POLITECNICO
MILANO 1863

Politecnico di Milano
A.A. 2016 – 2017

Software Engineering 2: "PowerEnJoy" Requirements Analysis and Specification Document

Marco Festa
November 19, 2016

Contents

1 Introduction

1.1 Purpose

The goal of the Requirement Analysis and Specification Document (RASD) is to define and entirely describe all of the system's functionalities and requirements. Specifications are set according to the customer's needs in order to create an accurate model for the system and simulate the typical use cases and scenarios occurring after development.

1.2 Scope

The aim of the project is to develop a digital management system for a car-sharing service that exclusively employs electric cars. All registered users must be verified and provide valid driving license in order to get access to the system. Once they get approved they may use all of the most common features available for every existing car sharing service. GPS and other positioning systems are crucial to provide the customer with the best experience possible, ensuring accurate car localization and position related features. The eco-friendly attitude of the company focuses on fuel-efficiency and smart-transportation topics imposing the system to capture specific customer behaviors and eventually encourage them through bonuses or lowered rates. The iteration between the customer and the system is brought out through 3 different platforms, the web interface, the smart phone app, and the on-board display. These applications are in continuous communication with the centralized system which keeps track of all sensitive information.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- *User (or Registered User):* a person registered to the system. The driving license has been verified and the customer info is correctly added into the database. Registered users are the only entities eligible for car riding.

- *Driver:* the user who physically unlocks an electric car using his own credentials and starts driving it becomes automatically the driver. Registered users are actually the only possible persons to potentially become drivers.
- *Guest:* a person that is not necessarily registered to the system.
- *Safe area:* geographical area where cars are authorized to be parked giving the user the chance to end the ride.
- *Power Grid Station:* inside the safe areas are located several power stations where electric cars can be plugged in and have the central battery recharged.
- *Free Car:* a car which is not being used by any registered user and is not under any pending reservation is considered available or free.
- *Reserved Car:* each registered user has the ability to choose a free car from the smartphone app or web interface and have it reserved for at most one hour. During this phase the car disappears from the list of free cars and can only be opened and unlocked by the user who made the reservation. The reservation state of a car ends either when the car is unlocked by the user who automatically becomes the driver, the one-hour limit is reached, or the state is ended by the user itself.
- *Opening procedure:* a user can open a free car, or one he had reserved, by using the dedicated feature on his smartphone app. An opened and locked car has to be considered a reserved car.
- *Unlocking procedure:* car unlocking is achieved via a PIN code entered through the touch screen display inside the vehicle.
- *PIN code (or PIN):* a 4-digit secret code chosen by the user on his first car ride.
- *In-use Car:* a car that has been unlocked by a registered user and is now able to be turned on by the driver.
- *CAN bus:* the physical network connecting each present electronic device: sensors, micro-controllers, actuators and instruments.

1.3.2 Acronyms

- **RASD**: Requirement Analysis and Specification Document.
- **DB**: Database
- **PGS**: Power Grid Station
- **CAN bus**: Controller Area Network bus
- **API**: Application Programming Interface: a common way to communicate with other systems.

1.3.3 Abbreviations

- **[Gx]**: Goal
- **[RE.x]**: Functional Requirement
- **[UC.x]**: Use Case

1.4 Reference Documents

- ISO/IEC/IEEE Std. 29148:2011, "Systems and software engineering - Life cycle processes – Requirements engineering"
- Specification document: "Assignments AA 2016-2017.pdf"

1.5 Document Overview

Document structure:

- **Section 1** – **Introduction**: presentation of the document and product.
- **Section 2** – **Overall Description**: specifies the background of assumptions and constraints necessary to describe the software product.
- **Section 3** – **Specific Requirements**: this section gets deeper into product requirements. Lists a variety of possible scenarios and implemented features. Use cases, UML diagrams and eventually mockups are shown to give a clearer vision of the final product aspect and functionality.
- **Section 4** – **Appendix**: in this section is shown an alloy model generated on the application domain, various tools used to write the document and the work hours of all the document relators.

2 Overall Perspective

2.1 Product Perspective

2.1.1 Integration with external systems

Being the system practically a standalone implementation, integrations with external resources are few. Basic API interfaces are used to handle driving license verification and payment processing. All of the necessary sensors and instruments are already configured and linked to the CAN bus of the electric cars ready to be handled and interpreted by our custom software installed on the onboard device.

2.1.2 Domain model

State chart and class diagram are shown below. The car state chart explains in detail the various possible car states and the transitions between them

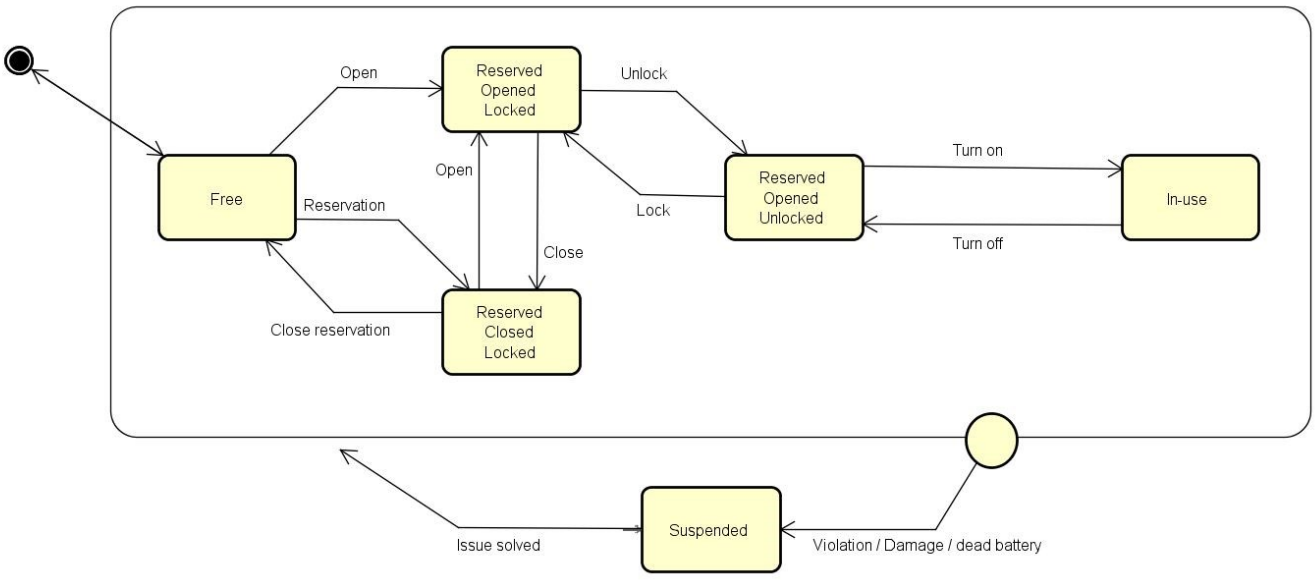

Figure 1: Car State Chart

The class diagram shown below displays the class structure chosen to manage all sensitive information for the system in order to make it reliable and efficient.

