-level description of the software requirements, not describing specifically the main aspect of them, but only providing a background of those requirements. The main purpose of this section is make the requirements easy to understand.
- Specific Requirements: Shows all of software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test the system requirements.
- Appendix: it provides informations that are not considered part of the actual RASD. It includes: software and tools used, alloy implementation, project group organization

Chapter 2

Overall description

2.1 Product perspective

The software *PowerEnJoy*is a completely new product, not based on previous ones. However, it relies on location data received via Internet from each user smartphone application: all the involved smartphones already have a GPS antenna installed inside, that communicates their position to the service. Being distributed application, *PowerEnJoy*requires a Internet connection in order to work properly. This project provides interesting features for users who are logged-in. Non registered passengers can only register to *PowerEnJoy*, the procedure and the definition of the different actions done by the users are listed and well explained in the chapter bellow.

2.2 Product functions

The system allows different kinds of user to perform different actions. In particular:

- Visitors can simply register or log in.
- Logged user can reserve and rent a car, plug a car into a safe area and finally communicate with the customers service in case of any problems during one of this operations

The user's action as the all the sequence is it possible to do with them is explain in the figure belove

2.2.1 Goals

Here are described the software's high level goals:

G_1 Allow visitor only to sign-up or sign-in

G_2 Allow user to log in

G_3 Allow user logged in to rent a car

G_4 Allow user logged in to book a car in a certain location

G_5 Allow user logged in to see the reservation's confirmation and the time of expiration

G_6 A non registered users can only register once to the service.

G_7 A registered user can login to the service only when not logged in.

G_8 A registered user can logout from the service only when logged in.

G_9 A user who request a rent can abort the process when ever he/she wants

G_10 A user can get discount or overtaxes from his/her last rent

G_11 Further services can be built on the top of the existing one through a set of given APIs.

2.2.2 UML

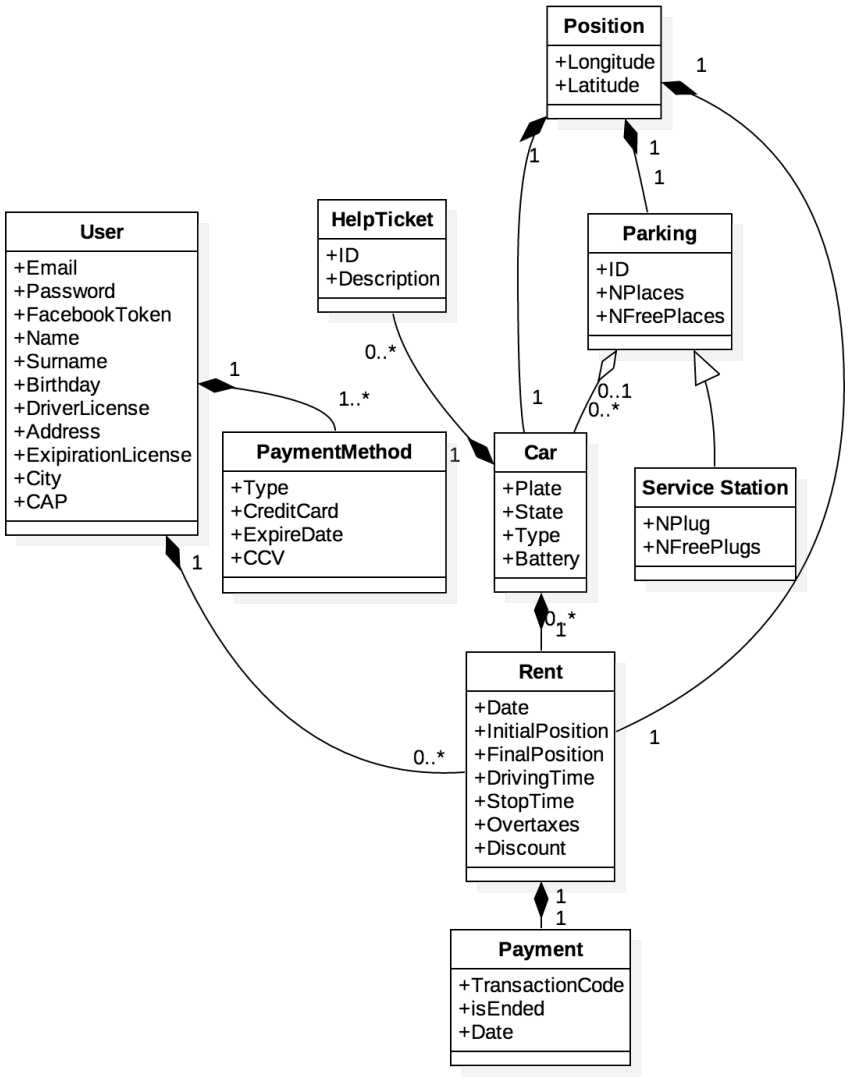

Figure 2.1: UML Class Diagram

2.3 User characteristics

The system wants to give to the users an easy way to interact with it. This without undermine the complexity of the problem and achieving all the goal imposted by the stakeholder. To do that, users must be able to install the mobile application from the store, besides, their phone must be provided with a GPS, a camera and an internet connection.

2.4 Constraints

2.4.1 Regulatory policies

PowerEnJoy is a service provided by a private company. The user, who reaches this service has to agree to License Agreement rather than Privacy policy and Terms of use at registration.
The user access and use of the services constitutes his/her agreement to be bound by these Terms, which establishes a contractual relationship between him/her and PowerEnJoy. If user does not agree to these Terms, he/she may not access or use the services. PowerEnJoy may immediately terminate these Terms or any services with respect to him/her, or generally cease offering or deny access to the Services or any portion thereof, at any time for any reason.
PowerEnJoy collects the information provided by the user, for example when creating or making changes to services on demand, through contact with customer service or during other communications. This information may include: name, email, phone number, mailing address, payment method, delivery receipts and other information user choose to provide. The personal data will be used only to provide the services requested.
User is responsible for obtaining the data network access necessary to use the services. User mobile network's data and messaging rates and fees may apply if he/she accesses or uses the services from a wireless-enabled device. User is responsible for acquiring and updating compatible hardware or devices necessary to access and use the service and applications and any updates thereto.
PowerEnJoy does not guarantee that the services, or any portion thereof, will function on any particular hardware or devices. In addition, the services may be subject to malfunctions and delays inherents in the use of the Internet and electronic communications.

2.4.2 Hardware limitations

PowerEnJoy defines the minimum requirements for using web and mobile applications.

- *Web application*
  See that the web application is a static web site used as advertisement it does not require a minimum access speed and it is built to Browse from

Internet explorer 8 or more, intending all the different version of Chrome, Firefox and Safari exit during this time

- *Mobile application*
  Operating system: Android, iOS, Windows Phone
  Memory: 512MB RAM
  Hard drive: 40MB of free space
  GPS navigation system
  Web access at the minimum speed of 1Mbps

2.4.3 Interfaces to other applications

PowerEnJoy provides APIs to enable development of additional software on this platform. Furthermore PowerEnJoy manages the communication between itself and the payment management system, to do so the system use the API of the most common payment services used on the web. It is request a system that can work automatically without asking each time the permission to the user. The system sends a payment request with the information of the rent and the amount to pay and the external payment method has to answer with a code that indicate the result of the transaction. Related to the policy of the external system the money that PowerEnJoy earn from the users has to be returned from the payment system no more than a moth after the rent.

