RASD - Software Engineering 2

POLITECNICO
MILANO 1863

PowerEnjoy

Marini Alberto
862838
alberto2.marini@mail.polimi.it

Marrone Matteo
810840
matteo.marrone@mail.polimi.it

Sabatelli Antonella
875666
antonella.sabatelli@mail.polimi.it

November 13th, 2016

Politecnico di Milano

Contents

Chapter 1

1 Introduction

1.1 Purpose

This document contains the **RASD** (*Requirement Analysis and Specification Document*), aimed to describe a car sharing system named PowerEnjoy which only employs electric cars, characterized by an AI developed by the car producer. No other previous software infrastructure exists. Conforming to the *IEEE-STD-830-1993* standard for RASD documentation this will include an overview of the functional and the non functional requirements of the system, showing the constraints and the limits of the software and simulating typical post-development scenarios through corresponding use cases.

This document is intended for all developers and programmers who have to implement the requirements, to system analysts who want to integrate other systems with this one, as well as for stakeholders interested in learning how the developed solution matches their request.

1.2 Scope

The project is aimed at creating a product that will allow end users to rent electric cars for any amount of time. Generic users should be able to register to the system by providing their credentials and payment information, receiving at the end of the process a password usable to access the system. Once accessed, the system should point to the users the locations of nearby cars available for reservation, which theyll be able to access within one hour after the reservation time. Once arrived on the site where the car is parked within the allotted time, a user should be able to prove its identity as the booker of the car, which would then be unlocked. The user should be charged while driving, in proportion to the time spent driving, until the car is parked in a safe area, whose set is pre-defined by the system. Once the user is out of the car, the system would lock the car and charge the user which would then be ready to be used by another user.

In case of expiration of the reservation, the user is charged of a predetermined fee. Economical incentives and penalties should be awarded in proportion to how eco friendly the user's behaviour has been.

1.3 Actors

- **Guest**: a guest is a user who has yet to join the service; as such, it can only consult the information regarding the service available on

the homepage and navigate between it and the registration and the login pages. To become a registered user the guest will have to fill in a registration form and provide payment method data as well as a photo of its driving license; the system will then analyze the data and confirm the success of the registration process within 3 days.

- **Registered user**: a registered user is able to log in.
- **Logged-In User**: a user allowed to search for nearby cars and make and (within 15 minutes from its making) cancel a reservation for one of them at a time. After the reservation it will be enabled to open the reserved car.
- **Car AI:** through its sensors, the car detects information regarding the ride that will be sent to the system, unlocks and locks itself and displays to the user infos about the surrounding area.
- **Power Enjoy System**: the Power Enjoy System primarily keeps a record of the users, the cars and the reservations updating the status of each of them according to the input from the users and the cars AIs. It also calculates the price for each ride and coordinates itself with payment systems interfaces to finalize money transfers, and warns the staff about exceptional situations such as ones where a car has been damaged, or abandoned in a discharged state.

1.4 Goals

List of the goals of PowerEnjoy application:

- [G0] Allow a guest to visualize information regarding the service and become a registered user.
- [G1] Allow a registered user to log in.
- [G2] Allow logged in user to check the location of nearby cars.
- [G3] Allow logged in user to reserve a car.
- [G4] Allow logged in user to cancel its reservation until 15 minutes from the reservation time.
- [G5] Allow logged in user to open the car reserved.
- [G6] Carry out the transaction at the end of the ride, after the calculation of discounts and penalties.
- [G7] Allow logged in user to ride the car.
- [G8] Allow logged in user to enable money saving option when in the car.

1.5 Definitions, Acronyms, Abbreviations

1.5.1 Definitions

- **Pairing Code**: A code provided to the portable device by the system after the reservation, to be used to open the car exploiting bluetooth technology. When the car is reached by the user its portable device and the car join what is called a trusted pair through a discovery and authentication process where the matching of the code in possession of car and device is verified, and at the end of the process the car opens.
- **Blocked User**: a registered user who is able to access the service yet unable to reserve a car since the money transfer related to the last transaction failed.
- **Safe Area**: An area where the car can be parked without being subject to further time-based charging. It corresponds to a collections of parking spots in the city of Milan reserved by the service for the users.
- **Grid Station**: A safe area where it is possible to plug an electric car.