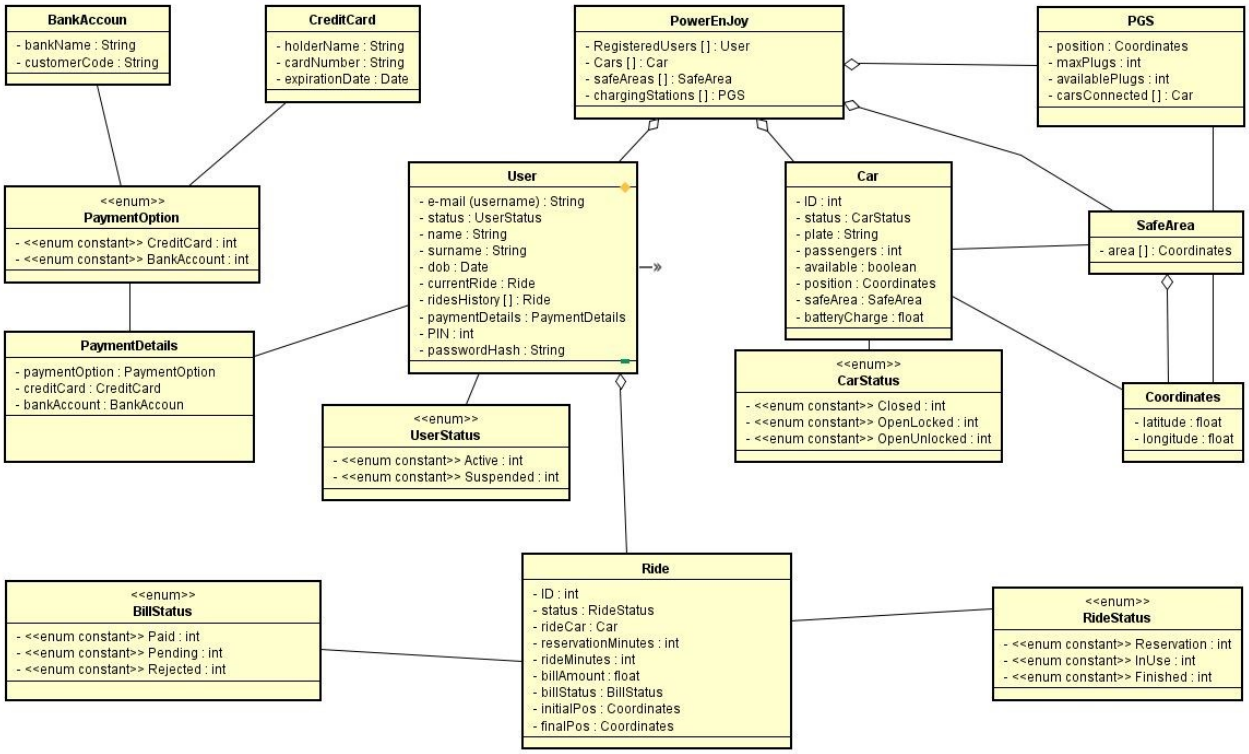

Figure 2: Class Diagram

2.2 Product Functions

In this section are listed all the goals we aim to accomplish through the different functionalities we plan to implement.

[G1] Users must be able to register to the system by providing all necessary credentials and payment information. They will receive back via e-mail a password to access the system.

[G2] User are able to login using their personal login credentials (e-mail/password).

[G3] User must be able to search for available cars nearby or close to every chosen location. For each available car battery charge and specific safe area must be visible.

[G4] User must be able to reserve a single car for up to one hour before they pick it up. If the reserved car is not picked-up within the one-hour limit reservation in cancelled and the user pays a 1 EUR fee.

[G5] A user has the chance to cancel his reservation at any given moment.

[G6] A user near to a free or reserved (by him) car must be able to communicate his position in order to let the car open if he's detected close to it.

[G7] Once the car has been opened the user must be able to unlock it using his private 4-digits PIN code.

[G8] The first time a registered user gets into a car he has to be prompt with a message allowing him to choose his personal PIN code.

[G9] As soon as the car is unlocked the "reservation" state ends, the engine can now be turned on and the system starts charging the user for a given amount of money per minute.

[G10] At each moment the user has to be notified with the current amount he's being charged.

[G11] Within the car safe area, the user must be able to stop the vehicle, lock it and close it to finish his ride.

[G12] The system must be able to close the car automatically once the user is detected outside the vehicle.

[G13] The system stops charging the client as soon as the car is empty and closed.

[G14] The system must apply a discount of 10% on the last ride if the user took at least two other passengers onto the car.

[G15] The system must apply a discount of 20% on the last ride if the car is left with more than 50% of remaining battery charge.

[G16] The system must give the user a way to notify if there are any problems with the chosen car.

[G17] The system has to suspend a user if the ride payment fails and restrict access to cars until the issue is solved.

2.3 User Characteristics

All of the registered users are in possession of a valid driving license and have accepted all of the terms and conditions imposed by the company during registration phase. They have received valid credentials to login the system and start searching for available cars. Since the only way to open PowerEnJoy electric cars is via the smartphone app each user is required to have an Android or IOS device.

2.4 Constrains

2.4.1 Regulatory policies

All sensitive data and user information are acquired by the company under the accepted terms and conditions. This data is stored in the company DB; it's use and transmission to third parts society is regulated in accordance to the law.

2.4.2 Hardware limitations

Except for an internet connection and a compatible smartphone, the system does not require any specific hardware limitation.

2.4.3 Interfaces to other applications

Specific API integrations is used to perform credit card payments and check the user's driving license validity in real time.

2.4.4 Parallel operations

PowerEnJoy system must support parallel operations from different users when working with the DB in order to avoid collision or any other integrity issue.

2.5 Assumptions and Dependencies

2.5.1 Dependencies

Car hardware: in order to provide all the services and functionalities specified all of the electric cars must have:

- GPS tracker;
- Mobile internet connection;
- A powerful embedded system to provide all the required functionalities and communicate to the central system;
- A touch screen monitor to handle user input to the embedded device;
- Accurate sensors to collect data from engine, battery and instruments, a weight sensor positioned under each seat to calculate the number of present passengers.
- A hardware connection between the embedded system and the CAN bus of the car to control door locking/unlocking and receive data from the sensors.

Mobile application: access to the positioning system has to be granted by the user in order to open the car. Internet connection is also mandatory to communicate the position to the central system to manage the request.