2.4.4 Parallel operation

PowerEnJoy supports parallel operations cause of the nature of service. Many users can access to the service at same time thus system and database have to work with parallel requests.

2.4.5 Reliability requirements

PowerEnJoy relies on network connections thus reliability issues are equivalent to performance issues. However, the application should not corrupt server data as a result of its actions. The system has to guarantee whole-time availability.

2.4.6 Criticality of the application

PowerEnJoy relies on network systems and servers. Scheduled downtime is acceptable. This system requires a generator backup and redundant power in the event of failover.

2.4.7 Safety and security considerations

PowerEnJoy guarantees secure communications through AES encryption algorithms.

2.5 Assumptions and dependencies

This assumptions are related to the domain and to the environment where the user act.

A_1 The phone must have a camera and the application must have the right to use it

A_2 The user who reserve and/or rent the car is legally responsible for everything that happens during this time

A_3 The validity of the payment method is verifiable as the correct end of a payment.

A_4 The car is furnished by some device that allow the system to completely control and communicate with it. As open it from distance, control the passengers number, check its position, check its battery state.

A_5 The user's GPS must be switched on using the mobile app

A_6 The user's GPS must work rightly

A_7 The users must have a working internet connection

A_8 There are a method to charge the battery of the car

Chapter 3

Specific requirements

In this chapter are analyzed all the requirements related to the system. Each section corresponds to a specific category of requirement. In particular the section 3.1 explain the requirements and the constraints the system must to respect to achieve the goal imposed by the customer, as the user interface.

3.1 External interface requirements

3.1.1 User interfaces

The interface of PowerEnJoy can be both for web application and mobile application. From figure 3.1 to figure 3.4 are presented some of the most important pages and screens of PowerEnjoy. In particular in figure 3.3 is showed the home page of the PowerEnJoy's application, it is composed by a map and some marker located on the map depending of the position of the car. It's important to understand that only the car in a free state are showed within the map. The figure 3.4 appears whereas when the user clicks on a marker in the map and shows the information of the selected car and a button to reserve it.

- Login Page

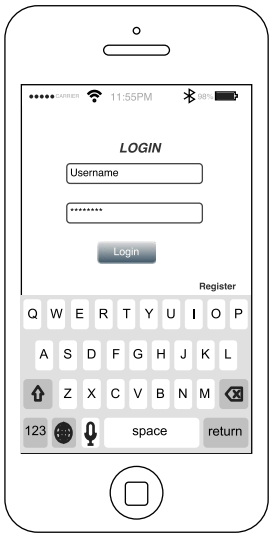

Figure 3.1: Login page

- Registration page

Figure 3.2: Registration page

- Home page or Map page

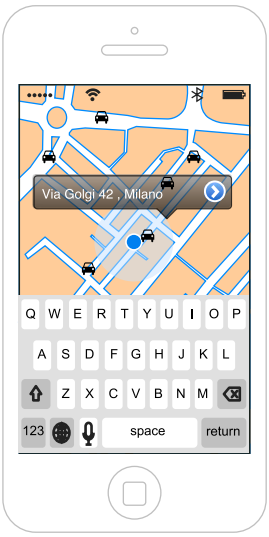

Figure 3.3: Home page or Map page

- Car information page

Figure 3.4: Car information page

3.2 System Features

3.2.1 Registration

Visitors can register to PowEnJoy through mobile application. This operation requires the visitor to fill a registration form with personal data and accept PowEnJoy terms and conditions, including personal data policies, according to local law. The system requires the visitor personal information as name, surname, and birthday, payment information (as a credit card or a paypal account) and proof of the possesion of a valid driver license. If any of the previous requirements are not met or any input is invalid, the registration fails and the system asks the visitor to repeat the process. Otherwise, a verification email containing the password of the account is sent to the provided email address. To validate his account the visitor needs to login one time with the provided password.

3.2.1.0.1 Use Case

Name: User registration

Actors: Visitor

Entry conditions: There are no entry conditions

Flow of events:

- The visitor arrives to the home page of the application, as is not logged in ,is redirected to the login/registration page
- The visitor enters his personal information, his driver license, a photo of his driver license and some payment method
- The visitor clicks on the confirm button
- The application suggests the user to read his emails to receive the password
- The visitor logs-in after reading the password

Exit conditions: The visitor is redirected to the home page of the application

Exception: The information furnished by the visitor are not correct or ambiguous as the following case:

- The Email has not format-correct
- The Birthday is not at least eighteen years ago
- The Payment method is not valid
- The informations of the driver license don't correspond with the informations furnished by the visitor

- The Driver license is not valid

Also the visitor could had forgot to enter some requested camps or to accept the Terms and Conditions. In all this case, the system does not send any mail to the visitor but notifies him that an error has been made and allows to input the incorrect data again

3.2.1.0.2 Scenario

Meg is a student. She has heard about PowEnJoy and, finding it an easy and ethical way to travel, wants to subscribe to it. Therefore, she downloads the mobile application from the store and clicks on Register in the main screen. She fulfils the form, accepts the term and conditions and she click Confirm. However, the system cannot verify Meg's driver license because she forgot to put the photo that prove the possession of it. It therefore asks Meg to take the picture from her mobile's camera. Once she has enter everything correctly she click on Confirm, this time the application valid his credential and tell to Meg to check her emails, she will find the confirmation of the correct registration and the given by the software. Meg read her emails and can finally open the application again and login with the given password and the email she gave before.

3.2.1.0.3 Description and Priority

This feature is of high priority, it is essential for the system to be.

3.2.1.0.4 Functional requirements

- **FR1:** Visitor can abort the registration process at any time.
- **FR2:** The password in the email must be used within 1 day, otherwise the registration is deleted along with the visitor's info.
- **FR3:** Registration form contain the following information (fields):
  - Email address.
  - First name.
  - Surname.
  - Address.
  - City.
  - Postal Code.
  - Credit card code.
  - Expiration date of the credit card.
  - Secure code of the credit card.

  - Driver license code.
  - Expiration of driver license.
  - Photo of a driver license.

- **FR4:** Email address cannot be the same as ones from other PowerEnJoy users.
- **FR5:** The photo of the driver license must be taken by the camera of the mobile.

