Version 1.8

Software Engineering Project

Requirements Analysis and Specifications Document

POLITECNICO
MILANO 1863

Vianney Payelle - Rémi Rigal - Noëlie Ramuzat

Revision Notice

Versions	Date	Description	Modifications
V1.1	25/10/2016	Creation of the document	_
V1.2	2/11/2016	Implementation of Scenarios	Scenarios addition
V1.3	4/11/2016	Creation of: - Goals - World domain - First requirements - Use cases	- Goals, assumption, requirement and Use cases addition - Update of Scenarios
V1.4	6/11/2016	Implementation of: - UML diagrams - Scenarios diagram	Addition of : - Class diagram - State diagram - Sequence diagrams - Activity diagram - Scenarios diagram
V1.5	8/11/2016	Creation of the Introduction part	Update/ Rephrase of : - Goals - Requirements (delete of the breath-alcohol analyser, used to start the car) - Assumptions Addition of: - Constraints - Definitions - Purpose - Scope - Overview
V1.6	10/11/2016	Creation of the Specific Requirements	Addition of : - User interfaces requirements - Performance requirements - Use cases requirements Definitions updates : - Merge of the Driver and User definition - Only 3 stations types : safe, power grid or unsafe - Last ride defined as current ride
V1.7	11/11/2016	Creation of : - Alloy Model - Design constraints - Application mock-up	Addition of: - Design constraints part - Alloy model code - Smartphone application mock-up Update of Use cases
V1.8	12/12/2016	- Simulation of the Alloy world - Validation of the document	Addition of : - Alloy world - Human factors requirement - Software system attributes - Assumptions (one car per driver)

Table 1 : RASD versions

Table of Contents

List of tables

List of figures

1. Introduction

These days the mobile industry is in constant development and their technologies don't stop to improve. In particular, the mobile applications sector has become one of the more active in the market. Indeed, those applications are easy and fast to use, everywhere and at any time, appealing the customers.

The Enjoy Company provides a real-time car reservation service, where its clients can reserve electric cars near a given address and pick it within an hour. In order to satisfy their customers and ease the use of their services, Enjoy Company decided to build a mobile application: PowerEnjoy. This one creates a portable interface between the driver, his account and the available cars, enhancing the efficiency of the services.

1.1. Purpose

The Requirements Analysis and Specifications Document is the support piece of this project. The aim of this document is to presents the different requirements, with explanations about the application domain of the system and its development. It is the starting point for all the future evolution of the software, providing a basis for estimations (size and cost of the project), for testing process and for requirements changes.

1.2. Scope

The scope of the project PowerEnjoy, which is a service based on mobile application, is to manage, design, build, and implement a service aimed at facilitating public transportation.

The application provides to its target, the client, a way to research an electric car near a position, reserve it and pick it for a ride. At the end the application sends the ride's bill to the client.

The PowerEnjoy application needs the client to be registered in its database before he can reserve a car, for security and payment reasons (credential information, driver license, and identity card). After he logs in, the mobile application allows him to reserve a car around an address or his GPS position. Then it provides him details about his reservation on the main page such as a way to reach the car. The application also allows the client to cancel his reservation, unlock the reserved car when he his near it, and access his account details and modify it. After the ride the application locks the car and sends an email to the client with the bill of the ride. The mobile application can moreover give discounts and charges in function of the client's ride such as sharing the car or plug it in power grid station.

The PowerEnjoy application is built in order to ensure an easy and clear reservation service as well as an effective data collecting and saving. This refers the PowerEnjoy application simplifies the customer's uses, optimises the time to reserve a car and adjusts the price of the ride according to the driver.

1.3.Goals

Here are the main features that PowerEnjoy will grant its users.

- *G1*: Allow driver to register to the system by providing credential and payment details
- *G2*: Allow driver to log into the system with provided password
- *G3*: Driver can locate available cars within a certain distance around him or an address
- *G4*: Driver can reserve a car for up to 1hour before pick it up
- *G5*: A reserved car but not picked up within one hour generate 1€ fee for the driver
- *G6*: A driver close enough to a car reserved by him must be able to open it
- *G7*: The system start charging once engine ignite
- *G8*: The set of safe area is predefined by the management system
- *G9*: The system stop charging once he leaves the car parked in a safe area
- *G10*: 10% discount on last ride if the car detect at least two passengers
- *G11*: The system apply 20% discount on the last ride if the car is left with more than 50% battery (over full battery)
- *G12*: If the driver park the car in a special parking area, where the car can be charged, and take care to plug the car. The system apply 30% discount on the last ride.
- *G13*: If the car is left with less than 20% battery or at more than 3km from the nearest power grid station, the system charges 30% more on the last ride.
- *G14*: The driver can enable the saving money option and so by giving his final destination to the system, this one is able give him a station as destination (based on distribution of cars algorithm and available plugs in the station as well as the final destination of the driver) and so if the driver leave the car at this place he will get a special discount.

1.4.Definitions, acronyms, and abbreviations

Name	Definition
RASD	Requirements Analysis and Specifications Document
System	Element which regroup each software pieces of the project
Management system	Part of the system which take care of drivers an cars
Application	Program providing an interface between the system and the driver through a smartphone
Driver	Any people interacting with the system through the car or the application
Car	Any electric car involved in the project through the system
Car status	Attribute of a car to define if it is available or not (boolean)
Driver status	Attribute of a driver that define if he reserved a car, is driving one or forbidden to reserve a car (and for how long)
Safe area	A zone defined by a set of position with special characteristics
Power grid station	A place where an electric car can be charged
Last ride	The current ride of the driver

Table 2 : Glossary of the RASD

1.4. Reference Documents

Name	**Publication Date**	**Authors**	**Contents**
Assignment AA 2016-2017 Software Engineering 2	14/10/2016	Elisabetta di Nitto	Project goal, schedule and rules
IEEE standard on requirement engineering	12/01/2011	Institute of Electrical and Electronics Engineers	Systems and software engineering, life cycle processes, requirements engineering

Table 3 : Reference Documents used in the RASD

1.5. Used Tools

Name	Use
Github & SourceTree	Control the document versions
Modelio	Create the UML models
Alloy Analizer 4.2	Prove the consistency of the model
Adobe Acrobat Reader DC	Create the RASD PDF
Android SDK	Create the smartphone app mock-up
Adobe Illustrator	Create the scenarios diagram

Table 4 : Description of the tools used to create the RASD

1.6. Overview

In the following parts of the document are describing the product with its main features, then the specific requirements, the alloy modelling and finally the future development.

Description of the product

This section introduced the perspective of the product with the class and state diagrams. It presents also the application's major requirements, the characteristics of its users, its constraints and assumptions.

Specific Requirements

In this second part, the external, functional and performance requirements are provided, with the scenarios, use cases and system attributes definitions.

Alloy Model

This section provides the implementation of the application's Alloy model and presents the world generated.

Future development

In this part is given the possible future improvement of the system.

Finally the hours of work repartition follows this section.