Web application: there are no specific dependencies other than an internet connection to use the web application. No specific plugin or browser extension are required. Browser location detection is optional to find "around me" cars.

Power grid station: each power grid station is connected to the central system through and internet connection to send information about free recharging spots or possible damaged plugs.

2.5.2 Assumptions

Users: the user who has reserved or directly opened the car is the only user entitled to drive it. All users are considered honest and not interested in cheating the weight sensors to achieve the tariff discount. Users always reports damages or bad interior conditions for each car they use.

Cars: cars are always connected to the internet and the GPS signal is to consider stable. Each car has its own saved safe area

where they are allowed to be parked. Different cars of course can share the same safe area. An external company is in charge to provide the following services to PowerEnJoy: cleaning the car if necessary, recharging the batteries if cars are left outside a PGS and run out of charge, reintegrate abandoned cars inside the specific safe area. A car has the ability to maintain enough battery charge to perform crucial operations: closing itself if abandoned, send to the central system the current position, send as much data as possible to the central system if any misuse is detected. For safety reasons a car cannot close itself if any passenger is detected inside the vehicle.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

PowerEnJoy system proposes 3 different user interfaces. Each of them is accessible via a different device. The web application is accessible both on a standard browser window on a personal computer or a smartphone device. The dedicated apps are installed on either a IOS or an Android device. The car custom display also presents a graphical user interface to interact with its internal functionalities.

Web Application UI: the responsive web interface is compatible with each screen size and orientation; it's intuitive and easy to use. The registration process is available only from this platform which instead lacks the "open now" feature. In the pages below are shown a couple examples of what the final aspect of the interface should be.

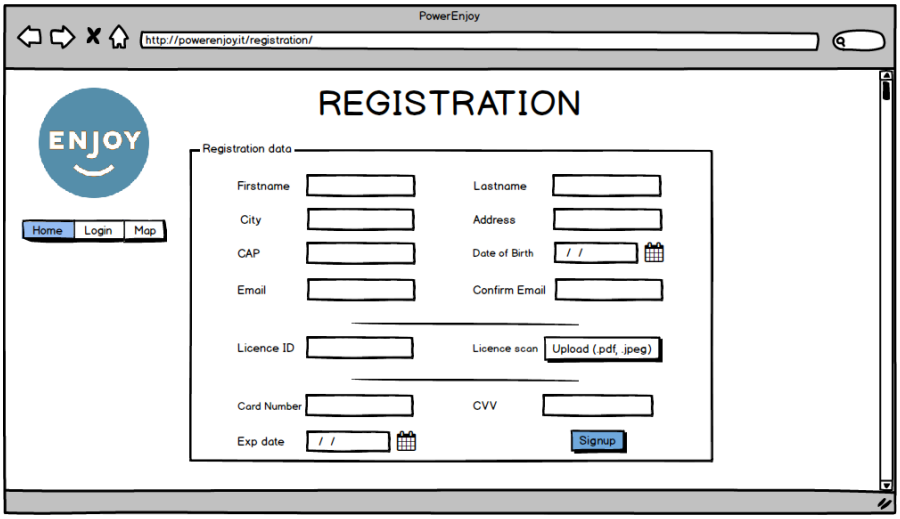

Figure 3: registration page

Figure 4: login window

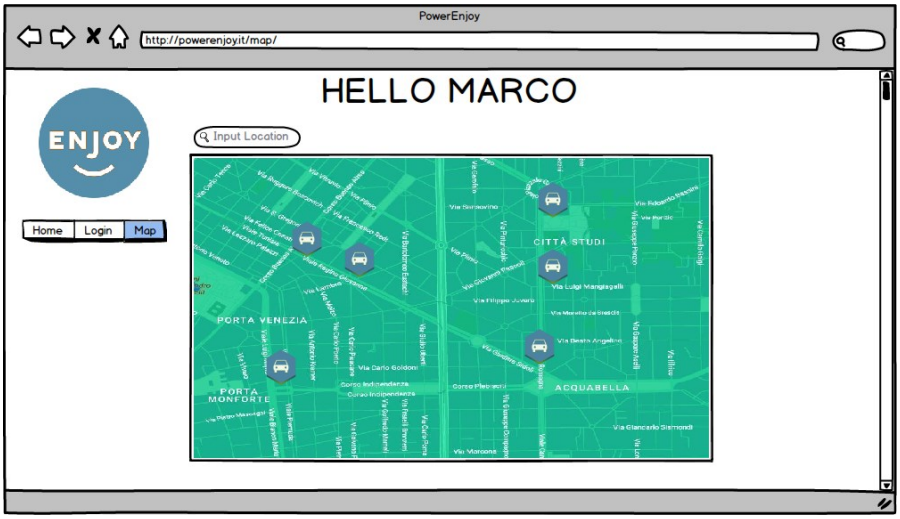

Figure 5: city map page

Figure 6: reservation window

Mobile App UI: the mobile UI allows the user to log with his credential and save them. The user has the chance to search car around his current position and open them once he's detected sufficiently close to it.

Figure 7: search for a car

Figure 8: reserve or open a car

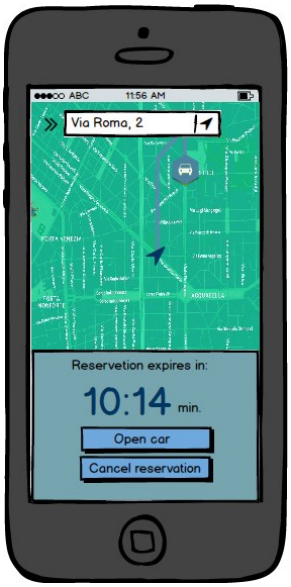

Figure 9: get directions to the car

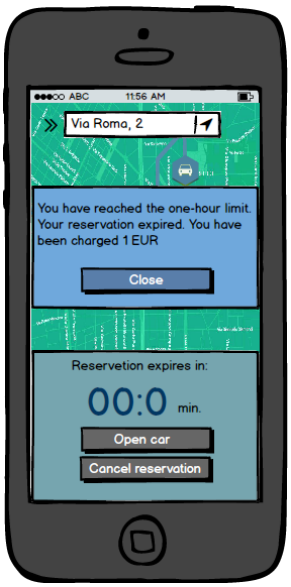

Figure 10: reservation expired

On-board touch screen application: the embedded system installed in the car is connected to a 10-inch touch screen monitor to read user input and display information through a simple and user-friendly interface.

