POLITECNICO
MILANO 1863

Software Engineering 2 project: *PowerEnJoy*

A.A. 2016/2017 - Prof. E. di Nitto

RASD

Requirements **A**nalysis and **S**pecification **D**ocument

Version **1.0** - 2016/11/13

Pietro Avolio Mat 878640
Guido Borrelli Mat 874451

Contents

1 Introduction

1.1 Purpose

PowerEnJoy is a new company that wants to enter the car sharing market and wants to provide a service based on electric vehicles only.
The way they want to implement their service is classical and very common to other car sharing services: it will be available in a specific geographical area (called safe area) for each city where the service will be activated. Users will be able to reserve and then rent a vehicle, use it for as much time as they desire and then be charged based on the rental time.

As the car sharing service is based on electric vehicles only, the system must provide some specific functionalities in order to handle electric vehicles behaviours and requirements. For example, the company owns some electric recharging stations spread among the safe area and users should be encouraged to terminate their rentals in these stations.

PowerEnJoy needs a digital management system in order to support all the activity for both their customers and their staff operators.

1.2 Scope

Following the "The World & Machine" approach by M. Jackson and P. Zave we can identify real word entities that interact with the system ("the World"), specific system entities ("the Machine") and the intersection between them ("the Shared Phenomena").

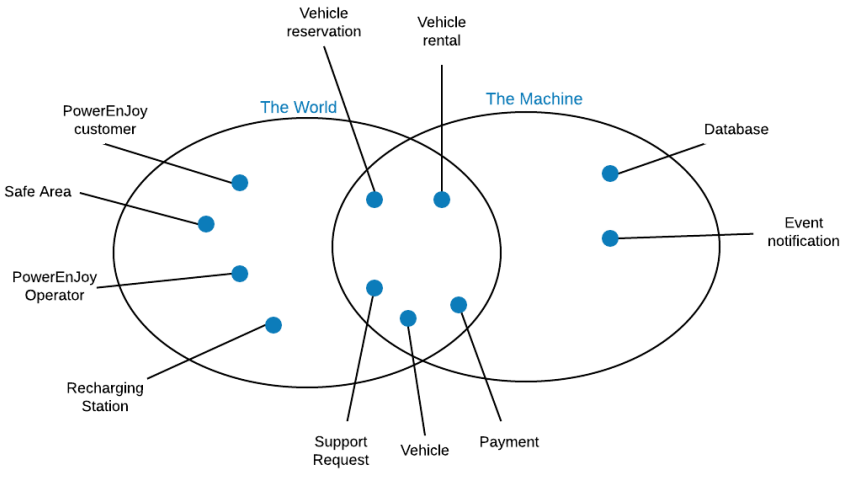

Figure 1: "The World & Machine" Venn diagram

The system will be composed of three main components with different roles and purposes:

PowerEnJoy Mobile App This component is intended to be used by PowerEnJoy customers and will offer the functionality to see available vehicles in a specific geographical area, to reserve and then to rent a vehicle. It will also allow users to unlock a vehicle they have reserved when they're nearby. Through the app, users will also be able to send support requests to PowerEnJoy support staff.

PowerEnJoy Control Room This component is intended to be used by PowerEnJoy operators. It will provide a real-time blueprint of the system, the functionalities to handle support requests and the possibility to be notified for some specific events (e.g. an automatic payment failure, a car that runs out of battery).

PowerEnJoy Core This component is intended to contain all the system logic. It will work as a connection node between all the other components as well as connection point to vehicles. It will handle user's payments through a payment gateway and it will perform automated tasks.

A list of high level goals that the system should accomplish is the following:

|G1| The system should know vehicles position and information such as model, battery percentage, mechanical issues and damages.

|G2| The system should be able to show on a map the position of available vehicles together with some selected status information (e.g. battery percentage) and estimated information (e.g. kilometers autonomy).

|G3| Users should be able to register and insert a payment method.

|G4| Users should be able to reserve a vehicle, unlock it when they're close to its position and start the rent.

|G5| The system should charge the users after each rent with fees based on company policies.

|G6| Users should be able to send support requests.

|G7| Company operators should be able to handle support requests forwarded by users.

|G8| Company operators should be able to see and handle notifications sent by the system.

|G9| The system should be able to notify company operators for some specific events.

1.3 Definitions, acronyms and abbreviations

In order to avoid ambiguity and possible misunderstanding here are formally listed some recurrent terms and acronyms used in this document.

The System	The digital management system to be developed.
PowerEnJoy	The company to develop the system for. Also referred as 'the company'.
Vehicle	A vehicle owned by the company that can be used in the car sharing service.
User	A person who wants to use the system and he's not a member of the company.
Logged-in user	A user who has completed the log-in process so that he can be associated with a real identity and payment information.
Customer	A user or a logged-in user.
Operator	A person who wants to use the system and he is authenticated as a member of the company.
Availabe vehicle	A vehicle that is not reserved by anywan and can be reserved by a logged-in user.
Reserved vehicle	A vehicle that has been reserved by a logged-in user.
Rented vehicle	A vehicle that is currently rented by a logged-in user.
Expired reservation	A reservation that has not been cancelled or transformed into a rental after a certain amount of time defined by company policies.
Safe Area	The geographical area in which a vehicle can be parked and rental terminated.
DBMS	Database Management System
TBD	To Be Determined

1.4 Reference documents

|REFD1| Assignments AA 2016-2017

|REFD2| ISO/IEC/IEEE 29148, first edition, 2011-12-01

|REFD3| Requirements Engineering Part III

|REFD4| IDC Research Inc analysis on mobile operating systems market share `http://www.idc.com/prodserv/smartphone-os-market-share.jsp`, 2016 Q2

|REFD5| Android distribution per version `https://developer.android.com/about/dashboards/index.html`, as of 2016-11-02

|REFD6| iOS per version distribution `https://david-smith.org/iosversionstats/`, as of 2016-11-02

1.5 Overview

This RASD document is made of the following parts:

1. Introduction: this section provides a general and low-detailed description of the system-to-be.
2. Overall description: this section provides general aspects of the system, like interfaces, constraints, domain assumptions, software dependencies and users' characteristics.
3. Specific requirements: this section provides scenarios, use cases and a set of diagrams in order to represent the functionalities of the system-to-be.
4. Appendix: this section contains the Alloy model used to model and verify the system, a list of the tools used to develop this document and the working hour tracking table.

2 Overall Description

2.1 Product perspective

The central component of the entire system is the PowerEnJoy Core component which is where all the logic is put. This component of the system is in charge of providing an interface to vehicles on-board information system, of processing requests from the other components of the system, of accessing company database and of interfacing with the payment and SMS gateways.

All the other components the system is made of (i.e. PowerEnJoy Mobile App and PowerEnJoy Control Room) should interface with the core component in order to accomplish required tasks (e.g. user's registration and login, vehicle reservation request, etc). They can be considered very thin clients.

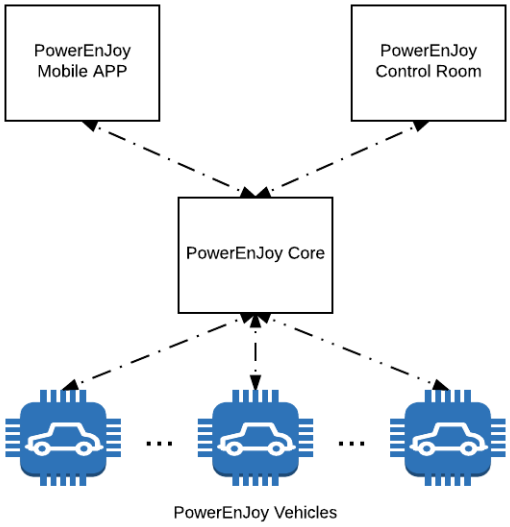

Figure 2: Block diagram representing the system structure

2.1.1 User Interfaces

PowerEnJoy Mobile App This component will be implemented as a mobile application and it's intended for costumers. It's important that the most used commands are visibile and easy navigable. Secondary commands of less frequent usage can be organised in secondary less visible menus.

In order to avoid mistakes between commands it's better not to have too many selections on every page; text explainations should

be clear and concise. Error windows should show short error descriptions together with the error identification code.

The application should follow the design guidelines for the different operating systems it will be implemented for (i.e. Material design for Android, ModernUI for iOS) and support all the screen resolutions between ldpi and xxdpi.

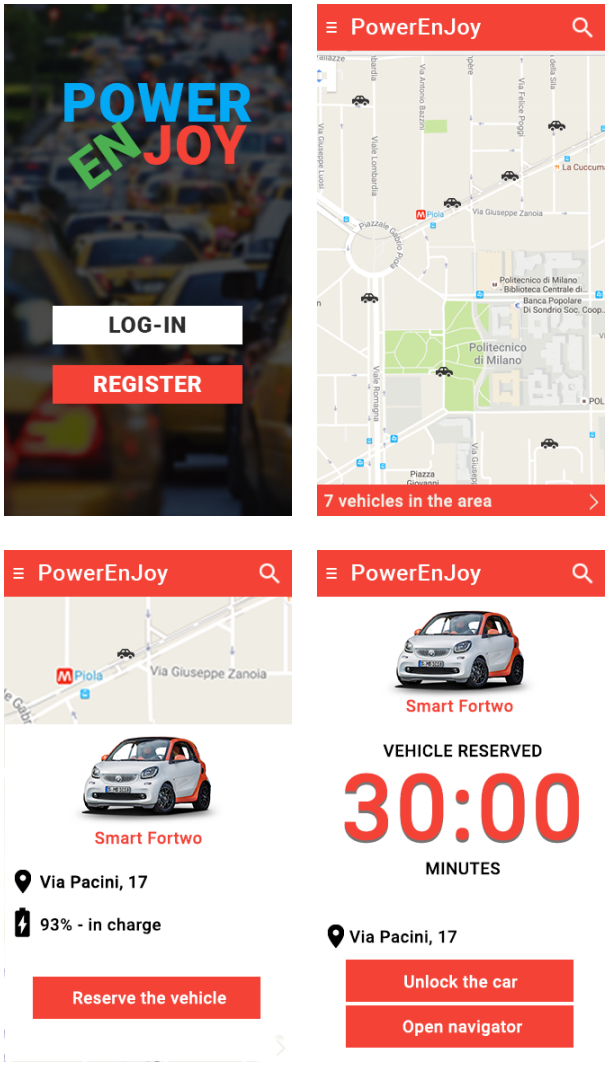

Figure 3: PowerEnJoy Mobile App mockups

PowerEnJoy Control Room This component will be implemented as a web application and it is intended for operators. All the functionalities should be well categorized and unambiguous. Most frequently used functions should be as easy and immediate as possibile in order to

speed-up operators' work and reduce customers' waiting time. Some macros could be present in order to reduce repetitve tasks.
It should be accessible through a common internet connection and it should be well optimized for a single chosen browser and screen resolution. Error windows should show a long error message in order to easily recover.