2. Overall Description

2.1. Product perspective

2.1.1. Class diagram

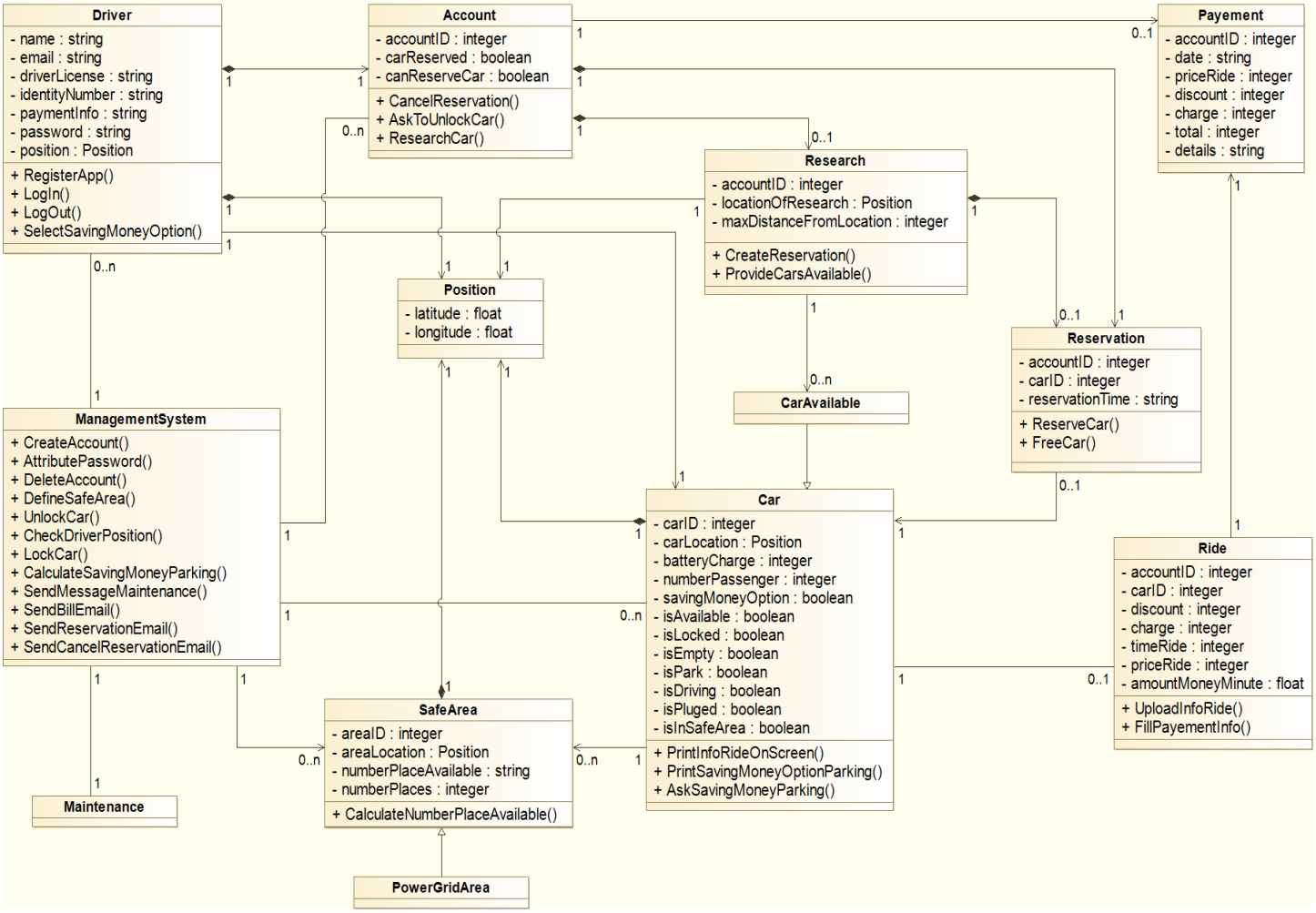

Figure 1 : Overview diagram of the System

2.1.2. State diagrams

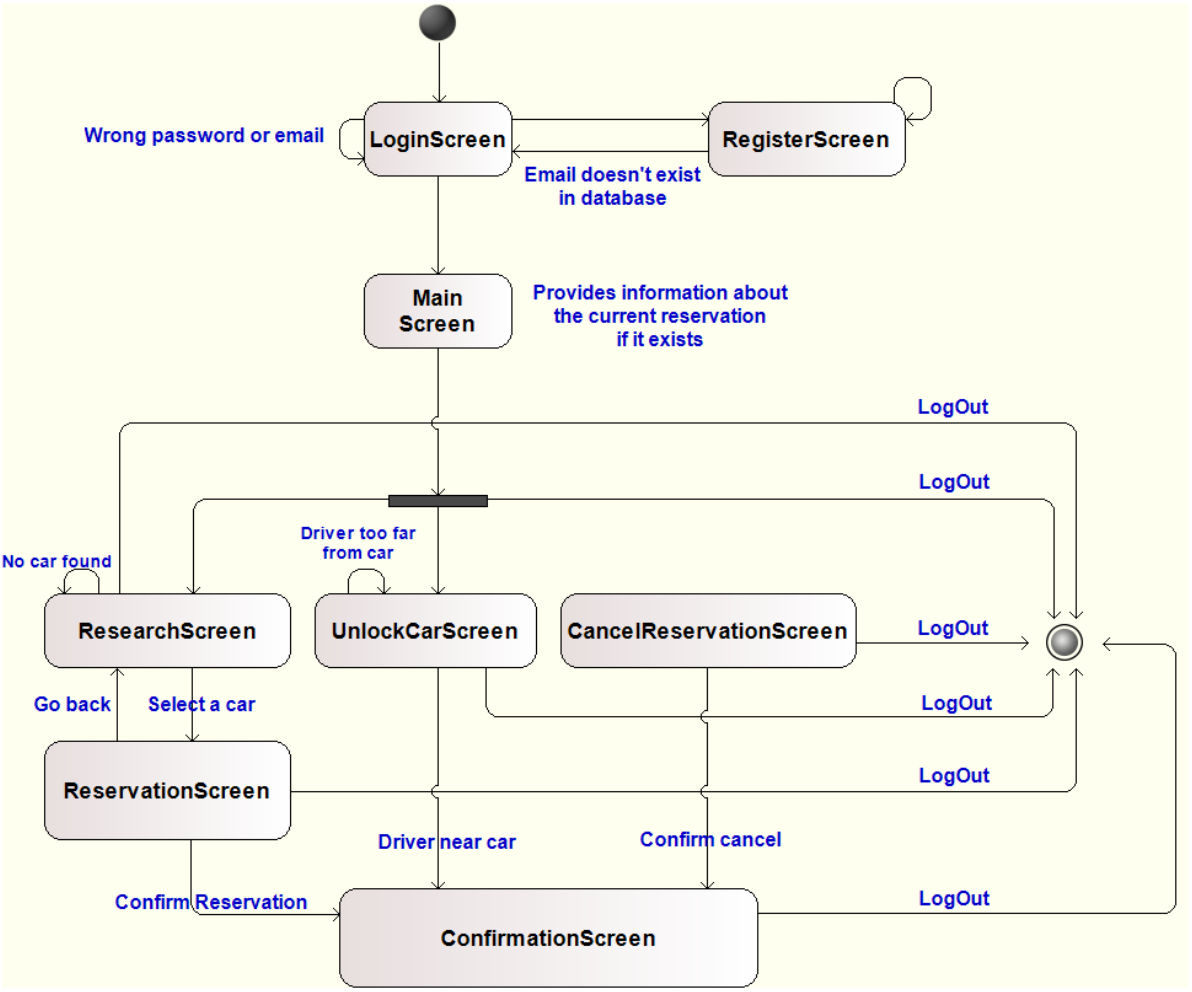

Figure 2 : State Machine Diagram of the Application

2.2. Product functions

Requirements

G1: Allow driver to register the system by providing credential and payment details

- Store provided details
- Link payment credential to payment system
- Ensure confidentiality and security
- Return password to log into the system
- Check already existing driver

G2: Allow driver to log into the system with provided password

- Check credential
- Allow access if the details are right

G3: Driver can locate available cars within a certain distance around him or an address

- Driver can input address and/or range
- The management system knows if a car is available or not
- The system manage a database of the cars of the society
- The management system can compare cars locations with the provided address or the driver's locations
- The application can show those cars and their locations

G4: Driver can reserve a car for up to 1 hour before pick it up

- Driver cannot reserve the car for more than one hour later than the current time.
- Driver must pick a car among the available ones
- The system must change the status of the car from available to unavailable

G5: A reserved car but not picked up within one hour generate 1€ fee for the driver and forbidden him to reserve another one within 3 hours

- The system charges the driver who reserved the car for 1€
- The system changes the status of the car from unavailable to available
- If the driver cancelled the reservation before it expires, the system update his status so he can't book a car for the next 2 hours otherwise he can't book a car for the next 3 hours

G6: Driver can cancel his reservation within the hour after he reserves it.

- The system charges the driver who reserved the car for 1€
- The system changes the status of the driver from can reserve a car to cannot reserve a car for 2 hours
- The system changes the status of the car from unavailable to available

G7: A driver close enough to a car reserved by him must be able to open it

- The system can detect when a driver is close to the car that I reserved
- A driver close to his reserved car can open it

G8: The system starts charging once engine ignite

- The system can detect that the driver ignited the engine
- The charge is linked to the driver through the system, kept in memory, and grows based on rate
- The charging rate and the current amount of charge is displayed with an interface in the car, for the driver

G9: The set of safe area is predefined by the management system

- The system knows the position of safe area and can link them to the position of the vehicle (same positioning system)

G10: The system stop charging once he leaves the car parked in a safe area

- The system can detect if the vehicle is parked (engine stopped)
- The system stops charging the driver if the vehicle is parked in safe area

G11: 10% discount on last ride if the car detects at least two passengers

- The car can detect the amount of passengers in the car
- The system applies a 10% discount on the last ride of the driver if the car detects more than 2 passengers

G12: The system apply 20% discount on the last ride if the car is left with more than 50% battery (over full battery)

- The system can read the battery's level of the car
- The system applies a 20% discount on the last ride of the driver if the battery's level is higher than 50% when the car is left.

G13: If the driver parks the car in a power grid station, where the car can be charged, and takes care to plug the car. The system applies 30% discount on the last ride.

- The system knows the position of special parking area and can link them to the position of the vehicle (same positioning system)
- The system can detect if the driver park the car in a special parking area
- The system can detect if the driver plugged the car and so apply a 30% discount on the last ride of the driver.

G14: If the car is left with less than 20% battery or at more than 3km from the nearest power grid station, the system charges 30% more on the last ride.

- The system knows the position of the power grid stations
- If the car is left farther than 3km from a power grid station or with less than 20% battery, the system charges 30% more on the last ride of the driver.

G15: The driver can enable the saving money option and so by giving his final destination to the system, this one is able to give him a station as destination and if the driver leaves the car and plugs it at this place he will get a special discount.

- The driver can activate a saving money option and enters his final destination
- The system knows the number of available plugs per station
- The system has an algorithm to find the best station with available plugs in order to be as close as possible from the final destination of the driver and optimize an uniform distribution of the cars among the city

2.3. User characteristics

The target users are clients who would use the PowerEnjoy application quickly in town, to reach specific places without using common transports or taxis. In the document the user is called the 'Driver'.

2.4. Constraints

- Ensure safety of driver details and transactions
- The system is compliant with local laws about drivers' details
- The car position can be known at anytime, anywhere
- The system can work with most of online mean of payment

2.5. Assumptions and Dependencies

Domain assumption

- We assume that the amount of safe areas and power grid stations is enough to cover efficiently the entire city in which the system evolves
- We know the position of the car at anytime, anywhere
- The car can count the number of passengers
- We can communicate with the car at anytime, anywhere
- The driver has a driver license, piece of identity and mean of payment
- The driver has a smartphone with localisation system
- The driver is legally responsible for his actions
- There is an interface in the car to communicate with the driver
- The society has a maintenance and insurance service for cars
- A driver who unlocks a car will ignite the engine
- You can only use the smartphone application to access the system

Explicit assumption

- A user can only drive one car at a time
- A user cannot drive a car that is charging
- Two cars cannot have the same position
- We can trust the details provided by the driver

Text assumption

- We assume that the amount of safe areas and power grid stations is enough to cover efficiently the entire city in which the system evolves
- We know the position of safe areas, special parking areas and power grid stations
- A Driver cannot book a car with less than 20% battery
- A damaged car is manually set to unavailable by the maintenance service (a driver cannot book a damaged car)

3. Specific Requirements

3.1. External Interface Requirements

3.1.1. User Interfaces

All drivers will first be interacting with the system through the smartphone application. Then the driver will interact with the in-board screen of the car.