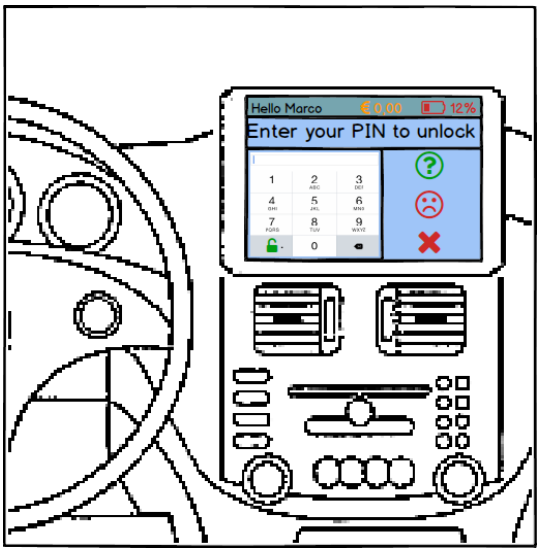

Figure 10: PIN unlock screen

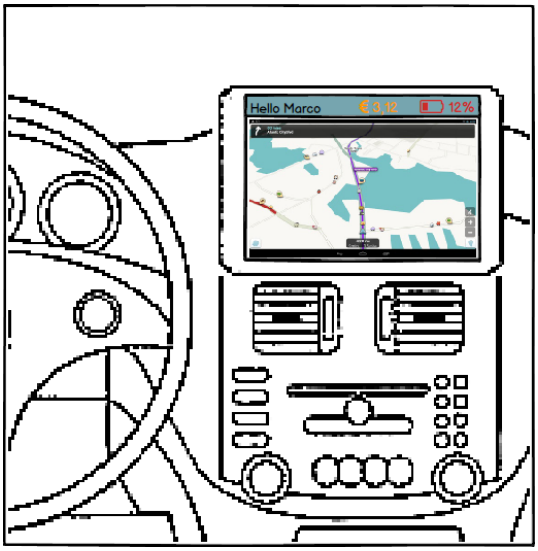

Figure 11: GPS navigation

3.1.2 Hardware Interfaces

The embedded system installed inside the car is connected to the CAN bus to collect data from all sensors and instruments. All of this data is either shown to the user through the on-board monitor or sent to the central system via the internet connection. The embedded system also has the ability to upgrade its internal software when the central system dispatches security or functionality updates.

3.1.3 Software Interfaces

- Database Management System (DBMS)
  - Name: MySQL
  - Source: http://www.mysql.it
- Java Application Framework
  - Name: Glassfish 4.1
  - Source: https://glassfish.java.net
- Java Virtual Machine (JVM)
  - Name: JEE
  - Source: http://www.oracle.com
- Payment APIs
  - Name: Stripe
  - Source: https://stripe.com/it
- EUCARIS Verification APIs
  - Name: IDcheck.io
  - Source: https://www.eucaris.net/
- Operating System (OS)
  - Any OS supporting JVM and DBMS specified above.
  - Open source Linux based OS installed on the car embedded device.

3.1.4 Communication Interfaces

All of the communication between the various components of the whole system are handled through HTTPS protocol in accordance with the highest security standards. Web application and mobile application may also require Web Socket protocol to integrate advanced functionalities.

3.2 Functional Requirements

3.2.1 Requirements

[G1] Users must be able to register to the system by providing all necessary credentials and payment information. They will receive back via e-mail a password to access the system.

- [RE.1.1] The system must offer a registration form via a web application interface where the user fills all of his personal information.
- [RE.1.2] The system must check the validity of all the data provided by the registering user.
- [RE.1.3] Once the registration data is submitted to the central system a password is randomly generated, sent to the user via email and saved hashed in the DB.
- [RE.1.4] Once all documents and information are verified a mail is sent to the user to confirm his registration.

[G2] User are able to login using their personal credentials (e-mail/password).

- [RE.2.1] The system must offer an interface to for the user to enter his credentials.
- [RE.2.2] If matching credentials are found in the DB the user is now to be considered logged in and able to access all of the system features.
- [RE.2.3] The system must provide a way to keep the user logged in a permanent way without having to submit mail and password each time (e.g. using a session cookie.)
- [RE.2.4] If the submitted credentials are not found in the DB the system has to reject and notify the user.

[G3] User must be able to search for available cars nearby or close to every chosen location. For each available car battery charge and specific safe area must be visible.

- [RE.3.1] The system must offer an interface and display a digital map with all the available cars.
- [RE.3.2] The system's interface must present a button to center the map on the user location and display nearby cars.

[RE.3.3] The system's interface must present a text box to input a given address and display all the cars near the input location.

[RE.3.4] For each shown car in the digital map the system offers a way to check car battery level and the car's safe area.

[G4] User must be able to reserve a single car for up to one hour before they pick it up. If the reserved car is not picked-up within the one-hour limit reservation in cancelled and the user pays a 1 EUR fee.

[RE.4.1] A user must be able to select a car and reserve it.

[RE.4.2] The system has to mark the selected car as unavailable to every other user and start counting the reservation time.

[RE.4.3] After a one-hour period in which the car is not unlocked by the user the system has to cancel the user's reservation and notify the 1 EUR fee.

[G5] A user has the chance to cancel his reservation at any given moment.

[RE.5.1] Within the one-hour reservation limit the user has the chance to cancel his reservation through the web or mobile application

[G6] A user near to a free or reserved (by him) car must be able to communicate his position in order to let the car open if he's detected close to it.

[RE.6.1] During the reservation period the system has to constantly check the user position through the mobile application.

[RE.6.2] Once the user is detected near his reserved car the system enables the "Open Car" function on the mobile application.

[RE.6.3] Once the "Open Car" input is sent to the system, the electric switch inside the vehicle must activate and open the car doors.

[RE.6.4] The system must offer a way to open a car directly if the user is detected sufficiently near a free car.

[G7] Once the car has been opened the user must be able to unlock it using his private 4-digits PIN code.

[RE.7.1] The on-board device must prompt the user with the PIN request in order to unlock the car and start the engine.

[RE.7.2] The car embedded device must have the ability to detect a user entering the vehicle.

[G8] The first time a registered user gets into a car he has to be prompt with a message allowing him to choose his personal PIN code.

[RE.8.1] By communicating with the central system the car detects if this is a user's first ride.

[RE.8.2] A dedicated interface on the on-board touch screen monitor prompts the user to choose his private 4-digit PIN.

[G9] As soon as the car is unlocked the "reservation" state ends, the engine can now be turned on and the system starts charging the user for a given amount of money per minute.

[RE.9.1] The car must have a hardware switch to turn on the engine. This can be activated only if the car is in an unlocked state.

[RE.9.2] The system starts to charge the user the moment the car is unlocked.

[RE.9.3] Every minute the system charges a specific amount of money on the ride bill.

[G10] At each moment the user has to be notified with the current amount he's being charged.