3.2.1.0.5 UML Sequence Diagram

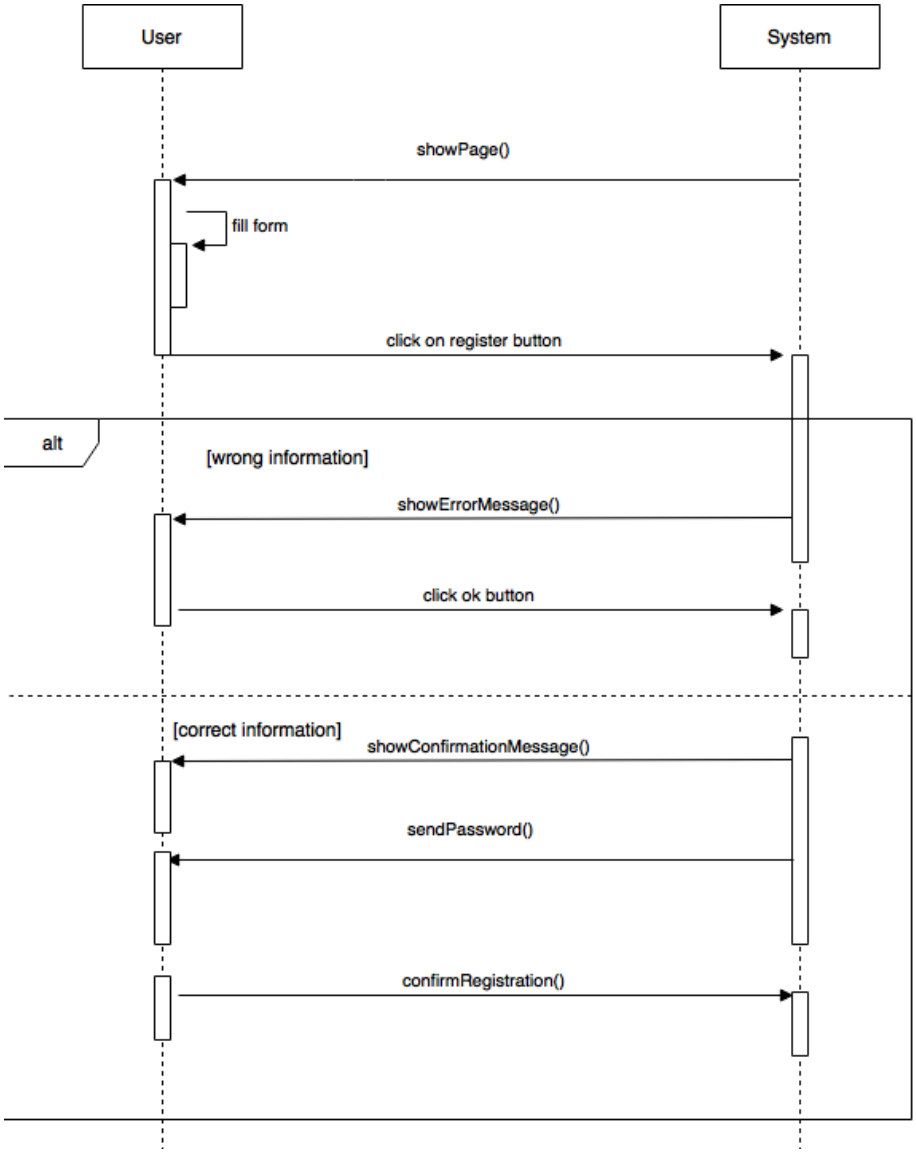

Figure 3.5: Registration sequence

3.2.2 Login

Visitors on PowerEnJoy mobile application may access to an existing registered user account providing its corresponding email address and password. In case that the submitted info do not match with any existing account info, the system notifies the visitor that the email address doesn't exist, or that it exists, but the submitted password is wrong. In case a user forgets his/her password, the system allows him/her to retrieve it, automatically creating a new password, setting it as the user's one and sending it to the provided email address.

3.2.2.0.1 Use Case

Name: User Login

Actors: User

Entry conditions: There are no entry conditions

Flow of events:

- The user arrives at the Login page of the mobile application.
- The user inputs his email address and his password.
- The user clicks on the log in button.
- The system redirects the user to the home page.

Exit conditions: The user is successfully redirected to the application home page.

Exception: The email and/or the password furnished by the user are not correct. In this case, the system does not redirect the user to the home page but notifies him that an error has been made and allows to input his email and password again. The user can also forget his/her password, in this case he/she can asks to generate another password and received it on his/her personal email address.

3.2.2.0.2 Scenario

1. Freddy is user of PowerEnJoy. He has already downloaded the application from the store and he has already done the registration from the application. He cannot remember the password given from the system during the registration time. Therefore he opens the app on the home page and he is redirect to the login page. He click then on the forget password link and the application ask him his email address. He insert the email address and then the application show another message telling the user to check the emails. Once he has received the new password, Freddy can finally open again the application and login with the his email address and the new password.

2. Eleonor is a lawyer familiar with the PowerEnJoy system, she have recently changed phone and she has already download the application again. She open the application and she is redirect in the login page. she fills both fields and clicks on "Log in". The system verifies her info: the operation ends successfully, and she gains access to the user homepage.

3.2.2.0.3 Description and Priority This feature is of high priority, it is essential for the system to be.

3.2.2.0.4 Functional requirements

- **FR6:** Visitors must fill the "email field" with an existing email address in order to successfully log in.
- **FR7:** Visitors must fill the "password" field with the only password correspond- ing to the submitted email address in order to successfully log in.
- **FR8:** The system will ignore log in requests if at least one of the "email" and "password" fields are left blank.
- **FR9:** The system allows visitors to retrieve their password if they forget it, by clicking "Forgot password".
- **FR10:** The system requires visitors to submit an existing email address in order to retrieve their password.
- **FR11:** The system will take care of assigning the user a new password, when he/she states to have lost the previous one.
- **FR12:** The system will take care of sending to the email address submitted by the visitor the new assigned password, when he/she states to have lost the previous one.
- **FR13:** The system allows visitors to retrieve their password once a day.
- **FR14:** The system remember the user's credential until the user decide to logout.

3.2.3 Reserve car

Logged user on PowerEnJoy can look for cars near his/her position, or next to a specify address, and reserve one for a rent. This operation is possible using the map on the home page of the application that indicates with a marker the position of the car, only the cars that are in a free state can be reserved and are visible on the map

3.2.3.0.1 Use Case

Name: Reserve a car

Actors: User,

Entry conditions: There is at least a car not reserved neither used.

Flow of events:

- The user arrives at the home page of the application that shows the map with the markers of the cars.
- The user choose a car.
- The user clicks on the marker of the car chosen.
- The user clicks on the "Reserve" button
- The application shows to the user the time remained to start the engine and the position of both the actors.
- The user arrives next to the car.
- The user clicks on the button "Open the car"
- The car is opened by the system
- The user get into the car and start the engine.

Exit conditions: The user successfully start the engine of the car

Exception: Two users reserve the same car in a really small difference of time, the system in this case will delete the reservation that is requested later.

3.2.3.0.2 Scenario

Francis needs to go home from a dinner with his friends. It is late and there are public transport anymore. He is already registered and successfully logged-in in the PowerEnJoy application. He decided to reserve a car using the application. He opens the application and he is directly redirect to the application home page that contains the map with the markers of cars near him. He chooses a marker and he click on it. The app shows him the information of the car as its battery charge and its position. The car is really close to him therefore he clicks on the

button reserve and he moves next to the car. Meanwhile the application shows him the a timer, the vehicle registration plate, his position and the position of the car. Once the Francis arrives next to the car the app shows him a button to open the car. The rent starts when Francis starts the engine of the car.