The smartphone application shall be composed of 5 different screens which are:

- *Login/Register screen*: a simple form the driver fills to log into the application or register into the system
- *Main menu*: this screen displays the current reservation of the logged driver if he has one, a menu drawer on the left provides buttons to log out or to reach the following features:
  - Car finder screen
  - Car unlocker
  - Reservation canceller
  - Account screen
  - Credentials screen

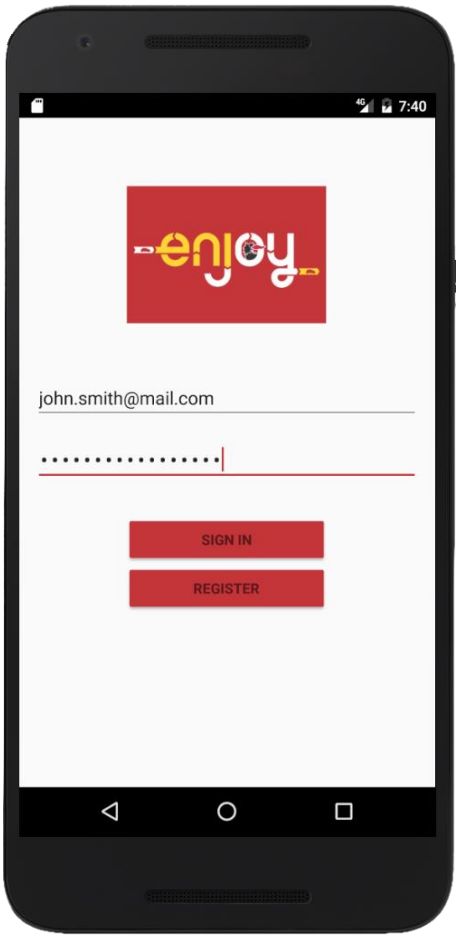

Figure 3 : Login/Register Screen

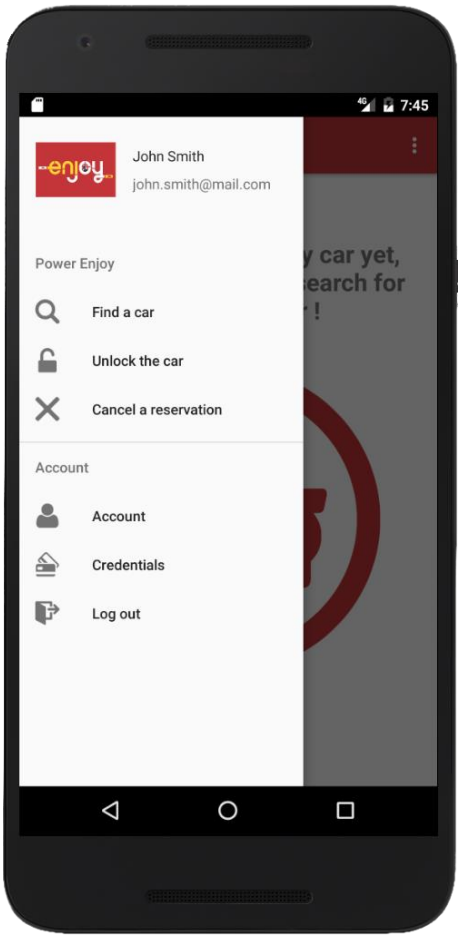

Figure 4 : Main Menu

- *Car finder screen*: asks the driver for an address (can use his current location) and gives him details about all cars around, he can then reserve one of them. This screen is not available when the driver has a running reservation.
- *Car unlocker*: provides a single button that can unlock the reserved car if the driver is near its location. This screen is available only when the driver has a running reservation.
- *Reservation canceller*: provides a button that can cancel the current reservation. This screen is available only when the driver has a running reservation.
- *Account screen*: the driver can consult the settings of his account and his reservation history
- *Credentials screen*: the driver can see his credentials details and modify them if needed

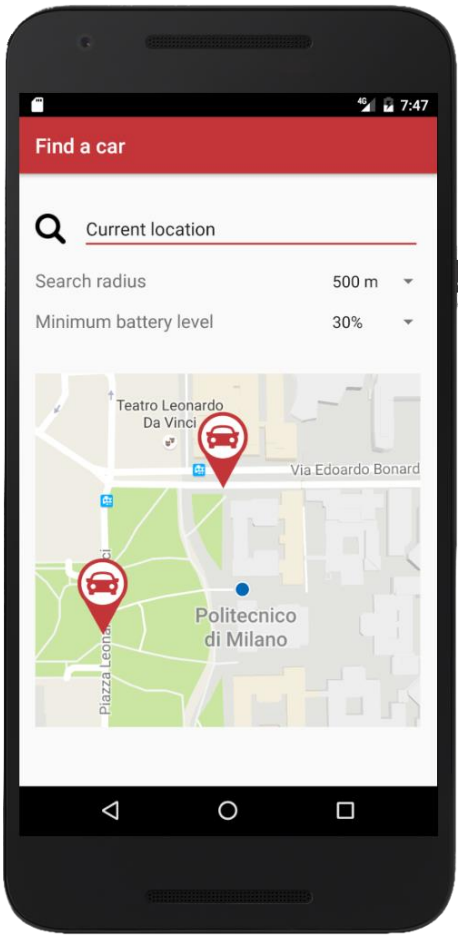

Figure 5 : Finder Screen

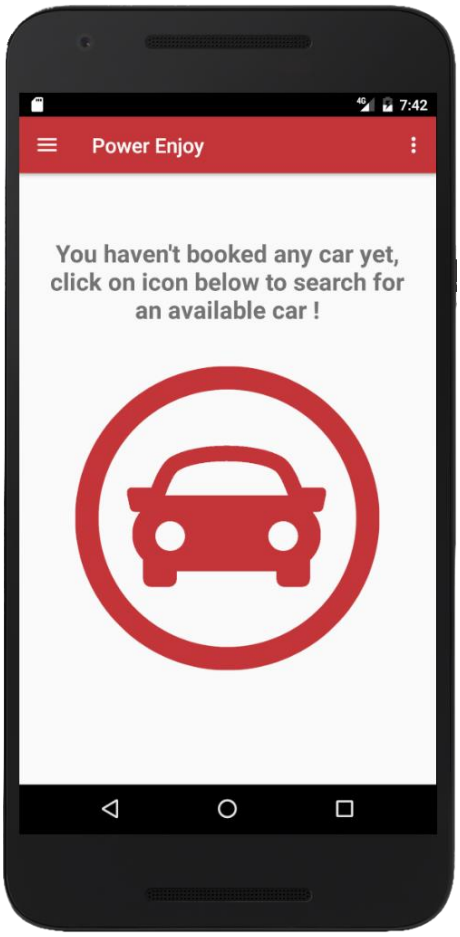

Figure 6 : Main Screen

The car screen is a complete navigation system in which the driver can choose to activate the "money saving" option and a parking station will be chosen according to its current destination.

3.1.2. Hardware Interfaces

The smartphone application shall be compatible with all devices running on iOS (from iOS 6), Android (from version 4.1) and Windows Phone in order to have a large target audience.

3.1.3. Software Interfaces

The smartphone application will use the Google Maps API, compatible with all the operating systems above, to provide a map of the available cars around a given position to the driver.

3.2. Functional requirements

3.2.1. Scenarios identifying

3.2.1.1. Scenarios Diagram

Figure 7 : Scenarios diagram

3.2.1.2. First Scenario

- John wants to visits the Duomo with Claire and Marie. He has heard about the PowerEnjoy app by a friend, so he wants to try and downloads it on his mobile.
- He registers on the app, giving his name, his email, his payment details, his driver license and a copy of his identity card.
- He receives a confirmation of his registration and his driver password by email.
- He logs in the app, selects the reservation mode and chooses 1.5km as maximal distance from his position where he wants to find a car.
- PowerEnjoy provides him a map with the stations around him, according to the distance, with the indication of the available cars.
- He selects a car in the nearest station, at 500m from him.
- He validates at 2pm. He receives an email, confirming that the car is locked for John until 3pm.
- At 2.05pm John walks to the car and selects the unlock mode of the app, and then the car unlocks.
- He gets in the car and sees on the navigation screen the price of the ride: 0€, the actual discount: 0€ and the amount of money/min: 0.8€.
- Then he ticks the money saving option and put his destination. On the car interface, the app indicates him the nearest station from the Duomo where he can have a discount.
- He drives and takes Claire and Marie on the road, the system detects them and the screen prints a discount of 10%.
- Reaching the station after 20 minutes, John parks the car and plugs it. The battery is 40% empty. The price is 16€ and the discount 60% on the screen.
- John, Claire and Marie leave the car, the car locks and is available again.
- John receives an email with the amount of money he will pay, 7€ and the discount he has: 60%.