[RE.10.1] The on-board device constantly has to show the amount of money the user is being charged for the current ride.

[G11] Within the car safe area, the user must be able to stop the vehicle, lock it and close it to finish his ride.

[RE.11.1] The system must detect at any moment the position of the car and if its position is respecting the specific car's safe area.

[RE.11.2] If the car is inside the safe area the user can use a specific interface to switch off the car, lock it, leave it and close it.

[G12] The system must be able to close the car automatically once the user is detected outside the vehicle.

- [RE.12.1] At any moment the system has to ensure that if a car is left empty it has to lock and close itself as soon as possible.
- [RE.12.2] If a car gets locked and closed automatically outside the safe area this is considered a misuse of the system and a violation of the accepted terms on behalf of the user. A report case is filled and left to the company to handle.

[G13] The system stops charging the client as soon as the car is empty and closed.

[G14] The system must apply a discount of 10% on the last ride if the user took at least two other passengers onto the car.

- [RE.14.1] The weight sensors detect the presence of two or more passengers. The car internal system sends this information to the central system.
- [RE.14.2] When the ride is ended by the user and no misuse has been detected the system applies the discount rate to the ride bill.
- [RE.14.3] The on-board screen interface notifies the user the discount has been applied.

[G15] The system must apply a discount of 20% on the last ride if the car is left with more than 50% of remaining battery charge.

- [RE.15.1] At the beginning of each ride the embedded system memorizes the state of the battery reading this data from the dedicated battery sensors.
- [RE.15.2] At the end of the ride the embedded system calculates the amount of energy spent and notifies the central system.
- [RE.15.3] If the discount requisites are satisfied this is applied and notified to the user.

[G16] The system must give the user a way to notify if there are any problems with the chosen car.

- [RE.16.1] A dedicated interface on the on-boar monitor allows the user to contact the company's client service department.
- [RE.16.2] The system must have a dedicated section of its DB to save various users reports.

[G17] The system has to suspend a user if the ride payment fails and restrict access to cars until the issue is solved.

[RE.17.1] The system must be able to mark a user's state to suspended.

[RE.17.2] The system must restrict suspended users from reserving or opening any of the electric cars.

3.2.2 Use Cases

In this section is proposed a series of possible use cases for the custom central system software.

- [UC.1] A guest registers to PowerEnJoy.
- [UC.2] A user logs in the PowerEnJoy system.
- [UC.3] A user searches for free cars.
- [UC.4] A user reserves a car.
- [UC.5a] A user opens a reserved car.
- [UC.5b] A user opens a free car.
- [UC.6] A user enters his PIN to unlock a car.
- [UC.7] A user ends a ride.

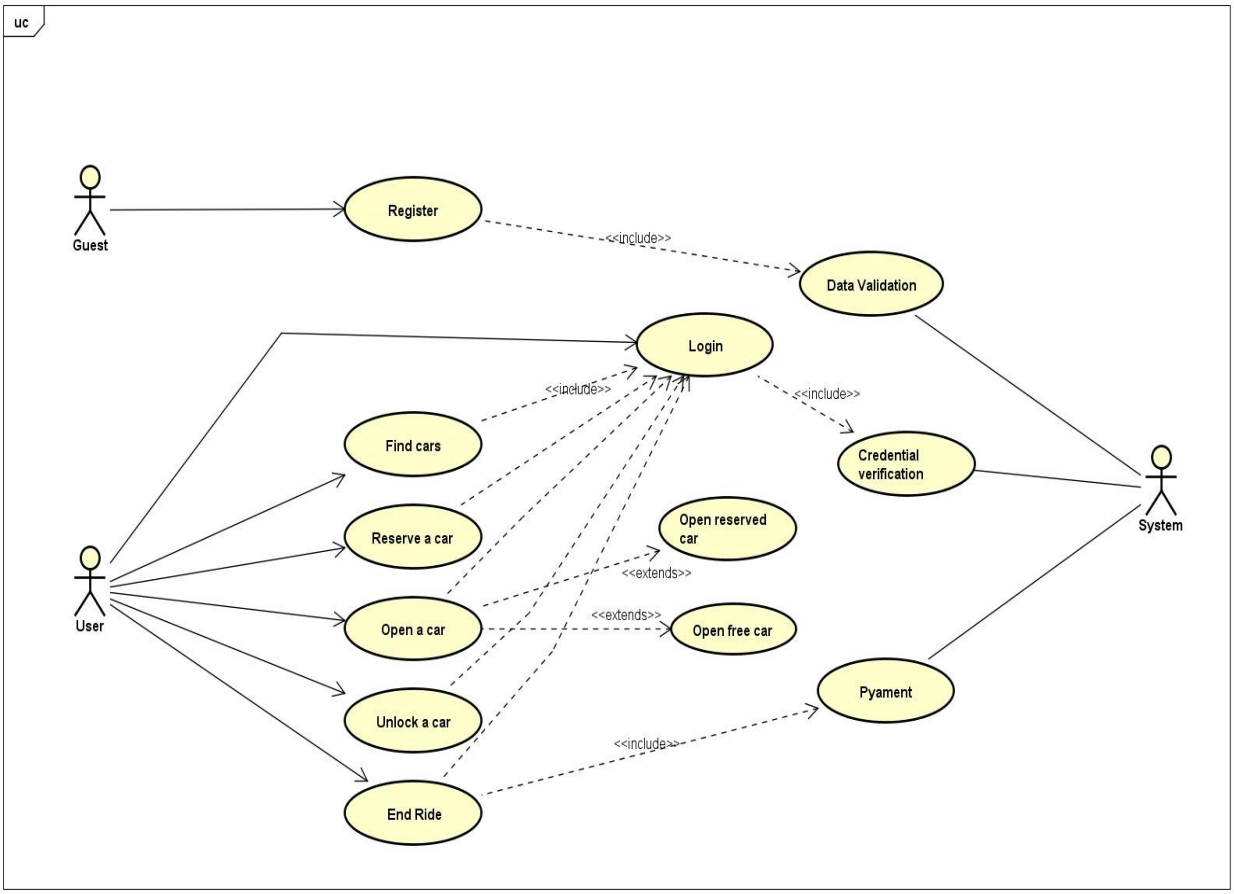

Figure 12: Use case diagram

[UC.1] A guest registers to PowerEnJoy.