1.5.2 Acronyms

- **RASD**: Requirements Analysis and Specication Document
- **DB**: DataBase
- **DBMS**: DataBase management system
- **API**: Application Programming Interface
- **OS**: Operating System
- **JVM**: Java Virtual Machine
- **JEE**: Java Enterprise Edition
- **AI**: Artificial Intelligence
- **GPS**:Global Positioning System
- **MB**: Megabyte
- **MSO**: Money Saving Option

1.5.3 Abbreviations

1.6 Reference Documents

- Assignment document: AA 2016-2017 Software Engineering 2 -Project goal, schedule, and rules
- SO/IEC/ IEEE Std 29148:2011(E) Systems and software engineering - Life cycle processes - Requirements engineering

1.7 Document overview

This document is divided in four sections:

- Section 1: Introduction, it consists of a generic introduction of the document aimed at providing the basic info necessary for the comprehension of the other sections contents.
- Section 2: Overall Description, gives general information about the software product with a special focus on constraints and assumptions.
- Section 3: Specific requirements, this part mainly includes different representations of functional requirements and also nonfunctional requirements.
- Section 4: Appendix, it cointains a modelization of the requirements created by using alloy language.

Chapter 2

2 Overall Description

2.1 Product perspective

The chosen solution is a web-based application meant for portable devices, which will take up to 20MB of free space on the device memory and will make use of Bluetooth technology for the car unlock mechanism as well as of GPS technology to detect the car closest to the user. Only interfaces to external systems of financial institutions and country authorities meant to verify the existence of the person associated to the document provided will be developed, since, as stated before, no other previous software infrastructure exists (bar the car AIs) and the the application will be self contained.

2.2 User characteristics

We expect our user pool to be comprised of rather young individuals (age varying between 18 and 35), owning a driving license but still not in possession of a car and/or with a preference for electric cars due to environmental reasons. The user is expected to be used to exploiting mobile apps and should not require a particularly user-friendly interface.

2.3 Constraints

2.3.1 Regulatory policies

- As previously explained, to utilize the service a user must provide proof of its driving capability. No other legislation binding car sharing services exists.

2.3.2 Hardware limitations

No hardware limitations exist.

2.3.3 Interfaces to other applications

Interfaces to the systems of financial institutions responsible for the money transfers connected to the transactions are to be developed.

2.3.4 Parallel operation

PowerEnjoy must support parallelism since it is bound to face reservation and access to the service requests from multiple users at once on a 24 hour basis.

2.4 Assumptions and Dependencies

2.4.0.1 Assumptions concerning the user

- Users do not seek to trick the sensors of the car to trigger the application of a discount (for example carrying around weight to mimic the presence of passengers).
- No user who reserves a car and actually reaches it does not inform the system about damages seemingly previously incurred by the car, even when though damaged it is still possible to drive it.
- Every user is held responsible for the damages incurred by the car driven and accountable for every fine received while driving it, being informed of this policy within the terms of use.
- The user never approaches another car that he didn't have reserve.
- Users always provide an address corresponding to their actual location.
- The user either plugs the car within 5 minutes from getting out of the car or he doesnt plug it.

2.4.0.2 Assumptions concerning the cars

- All the cars used are identical.
- Cars AI source of power is not the same as the engines.
- Cars' doors are locked while the engine is turned on.
- If the engine is turned on, the locking mechanism turns itself off only in case of accident.
- The car doors close automatically if the user, after having gotten out of the car, does not do it within 10 seconds.
- A car undergoing charging is not displayed whenever a search is carried out by the user, if its battery level is below 75%.

2.4.0.3 Assumptions concerning the operating space

- The service is presently active in Milan and nowhere else. Attempts to lead the car beyond the city boundaries will succeed, yet no safe areas nor grid stations are defined outside of Milan.
- Safe areas are evenly distributed around the city.
- Every grid station is a safe area.