PowerEnJoy Core This component does not provide a direct user interface since it's intended to be managed and maintained by qualified and specialized staff. It should provide very detailed activity logs.

2.1.2 Hardware interfaces

PowerEnJoy Mobile App This component should obtain device location by its operating system. If GPS position is unavailable or insufficiently precise, a localization via WiFi network could be possibile. If both methods fail or can't provide the desired precision the application can't work properly.

PowerEnJoy Control Room This component does not have any hardware interface.

PowerEnJoy Core This component does not have any hardware interface.

2.1.3 Software interfaces

PowerEnJoy Mobile App Android and iOS together cover about the 99% of the mobile operating system market share[1] and these are the systems we are going to focus on.
After analysing per operating system distribution and share, a good compatibility tradeoff could be the following.

Operating System	Min. version
Android	4.1 (API level 16)
iOS	9

With this minimal requirements PowerEnJoy Mobile App will support 97% of the android devices[2] and 92% of the iOS devices[3]

[1] REFD4
[2] REFD5
[3] REFD6

PowerEnJoy Control Room This component will require a Java EE application server to be executed.

Application server	
Name	Glassfish
Version	4.1.1

It requires an interface to a support ticket system in order to handle support requests forwarded by users.

Support ticket system	
Name	osTicket
Version	OSE 1.10

PowerEnJoy Core This component requires a Java EE application server to be executed.

Application server	
Name	Glassfish
Version	4.1.1

It requires an interface to vehicles on-board information system, to the payment gateway, to the SMS gateway and to the DBMS in order to access the company database.

Payment Gateway	
Name	GESTPAY
Version	Professional

DBMS	
Name	MySql C.E.
Version	5.7.6

Vehicles information system	
Name	PowerEnJoy Vehicle I.S.
Version	1.0

SMS Gateway	
Name	Trendoo
Version	-

2.1.4 Communication Interfaces

All components should communicate with the core component in order to complete tasks, thus the core component can be seen as an internal API provider. All the communications between the components are bidirectional and could be implemented using the TCP/IP protocol with a REST approach. Responses could be implemented using the JSON format.

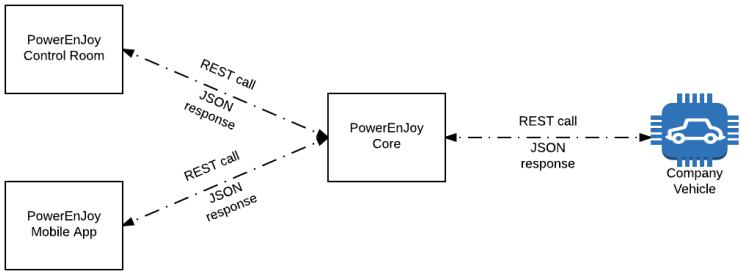

Figure 4: Communication between system components

Communication between the core component and the DBMS works using the TCP protocol on port 3306.

Communication between the core component and the payment gateway works using the TCP/IP protocol on an arbitrary port.

Communication between the core component and the SMS gateway works using the TCP/IP protocol.

The PowerEnJoy control room can be accessed by the operators using the HTTPs protocol on port 80 and a common web browser.

2.1.5 Memory constraints

Users must have enough space to install PowerEnJoy Mobile App application on their own devices. The size of the application is still unknown but it can be estimated in less than 100MB.

The system on which the core component will be run requires enough primary memory space in order to install required softwares (section 2.1.3). Other 5GB of primary memory space are required for the database and the software itself.
A good amount of secondary memory (i.e. 32GB) will result in better performances for both the software itself and the DBMS.

2.2 Product functions

In this section are listed all the functionalities that the system-to-be is going to provide.

2.2.1 Functional requirements

Functional requirements are listed per system component:

PowerEnJoy Mobile App

- Users can register.
- Users can log-in.
- Logged-in users can edit their account information and payment method.
- Users can see available vehicles on a map.
- Users can search for available vehicles in a specific area using their GPS position or inserting an address.
- Users can see specific vehicle information (i.e. vehicle model, battery percentage, estimated kilometers of autonomy).
- Logged-in users can make a vehicle reservation.
  - Logged-in users can have only one reservation at a time
- Logged-in users can cancel an existing vehicle reservation.
- Logged-in users can unlock a vehicle they have reserved when they are nearby.
  - Logged-in users can have only one rental at a time.
  - Logged-in users can rent a vehicle if and only if they have a reservation on that vehicle.
- Logged-in users can not have a reservation and a rental at the same time.
- Logged-in users can be guided by a navigator to their reserved vehicle.
- Logged-in users can send a support request.

PowerEnJoy Control Room

- Operators can log-in.
- Operators can see on a map the position of all the vehicles together with their information.
- Operators can see users' account information.
- Operators can disable or enable a user's account.
- Operators can see user's last activity.
- Operators can see users' support requests.
- Operators can change the status of a user's support request into $OPEN, HANDLING$ or $CLOSED$.
- Operators can see system notifications.

- Operators can reply to a user's support request.
- Operators can change the status of a system notification in *OPEN*, *HANDLING*, *CLOSED* or *CRYTICAL*.
- Operators can change the status of a vehicle into*AVAILABLE* or $NOT - AVAILABLE$.
- Operators can charge users for driving fines.

PowerEnJoy Core

- The component can handle rent events sent from vehicles (i.e. *RENT STARTED*, *RENT TERMINATED*).
- The component can send commands to vehicles (e.g. *UNLOCK DOORS*).
- The component can charge users on their payment method applying company pricing policies.
- The component can handle payment failures.
- The component can access and manage the company database.
- The system can send communications to users via email on their email address.
- The system can send communications to users via SMS on their mobile phone number.
- The system can send communications to users via push notifications on their smartphone.
- The system can generate monthly invoices and sent them to the users on their email address.
- The system can detect when a vehicle reservation expires.
- The system can mark a vehicle as $NOT - AVAILABLE$ when its battery level is less than 20% and it's not rented by any user.
- The system can mark a vehicle as *AVAILABLE* when its status is $NOT - AVAILABLE$ and its battery level is higher than 20%.

2.2.2 Non-functional requirements

- Users should be encouraged to use the service with their friends.
- Users should be encouraged to plug vehicles in a recharging station after their rentals.

2.2.3 Company pricing policies

The company has defined some specific pricing policies to be applied in correspondence to some specific user behaviours:

- Expired reservation has a cost of 1EUR.
- The rental starts as the user turns on the vehicle engine.

- The rental stops as the user parks the vehicle within the safe area and all the passengers leave the vehicle.
- Rental fee is per minute.
- A 10% discount is applied on the total rental fee if there are at least 2 other passengers in the vehicle for more than the 50% of the rental time.
- A 20% discount is applied on the total rental fee if the vehicle is parked with more than 50% of battery capacity available.
- A 30% discount is applied on the total reantal fee if the vehicle is parked in a charging station and the user plugs-in the charging cable.
- A 30% extra fee is applied on the total rental fee if the vehicle is left more than 3KM away from a rechargin station or if the vehicle is left with less than 20% of battery capacity available.
- If more than a discount or an extra fee must be applied, extra fees are applied first in ascendant order, discounts are applied later in descendant order.

Policies are applied to calculate the final price for the users at the end of each rental.

2.3 User characteristics

We distinguish between two main categories of users:

PowerEnJoy Customers Customers who want to use the service in order to rent a vehicle. Divided into:

- Not logged-in users: They can only see the position of available vehicles. Anyone that uses the app without logging-in.
- Logged-in users: They can access to all the functionalities of the service. They are associated with a real identity and a payment method. Logged-in users should possess a valid driving license, which should be indicated during the registration process. People with disabilities that do not allow them to drive a normal vehicle cannot register and use the service: this check is done validating driver license data during the registration process.

PowerEnJoy Operators Clerks of the company. They have a good knowledge about the service and they will attend a training on how to use PowerEnJoy Control Room software.

2.4 Constraints

2.4.1 Regulatory policies

The system needs to store in the company database customers' personal information (e.g. name, surname, address, email address, telephone number) as well as to access vehicles locations in order to track them. By tracking vehicles locations users are tracked too. Sensible data treatment should respect local laws belonging to each area where the service will be available.

2.4.2 Safety and security considerations

Communications between system components should be secure and use advanced security protocols in order to avoid man-in-the middle attacks or other kind of informatic attacks that could lead to data loss.
Communications towards vehicles should be secured in order to avoid illegal vehicles unlocks performed by malicious users or any other type of illegal behaviour.

2.5 Assumptions and dependencies

The system is based on the following domain assumptions:

- Information provided by vehicles on-board information system to the core component are always available and accurate.
- Vehicles on-board information system are always able to detect and provide: battery percentage, number of passengers, mechanical failures, GPS location.
- If a vehicle cannot obtain location by GPS or has not internet connection it is considered unavailable and it is not shown to users.
- Vehicles always unlock when the system tells them to unlock. This is true for each type of command the system sends to vehicles.
- If a user owns a valid driving license - so his driving license was successfully evaluated during the registration process - then he can access the service.
- Users always unplug a vehicle from the recharging station before they start a rental.

2.6 Further developments

- Allow the creation of different groups of users in order to have different system behaviours based on the group (e.g. different pricing policies for different groups).
- Allow the creation of coupons that users can use in order to have special discounts.

3 Specific requirements

3.1 Functional requirements

3.1.1 Scenarios

In order to better describe functional requirements listed in the section 2.2.1 here is a list of possible system usages (scenarios) from various viewpoints.

Scenario 1 **User registration and log-in**

Luca is a university student who moved to Milan for studying. He can't afford to buy a car but sometimes he needs to move even when public transports are unavailable (i.e. during the night). He decided to try a car sharing service and downloaded PowerEnJoy Mobile App in order to use it during the incoming weekend. Once the download is complete and the app is installed on his smartphone, Luca starts the registration process: first he needs to create an account filling-in his name and email address, his mobile phone number, his home address and choosing an username. Email address is not yet used and username is available, thus he can continue inserting his driving license ID: he missed a character so the application shows an error message. Luca fixes the mistake and moves on to the latest step in which he is requested to insert his credit card information. A fee of 0,01EUR is charged on his credit card in order to validate it and the registration process is complete.
Luca receives on his email address his password and can now log-in.

Scenario 2 **Search for a vehicle and make a reservation**

Martin is a businessman and he is just landed in Milan Malpensa airport. He is going to have an intensive day and he needs to move fast in the city center. He is taking a train from Malpensa to Stazione Centrale station and during the 30 minutes time trip he plans to reserve a vehicle near Stazione Centrale in order to do not waste time when he will get off the train. He is a PowerEnJoy user and he starts the application on his smartphone: he fills in the address and sees available vehicles on the map. He finds a vehicle with enough battery percentage according to his needs and away only three minutes walking from the station. He decides to reserve this vehicle and clicks the button on the application. Now Martin can relax until he reaches Stazione Centrale.