3.2.1.3. Second Scenario

- Katia is already registered on the PowerEnjoy app. She uses the reservation mode to find a car near her flat to go to the cinema; she enters her address on the app.
- She enters 700m as the maximal distance from her flat where she wants to find a car.
- PowerEnjoy provides her a map with the stations around her according to the distance, with the indication of the available cars.
- She selects a car in the tram station and she validates at 4p.m.
- She receives an email, confirming that the car is locked for her until 5pm.
- But finally Katia invites Karl to the cinema and he drives them. So Katia was not at the station after 6p.m. At this moment the car is available again at the station.
- Katia receives an email with the amount of money she has to pay: 1€, the app informs her that she cannot select a car again in the next 3h.

3.2.1.4. Third Scenario

- Steve selects the reservation mode of the app and enters 1km as maximal distance to find a car.
- The app provides him a map with the station around him according to the distance, with the indication of the available cars.
- He selects a car with 50% of battery empty in the 3rd station and validates at 5p.m.
- He receives an email, confirming that the car is locked for him until 6pm.
- At 5.40pm, Steve walks to the station, stops near the car and selects the unlock mode of the app. The car is unlocked.
- He gets in the car and sees on the screen the price of the ride: 0€, the actual discount: 0€ and the amount of money/min: 0.7€.
- He drives 1h until reach his grandparent's small town. There is only one safe station given by the app, without power grid, the nearest is at 4km from the station.
- He parks there; the battery is 81% empty. The screen prints a charge of 60%.
- He leaves the car, which locks and is available again.
- Steve receives a message with the amount of money he will pay: 67€ and the charge he has on his last ride: 60%.

3.2.1.5. Fourth Scenario

- Amelie selects the reservation mode of the app, and enters 2km as the maximal distance from her she wants to find a car.
- The app provides her a map with the station around her, according to the distance, with the indication of the available cars.
- She selects a car in the nearest parking and validates at 7p.m.
- She receives an email, confirming that the car is locked for her until 8p.m.
- But her mother decides at the last moment to come at her house.
- So Amelie cancels her booking on the app at 7.30p.m.
- She received an email confirming it, she pays 1€ and the app informs her that she cannot select a car again in the next 2h.
- The car is available again immediately on the app.

3.2.1.6. Fifth Scenario

- Anna is already registered on the PowerEnjoy app. She selects the reservation mode of the app, and enters 1km as the maximal distance from her position she wants to find a car.
- The app provides her a map with the station around her, according to the distance, with the indication of the available cars.
- She selects a car in the nearest parking and validates at 6a.m.
- She receives an email, confirming that the car is locked for her until 7a.m.
- Then her sister wants to take a car too, she is used to take the account of Anna on PowerEnjoy. She tried to use the reservation mode on her phone but she can't select it.
- She can only access to the cancel mode.
- She asks to Anna if she reserved a car, and if she can drive her to the school. Anna agrees and they live their house.

3.2.1.7. Sixth Scenario

- Simone has reserved a car with the *PowerEnjoy* app. He has picked it in time and has driven until his girlfriend's house. The price on the screen is 22€.
- There is no safe area around the house and Simone decides to parks in the street.
- He leaves the car, which locks and is available again.
- Simone receives a message with the amount of money he will pay: 44€.

3.2.2. Use cases diagram

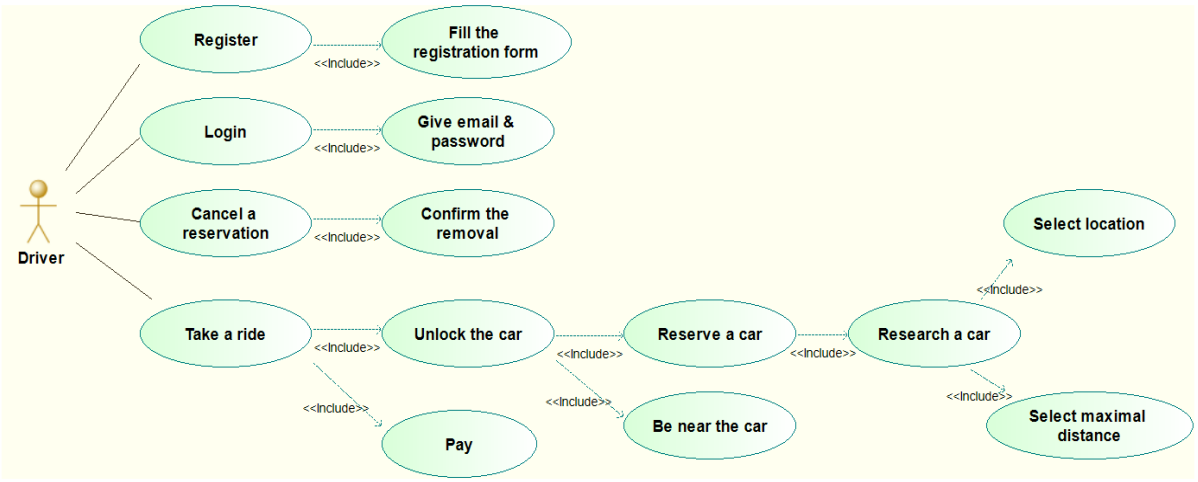

Figure 8 : Use Cases diagram

3.2.3. Use cases

Use case 1

Use case name: AppRegistration

Participating actors:
- Driver

Entry condition:
- The Driver selects the "Registration" function of the PowerEnjoy app

Flow of Events:
- The Driver fills in the registration form: The Driver gives his name, his email, his payment details, his driver license and an identity document.
- The Driver submits his details.
- The System checks if the email is not in the database. Then it creates a new Driver in the database, with the details given and a new password.
- The System sends a registration confirmation to the Driver, with his password, by email.

Exit condition:
- The Driver received his password by email.

Exceptions:
- If the connection is lost, the app page is reloaded and the Driver has to fill it again
- If the email is already in the database, the app page is reloaded with an error message

Functional requirements
- The application provides a register interface
- The details are safely transmitted
- The database can store properly Driver's details
- Lost connection management
- Check if the email provided is already in the database
- Return password by email
- Link payment details to payment system

Use case 2

Use cases name: DriverLogIn

Participating actors:
- Driver

Entry condition:
- The Driver selects the "Login" function of the app

Flow of Events:
- The Driver fills in the email and password entries and submits.
- The System checks the email and password in the database and logs the Driver.

Exit condition:
- The Driver logs in the app.

Exceptions:
- If the connection is lost the app page is reloaded and the Driver has to fill it again.
- If the email is not in the database, the app page is reloaded with an error message.
- If the password does not correspond to the email in the database, the app page is reloaded with an error message.
- If the Driver selects the "forget password" function, the Driver has to fill in the email entries and submits.
  The System checks the email in the database and sends the associated password by email to the Driver or prints an error.

Functional requirements
- The application provides a login interface
- Check that the details provided match the database's details
- Lost connection management
- The details are safely transmitted
- The system must provide a safe way for the Driver to get back his password

Use case 3

Use cases name: CarReservation

Participating actors:
- Driver

Entry condition:
- The Driver logs in and selects the "Reserve car" function of the app

Flow of Events:
- The Driver enters his location, by giving an adress or selecting his GPS position.
- The Driver chooses a maximal distance from the selected location to find a car.
- The System finds the station around the location, according to the distance and the available cars. It sends them on a map to the Driver's mobile.
- The Driver selects a car in a proposed station and validates the reservation.
- The System registers the reservation in the database, by locking the car for the Driver during 1hour: the car is unavailable on the PowerEnjoy app during 1hour.
- The System sends an email of confirmation to the Driver.

Termination condition:
- The Driver received the reservation confirmation by email.

Exceptions:
- If the connection is lost the app page is reloaded and the Driver has to fill it again.
- If the Driver had already reserved a car or is forbidden to reserve one, the "Reserve car" function is not available.

Functional requirements
- Lost connection management
- A driver can only have one car linked to him, either driving or reserved
- A driver cannot reserve a car if his status prevents him
- As soon as a car is reserved, her status move from available to unavailable
- The system knows the car and the driver status
- The system knows and can compare the position of the user with cars
- An algorithm can return the available cars within a given distance from the position given by the user, on a map

Use case 4

Use cases name: CancelReservation

Participating actors:
- Driver

Entry condition:
- The Driver logs in and selects the "Cancel Reservation" function of the app before the end of the reservation.

Flow of Events:
- The System puts the car as available in the database and sends an email confirming the cancel & giving the bell of the Driver reservation: 1€.
- The Driver is notified by the System that it is impossible for him to select a car again in the next 2h.
- The System registers the Driver as forbidden to reserve a car for the next 2hours in the database.

Exit condition:
- The Driver received the cancel confirmation by email and the notification.

Exceptions:
- If the connection is lost the app page is reloaded and the Driver has to fill it again.

Functional requirements
- Lost connexion management
- A cancelled reserving prevent the driver to reserve a car for 2 hours and set the car to available
- An email is send to the driver to confirm the cancellation
- The driver is charged for 1€

Use case 5

Use case name: CarUnlocking

Participating actors:
- Driver

Entry condition:
- The Driver arrives near the car, logs the app and selects the "Unlock car" function

Flow of Events:
- The System accesses to the location of the Driver by using the GPS of his mobile.
- The System compares this location with the car's one and unlock it.
- The Driver gets in the car and sees on the navigation screen the price of the ride, the amount of money/min and the discount or charge of the ride.

Exit condition:
- The Driver starts the car.

Exceptions:
- If the connection is lost the app page is reloaded and the Driver has to fill it again.
- If the Driver is too far from the car, The System doesn't unlock the car and sends an error message to the Driver.
- If the Driver is not detected at the station one hour after the reservation or is detected after this time, then the System registers the car as available again in the database with its new details.
  The System sends an email giving the bell of the Driver reservation: 1€, and notified him that he can't select a car again in the next 3h.
  The System registers the Driver as forbidden to reserve a car for the next 3hours in the database.