3.2.3.0.3 Description and Priority

This feature is of high priority, it is essential for the system to be.

3.2.3.0.4 Functional requirements

- **FR15:** A car changes is state from *"Reserved"* to *"In use"* only when the engine starts
- **FR16:** A car can be reserved and showed on the map only if its state is *"Free"*
- **FR17:** A car stays in the *"Reserved"* state for at maximum one hour, if it's not picked-up it return to the state *"Free"*
- **FR18:** Each car have a precise position
- **FR19:** An user can open the car through the app only if it is near to it
- **FR20:** Each user can reserve only one car at the same time
- **FR21:** A car can be reserved from only one user at the same time

3.2.3.0.5 UML Sequence Diagram

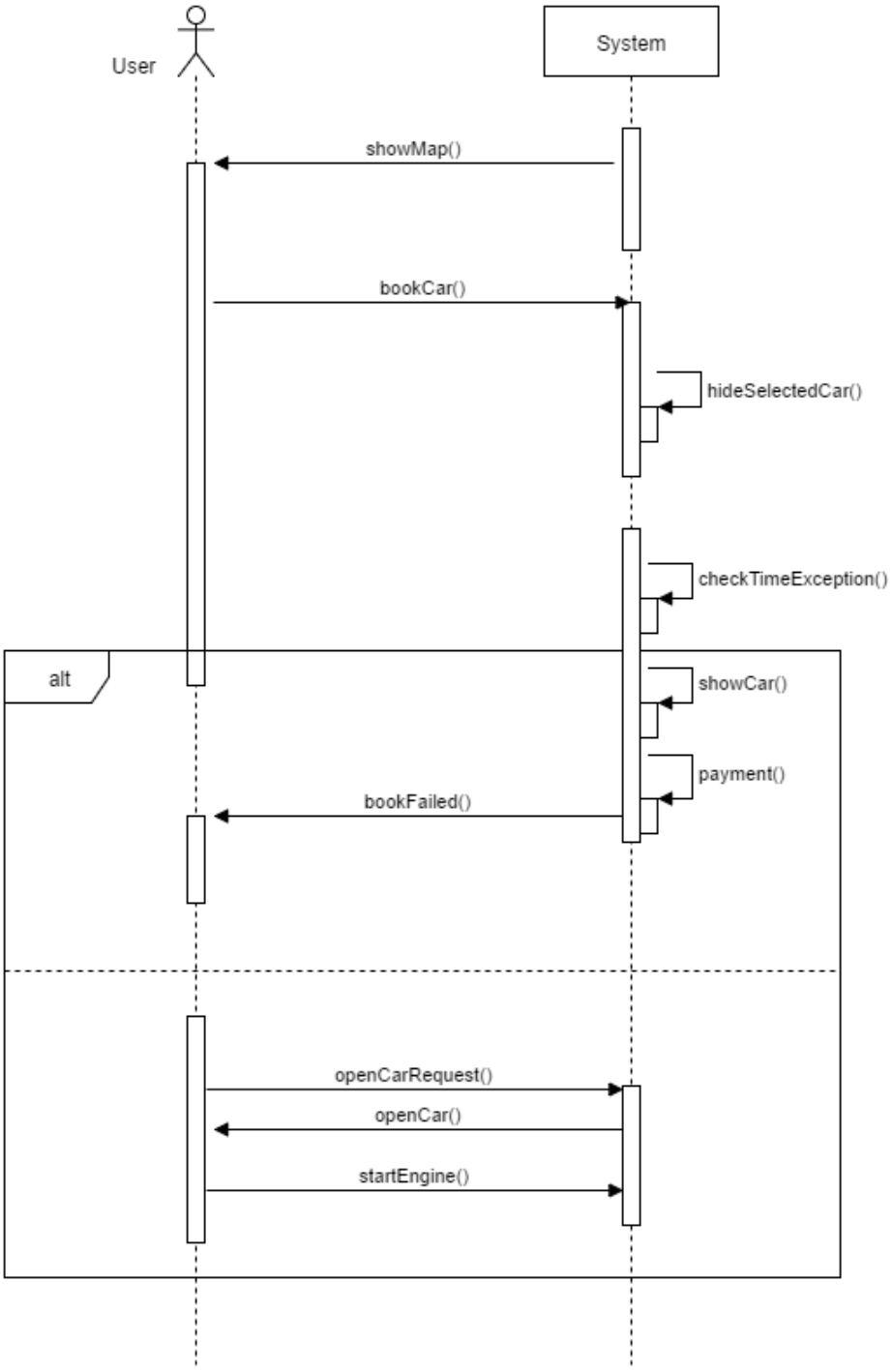

Figure 3.6: Car reservation

3.2.4 End a rent

Once the user has finished his/her ride he/she have to park the car in a safe area or a parking, stop the engine and get off the vehicle. The system checks if all the requirements to end the rent are respected and closes the car. The system waits 5 minute to let the user plug the car if he/she wants to. After the wait of time the system sends the payment request to the external payment system. The end of a rent can also provide some discount on the final payment or the add some overtaxes.

3.2.4.0.1 Use Case

Name: End a rent

Actors: User, Car

Entry conditions: The car is in the state In Use

Flow of events:

- The user stop the engine of the car and get out of the vehicle.
- The car check if all the doors are closed and there is no one into the it.
- The system close the car
- The system wait 5 minute
- The system send the request for payment to the external system
- The external system answer to the request
- The system notify the user about the end of the rent with a message on the app

Exit conditions: The user successfully ends the rent

Exception:
- The car is not parked in predefined area, in this case the system will not allow the user to end rent and close the car.
- The payment request failed, in this case the system blocks the user and he/her will not be able to use the system again until he complete the payment
- The car is stopped but the user still inside or a door is open, in this case the rent will not end and the user is charged as he/she would still driving.

3.2.4.0.2 Scenario

1. Isa is an habitual user of PowerEnJoy. She picked-up a car to cross the city and be ecologic with the system. She has finished her ride and she wants to end the rent. She parks the car near a safe area near her destination and she stops the engine of the car. The safe area, as defined, has a plug to recharge the car so Isa plugs the car just after she has stopped the car. The system notify Isa of the correct end of her rent and show her the final bill, that contain a discount of 30%, with a message on the application.
2. Laura took a car of PowerEnJoy to get home with her family, her husband and her tow child. She park the car in a parking but unfortunately the battery of the car is at 10% and there are no safe area next to Laura's house, the most near is 3.2 Km away. Laura end her rent stopping the engine of the car and receives a message from the app that shows her an overtaxes of 30% due to the position of the parking and the state of the battery life of the car.
3. Mary rented a car to go home from her university, once she arrives next to her destination she park the car into a parking, she stop the engine and get off the car. Unfortunately the chosen area is not an allowed parking area and the system does not let the her close the car even if nobody is into it. Therefore she moves the car into another parking area and this time, once she followed the procedure, the system close the vehicle and wait 5 minute before charging Mary for her rent. Once the payment is successfully ended the system send a notification to Mary with the resuming of her rent and the bill.