- In Milan there is at least a grid station per 5 km area.
- In grid stations parking spots that have been booked through MSO are marked through led lights.

2.4.0.4 Assumptions concerning payments and discounts policy

- The system checks the validity of the payment method provided by the user only once during the registration process.
- If a transaction is not carried out successfully, the user is blocked and becomes unable to reserve other cars until the debt to the service is repaid.
- The service calculates the price of a ride and charges the user only after 5 minutes it has left the car, so to allow him to plug the car and exploit the potential related discount.
- The user can visualize from the map how much battery is left in each car found before reserving one.

2.4.0.5 Assumptions concerning the rides

- Every user has 30 seconds to enter the car it has reserved once the lock has been lifted.
- A ride is deemed ended when the user parks in a safe area, the engine is turned off and the user has got out of the car. If the engine is turned on again before the user has got out of the car when that is parked in a safe area the counter restarts using the ride price previously calculated (that is the amount previously shown on the car display up to a second before parking) as zero value.
- It is thus not possible to stop the car, exiting, closing the car doors and reentering the car and then keep driving under any circumstance.
- When the car battery fall to 5% a warning.
- If a car battery falls to 0% while the car is being driven and the car stops outside of a safe area as a consequence, the ride will be deemed concluded, but the driver will be held responsible for any accident or disturbance caused not having properly recharged the battery before the beginning of the ride and will be expected to move the car to a suitable location through any means, failure in doing so resulting in a violation of the terms of use.
- Pairing code and reservationID are one and the same.

Chapter 3

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

3.1.1.1 Download app This page represents the official page inside the app store, from where an user can download our PowerEnjoy app.

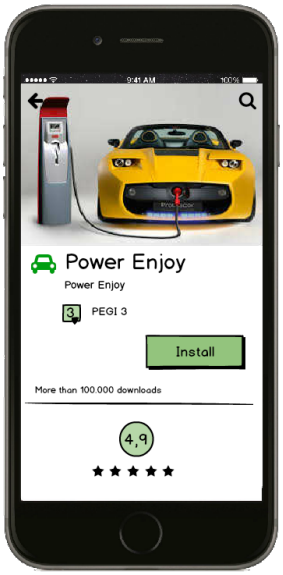


Figure 1: Download app

Before the installation procedure starts, PowerEnjoy app needs the users permission to access to some functionalities, like *GPS position*, *Bluetooth*, *Camera* and *Camera Roll*.

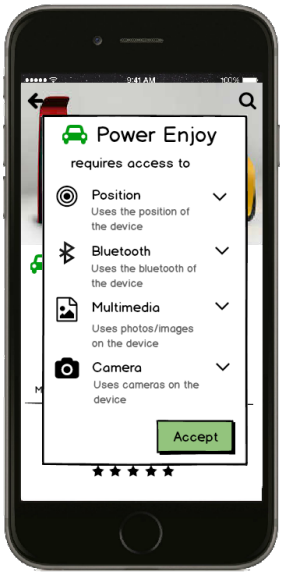


Figure 2: Download app permissions

3.1.1.2 Homepage Starting from the homepage, the user can easily register to the PowerEnjoy service or login.

Figure 3: Homepage

3.1.1.3 Registration Form In the registration form, the user must complete all the mandatory fields, accept the use term and upload a driving license picture.

Figure 4: Registration page 1

If one of the previous conditions won't be satisfied, the user must repeat the registration procedure.

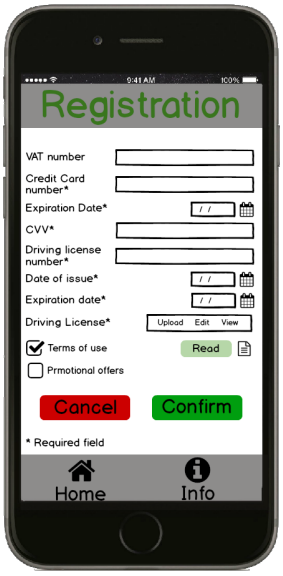


Figure 5: Registration page 2

3.1.1.4 Login Inside the login page, the user can easily login using a username and a password.

Figure 6: Login page

3.1.1.5 Personal Page This page can be seen after the login. The user can find statistics, a button for checking previous rides and a button to look for a new car.