Functional requirements
- When a user doesn’t pick his reserved car within 1 hour, the system charge him for 1€, set the car to available, and prevent the user to reserve a car for next 3 hours
- Lost connexion management
- The system can detect if the user is close to the car when he asks to unlock the car
- There is an unlock function on the interface of the application

Use case 6

Use cases name: RideAndCarLock

Participating actors:
- Driver
- Maintenance

Entry condition:
- The Driver starts the car.

Flow of Events:
- The System starts the charging of the money on the navigation screen.
- The Driver takes a ride until the destination.
- The Driver parks in a safe area given by the navigation screen, stops and exits the car.
- The System is notified that the car is empty and parked in a safe area with the GPS of the car. It stops charging the price, locks the car and registers it as available at the new station in database with its new details.
- The System sends an email to the Driver with the price of the ride and the discount or charge he will have on his last ride.

Exit condition:
- The Driver received the bell of the ride.

Exceptions:
- If the Driver doesn't park in a safe area, the System stops charging the price, locks the car and sends an email to the Maintenance with the location of the car. Then it doubles the ride price of the Driver.
- If the Driver selects the saving money option on the car's navigation screen and gives his destination, then the System provides him, on the screen, a safe area where he can have a discount.
- If the car detects more than 2 passengers, it informs the System which increases the discount of the last ride by 10% on the navigation screen & in the database.
- If the battery level is > 50%, the car informs the System which increases the discount of the last ride by 20% on the navigation screen & in the database.
- If the Driver plugs the car on a power grid within 5minutes after leaving the car, the System is notified and increases the discount of the last ride by 30% on the navigation screen & in the database.
- If the battery level is < 80%, the car informs the System which increases the charge of the last ride by 30% on the navigation screen & in the database.
- If the Driver parks further than 3km from the nearest power grid station the System detects it and increases the charge of the last ride by 30% on the navigation screen & in the database.

Functional requirements
- An algorithm can compute the total price of the ride based on the standard price of the ride and extra charge/discount based on details provided by the car: location, battery's level, passengers etc. …
- The final price is displayed in the car to inform the Driver
- If the Driver plugs the car within 5 min after left it, he got an additional 30% discount and then the system use Driver's payment details to charge him and send him an email with the bill and payment confirmation
- Once the car is left, it is set to available

- If the car is left with less than 20% battery, in an unsafe area or farther than 3km from the nearest power grid station, the car is set to unavailable and the Maintenance service is called to move it to a power grid station and plug it
- Cars that have been moved by the Maintenance are set available if they are at least 30% of battery