Actor	Guest
Goal	**[G1]**
Preconditions	Guest must not be already registered
Execution Flow	1. Gest clicks on the "register" link from the homepage of the web application. 2. Guest fills in all necessary information in the registration form. 3. The guest must upload a picture of his document and driving license. 4. The guest fills in the payment information required. 5. Payment method is immediately tested by the system. 6. The guest has to accepts rights and terms and all the necessary conditions. 7. The driving license data is verified through the dedicated API. 8. If verification is successful, the system registers all of the user's data in his DB. 9. The system sends a mail to the user with a newly generated password. 10. The systems inform the user on the web application interface that the login password has been sent to his personal e-mail address and registration is complete.
Postconditions	The guest completed the registration process and is now able to ride the electric cars.
Exceptions	1. Guest is already registered. 2. Mail is already in use.

[UC.2] A user logs in the PowerEnJoy system.

Actor	User
Goal	**[G2]**
Preconditions	User must be registered to the system.
Execution Flow	1. The guest opens PoweEnJoy web application or mobile app. 2. The guest types email and password. 3. The system searches for said credentials in its DB. 4. The user is authenticated and redirected to the map interface. 5. The user receives an authentication token by the system to prevent him from logging in every day.
Postconditions	The user is logged into the system and has access to all of its functionalities.
Exceptions	1. Credentials are not found in the DB. Login fails.

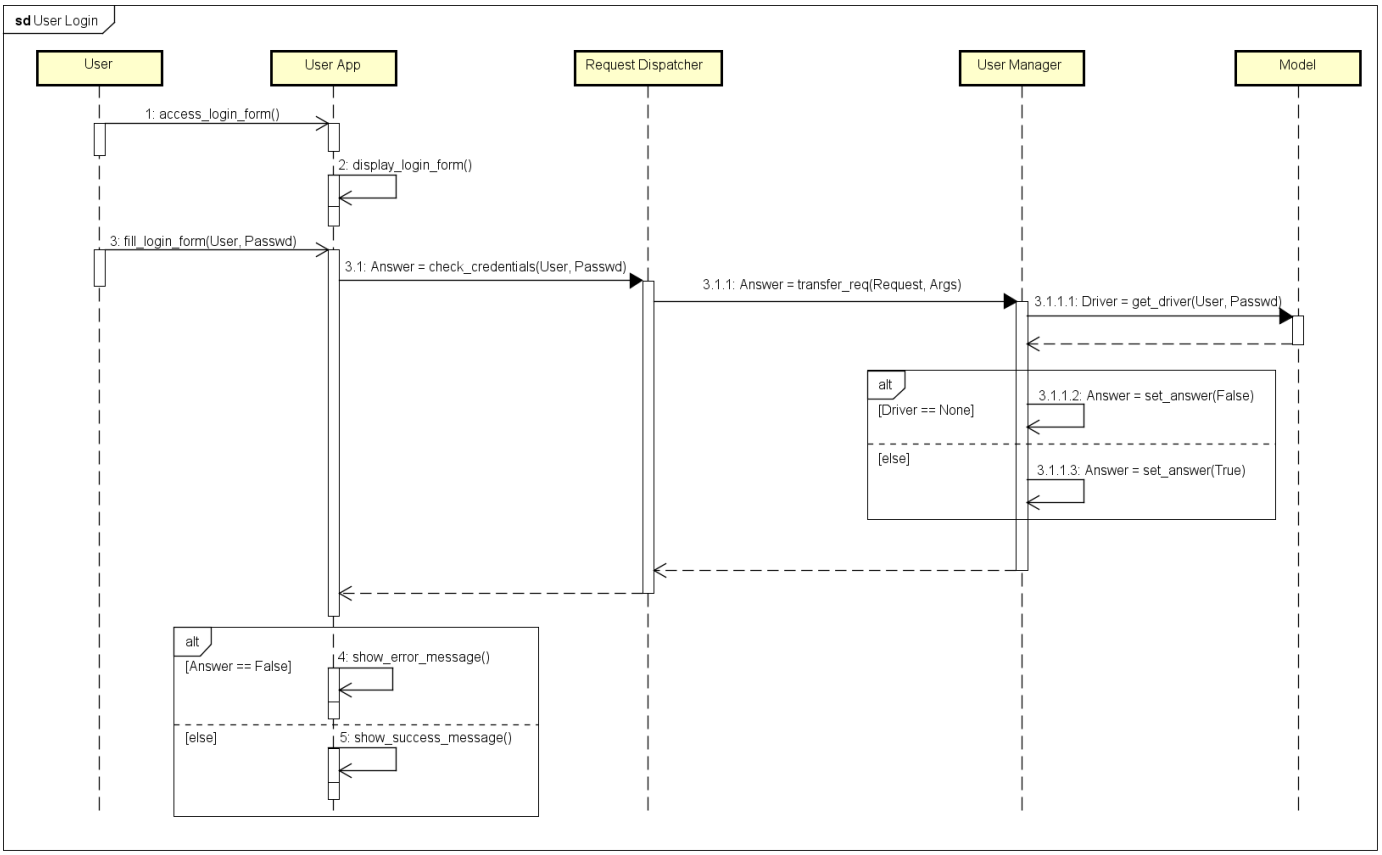

Figure 13: Login sequence diagram

[UC.3] A user searches for a free car.

Actor	User
Goal	**[G3]**
Preconditions	User must have logged into the system.
Execution Flow	1. User goes to the main map page of the application. 2. User either inputs an address or presses the "around me" button. 3. In the case of an input address the system verifies the correctness and eventually proposes a correction. 4. The system centers the map to the said location and displays all available cars.
Postconditions	All the available cars are shown on the map in the application.
Exceptions	1. There are no available cars. In this case, the system suggests to attempt a new research.

[UC.4] A user reserves a free car.

Actor	User
Goal	**[G4]**
Preconditions	User must have logged into the system and chosen an available car.
Execution Flow	1. The user selects the desired car from among the ones shown in the digital map. 2. The system shows the battery level and the exact address of the car. 3. The user clicks on the reserve button to reserve the car. 4. The system generates a ride object to save the reservation information.
Postconditions	The selected car availability is set to "False" by the system. The ride status is set to "Reservation"
Exceptions	1. In the time the user selected the car and pressed the reserve button someone else has reserved the same car.