Scenario 3 **Unlock a vehicle and start the rent**

Marco is walking home after his evening gym session when he spots a PowerEnJoy car parked along the street. He is tired by his training session so he decides to check if the car is available. Marco starts the application on his smartphone and checks for available vehicles nearby using his GPS location. The vehicle along the street is available so Marco reserves it and, being less than 5 meters away from the car, he is able to unlock it using the relative button on the application. Marco jumps into the car, secures the fast belt and starts the engine: the rental starts and the price to be paied is shown on the screen installed in the car.

Scenario 4 **User terminates a rent**

Mario and his friends are planning to go the the cinema tonight but the place they want to reach is not covered by city public transports. Mario is an old PowerEnJoy customer and he knows that the place is inside the safe area and that PowerEnJoy applies a special discount on the rental fee if there are passengers. He rents a vehicle and reaches the cinema: once there, he stops the engine and terminates the rent. An SMS sent to his mobile phone number informs him of the final price, which was influenced by a number of company policies. 9EUR is the base price, plus he has got a 10% discount because he traveled with friends, a 20% discount because he left the car with more than 50% of the battery capacity available and a 30% extra fee because there are no recharging station in a 3KM range from the cinema. Mario thinks that 8,43EUR is a very good price to spend a cinema night with friends (with which he will divide the cost later!).

Scenario 5 **User sends a support request and operator handles it**

Mario had a really busy day and he is now looking for an available PowerEnJoy car so that he can reach his friends at the restaurant. Mario uses the application on his smartphone to reserve a vehicle but when he reaches the car he has an unexpected surprise: the car has no wheels because they have been stolen. He uses the application to open a support request describing the problem. The operator sees the request in the PowerEnJoy Control Room software and gives instruction on what to do to the client. Then, the operator marks the car as unavailable and sends a notification to the mechanics team.

Scenario 6 **User tries to terminate the rent outside a safe area**

Marta is new to the car sharing services and it is the first time she rents a PowerEnJoy car. She wants to reach Rho, a town near Milan, but it is outside the safe area and she has not read information documents. Once she reaches her destination, she stops the engine and tries to terminate the rental: a message on the screen installed in the car and an SMS sent to her mobile phone number notifies her that she can't terminate the rent in that location and that she will be charged until she will terminate the rental inside the safe area. Marta checks the map and starts the engine again in order to move the car inside the safe area, which is just 700mt away from her destination.

Scenario 7 **User's payment failure**

Carol has just terminated a rental but she has no money left on her prepayed credit card. As soon as the PowerEnJoy Core component tries to make a payment and it is refused, it disables Carol's account and sents her a notification both via SMS and email. She will not be ablo the use her account to rent a vehicle until she will update her payment information and pay the given amount.

Scenario 8 **User took a driving loan**

A driving loan has just been received at the PowerEnJoy headquarter. The operator takes the loan and starts the procedure to forward it to the user that took the loan. The operator just inserts the date and the time indicated on the loan and the system returns the user that took it. The operator charges the user for the forwarding fee and sends him the loan via post.

Scenario 9 **A vehicle is parked with a very low battery level**

Francesco is in a rush and he parks the car that he rented with only 5% of the battery available and far away from a charging station. Once the rental is terminated the PowerEnJoy Core notifies the operators via the PowerEnJoy Control Room software and marks the vehicle as unavailable so that other users can not rent it. Operators handle the notification and forward it to the mechanics staff that will handle the situation.

Scenario 10 **User's documents are about to expire** PowerEnJoy Core detected that at least one of Mirco's document is going to expire. Mirco is at work when he receives an app notify and clicks it. A message in the app appears and informs Mirco that his credit card is going to expire this month and that he must provide a new one in order to be able to use PowerEnJoy service in the next months. Mirco inserts the new credit card information and saves. A fee of 0,01EUR is charged on his new credit card in order to validate it. Some weeks later, the same notify appears in Mirco's phone and it surprises him. Mirco discovers that his driving license is going to expire in the next month too. He must insert new driving license information once he will get a renewal.

3.1.2 Class diagram

This class diagram is a first analysis of the classes that will compose the system and the way they interact one with each other.

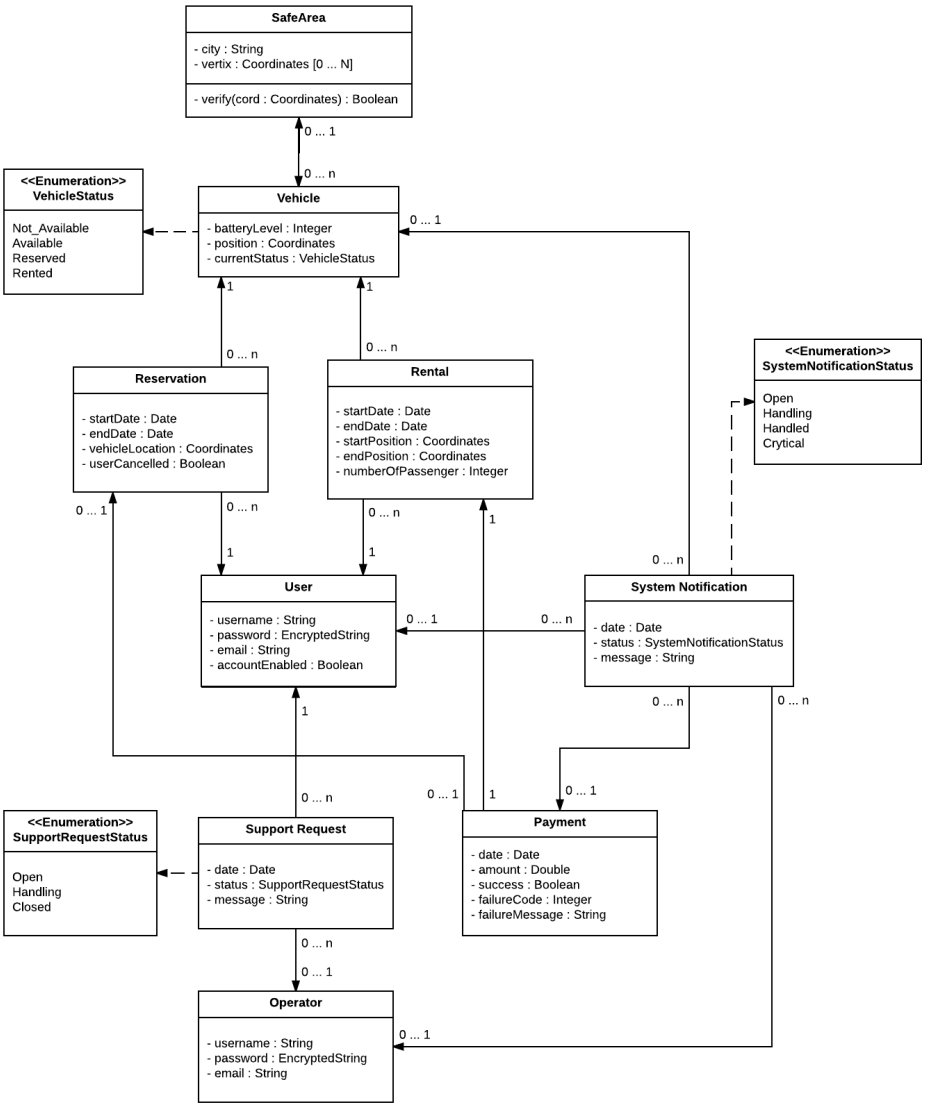

Figure 5: Proposed class diagram

3.1.3 State diagrams

State chart of a vehicle

When a vehicle is inserted into the system its status is *AVAILABLE* and it can be reserved by any user. When a reservation comes, the status of the vehicle changes into *RESERVED*: it can not be reserved by any other user and can be rented only by the user who reserved it. If the user cancels the reservation or the reservation expires the status of the vehicles returns back to *AVAILABLE*. When the user procedes with the rental the status changes into *RENTED* and the vehicle can not be reserved by any other user. Once the user terminates the rental the status returns back to *AVAILABLE*.
If the vehicle encounters a mechanical issue or if the battery level is too low when a rental is terminated its status is changed into *NOTAVAILABLE*: it can't be reserved by any user. Once all problems are fixed the vehicle status returns back to *AVAILABLE*.

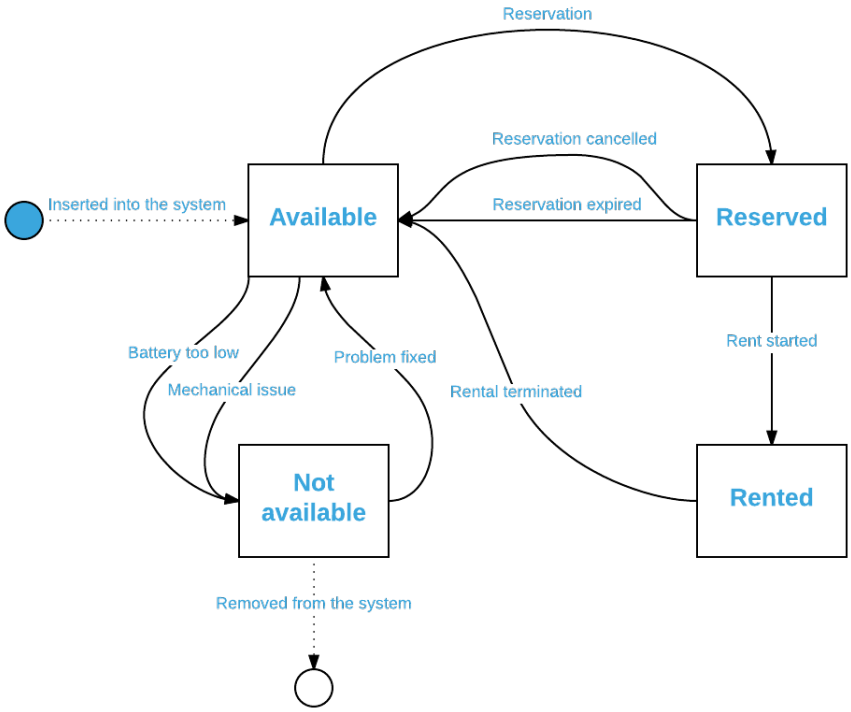

Figure 6: State chart of a vehicle

State chart of an user account

When an account is created and the user has not yet completed the log-in process the account is in the $NOT\ LOGGED-IN$ status. When in this status, the user can only access to a subset of the functions of the system. When the user successfully completes the log-in process the account status changes into the $LOGGED-IN$ status. In this status the user can access to all the functions of the system. Both the operators or the system can change the status of an account into *SUSPENDED*. In this status the user can not reserve o rent vehicles.