Figure 7: Personal page

3.1.1.6 History Inside the history page, a user can check informations about previous rides, in chronological order.

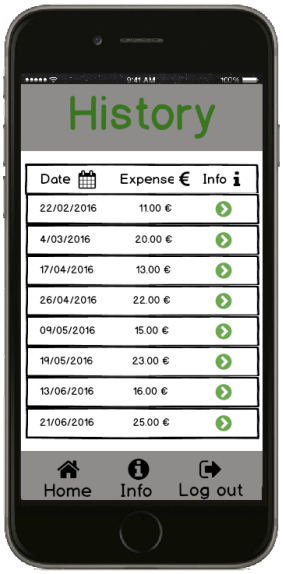


Figure 8: History 1

By selecting one element from the list, the app will provide many informations about the ride, including date, starting time, arriving time, final destination address, time, distance, total cost, discounts and penalties.

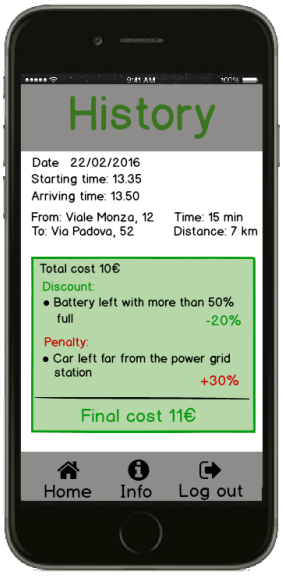


Figure 9: History 2

3.1.1.7 Info By pressing the 'Info' button placed on the task bar, the user can check the how discounts and penalties are applied.

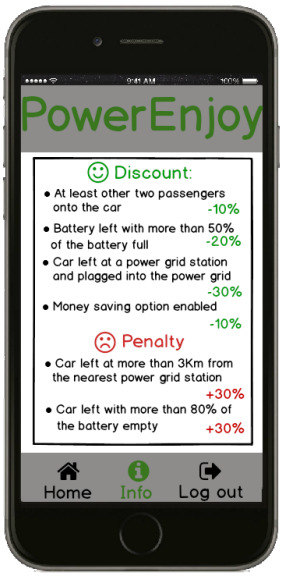


Figure 10: Info button shows penalties and discounts

3.1.1.8 Find a car From the Personal Page, by clicking on the 'Find a car' button, PowerEnjoy app will show all the cars available depending on user's position or inserted address (when a user doesn't give any permission for using the localization system).

Figure 11: Find a car button

In this map, the user can see his position and all the available cars in 1km radius.

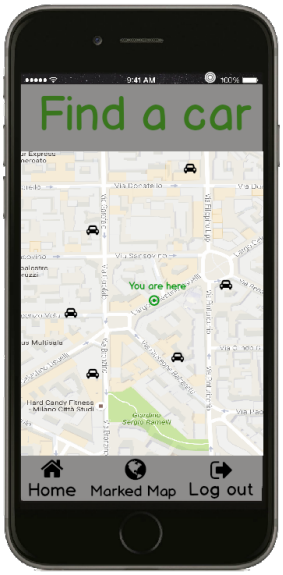


Figure 12: Find a car 2

By selecting a car, the user gets all the info and can reserve it. After pressing the 'Reserve' button, a popup will be shown to remember that the user has 1 hour for reaching the car and 15 minutes for canceling the reservation.