3.2.4.0.3 Description and Priority

This feature is of high priority, it is essential for the system to be.

3.2.4.0.4 Functional requirements

- **FR22:** If the system detects the user took at least two other passengers into the car, the system applies a discount of 10% on the last ride.
- **FR23:** If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride.
- **FR24:** If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.
- **FR25:** If a car is left at more than 3 KM from the nearest power grid station or with more than 80% of the battery empty, the system charges

30% more on the last ride to compensate for the cost required to re-charge the car on-site.

- **FR26:** The user has five minutes to plug the car if he/she wants a discount
- **FR27:** The rent end-up only if the car is closed, parked in a parking/safe area and there is anybody in the vehicle.
- **FR28:** The car can be closed only if it is turned off, there is no one in it and if it is parked into a safe/parking area
- **FR29:** The discount can be apply only if there are not any overtaxes
- **FR30:** Only one discount can be apply to the rent, the one who is the most convenient for the user.
- **FR31:** The user has 2 minute before the check of the requirements condition

3.2.4.0.5 UML Sequence Diagram

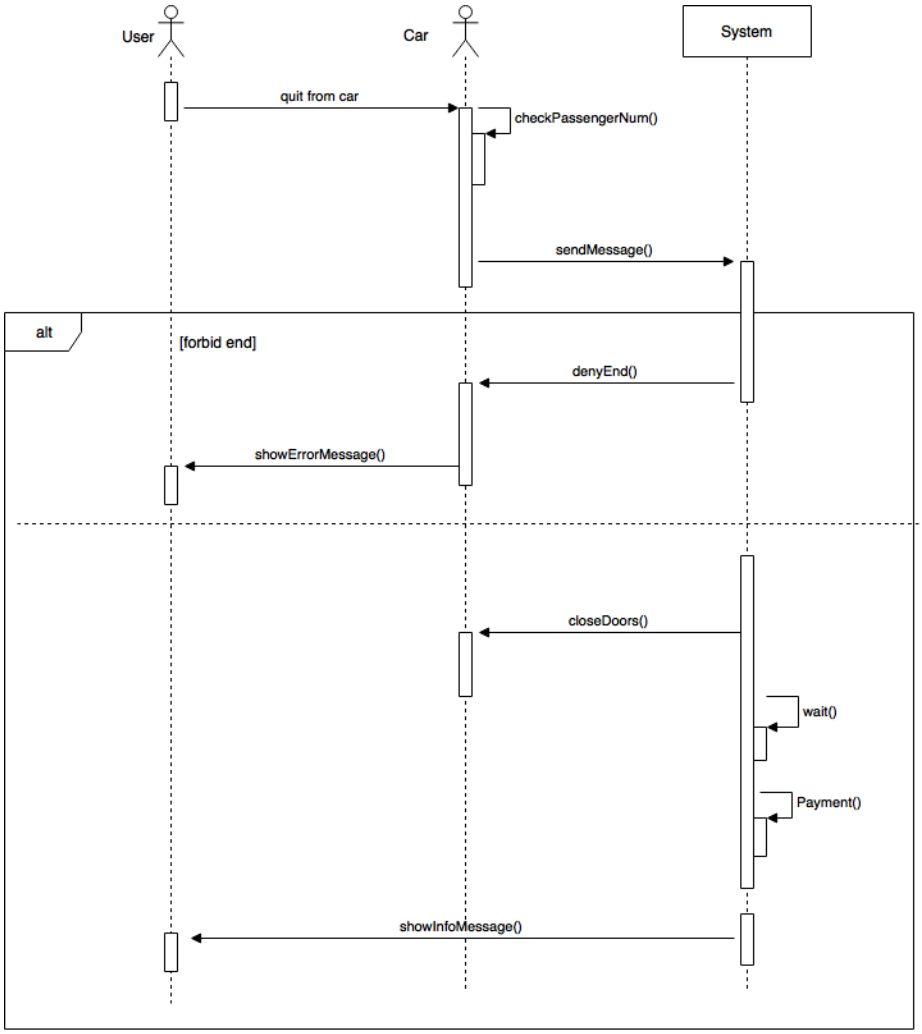

Figure 3.7: End rent sequence

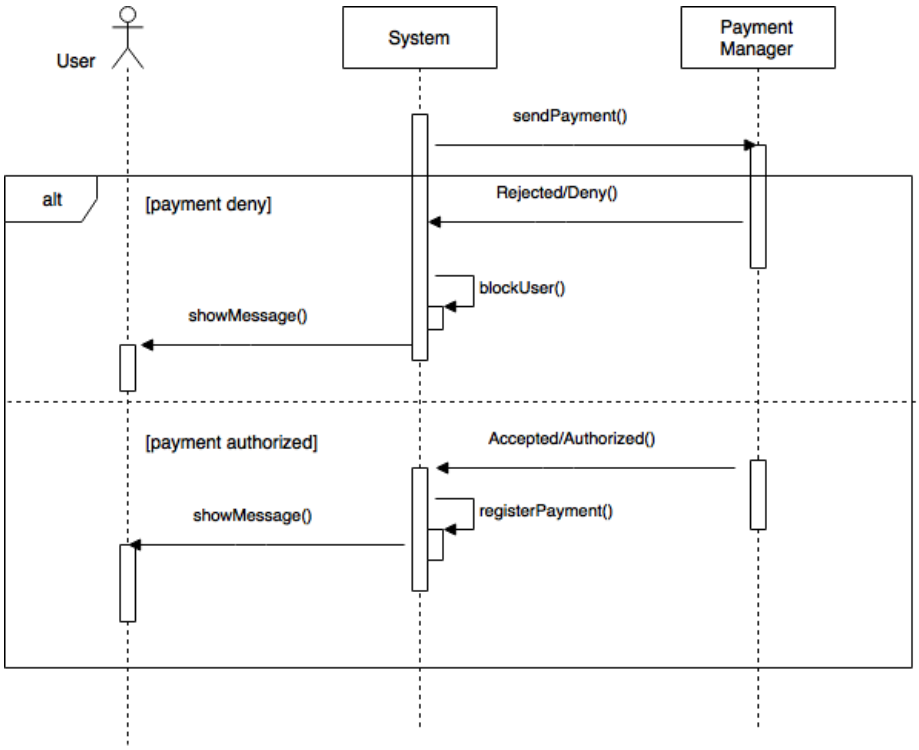

Figure 3.8: Payment sequence

3.2.5 Report problems

Every logged-in user can report a problem to the PowerEnJoy team during the all time of use of the system. In particular the user has a button to immediately contact the customer service during:

- The reservation of a car, in case the car is not opened by the system.
- The rent of a car, in case of accident or problem due to the system.
- The charge of a car, in case some safe area is not working correctly.
- The payment, in case of some error appeared during the payment time.

3.2.5.0.1 Use Case

Name: Report problems

Actors: User, External Customer service

Entry conditions: The user needs to be logged into the application and some part of the system must have a malfunction.