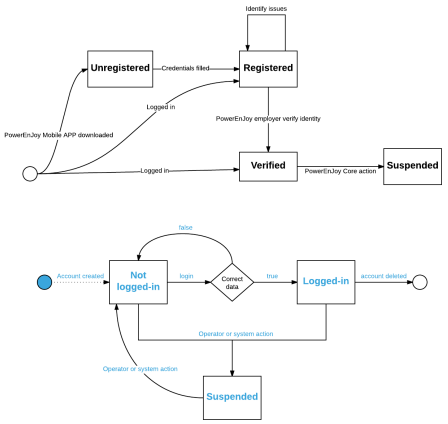

Figure 7: State chart of a user's account

State chart of payment

When a rental is terminated, the PowerEnJoy Core component automatically sends a payment request via the payment gateway for the due amount. If the response is positive the payment and it's information are just stored into the company database. If any error occours the payment enters into the *PAYMENT FAILED* status. As soon as the user updates his payment method information, the Core module sends a new payment request.

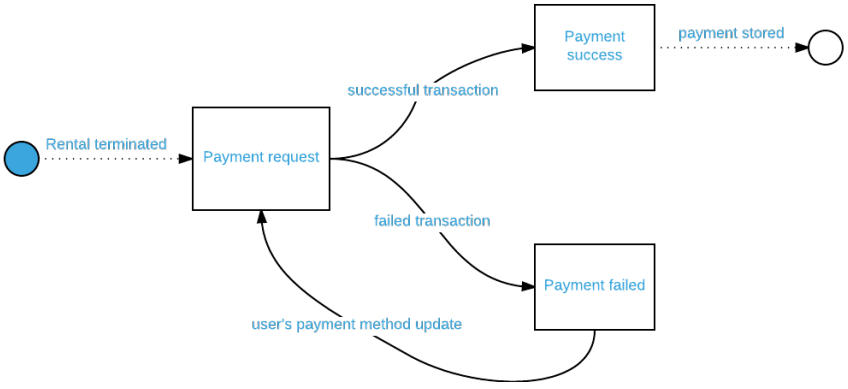

Figure 8: State chart of a payment

3.1.4 Use cases

The actors involved into the system are:

- Not logged-in User
- Logged-in User
- Operator
- Core component

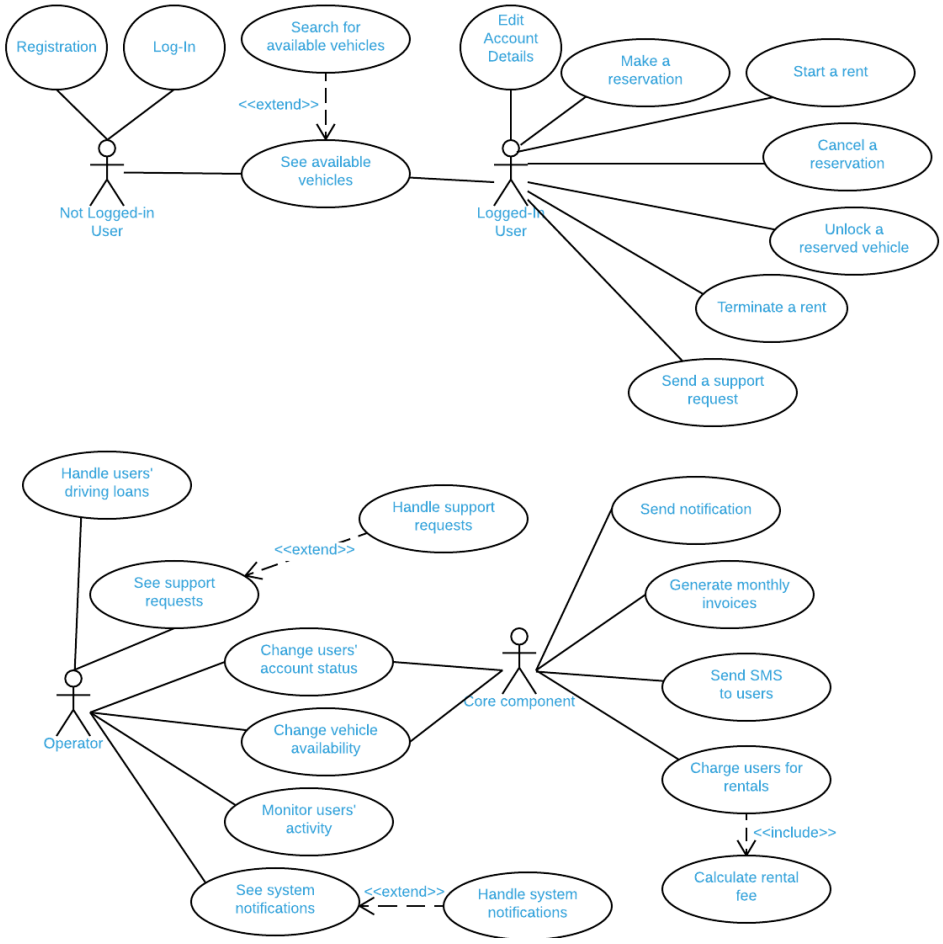

Figure 9: Use case diagram of the system involving all the actors

Detailed description for each use case

Not logged − in user registration

Actors	Not Logged-in user
Start conditions	None
Events flow	• The not logged-in user starts the registration process. • Step1: the not-logged in user is prompted to choose an username and to provide his email address, home address and mobile phone number. • The system checks that email address is not used by any other account and that the username is still available. • Step2: the not-logged in user must provide his driving license information. • The system validates the provided driving license information. • Step3: the not-logged in user must provide his payment information. • The system validates the provided payment information. • The systems generates a password for the user which is sent via email to the user. • The system shows a registration complete screen.
Exit conditions	Account details, driving license information and payment method are stored in the company database.
Exceptions	• The choosen username is not available. • The filled-in email address is used by another account. • The driving license information is not valid. • The payment information is not valid.
Goal	G3

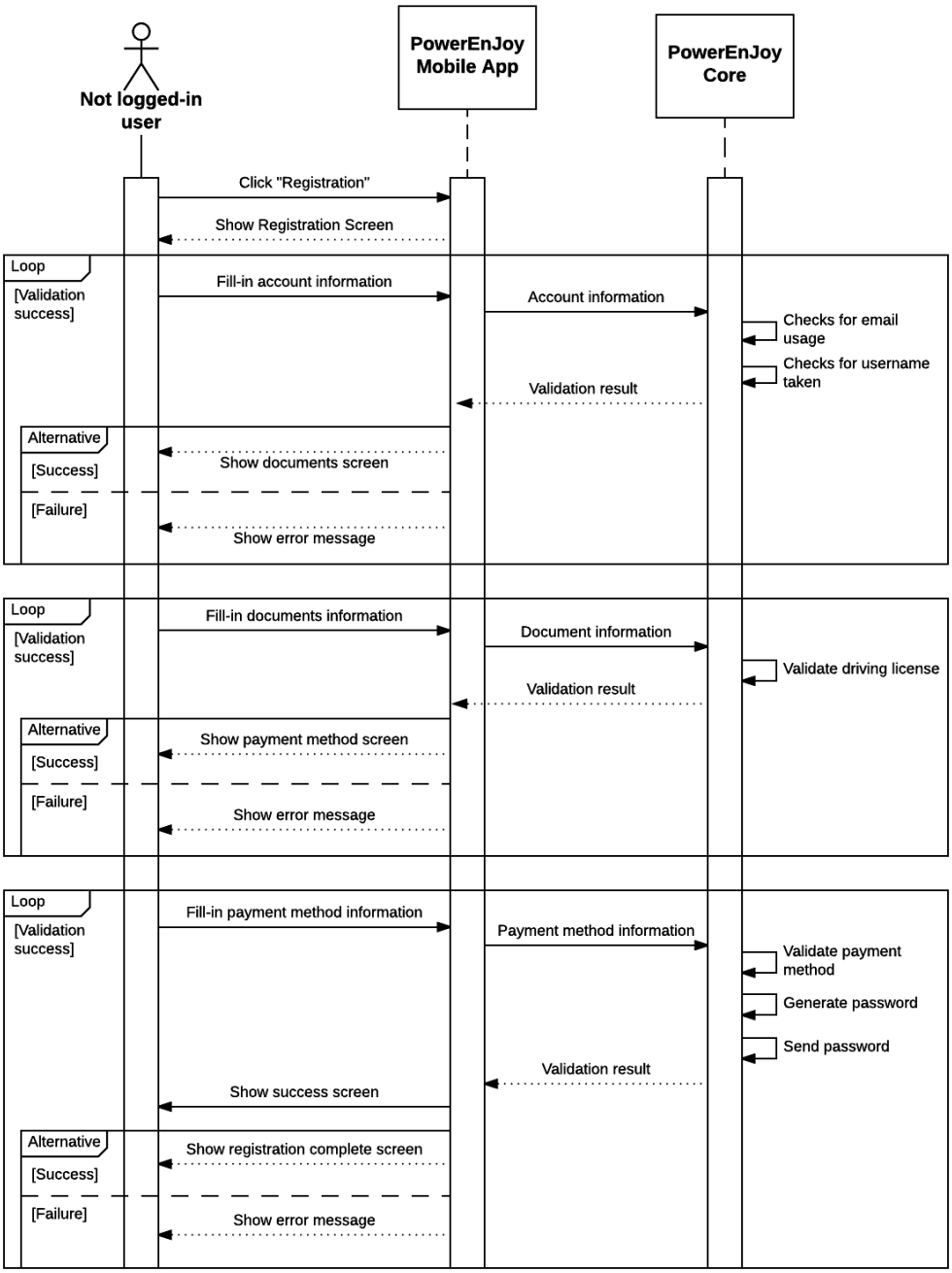

Figure 10: Sequence Diagram for the registration process

NOTE: The logged-in user that desires to register remains in a loop filling-in required information until the system validates them and gives a 'success' response. This happens for each step of the registration process.