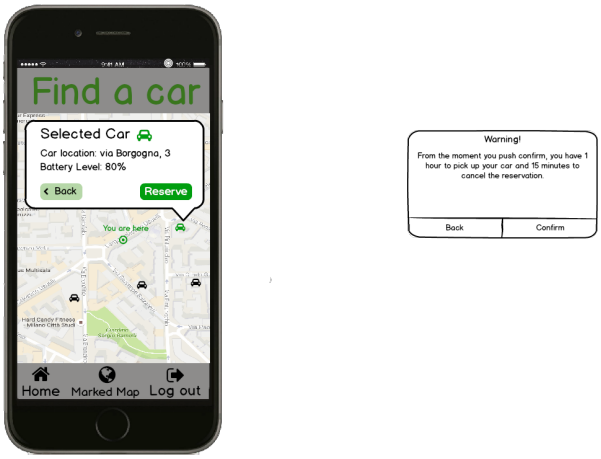


Figure 13: Book a car and popup

3.1.1.9 Marked map From the "Find a car" page, by pressing on "Marked Map", Power Enjoy app shows a map where there is the safe area and all the power grid stations. The user can also insert an address and take a look if it's inside or not a safe area.

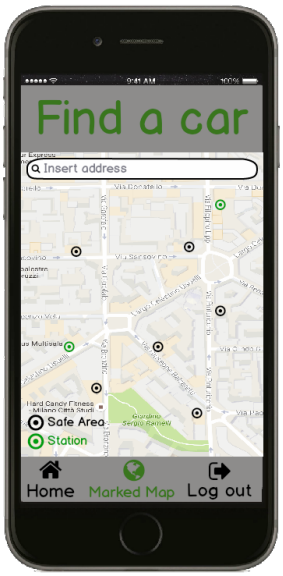


Figure 14: Marked map

3.1.1.10 Reserved Car The Reserved Car page is shown when a user has booked a car. The "Cancel" button is activated only for 15 minutes, after the booking.

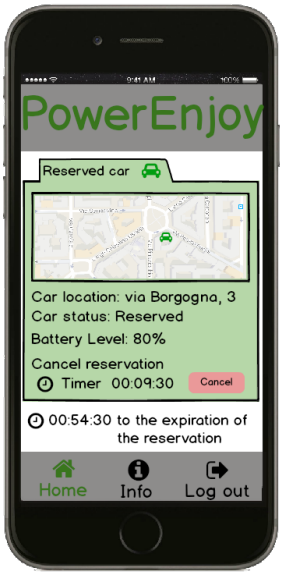


Figure 15: Reserved car

Here is represented the case when a user doesn't reach the car in one hour, starting from the booking time.

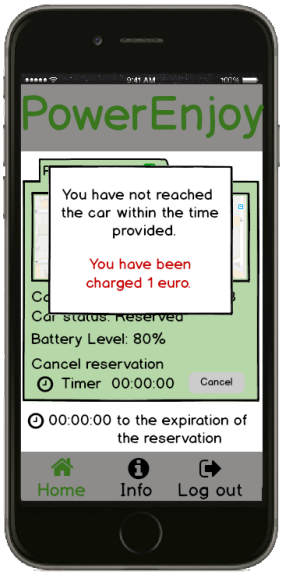


Figure 16: Reserved car not reached by the user in one hour

3.1.1.11 Reached Car The user has reached the booked car and the pairing process is going to start.

Figure 17: Reached car

The pairing procedure is concluded and the user can unlock the car or send a damage notification to the system.

Figure 18: Reached car and pairing process concluded

After pressing "Report damage" button, the user can take photos of the

car. When the damage notification procedure is completed. the user will be redirect to the "Find a car" page, where he can see the nearest available car.