3.2.4. Sequence diagrams

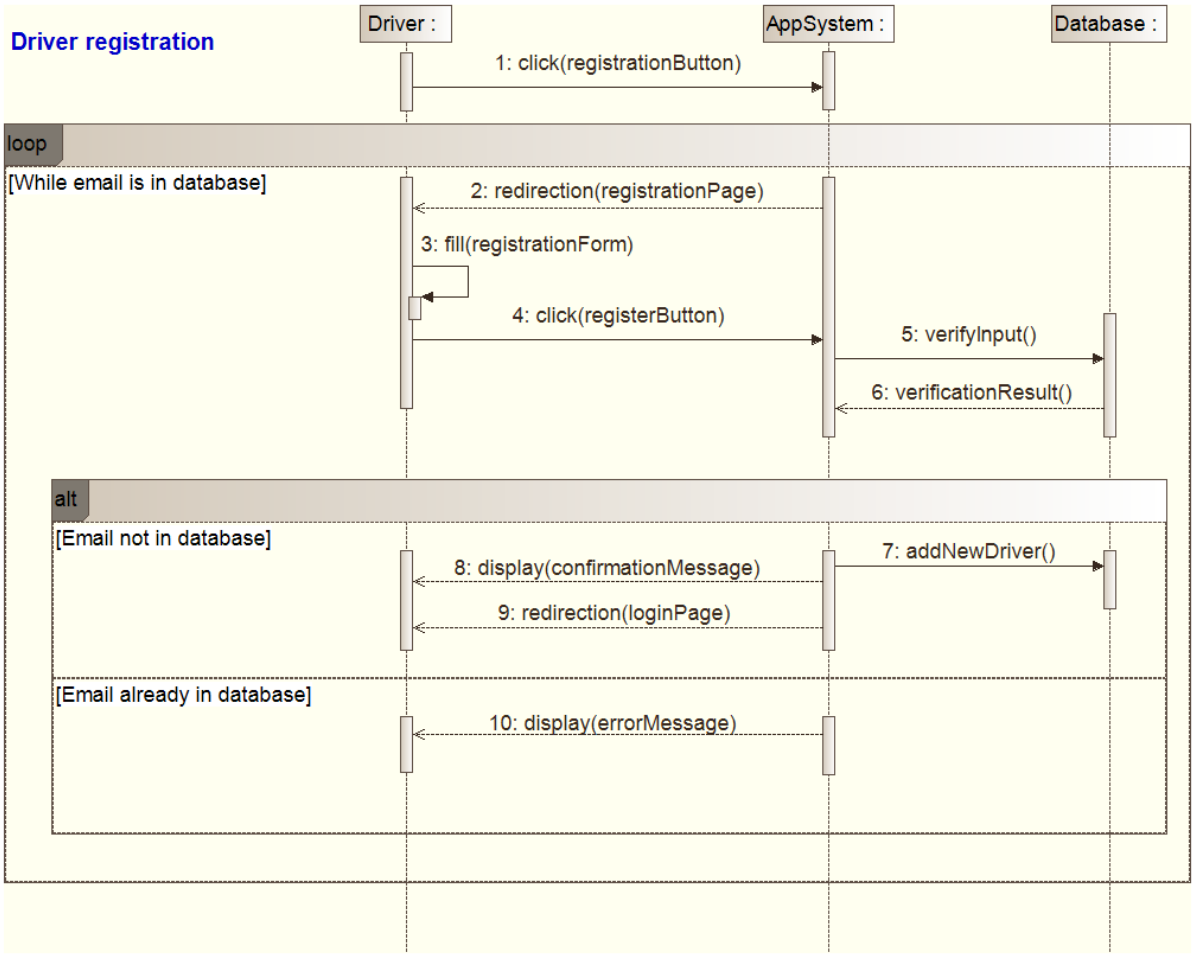

Figure 9 : Sequence diagram: Driver registration

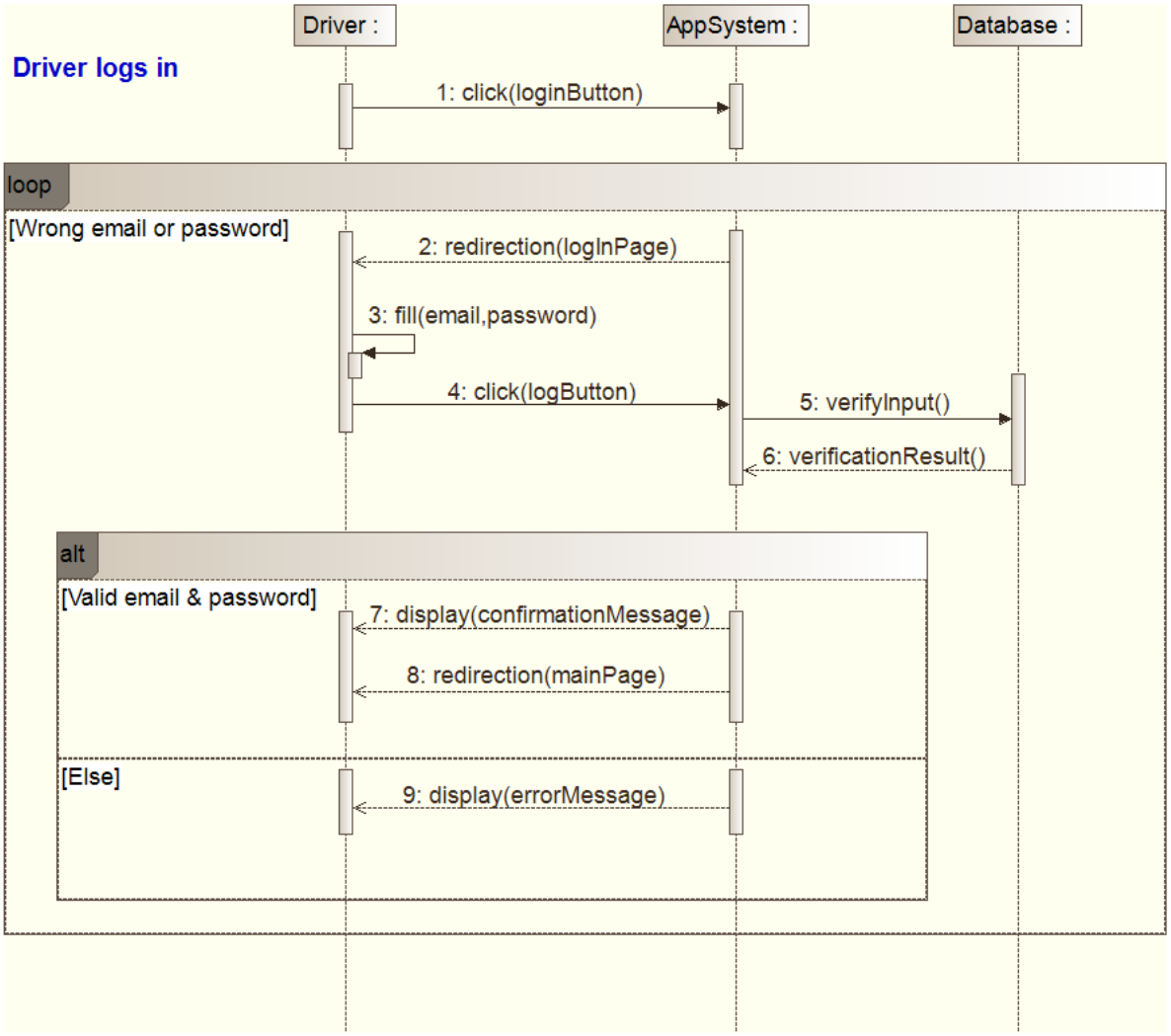

Figure 10 : Sequence diagram: Driver logs in

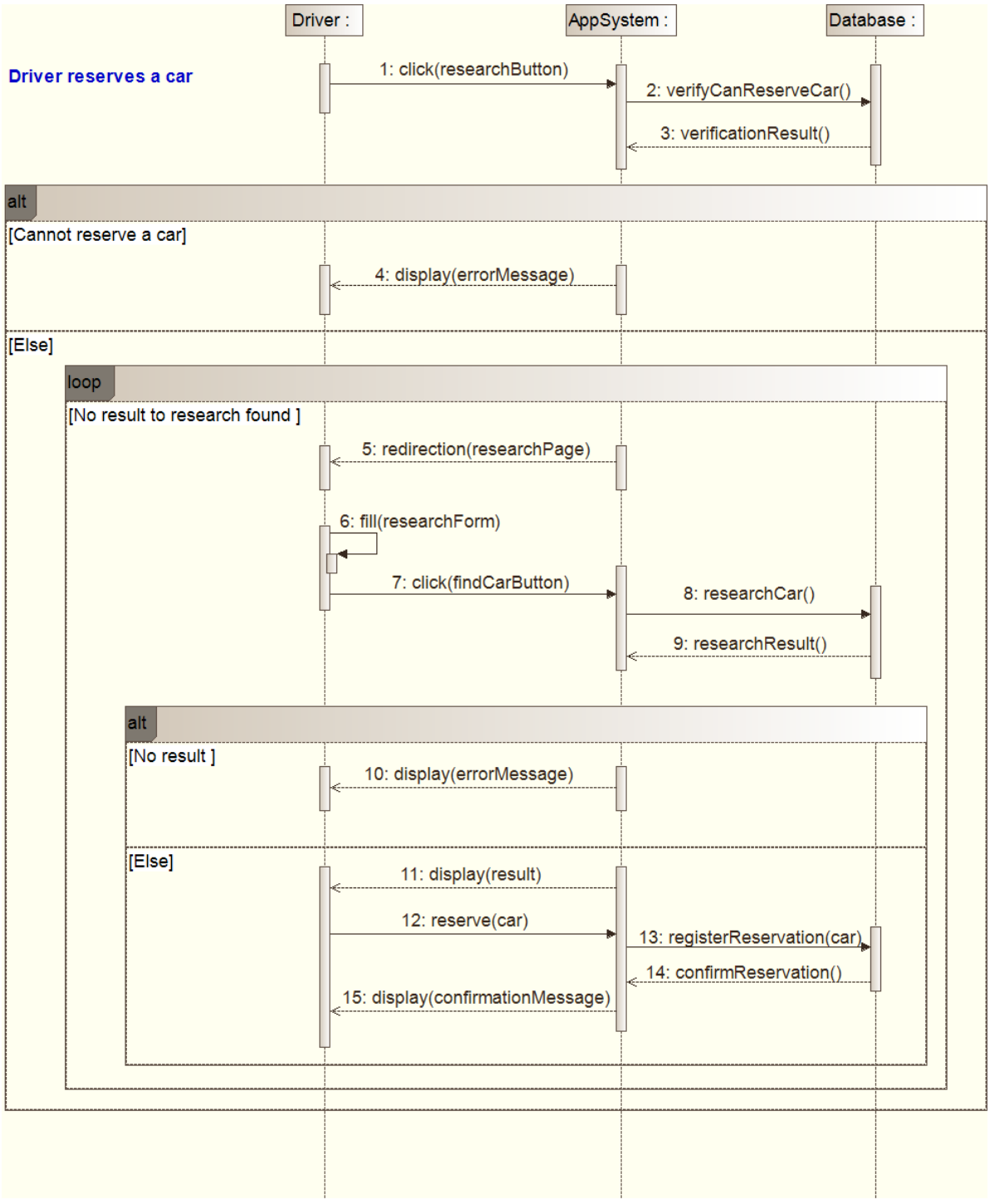

Figure 11 : Sequence diagram: Driver reserves a car

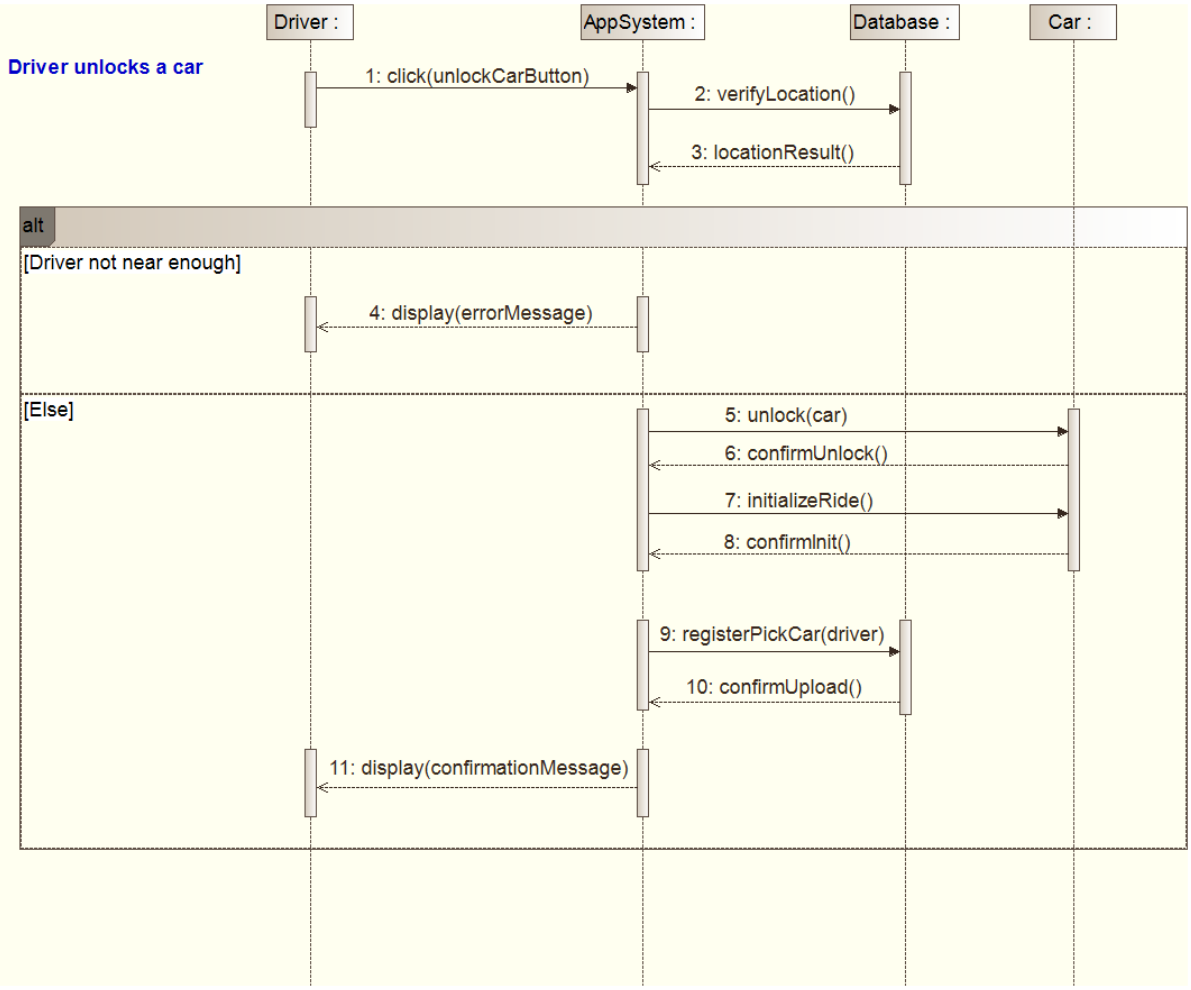

Figure 12 : Sequence diagram: Driver unlocks a car

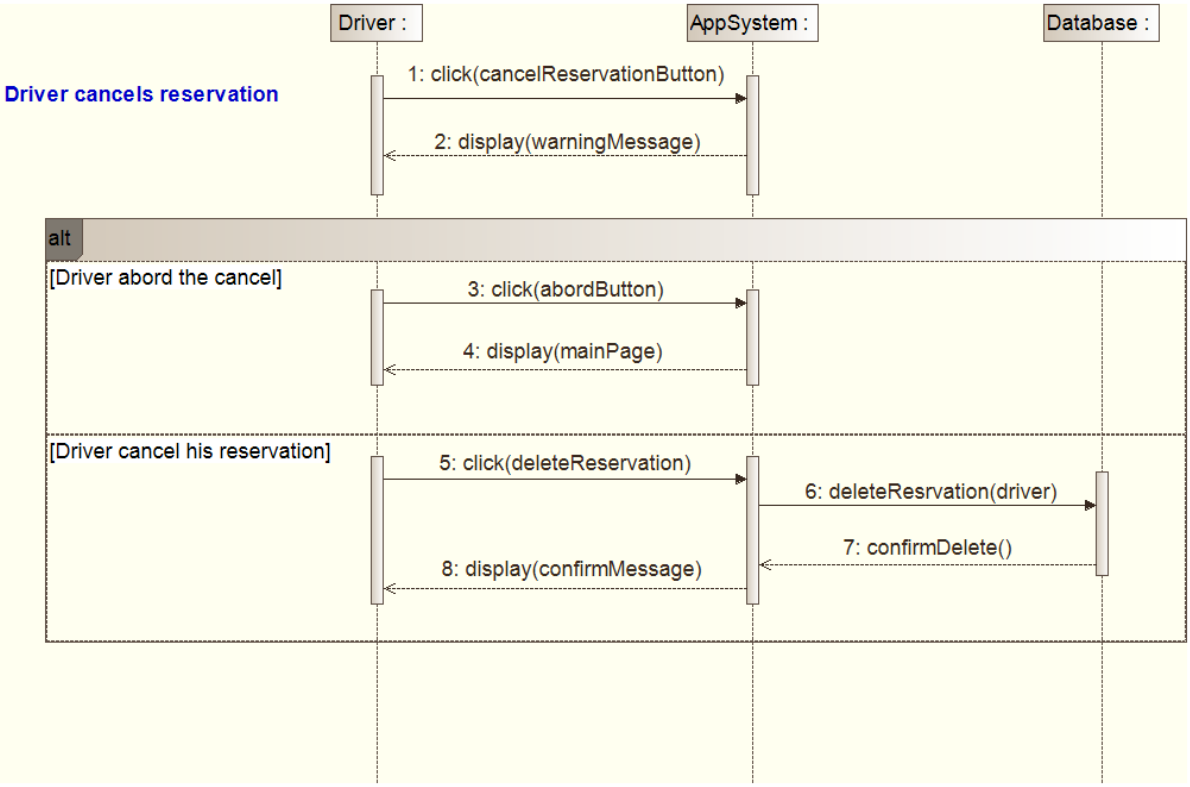

Figure 13 : Sequence diagram: Driver cancels his reservation

3.2.5. Activity diagrams

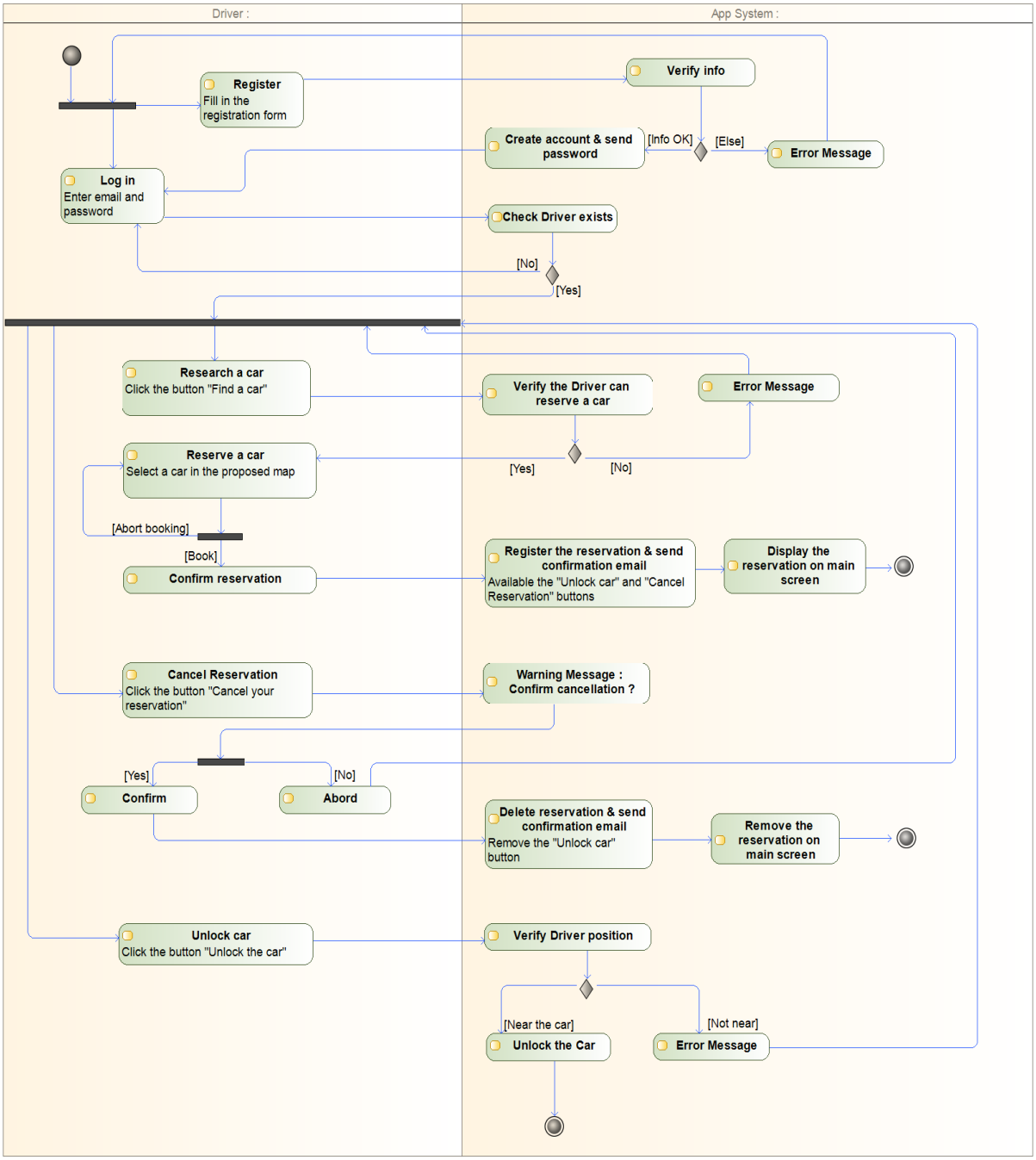

Figure 14 : Activity diagram of the system

3.3. Non-functional requirements

3.3.1 Performance requirements

The system shall be able to provide fast answers to the driver. The most critical point is the search for available car because the system will have to fetch and send details about all cars around the driver such as the precise position or the current charge of the battery.

At least 95% of these requests must be performed in less than 3 seconds. All others server tasks shall be processed in less than 1 second. The system must support 500 times the total number of provided car of simultaneous users.

3.3.2 Human factors requirements

Considering that every driver can make a mistake, the system shall allow every driver to update his account details at any moment.

3.4. Design Constraints

3.4.1. Standards compliance

Critical data such as payment details will be handled by the system, in order to ensure a maximum protection; every payment related changes applied to the database should be recorded in a trace file. For the same purpose, all the transactions between the server and the clients must be secured and traced.

3.4.2. Hardware limitations

PowerEnjoy doesn't have to meet any hardware limitations.

3.5. Software System Attributes

3.5.1. Reliability

Critical parts of the system: payment, car management and database must have a high reliability (in reference to high reliability of such like system on the market). Interaction with user over the application should be as reliable as most of such application on the market.

3.5.2. Availability

Car management system, payment system and global system should have a very high availability (99,999%). The system shall have a lower reliability for user management (99,9%).