Flow of events:

- The user clicks on the button help in the menu of the application
- The phone automatically call the customer service
- A customer service operator answer to the user and help him/her to solve the problem or open a repair request on the car

Exit conditions: The user successfully resolve or notify the problem

Exception: The information furnished by the user are not correct or ambiguous and the operator cannot help the user solving his/her problem. In this case if the operator will notify to the external system a reparation needed. The user have no credit in the phone to call the customer service, in this case the app will call the service through the internet connection

3.2.5.0.2 Scenario

1. Marc is a logged user who has already reserved a car. The rented car is really near to him and Marc wants to open it. Unfortunately the board computer of the chosen car, the one that able the system to open it, it is broken.Therefore he decides to call the customer service with the button in the reservation page. The customer service office answer to his call and let Marc abort his reservation without paying any additional feeds.

2. Claire is a user of PowerEnJoy who is renting a car, during her ride she rear-end another vehicle. Unaware about the procedure to follow she open the PowerEnJoy application and she click on "Customer service" in the menu. An employ of PowerEnJoy system answers and explains all the document Claire needs to complete before end her rent with the procedure to follow in her case. The operator opens also a Intervention request with the third-part company who is responsible to maintain and repair the cars of the system.

3. Jack is using a PowerEnJoy's car and he wants to park it because he is near to his destination. The battery charge is under 20% and he wants to plug the car in charge in order to let the next user able to use it for a longer period of time. Unfortunately the nearest safe area is broken and Jack cannot plug the car into charge. To not income in overtaxes he decides to call the Customer service. The employ answers to Jack's call and report the problem to the third-part agency. In plus the PowerEnJoy employ preserve Jack to receive overtaxes on his last rent but he also block every type of discount.

3.2.5.0.3 Description and Priority

This feature is of medium priority, it is not essential for the system to be.

3.2.5.0.4 Functional requirements

- **FR32:** The user must be able to contact the customer service 24h/24h

3.2.5.0.5 UML Sequence Diagram

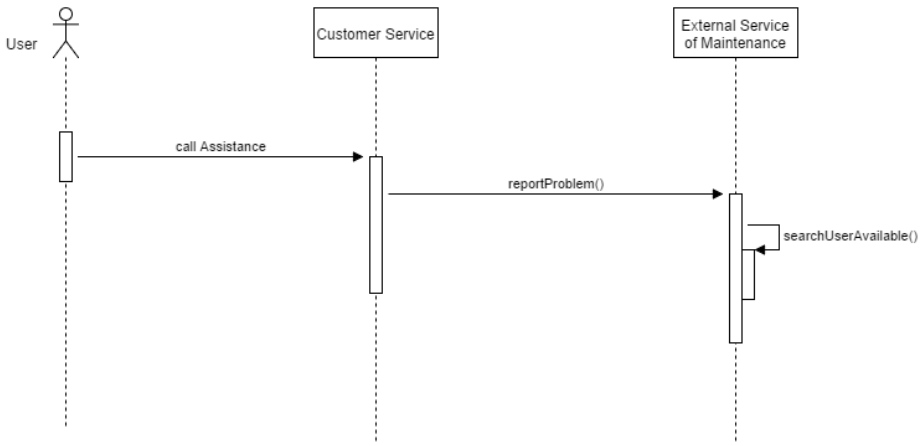

Figure 3.9: Report problem sequence

3.2.6 Profile settings

The system allows logged in users to view and modify their profiles at any moment, as long as they are logged in. While modifying email addresses, driver license or payment method must be unique in all the system, otherwise the system denies the modification request. In case of modify of email address, the system sends a confirmation email to the new address. Modification will successfully ends when the user clicks the link in the sent email.

3.2.6.0.1 Use Case

Name: Profile settings

Actors: User

Entry conditions: The user needs to be logged into the application

Flow of events:

- The user clicks on profile in the menu of the application
- The user changes his/her information that have changed
- The system notifies the user that the settings have been successfully updated.

Exit conditions: The user successfully saves his/her new settings

Exception: The information furnished by the user are not correct or ambiguous as the following case:

- The Email has not the correct format
- The Birthday is not at least eighteen years ago
- The Payment method is not valid
- The information's of the driver license don't correspond with the information furnished by the visitor
- The Driver license is not valid

3.2.6.0.2 Scenario

1. Zac periodically changes his account password, in order to increase protection. To do so, every 3 months, he opens PoweEnJoy on his mobile phone, chooses "Profile", then "Modify". He selects the password field, writes down a new one, then writes it again in the "Confirm password" field. Finally, he clicks "Confirm": the system informs him that his account password has successfully been updated.

2. Sailor is a user of PowerEnJoy and she has recently change her credit card because it was expired. She needs so to open PoweEnJoy on his mobile phone, chooses "Profile", then "Modify". She selects the old credit card and she writes all the new information about her new payment method. Finally she clicks "Confirm" and the system informs her that the account payment has been successfully updated.

3.2.6.0.3 Description and Priority

This feature is of low priority, it is not necessary but it is a service to create a better user expirience.

3.2.6.0.4 Functional requirements

- **FR33:** Account settings are accessible from the menu of the app, through the "Profile" button.
- **FR34:** The system allows users to view all their profile info, submitted during registration
- **FR35:** The system allows users to modify all their profile info, submitted during registration.
- **FR36:** Modifying the email address, the driver license requires that the new one doesn't match with the one of another registered user.
- **FR37:** Modifying the email address requires confirmation through an email sent to the submitted email address.
- **FR38:** The system allows users to abort modifications at any time.
- **FR49:** The system allows users to delete their account: confirmation is required to proceed.