Not logged − in user login

Actors	Not Logged-in user
Start conditions	None
Events flow	• The not logged-in user starts the log-in process. • The not logged-in user fills-in his username and password. • The systems searches for a correspondence in the company database. • The system shows a log-in complete screen.
Exit conditions	The not logged-in user is authenticated and his status changes into *LOGGED − IN USER*. He can use all the functionalities of the system.
Exceptions	• There is no correspondent tuple <username, password> in the company database. • The user's account has been suspended.
Goal	G3

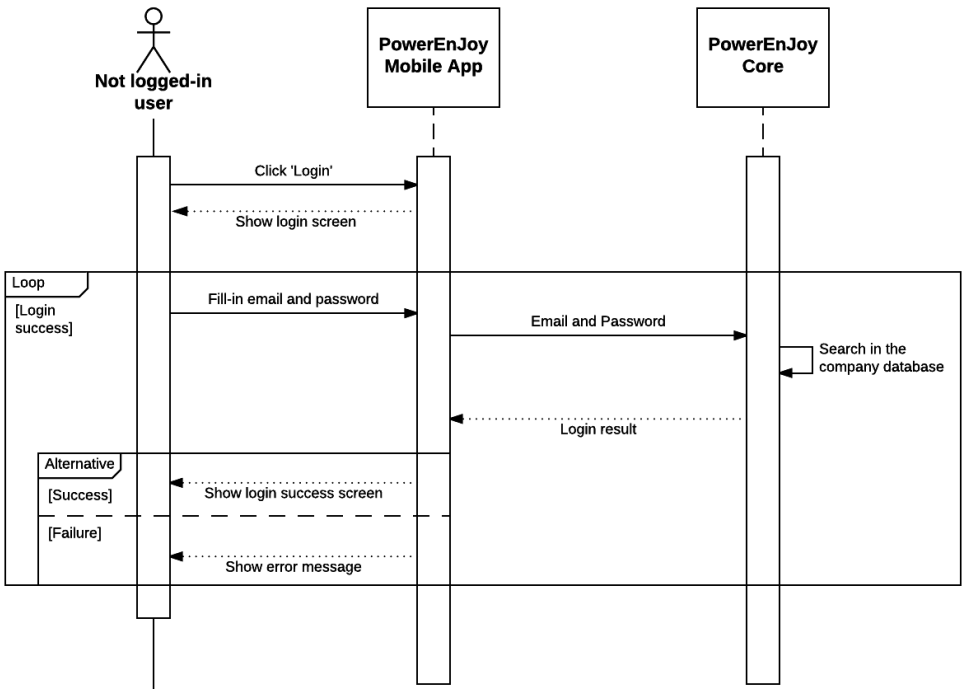

Figure 11: Sequence Diagram for the log-in process

See available vehicles

Actors	Not logged-in user, logged-in user
Start conditions	None
Events flow	• The actor accesses the available vehicles screen. • The system retrieves the position of all the available vehicles in the area shown in the screen. • The system prints on the map an icon for each available vehicle. • The actor clicks on a vehicle icon. • The system shows the details of the vehicle: model, battery level, exact position, estimated kilometers autonomy.
Exit conditions	None
Exceptions	None
Goal	G1, G2

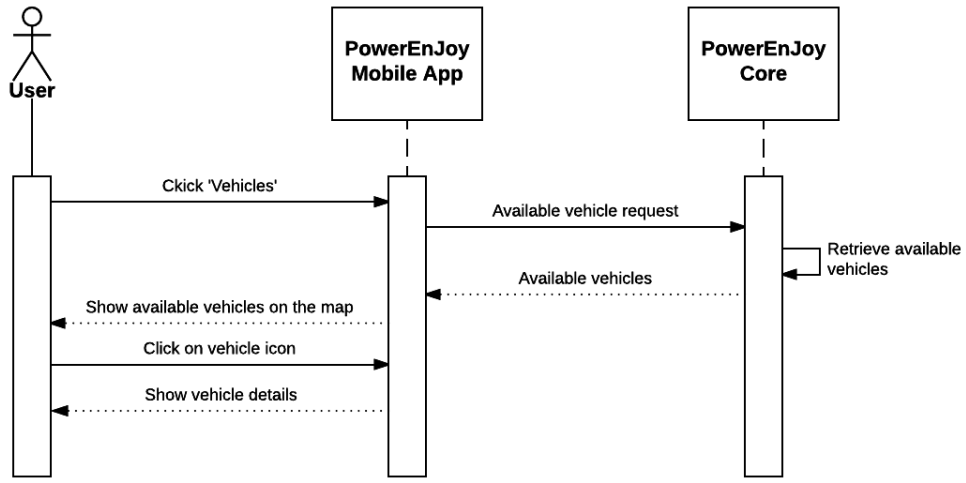

Figure 12: Sequence Diagram for the 'see available vehicles' use case

Search for available vehicles

Actors	Not logged-in user, logged-in user
Start conditions	None
Events flow	• The actor access the available vehicles screen. • The actor touches on the search-bar and types the address or chooses to be localized by its GPS position. • The system retrieves the position of all the available vehicles in the area choosen by the actor. • The system prints on the map an icon for each available vehicle. • The actor clicks on a vehicle icon. • The system shows the details of the vehicle: model, battery level, exact position, estimated kilometers autonomy.
Exit conditions	None
Exceptions	• System can not reverse locate the address filled-in by the actor. • The application can not retrieve actor's position by GPS.
Goal	G1, G2

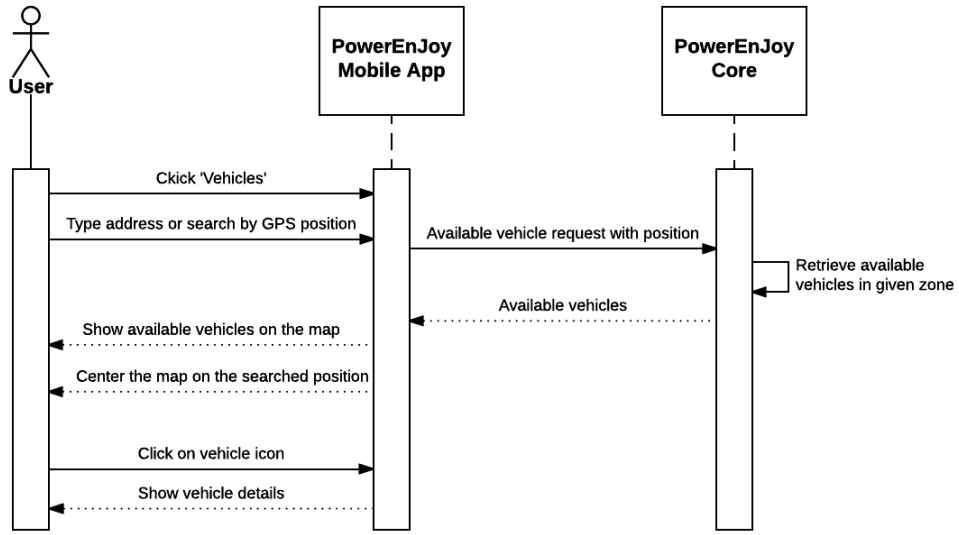

Figure 13: Sequence Diagram for the search for available vehicles'

Make a reservation

Actors	Logged-in user
Start conditions	• The user is authenticated. • The user is in the vehicle details screen.
Events flow	• The user clicks the 'Reserve Vehicle' button • The application asks for a confirmation and informs the user that an expired reservation has a cost. • The user proceeds by clicking 'confirm'. • The system changes the status of the vehicle into *RESERVED*. • The application shows a screen containing the address position of the vehicle and a countdown to reservation expiration.
Exit conditions	A reservation is made for the user over the choosen vehicle.
Exceptions	• Another user has made a reservation over the choosen vehicle in the meantime. • The user does not click 'confirm' when informed that an expired reservation has a cost.
Goal	G4

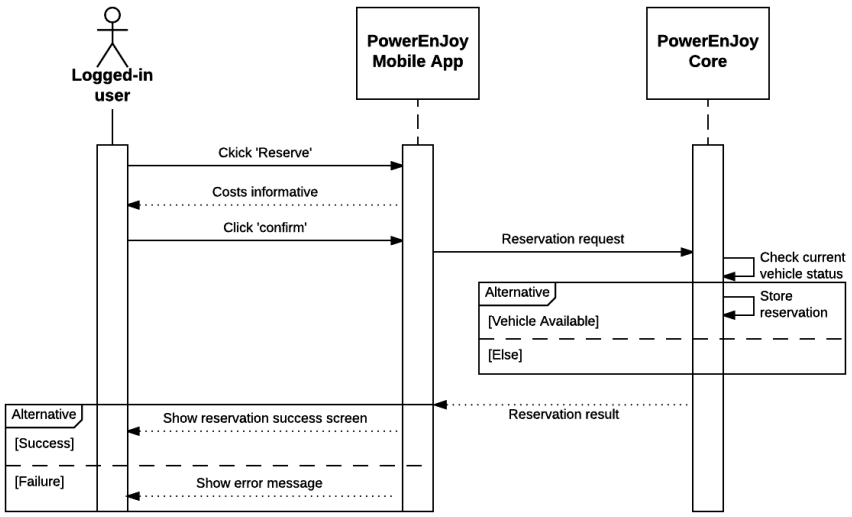

Figure 14: Sequence Diagram for 'make a reservation' use case

Cancel a reservation

Actors	Logged-in user
Start conditions	• The user is authenticated. • The user has reserved a vehicle. • The user is in the reservation details page.
Events flow	• The user clicks the 'Cancel reservation' button. • The applications informs the user that to cancel a reservation has a cost. • The user proceeds by clicking 'confirm'. • The system changes the status of the vehicle into *AVAILABLE*.
Exit conditions	The reservation that the user had over the vehicle is deleted.
Exceptions	• The user does not click 'confirm' when informed that to cancel a reservation has a cost.
Goal	G4

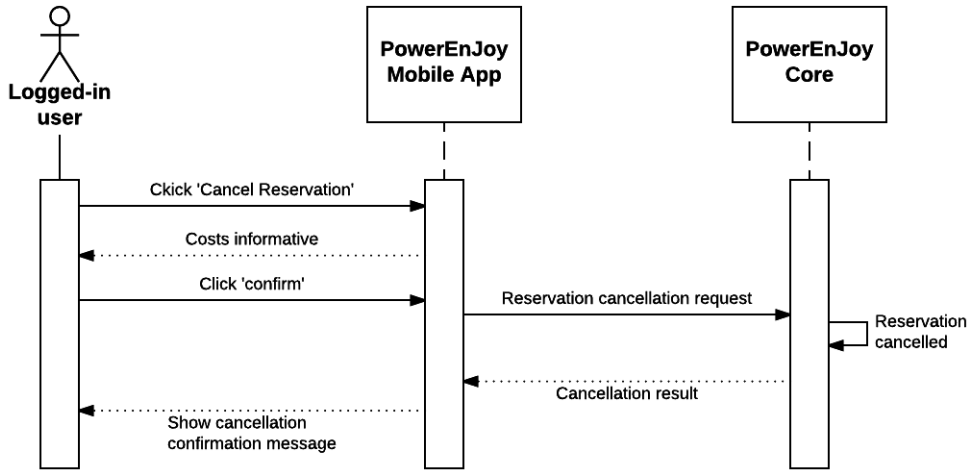

Figure 15: Sequence Diagram for 'cancel a reservation' use case

Unlock a reserved vehicle and start the rental

Actors	Logged-in user
Start conditions	• The user is authenticated. • The user has reserved a vehicle. • The user his in the reservation details page.
Events flow	• The user clicks the 'unlock vehicle' button in the application. • The application acquires user's position via GPS. • The system checks that the user is in a close range to the vehicle. • The system sends the *UNLOCK DOOR* command to the vehicle • The vehicle unlocks. • The user gets into the car. • The user starts the engine and the vehicle sends the $RENTAL - STARTED$ event to the system. • The system changes the status of the vehicle into*RENTED*. • The price of the rental is shown on the screen installed into the car.
Exit conditions	The reservation the user had over the vehicle is changed into a rental.
Exceptions	• The user is not nearby the vehicle. • The application can not acquire user's position via GPS. • The reservation the user had over the vehicle was expired.
Goal	G4