3.5.3. Security

Follow the most critical norms in each domain given the country where it has to be deployed. The car opening should be at least as safe as the current remote opening system (like key proximity detection).

3.5.4. Maintainability

The development of the system is fully documented and we must have the possibility to improve algorithms.

3.5.5. Portability

Main smartphone stores: Google Play Store, Apple Store, Windows Store. And most of current smartphones on the market (including those released up to 3 years ago).

4. Alloy modelling

4.1. Model

In this part is presented the Alloy model obtains by describing the UML class diagram, the assumptions and the requirements of the application. Some attributes have been commented to make the world simulation clearer.

4.1.1 Signatures

```
// Standard types
sig Float{}
sig string{}
abstract sig Bool{}
one sig True extends Bool{}
one sig False extends Bool{}

// Company entity
one sig EnjoyCompany{
	stations: set SafeArea,
	cars: set Car,
}

// Current date for reservation
sig Date{}

// Car reservation
sig Reservation{
	driver: one Driver,
	car: one Car,
	date: one Date,
}

// Cancel car reservation
sig CancelReservation{
	reservation: one Reservation
}

// Car unlocked
sig Unlock{
	reservation: one Reservation
}

// Position for car, driver and parking area
sig Position{
	latitude: one Float,
	longitude: one Float
}
```

```
// The user of the system
sig Driver{
	//name: one string,
	//email: one string,
	//drivingLicense: one Int,
	//idCard : one Int,
	//credential: one Int,
	//password: one string,
	position: one Position,
	//rideInfo: lone Ride
}

// Cars provided by the system
sig Car{
	position: one Position,
	//batteryCharge: one Int,
	passengersNb: one Int,
	navigationScreen: one NavigationScreen,
	isLocked: one Bool,
	isParked: one Bool,
	isPlugged: one Bool,
	plug: lone Plug
}
{passengersNb < 5 and passengersNb > -1}
sig NavigationScreen{
	savingMoneyOption: one Bool,
	destination: lone Position
}

// Parking areas
sig SafeArea{
	position: one Position,
	places: one Int,
}
{places >0}
sig ChargingStation extends SafeArea{
	capacity: set Plug,
	chargingCars: set Car
}
{#chargingCars <= #capacity}
sig Plug{}
```