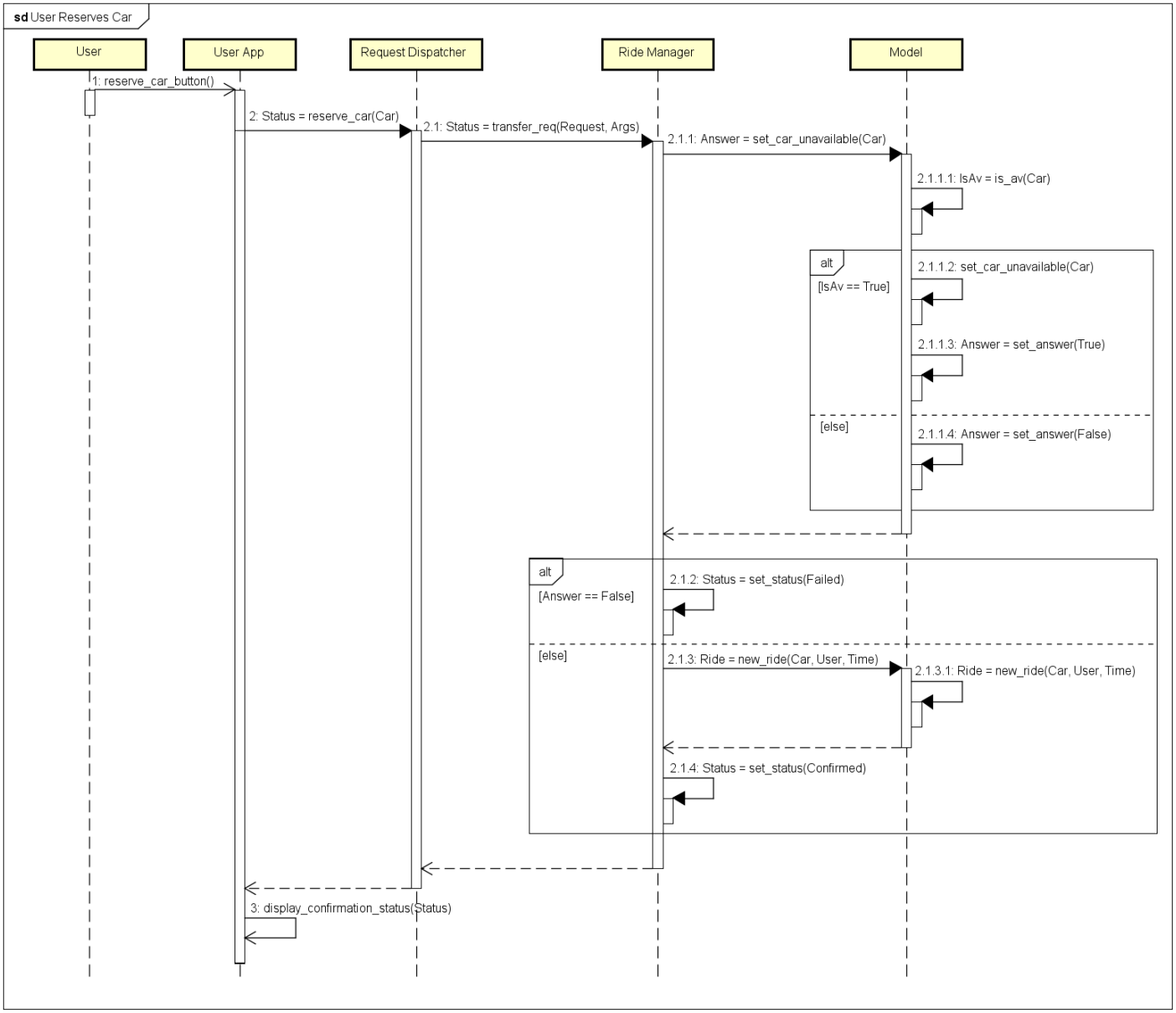

Figure 14: Reservation sequence diagram

[UC.5a] A user opens a reserved car.

Actor	User
Goal	**[G6]**
Preconditions	User must have logged into the system and reserved an available car. User must have his mobile device with him. User has to be near to the desired car.
Execution Flow	1. User pushes the open button on his mobile application. 2. The system verifies the position of the two entities (the user and the car) 3. The systems sends to the car the order to open its doors.
Postconditions	The car is now open and the user can enter the vehicle. The ride status is still marked as “Reservation”. Car status passes from “Closed” to “OpenLocked”
Exceptions	No exceptions

[UC.5b] A user opens a free car.

Actor	User
Goal	**[G6]**
Preconditions	User must have logged into the system. User must have his mobile device with him. User has to be near to the desired available car.
Execution Flow	1. User selects on the mobile app interface the car he desires to open. 2. The system checks the distance between user and car and enables the "open now" feature on the mobile app. 3. The user is prompted with the possibility to open the car directly. 4. The user pushes the "Open" button on his mobile application.
Postconditions	The car is now open and the user can enter the vehicle. The ride object is created and the ride status set to "Reservation". Car availability is set to "False". Car status passes from "Closed" to "OpenLocked"
Exceptions	No exceptions

[UC.6] A user enters his PIN to unlock a car.

Actor	User
Goal	**[G7]**
Preconditions	User has entered the electric car. User has already chosen his PIN code in a past ride with PowerEnJoy.
Execution Flow	1. The car display promptly presents the user the unlocking screen. (Figure 10) 2. The user types in his 4-digits PIN. 3. The sends the PIN to the central system to verify it. 4. The car unlocks. 5. The system starts charging the user.
Postconditions	The car status is now set to "OpenUnlocked". This status is the only one that allows the engine to be turned on. Ride status is set to "InUse"
Exceptions	1. The user inserts an invalid PIN.

[UC.7] A user ends a ride.

Actor	User
Goal	**[G11]**
Preconditions	User is in a "InUse" ride. The user is stopping in the car's safe area.
Execution Flow	1. The user stops the car. 2. The user presses the lock option on the car's display. 3. The system checks if there is any applicable discount and eventually notifies it to the user. 4. The user exits the car. 5. The car's internal system detects the emptiness of the car and locks it immediately.
Postconditions	The car status is now set to "Closed". Ride status is set to "Finished". Car availability is set to "True"
Exceptions	No exceptions

3.3 Performance Requirements

Performance is certainly a key aspect of the system since a lot of overhead and pending operations are expected from GPS and mobile communication. The response time of the software itself can be considered close to 0. Parallelism grants multiple requests to be satisfied in said response time.