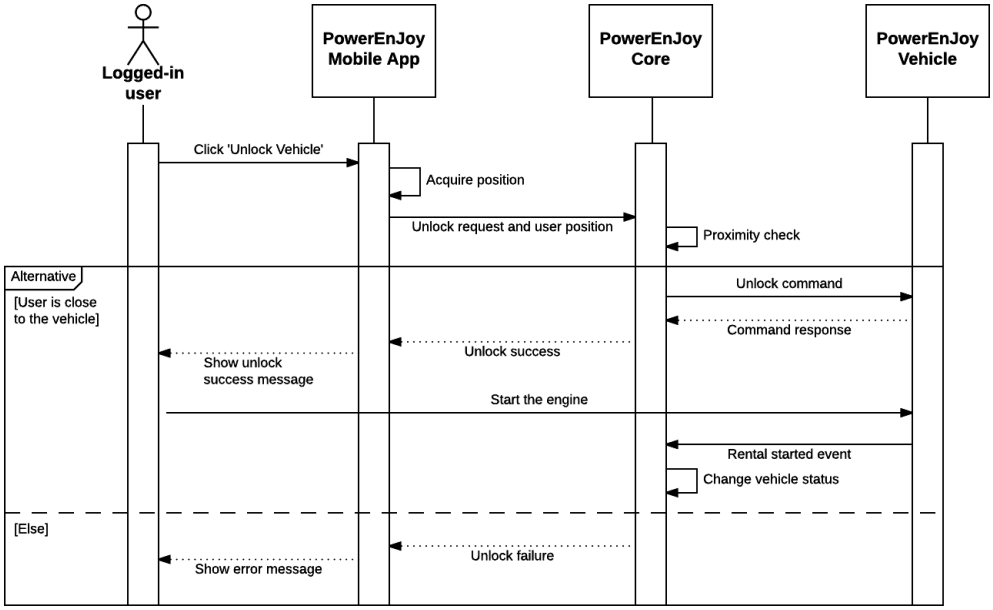

Figure 16: Sequence Diagram for 'Unlock and start the rental' use case

Terminate a rental

Actors	Logged-in user, core component
Start conditions	• The user is authenticated. • The user has rented a vehicle.
Events flow	• The user parks the car. • The user stops the engine. • The vehicle sends the *RENTAL TERMINATED* event to the system. • As all the passengers leave the vehicle locks itself. • The system changes the status of the vehicle into *AVAILABLE*. • The core component calculates the total rental fee based on the rental time and on the company pricing policies. • The core component makes the payment via the payment gateway. • The core module send an SMS to the user via the SMS gateway.
Exit conditions	The rental the user had over the vehicle is terminated.
Exceptions	• The battery level of the vehicle at the end of the rent is too low and the system does not change the status of the vehicle into *AVAILABLE* but into $NOT - AVAILABLE$. If the vehicle is not plugged to any recharging station the system issues a notification into the ControlRoom component. • The payment fails. The core component marks the payment as *PENDING* and changes the status of the user into *SUSPENDED*. • The vehicle is outside the safe area. The user can not terminate the rent.
Goal	G5

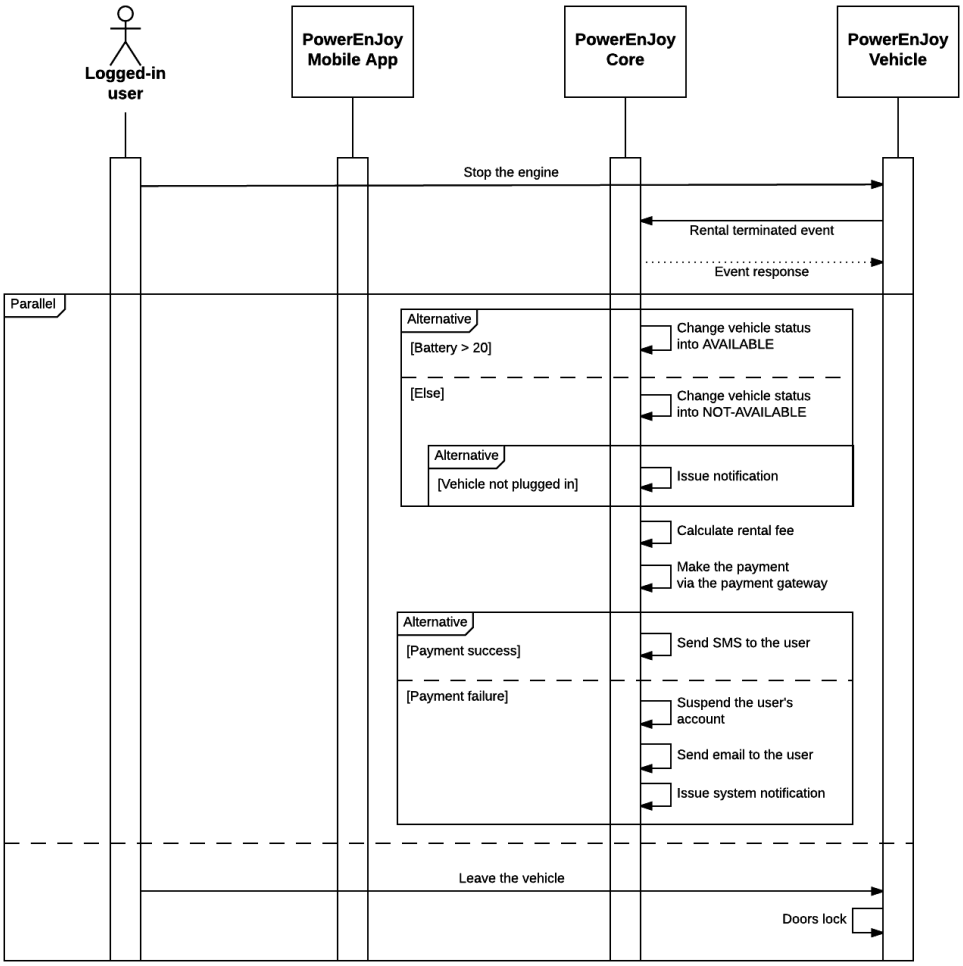

Figure 17: Sequence Diagram for 'Terminate a rental' use case

Send a support request

Actors	Logged-in user, Operator
Start conditions	• The user is authenticated.
Events flow	• The user enters the support section in the application. • The user clicks on 'Send Request' button. • The users fills-in a message containing the details of his request. • The user clicks the 'Send' button. • The system adds the support request into the company database and shows it to operators via the Control Room component. • The operator opens the request in the Control Room component. • The operator types a reply and sends it to the user. • The user receives the response both on his email address and on his smartphone.
Exit conditions	A support request is opened.
Exceptions	None
Goal	G6, G7

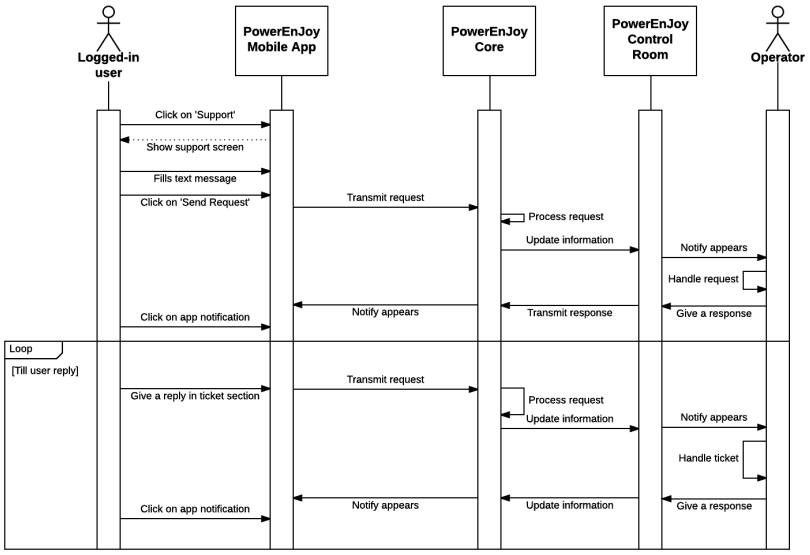

Figure 18: Sequence Diagram for 'send a support request' use case

Monitor user's activity

Actors	Operator
Start conditions	None
Events flow	• The operator accesses the 'User Activity' page. • The operator types-in the user's email address. • The system retrieves the user's last activity. • The system shows the last activity ordered by date descendantly.
Exit conditions	None
Exceptions	• The operator can change the visualization order.

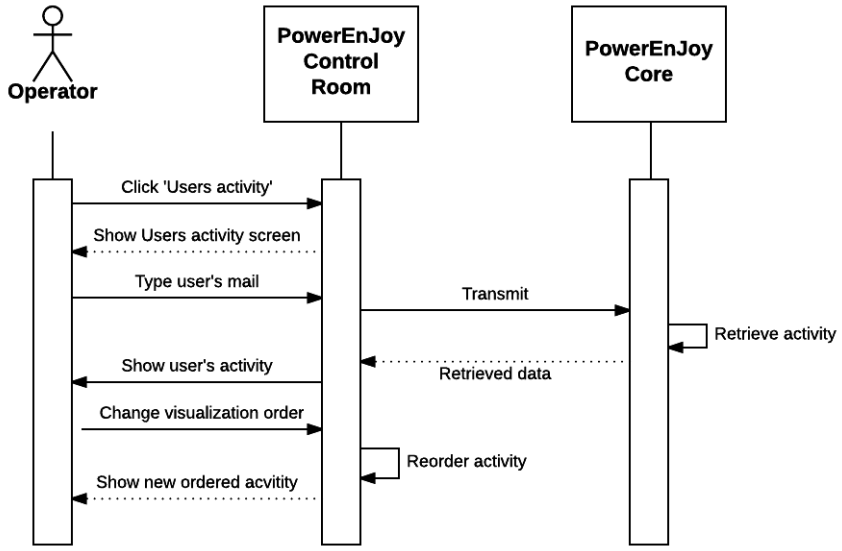

Figure 19: Sequence Diagram for 'Monitor user's activity' use case

Generate monthly invoices

Actors	Core component
Start conditions	• It is 00:01 of the first day of the month
Events flow	• The core component collects a list from the company database of all the users associated with a payment of the last month. • The core component generates an invoice containing all the payment for each user. • The core component sends the invoice in PDF format via email to the user.
Exit conditions	None
Exceptions	None

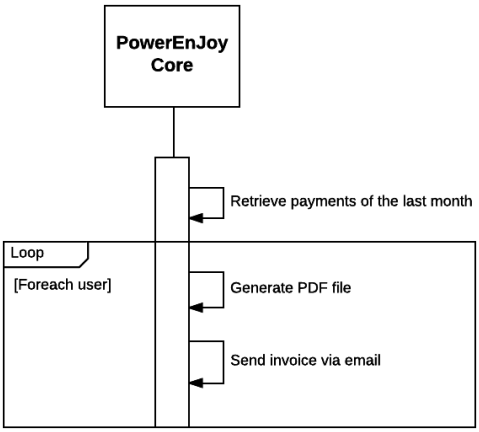

Figure 20: Sequence Diagram for 'Generate monthly invoices' use case

3.2 Software system properties

3.2.1 Reliability

The company forecasts a high amount of users, with an high growth curve especially in the first months. The system should support thousands of requests simultaneously. The system should be prepared to be scalable in order to increase the capacity of users it can handle as soon as it's required.