3.3 Alloy

In this section is presented the alloy model with its execution results

```
module PowEnj
//SIGNATURES
sig Position{}
sig Email{}
sig Code{}

sig Plug{}

abstract sig Bool{}
sig False extends Bool{}
sig True extends Bool{}

abstract sig State{}
sig FreeState extends State{}
sig ReservedState extends State{}
sig RentedState extends State{}

sig DriverLicense{
  code:one Code,
  expiration:Int
}
{
  expiration>0
}


sig User
{
  payInfo: some PaymentMethod,
  position: one Position,
  license: one DriverLicense,
  email:one Email
}

sig Reservation{
  user:one User,
  time:Int,
  endTime:Int,
  car:one Car
}
{
  time>0
  endTime>time
}

sig Rent
```

```
{
  startTime: one  Int,
  endTime:one Int,
  endRent:one Bool,
  applyDiscount:one Int,
  applyOvertax:one Int,
  totalCost:Int,
  passengers:one  Int,
  reservation:one Reservation,
  payment:one PaymentMethod
}{
  passengers>=0
  startTime>0
  endTime>startTime
  applyOvertax>=0
  applyDiscount>=0
}


abstract sig Station{
  parkedCar:set Car,
  positoin:one Position
}

sig  Parking extends Station{
  distanceFromCharge:Int
}{
  distanceFromCharge>0
}


sig SafeArea extends Station
{
  pluggedCar: set Car,
  plugAvailable: set Plug,
}


sig PaymentMethod
{
  transactionCode: one Code
}

sig Car
{
  position:one Position,
  battery: Int,
  plate: one Code,
  state:one State,
  readyToEnd:one Bool
```

```
}{
  battery>=0 and battery<=100
}

//FACT

fact{
  //No useless paymentMethod
  User.payInfo=PaymentMethod
  //No useless Plug
  SafeArea.plugAvailable=Plug
  //No useless Email
  User.email=Email
  //No useless Position
  Car.position+User.position=Position
  //No useless Code
  Car.plate+PaymentMethod.transactionCode+DriverLicense.
    code=Code
  //No useless State
  Car.state=State
  //No useless Bool
  Rent.endRent=Bool
  //No useless License
  User.license=DriverLicense

  //No PaymentMethod with the same code
  all p1,p2:PaymentMethod| p1!=p2 implies p1.code!=p2.code
  //No car with the same plate
  all c1, c2: Car | c1!=c2 implies c1.plate!=c2.plate
  //No users with same email
  all u1,u2:User| u1!=u2 implies u1.email!=u2.email
  //No users with same license
  all u1,u2:User| u1!=u2 implies u1.license!=u2.license

  //No reservation for the same car in a time<1
  all r1,r2:Reservation | (r1!=r2 and r1.car=r2.car)
    implies (r1.time>r2.time+1 or r2.time>r1.time+1)

  //If a car is reserved is in reserved state
  all r:Reservation,c:Car,rent:Rent|  (r.car=c and  !(rent
    .endRent=False and rent.reservation.car=c)) implies c.
    state=ReservedState

  //if a car is in use is in rentedState
  all c:Car,r:Rent| r.reservation.car=c and r.endRent=
    False implies c.state=RentedState

  //if a car is not rented or reserved is in free state
    and a reservation keep going for max 1 hour
  all c:Car,rent:Rent,res:Reservation|
```

```
    //if the car is not reserved
    ((res.car!=c
      or
        //or the reservation is expired
        (res.car=c and res.endTime>res.time+1))
    and
      //and there is no rent on the car
      (rent.reservation.car!=c
        or
          //or the rent is finished
          (rent.endRent=True and rent.reservation.car=c)))
     implies c.state=FreeState

//A rent can be done only after a reservation
all rent:Rent,res:Reservation| rent.reservation=res
  implies rent.startTime>res.time and rent.startTime<=
  res.time+1

//Two different rent can't have the same reservation
all r1,r2:Rent| r1!=r2 implies r1.reservation!=r2.
  reservation

//There is just one rent not finish for each car
all r1,r2:Rent|r1.endRent=False and r2.reservation.car=
  r1.reservation.car implies !(r2.endRent=False)

//A rent can end only in a station
all rent:Rent,c:Car,s:Station|rent.reservation.car=c and
    c.readyToEnd=True implies c.position=s.position

//A rent has been concluded only in a station
all rent:Rent,s:Station|rent.endRent=True implies rent.
  reservation.car.position=s.position

//Discount for battery state applyed only if the rent in
    ended and the battery state is >50
all rent:Rent| rent.reservation.car.battery>50 and rent.
  endRent=True implies rent.applyDiscount=20

//Discount for passengers is applyed only if the rent is
    finished and more than 2 passengers were found
all rent:Rent| rent.passengers>2 and rent.endRent=True
  implies rent.applyDiscount=20

//All the cars parked are free
all car:Car,s:Station| car.position=s.position and car.
  state=FreeState implies car in s.parkedCar

//If a car is plugged is also parked
all s:SafeArea,car:Car|car in s.pluggedCar implies car
```

```
    in s.parkedCar

  //the user get a discount parking and plugging the car
  all car:Car,s:SafeArea,r1,r2:Rent| car in s.parkedCar
    and car in s.pluggedCar implies r1.applyDiscount=30

  //Safe area overtax
  all car:Car,p:Parking, r1,r2:Rent | car in p.parkedCar
    and p.distanceFromCharge>3 implies r1.applyDiscount=0
    and r1.applyOvertax=30

  all car:Car| car.state=FreeState implies car in Station.
    parkedCar

  no s:SafeArea| #s.pluggedCar>#s.parkedCar

  //Overtaxes are 0 or 30
  all rent:Rent| rent.applyOvertax=0 or rent.applyOvertax
    =30

  //the bill is calculate only if the rent is finish
  all rent:Rent| rent.applyOvertax>0 or rent.applyDiscount
    >0 implies rent.endRent=True
}

//ASSUMPTION

//A car can be reused after a rent
assert CarReused{
  all r1,r2:Rent| r1.endRent=True and r2.endRent=False and
     r1.reservation.car= r2.reservation.car implies r2.
    startTime>r1.endTime
}

//There are as many driver license as users
assert DriverForEachDocument{
  #DriverLicense = #User
  #User=#Email
}

assert UserPaymentMethod{
  #User.payInfo<=#PaymentMethod
}

//The rent with taxes can't have a discount
assert RentOverTaxes{
  all rent:Rent| rent.applyOvertax>0 implies rent.
    applyDiscount<rent.applyOvertax
}
```

```
//the plugged car can't be in use
assert PluggedCar{
  all rent:Rent| rent.reservation.car in SafeArea.
    pluggedCar implies !rent.endRent=False
}

check PluggedCar

check RentOverTaxes

check UserPaymentMethod

check CarReused for 5

check DriverForEachDocument for 5


pred SimpleWorld
{
  #User <3
  #Car < 4
  #Station< 4
  #Reservation < 4
  #Rent <3
}

pred RealWorld
{
  #User > 2
  #Car > 3
  #SafeArea> 1
  #Parking > 2
  #Reservation > 2
  #Rent > 1
}

run SimpleWorld for 6

run RealWorld for 6
```

3.3.1 Alloy Model Execution Results

- **Alloy Console:**

```
Executing "Check PluggedCar"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11418 vars. 753 primary vars. 27008 clauses. 591ms.
  No counterexample found. Assertion may be valid. 51ms.

Executing "Check RentOverTaxes"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11775 vars. 753 primary vars. 28200 clauses. 208ms.
  No counterexample found. Assertion may be valid. 138ms.

Executing "Check UserPaymentMethod"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11357 vars. 750 primary vars. 26924 clauses. 170ms.
  No counterexample found. Assertion may be valid. 22ms.

Executing "Check CarReused for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  26022 vars. 1450 primary vars. 58969 clauses. 450ms.
  No counterexample found. Assertion may be valid. 249ms.

Executing "Check DriverForEachDocument for 5"
  Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20
  25311 vars. 1440 primary vars. 56854 clauses. 215ms.
  No counterexample found. Assertion may be valid. 251ms.

Executing "Run SimpleWorld for 6"
  Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
  35349 vars. 1842 primary vars. 77393 clauses. 189ms.
  Instance found. Predicate is consistent. 79ms.

Executing "Run RealWorld for 6"
  Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
  35380 vars. 1842 primary vars. 77519 clauses. 203ms.
  Instance found. Predicate is consistent. 383ms.

7 commands were executed. The results are:
  #1: No counterexample found. PluggedCar may be valid.
  #2: No counterexample found. RentOverTaxes may be valid.
  #3: No counterexample found. UserPaymentMethod may be valid.
  #4: No counterexample found. CarReused may be valid.
  #5: No counterexample found. DriverForEachDocument may be valid.
  #6: Instance found. SimpleWorld is consistent.
  #7: Instance found. RealWorld is consistent.
```

Figure 3.10: Alloy Console.