4.1.2. Facts

```
fact NoChargingCarInUse{
	no s: ChargingStation, r: Reservation | r.car.isLocked = False and
r.car not in s.chargingCars
}

fact NoCarsInSamePosition{
	no disj c1, c2: Car | c1.position = c2.position
}

fact NoSameReservation{
	no disj r1, r2: Reservation | r1.car = r2.car or r1.driver =
r2.driver
}

fact NoSameCancel{
	no disj c1, c2: CancelReservation | c1.reservation = c2.reservation
}

fact NoSameUnlock{
	no disj u1,u2: Unlock | u1.reservation = u2.reservation
}

fact NoSameScreen{
	no disj c1, c2: Car | c1.navigationScreen = c2.navigationScreen
}

fact NoSceenWithoutCar{
	#NavigationScreen = #Car
}

fact NoCarWithoutCompany{
	all c: Car | c in EnjoyCompany.cars
}

fact NoSafeAreaWithoutCompany{
	all s: SafeArea| s in EnjoyCompany.stations
}

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

fact CancelsReservation{
	all c: CancelReservation | c.reservation.car.isLocked = True and
c.reservation.car.isParked = True
}

fact NoPlugWithoutChargeStation{
	all p: Plug | p in ChargingStation.capacity
}

fact NoPassengerIfNoReservation{
	all c: Car | !(c in Unlock.reservation.car) implies c.passengersNb=0
}
```

```
fact LockImpliesPark{
      all c: Car | c.isLocked = True implies c.isParked = True
}

fact NoCancelReservationAndUnlock{
      no r: Reservation | r in CancelReservation.reservation and r in
Unlock.reservation
}

fact NoCarPlugInIfNoPlug{
      all c: Car | c.plug = none <=> c.isPlugged = False
}

fact NoSaveOptionIfLocked{
      all c: Car | c.isLocked = True implies
c.navigationScreen.savingMoneyOption = False and
c.navigationScreen.destination=none
}

fact UnlockCar{
      all u: Unlock | u.reservation.driver.position =
u.reservation.car.position
}

fact  NoTwoCarsOnSamePlug {
      no disj c1: Car, c2: Car | c1 != c2 and c1.plug = c2.plug and c1.plug
!= none
}

fact NoCarLockIfDriverUnlocked {
      no u:Unlock | u.reservation.car.isLocked = True
}

fact NoCarUnlockedWithoutUnlock {
      no c: Car | !(c in Unlock.reservation.car) and c.isLocked = False
}

fact NoCarPluggedIfNotParked {
      no c: Car | c.isPlugged = True and c.isParked = False
}
```

4.1.3. Asserts

```
assert NoCarUnlockIfDriverCancelReservation {
      no c:CancelReservation | c.reservation.car.isLocked = False
}
//check NoCarUnlockIfDriverCancelReservation

assert NoDriverOrCarWithMultipleReservations {
      no disj r1,r2: Reservation | r1.driver = r2.driver or r1.car = r2.car
}
//check NoDriverOrCarWithMultipleReservations

assert AllCarsAndParkingAreasAttachedToEnjoyCompany {
      all c:Car | c in EnjoyCompany.cars and
      all sa:SafeArea | sa in EnjoyCompany.stations
}
//check AllCarsAndParkingAreasAttachedToEnjoyCompany

assert NoReservationCancelledAndUnlocked {
      no r: Reservation | r in Unlock.reservation and r in
CancelReservation.reservation
}
//check NoReservationCancelledAndUnlocked

assert AllCarsLockedIfNoUnlock {
      no c: Car | !(c in Unlock.reservation.car) and c.isLocked = False
}
//check AllCarsLockedIfNoUnlock

assert NoSavingMoneyOptionIfCarLocked {
      no c: Car | c.isLocked = True and
c.navigationScreen.savingMoneyOption = True
}
//check NoSavingMoneyOptionIfCarLocked

assert PassengersOnlyAfterUnlock{
      no c: Car | c.passengersNb > 0 and
                        (c in CancelReservation.reservation.car or
                         !(c in Reservation.car) or
                         !(c in Unlock.reservation.car))
}
//check PassengersOnlyAfterUnlock
```

4.1.4. Predicate

```
pred show{
      #Driver = 2
      #SafeArea = 2
      #ChargingStation = 1
      #Reservation=2
      #Car= 2
      #Plug > 2
}
run show
```

4.2. Alloy result

Figure 15 : Alloy Result

4.3. World generated

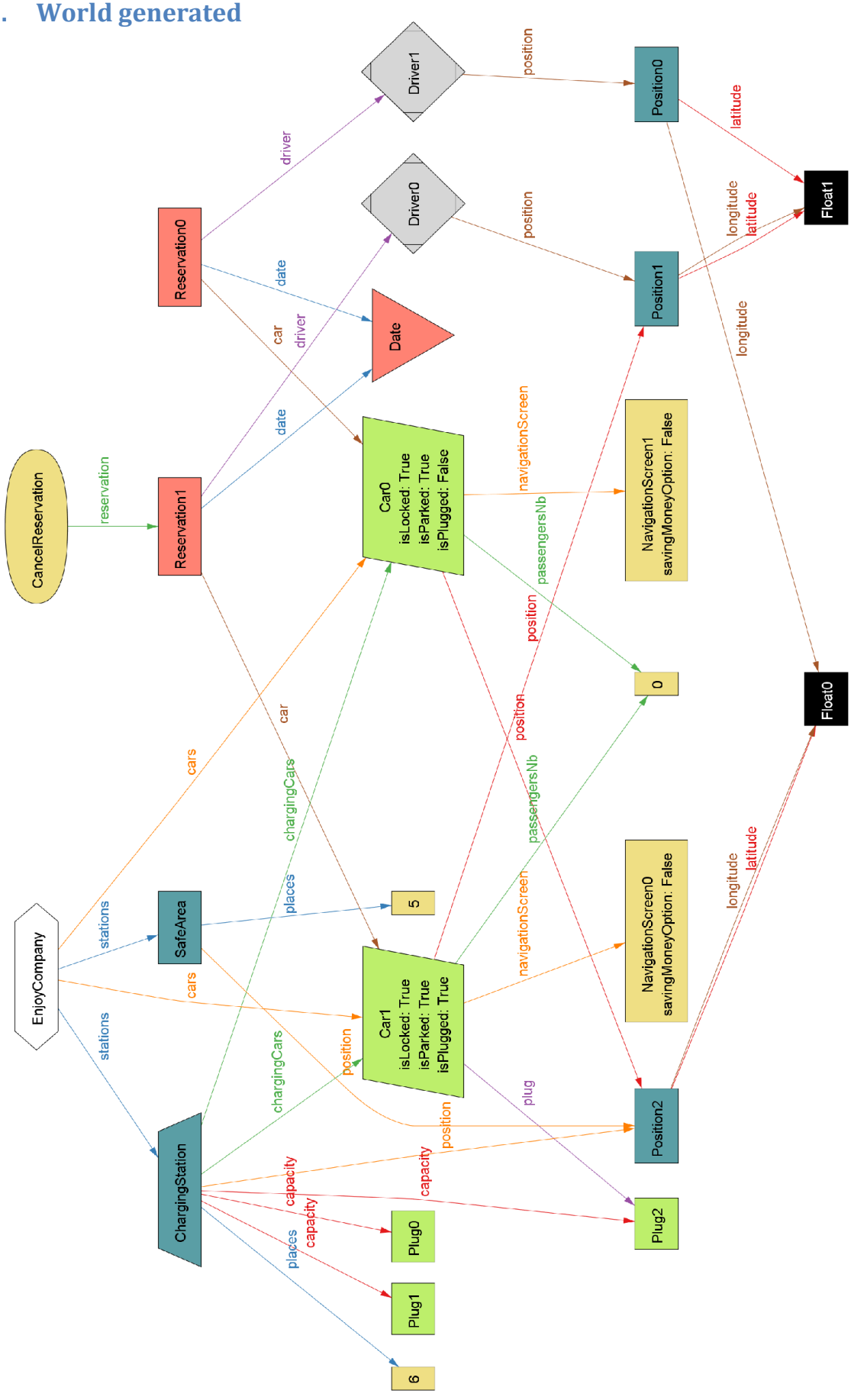

Figure 16 : World generated by the Alloy Model

5. Future development

- Include traffic management in saving money algorithm
- Improve user management availability and reliability
- Improve autonomy management in the user interface: Drivers should know if they will be able to reach their destination with the car that they want to use, and avoid more charges
- Analyse Driver's data and path to optimise car's distribution and management
- Improve booking system with algorithm based on planned destination and traffic: book a car that we know it should arrive soon for example
- Provide estimation on numbers of available cars within an area for a given time

6. Hours of work

6.1. Vianney Payelle

- 26/10 : 3h
- 29/10 : 2h
- 31/10 : 1h
- 01/11 : 1h
- 04/11 : 2h
- 06/11 : 2h
- 07/11 : 2h
- 09/11 : 1h
- 11/11 : 3h
- 12/11 : 2h

6.2. Rémi Rigal

- 26/10: 2h
- 27/10 : 1h
- 29/10 : 1h
- 02/11: 1h
- 04/11: 2h
- 06/11: 3h
- 10/11: 2h
- 11/11: 5h
- 12/11: 4h

6.3. Noëlie Ramuzat

- 23/10: 1h
- 24/10: 1h
- 25/10: 1h
- 26/10: 2h
- 02/11: 3h
- 03/11: 4h
- 04/11: 4h
- 05/11: 4h
- 06/11: 4h
- 07/11: 3h
- 11/11: 5h
- 12/11: 3h