3.2.2 Availability

Downtime means less revenues for the company as users can not access the service. For this reason, only few hours of downtime per year could be accepted. Maintenance sessions can be scheduled for low traffic moments (e.g. during the night).

3.2.3 Security

The system manages users' personal information: communication between components must be secured using the TCP/IP and TLS protocols together with encrypting communications data using a sophisticated encryption algorithm with public and private key.

Passwords stored in the company database must be hashed (e.g. using sha3 algorithm) and salted.

Vehicle must accept commands only from the core component in order to avoid malicious unlocks: this can be achivied using security certificates or private/public key mechanisms.

3.2.4 Maintainability

In order to facilitate further implementation we should adopt a MVC pattern and use best-practises.

Software must be well documented and associated with unit and integration tests.

Components should be implemented using a modular approach in order to facilitate maintenance and scalability.

4 Appendix

4.1 Alloy model

4.1.1 Model

```
//------ SIGNATURES ------
abstract sig Bool {}
one sig True extends Bool {}
one sig False extends Bool {}

//Signatura che rappresenta l'username di un utente
sig UserUsername {}{
        //Gli username generati sono necessariamente
            associati ad uno ed un solo utente
        all uname : UserUsername | one u : User | u.
            username = uname
}

//Signatura che rappresenta l'identificativo di un
    operatore
sig OperatorID{}{
        //Gli ID generati sono necessariamente associati
            ad uno ed un solo operatore
        all id : OperatorID | one o : Operator | o.ID =
            id
}

}
//Signatura che rappresenta ShareEnJoy one sig Company{
        vehicles: some Vehicle,
        availableVehicles: set Vehicle,
        notAvailableVehicles: set Vehicle,
        reservedVehicles: set Vehicle,
        rentedVehicles: set Vehicle,
        supportRequests: set SupportRequest,
        systemNotifications: set SystemNotification }
{
        //I veicoli del mondo sono solo veicoli della
            compagnia
        no v : Vehicle | vehicles - v = vehicles
        //I veicoli sono suddivisi in disponibili, non
            disponibili, prenotati, noleggiati e non
            esiste intersezione fra i gruppi
        (vehicles = availableVehicles+
            notAvailableVehicles+reservedVehicles+
            rentedVehicles) and (availableVehicles&
            reservedVehicles = none and availableVehicles&
            notAvailableVehicles = none and
            reservedVehicles&notAvailableVehicles = none
```

```
            and rentedVehicles&notAvailableVehicles = none
             and availableVehicles&rentedVehicles = none
            and reservedVehicles&rentedVehicles = none)
}

//Signatura che rappresenta una SafeArea
sig SafeArea {}

//Signatura che rappresenta un utente
sig User{
        username: one UserUsername
}

//Signatura che rappresenta un veicolo
sig Vehicle{
        battery: one Int,
        safeArea: lone SafeArea,
        plugged: one Bool
}{battery >= 0 and battery <= 100}


//Signatura che rappresenta una reservation
sig Reservation{
        user: one User,
        vehicle: one Vehicle,
        expired: one Bool,
        cancelled: one Bool,
        payment: lone Payment
}{
        expired = True => cancelled = False
        //Se e solo se la reservation è scaduta o
            cancellata allora è associata ad un pagamento
        payment != none <=> (cancelled = True or expired
            = True)
}

//Signatura che rappresenta un noleggio
sig Rental{
        user: one User,
        vehicle: one Vehicle,
        terminated: one Bool,
        payment: lone Payment,
        duration: one Int}
{
        duration > 0
        //Se il noleggio non è terminato non è associato
            ad un pagamento
        payment != none <=> terminated = True
}
```

```
//Signatura che rappresenta un pagamento sig Payment{
        amount: one Int,
        success: one Bool
}{
        amount > 0
        //I pagamenti vengono generati solo associati ai
            rental terminati o alle reservation scadute/
            cancellate
        all p : Payment | (one r : Rental | r.terminated
           = True and r.payment = p) or (one r :
            Reservation | (r.expired = True or r.cancelled
             = True) and r.payment = p)
}

//Signatura che rappresenta un operatore sig Operator{
        ID : OperatorID
}

//Signatura che rappresenta una richiesta di supporto
sig SupportRequest{
        sentBy : one User,
        handledBy : one Operator
}

sig SystemNotification {
        //Alcune notifiche possono essere associate ad un
             operatore
        handledBy : lone Operator,
        //Una notifica può riguardare un veicolo
        vehicle: lone Vehicle,
        //Una notifica può riguardare un pagamento
        payment: lone Payment
}{
        //Una notifica non può riguardare un veicolo e un
             pagamento contemporaneamente
        vehicle != none <=> payment = none
}

//------ FACTS ------
//Facts about vehicles
fact{
        //Non possono esistere veicoli disponibili o
            prenotati fuori da una safe area perché il
            noleggio può essere terminato solo nelle safe
            area
        no v : Company.vehicles | (v in Company.
            availableVehicles or v in Company.
            reservedVehicles) and v.safeArea = none
        //I veicoli con meno del 20% di batteria non sono
             disponibili e non può esistere una
```

```
            reservation su di loro
        all v : Company.vehicles | v.battery < 20 => (v
            not in Company.availableVehicles and v not in
            Company.reservedVehicles)
        //Se un veicolo è sotto carica allora non è
            noleggiato
        all v : Company.vehicles | v.plugged = True <=> v
             not in Company.rentedVehicles
}

//Facts about reservations
fact{
        //Per ogni veicolo nel set dei veicoli riservati
            esiste una ed una sola reservation non scaduta
             o cancellata
        all v : Company.reservedVehicles | one r :
            Reservation | r.vehicle = v and r.cancelled =
            False and r.expired = False
        //Per ogni reservation esiste uno ed un solo
            veicolo nel set dei veicoli riservati
        all r : Reservation | (r.cancelled = False and r.
            expired = False) => (one v : Company.
            reservedVehicles | v = r.vehicle)
}

//Facts about rentals
fact{
        //Per ogni veicolo nel set dei veicoli noleggiati
             esiste una ed una rental non terminata
        all v : Company.rentedVehicles | one r : Rental |
             r.vehicle = v and r.terminated = False
        //Per ogni rental esiste uno ed un solo veicolo
            nel set dei veicoli noleggiati
        all r : Rental | (r.terminated = False) => (one v
             : Company.rentedVehicles | v = r.vehicle)
        //Un noleggio non puo terminare fuori da una safe
             area
        no r : Rental | r.terminated = True and r.vehicle
            .safeArea = none
        //Non esiste un noleggio terminato non associato
            ad un pagamento
        no r : Rental | r.payment = none and r.terminated
             = True
}

//Facts abot users
fact{
        //Un utente non può prenotare più di un veicolo
            allo stesso tempo
        no disj r1, r2 : Reservation | r1.user = r2.user
```

```
            and r1.expired = False and r1.cancelled =
                False and r2.expired = False and r2.cancelled
                = False          //Un utente non può noleggiare
                più di un veicolo veicolo allo stesso tempo
                     no disj r1, r2 : Rental | r1.user = r2.
                user and r1.terminated = False and r2.
                terminated = False
        //Un utente non può avere un noleggio ed una
            reservation contemporaneamente
        no ren : Rental, res : Reservation | ren.user =
            res.user and ren.terminated = False and res.
            expired = False and res.cancelled = False
        //Un utente non puo avere due richieste di
            supporto allo stesso tempo
        no disj sr1, sr2 : SupportRequest | sr1.sentBy =
            sr2.sentBy
}

//Facts about payments
fact{
        //Un pagamento puo essere associato ad una sola
            reservation o noleggio
        all p : Payment | (one rental: Rental | rental.
            payment = p) => ((no rental2: Rental | rental2
            .payment = p) and (no reservation: Reservation
             | reservation.payment = p)) and (one
            reservation: Reservation | reservation.payment
             = p) => ((no reservation2: Reservation |
            reservation2.payment = p) and (no rental:
            Rental | rental.payment = p))
}

//facts about notifications
fact{
        //Esiste una notifica per ogni veicolo
            NonDisponibile
        all nav : Company.notAvailableVehicles | one n :
            Company.systemNotifications | n.vehicle = nav
        //Esiste una notifica per ogni pagamento fallito
        all fp : Payment | fp.success = False => (one
            notif : Company.systemNotifications | notif.
            payment = fp)
        //Se una notifica è associata ad un pagamento
            allora è un pagamento fallito
        all n : Company.systemNotifications | n.payment
            != none => n.payment.success = False
}
```