- **Alloy Simple World:**

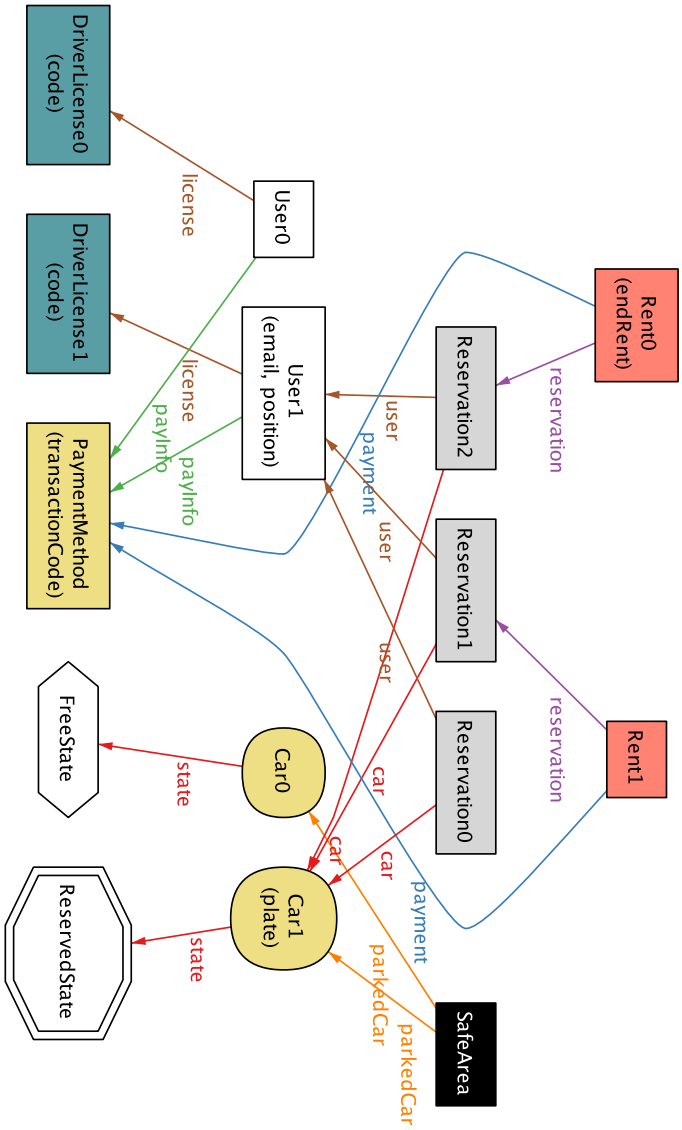

Figure 3.11: Alloy Simple World.

- **Alloy Real World:**

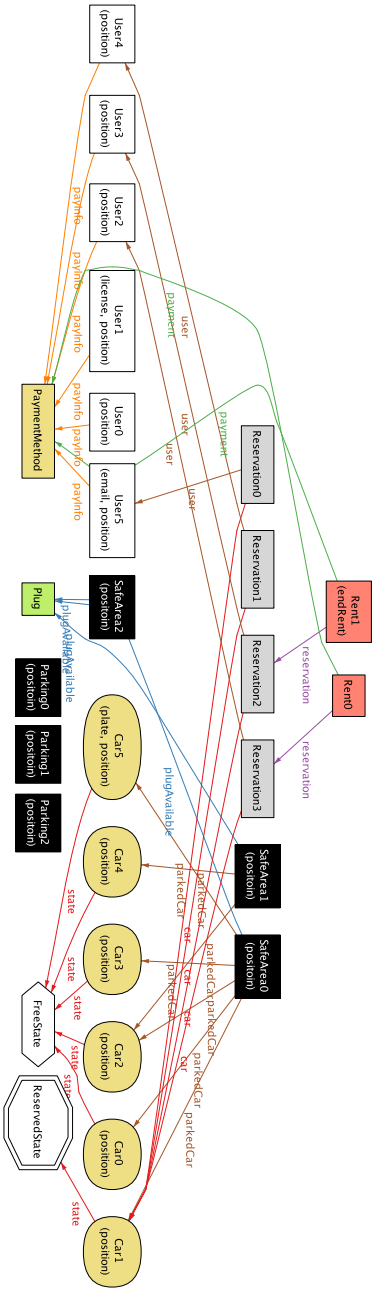

Figure 3.12: Alloy Real World.

3.4 Performance Requirements

PowerEnJoy will perform 95% of the operations within 4 seconds; the total amount of the operations within 10 seconds. The system should ensure at least 2000 passangers connected.

3.5 Design constraints

PowerEnJoy wants to reach most of users, requiring minimum specifications for devices. User, registered to the system, have to use their own devices provided with GPS navigation system to perform the service. Mobile applications have to offer backward compatibility.

3.6 Software system attributes

3.6.0.0.1 Reliability The mean time between failures (MTBF) shall exceed 3 months.

3.6.0.0.2 Availability In order to maintain the system up-to-date and secure, PowerEnJoy schedules downtime periods where will be executed routine operations. The service should be available 99% of the time.

3.6.0.0.3 Security myTaxiService to ensure service availability and data protection use:

- AES cryptography algorithm on network operations
- Data are encrypted and stored in backup drives to prevent system failure
- Login authentication. Users, after the registration, have to confirm their e-mail with the security code sent to the e-mail write in the registration form
- SQL injection detection

Server architecture will be implemented separating data from application. Application server must be separated from database and from the web server. All architectures are divided by firewalls.

3.6.0.0.4 Maintainability To ensure an easy maintenance of the software, it must be well-documented and written following coding patterns.

3.6.0.0.5 Portability Web programming ensures a wide target of browser. Mobile applications instead, cause of different languages and devices, have to be written following coding patterns for easy portability. Availability of the service is ensured by hardware and software limitations in Section 2.4.2.

Appendix A

Appendix

A.1 Tools

- **TeXstudio:** LaTeX editor used to write the document.
- **Alloy Analyzer 4.2:** Used to build an Alloy Model and to check its consistency.
- **StarUML:** Used to build UML Class Diagram and UML Use Case Diagrams.
- **draw.io WebSite:** Used to design the UML Sequence Diagram and the Mockup (wep portal and smartphone app).

A.2 Hours of work

In the following are listed the hours of work that each member of the group did:

1. Marco Redaelli: 41 *hours*
2. Francesco Zanoli: 41 *hours*

A.3 Version History

In the following are listed the differences between versions:

1. **13/11/2016:** First version
2. **11/12/2016:** Requirement enumeration changes, Edit version' table
3. **27/02/2017:** Fixed diagram UML