4 Appendix

4.1 Alloy

The complete alloy file (.als) is included in the repository. The following alloy model is created following the class diagram.

```
open util/integer as integer

//---SIGNATURES---

abstract sig Boolean {}
one sig True extends Boolean {}
one sig False extends Boolean {}

one sig PowerEnjoy {
    registeredUsers : set User,
    cars : some Car,
    safeAreas: some SafeArea,
    chargingStations : some ChargingStation
}

abstract sig UserStatus{}
sig Active, Suspended extends  UserStatus{}

sig User{
    status: one UserStatus,
    currentRide : lone Ride,
    ridesHistory : set Ride,
}{
    status = Suspended implies currentRide = none
}

abstract sig CarStatus{}
sig Closed, OpenLocked, OpenUnlocked extends
CarStatus{}

sig Car{
    status : one CarStatus,
    passengers : one Int,
    available : one Boolean,
    position: one Position,
    safeArea : one SafeArea,
    battery : one Int
}
{
    passengers >= 0 and passengers <= 5
    (status= OpenLocked or status = OpenUnlocked)
implies available = False
    battery >=0 and battery<=10
    passengers != none implies status = Closed
    battery = 0 implies available = False
}
```

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

sig SafeArea{
    area : some Position,
}{
    #(area)>2
}

sig ChargingStation{
    position : one Position,
    maxPlugs : one Int,
    availablePlugs : one Int,
    carsConnected : set Car
}{
    maxPlugs >= 0
    availablePlugs >= 0
    maxPlugs >= availablePlugs + #(carsConnected)
}

sig Ride{
    status : one RideStatus,
    car : one Car,
    reservationMinutes : one Int,
    rideMinutes : one Int,
    billAmount : one Int, //int?
    billStatus : one BillStatus
}{
    billAmount >= 0
    reservationMinutes >= 0
    rideMinutes >= 0
}

abstract sig BillStatus {}
sig Paid, Pending, Rejected extends BillStatus{}

abstract sig RideStatus{}
sig reservation, InUse, Finished extends RideStatus{}

//---FACTS---

fact{
    no disjoint x, y : User | x.currentRide =
y.currentRide
}

fact {
    all u: User | lone r:Ride | (r in u.ridesHistory
and r.billStatus != Paid)
}
```

```
fact{
    all u : User | u in PowerEnjoy.registeredUsers
}
fact{
    all us : UserStatus | us in User.status
}
fact{
    all c : Car | c in PowerEnjoy.cars
}
fact {
    all sa : SafeArea | sa in PowerEnjoy.safeAreas
}

fact{
    all cs : ChargingStation | cs in
PowerEnjoy.chargingStations
}

fact {
    no disjoint p1,p2 : Position|
p1.latitude=p2.latitude and p1.longitude=p2.longitude
}

fact{
no disjoint sa1, sa2: SafeArea,  p: Position| p in
sa1.area and p in sa2.area
}

fact{
no disjoint cs1, cs2: ChargingStation,  p: Position|
p=cs1.position and p=cs2.position
}

fact{
all p: Position | (p in SafeArea.area or p in
Car.position or p = ChargingStation.position)
}

fact{
    all rs : RideStatus | rs in Ride.status
}

fact{
    all bs : BillStatus | bs in Ride.billStatus
}
fact{
    all cs : CarStatus | cs in Car.status
}


fact{
    no disjoint x, y : Ride | x.car = y.car
```

```
}


//---PRED---

pred show(){
#(User)>2

}

run show for 4 but 7 Int
```

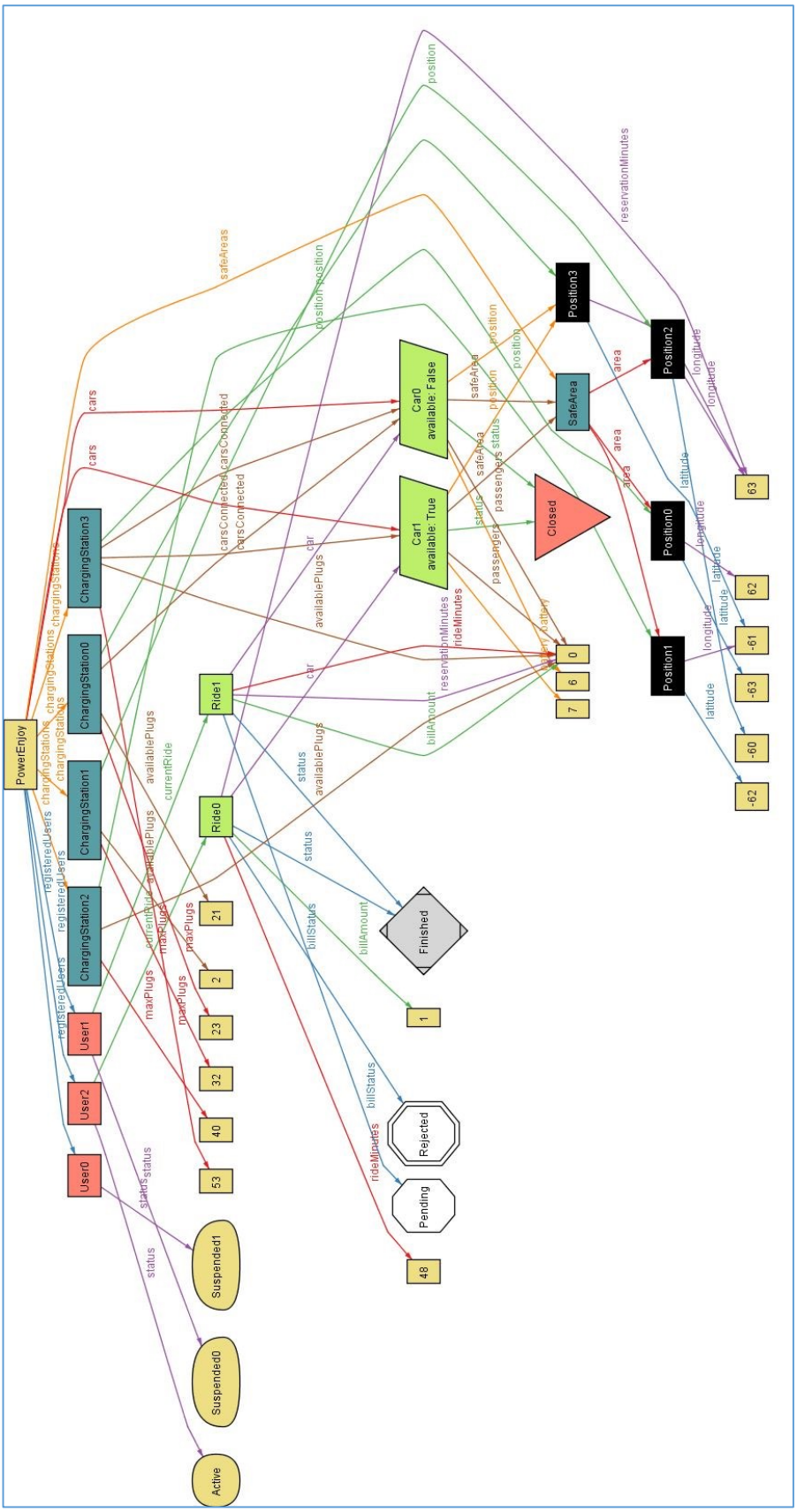

Figure 15: generated Alloy world

4.2 Tools

- Astah professional for all Diagrams
- MS Word for the whole Document
- Git for version control.
- Balsamiq for mockups.
- Alloy 4.2.

4.3 Effort spent

Marco Festa: 55 hours

4.4 Revisions

- RASD v1.0 published November 21, 2016.