4.1.2 Assertions

Assertions were used to validate the given model.

```
//------ ASSERTIONS ------
//Non esistono due utenti con lo stesso username
assert noSameUsername {
        no disj u,v: User | u.username = v.username
}
check noSameUsername for 5

//Non esistono due operatori con lo stesso ID
assert noSameID{
        no disj o1, o2: Operator | o1.ID = o2.ID
}
check noSameID for 5

//Tutti i noleggi terminati sono associati ad un
    pagamento
assert noTerminatedRentalWithoutPayment {
        no r: Rental | r.terminated = True && r.payment
            != none
}
check noTerminatedRentalWithoutPayment for 5

//Tutte le reservation scadute o cancellate sono
    associate ad un pagamento
assert noTerminatedReservationWithoutPayment{
        no r: Rental | r.terminated = True && r.payment
            != none
}
check noTerminatedReservationWithoutPayment for 5

//Un pagamento è associato ad un solo evento
assert onePaymentOneEvent{
        no disj r1, r2: Rental | r1.payment = r2.payment
        no disj r1, r2: Reservation | r1.payment = r2.
            payment
        no res: Reservation | some rent: Rental | res.
            payment = rent.payment
}
check onePaymentOneEvent for 5

//Un utente non può avere due reservation attive, un
    veicolo non può essere associato a due reservation
    attive
assert noMoreThanOneReservation{
        no disj r1, r2: Reservation | (r1.user = r2.user
            or r1.vehicle = r2.vehicle) and r1.cancelled =
             False and r2.cancelled = False and r1.expired
             = False and r2.expired = False
}
check noMoreThanOneReservation for 5
```

```
//Un utente non può avere due noleggi attivi, un veicolo
    non può essere associato a due noleggi attivi assert
    noMoreThanOneRental{
assert noMoreThanOneRental{
        no disj r1, r2: Rental | (r1.user = r2.user or r1
            .vehicle = r2.vehicle) and r1.terminated =
            False and r2.terminated = False
}
check noMoreThanOneRental for 5

//Un utente non può avere un noleggio ed una reservation
    attive allo stesso tempo, un veicolo non può essere
    associato ad una reservation ed un noleggio allo
    stesso tempo
assert noReservationAndRental{
        no u: User |
                (some rental: Rental | rental.user = u
                    and rental.terminated = False) and (
                    some reservation: Reservation |
                    reservation.user = u and reservation.
                    cancelled = False and reservation.
                    expired = False)
        no v: Vehicle |
                (some rental: Rental | rental.vehicle = v
                     and rental.terminated = False) and (
                    some reservation: Reservation |
                    reservation.vehicle = v and
                    reservation.cancelled = False and
                    reservation.expired = False)
}
check noReservationAndRental for 5

//Non può esistere un veicolo in carica e noleggiato
    assert
assert noChargingWhileDriving{
        no v: Vehicle | v in Company.rentedVehicles and v
            .plugged = True
}
check noChargingWhileDriving for 5

//Non può esistere un veicolo disponibile fuori da una
    safe area
assert AlwaysSafe{
        no v: Vehicle | v in Company.availableVehicles
            and v.safeArea = none
}
check AlwaysSafe for 5
```

4.1.3 Predicates

```
----- PREDICATES -----
pred show1{
        #User > 1
        #Reservation > 1
        #Rental > 1
        #Operator = 1
        #SupportRequest = 1
        #SystemNotification = 1
        some v : Vehicle | v.battery < 20
        some v: Vehicle | v.safeArea = none
        some v: Vehicle | v.plugged = True
        some p: Payment | p.success = False
}

pred smallScenario{
        #Vehicle <= 4
        #User <= 4
        #Operator = 1
        #SupportRequest = 0
        #Rental <= 4
        #Reservation <= 4
        some r : Rental | r.terminated = False
        some r : Reservation | r.cancelled = False and r.
            expired = False
}

run show1 for 6 but 8 int
run smallScenario for 6 but 8 int
```

The predicate show1 is intended to create a very complex simulation involving all possibile exceptions and signatures.

The predicate smallScenario is intendend to create a smaller simulation easily readble also via the Alloy visualizer.

4.1.4 Consistency results

The model is consistent by the Alloy Analizer 4.2_2015-02-22.

```
10 commands were executed. The results are:
#1: No counterexample found. noSameUsername may be valid.
#2: No counterexample found. noSameID may be valid.
#3: No counterexample found.
    noTerminatedRentalWithoutPayment may be valid.
#4: No counterexample found.
    noTerminatedReservationWithoutPayment may be valid.
#5: No counterexample found. onePaymentOneEvent may be
    valid.
#6: No counterexample found. noMoreThanOneReservation may
     be valid.
#7: No counterexample found. noMoreThanOneRental may be
    valid.
```

#8: No counterexample found. noReservationAndRental may be valid.
#9: No counterexample found. noChargingWhileDriving may be valid.
#10: No counterexample found. AlwaysSafe may be valid.
#11: Instance found. show1 is consistent.
#12: Instance found. smallScenario is consistent.

4.1.5 Generated words

In this section some words generated using the Alloy visualizer tool. All the words refer to the smallScenario predicate. In some words Integer objects are omitted in order to increase readability.

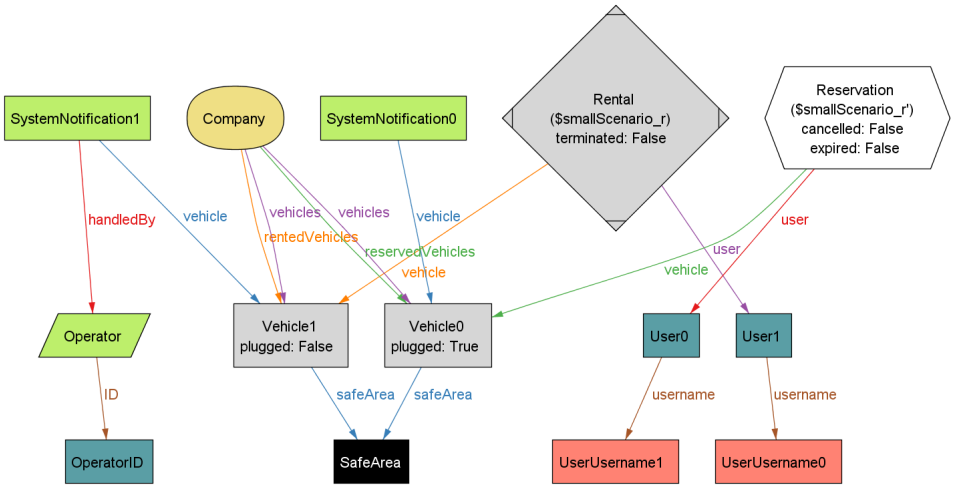

Figure 21: Alloy generated world #1

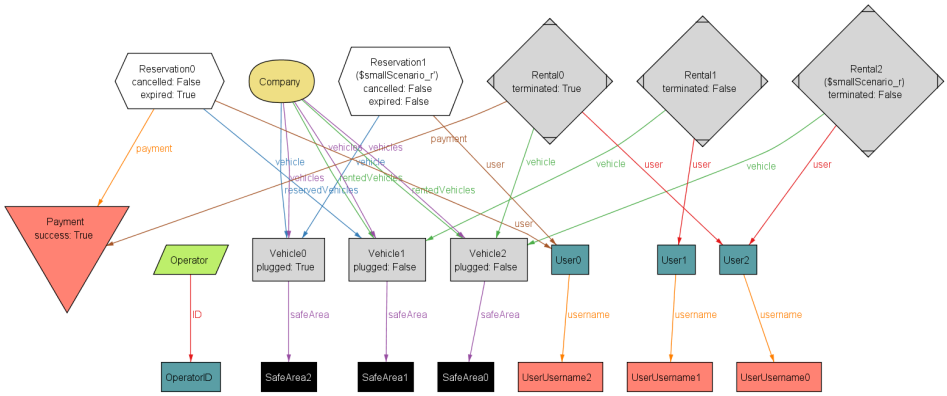

Figure 22: Alloy generated world #2

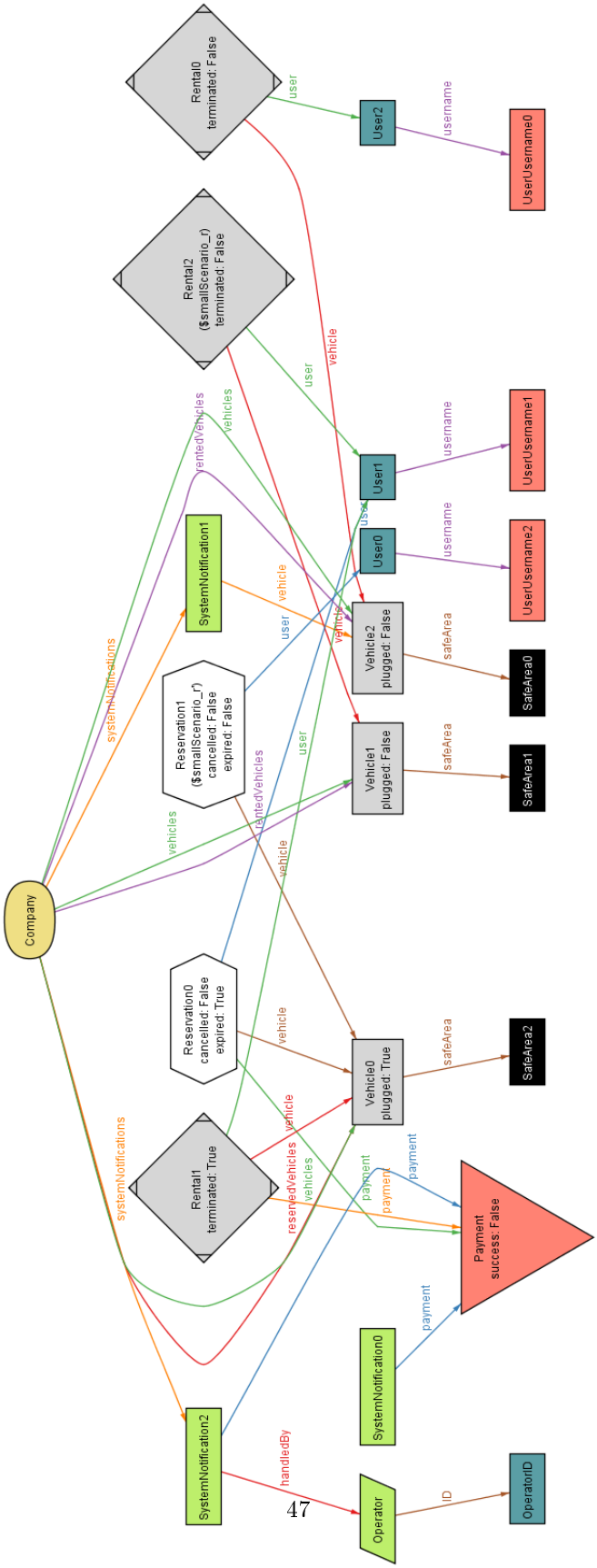

Figure 23: Alloy generated world #3

4.2 Used tools

- LyX as LaTeX editor.
- Photoshop for mockups.
- Lucidchart for diagrams.
- Alloy Analizer to validate the model of the system.
- Github as version controller.
- Google docs for beta documents sharing.

4.3 Working hour tracking

Date	Pietro Avolio	Guido Borrelli
27/10/2016	3.25	2
29/10/2016	-	2
30/10/2016	-	2
31/10/2016	2.5	2.5
1/11/2016	-	2.5
2/11/2016	6	-
3/11/2016	2	-
5/11/2016	1	3
6/11/2016	2.5	5
7/11/2016	2.5	2
8/11/2016	2	2
9/11/2016	-	-
10/11/2016	3	-
11/11/2016	7	3
12/11/2016	5	5
13/11/2016	7	10
Total	43 h 45m	41 h