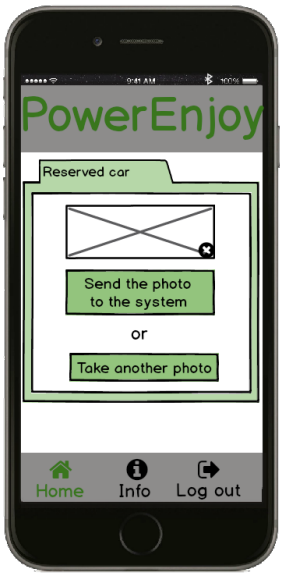


Figure 19: Send photos about damage

3.1.1.12 End ride Page shown to a user when the destination has been reached.

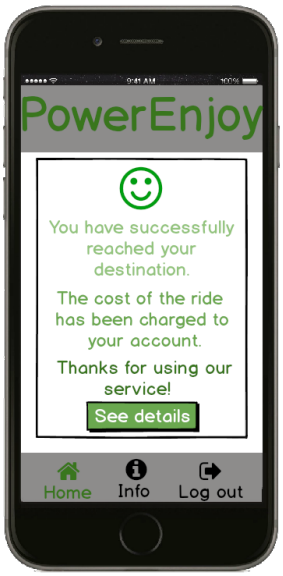


Figure 20: End ride

Once the user comes out from the car, by pressing on "View details" button, PowerEnjoy app will show all infos about discounts, penalties, date, stating time, ending time and price.

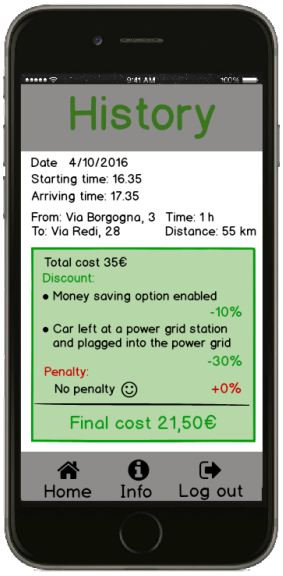


Figure 21: Send photos about damage

3.1.1.13 Navigation system This interface can be seen by the user while he is driving. Time, map and price are updated in real time. The map also shows where is the car and where is a safe area.

The "Saving money option" button is disabled, because it can be activated only before starting a ride.

3.1.1.14 Money saving option Before starting the ride, the user can activate the MSO from the navigation device. If MSO button will be selected, the user should insert the final destination address.

The navigation system will show to the user a map with arrows pointing to the closest power grid station, to a given address.

Figure 22: Money saving option

3.1.1.15 Battery warning The battery warning is shown on the car's GPS display when the car's battery is under 5%.

Figure 23: Battery warning

3.1.1.16 Battery empty The battery empty is shown when the car's battery is totally empty.

Figure 24: Battery empty

3.1.1.17 Boundaries warning The boundaries warning is shown, when the car's GPS understands that the user is driving outside the PowerEnjoy service area.

Figure 25: Boundaries warning

3.1.2 API Interfaces

For the map part of the PowerEnjoy application, we use **Google Maps API**. As well described on the website (https://developers.google.com/maps/), this API gives access to the location map in Italy for Android and iOS apps. Other companies like Runtastic, are using Google Maps API for their service, too. By paying Google a fee, its possible to get also tracking implementation for remote monitoring.

3.1.3 Hardware Interfaces

This project does not support any hardware interfaces.

3.1.4 Software Interfaces

DataBase Management System (DBMS)

- Name: MySQL
- Version: 6.0
- Source: https://www.mysql.it

Java Virtual Machine (JVM)

- Name: JEE
- Version: 7.0
- Source: http://www.oracle.com/

Application Server

- Name: Glassfish
- Version: 4.1.1
- Source: https://glassfish.java.net

Operating System:

- Any OS which supports JVM and DBMS specified in this document.

3.1.5 Communication Interfaces

Table 1: Communication Interfaces

Protocol	Application	Port
TCP	HTTPS	443
TCP	DBMS	3306 (default)

3.1.6 Memory

- Primary Memory: 4GB or higher (for the app back-end)
- Secondary Memory: 64GB or higher (for the app back-end)
- Device: 20 MB

3.2 Functional Requirements

3.2.1 [G0] Allow a guest to visualize information regarding the service and become a registered user.

- **[R0]** Until he has properly joined the service a guest can but access the registration and log in interfaces, and visualise important info related to the service.
- **[R1]** A registered user cannot join a second time.
- **[R2]** To carry out successfully the registration, the guest cannot choose a username already chosen by another user.
- **[R3]** To carry out successfully the registration, the guest has to provide a valid credit card number.
- **[R4]** To carry out successfully the registration, the guest has to provide a CVV matching the credit card number inserted.
- **[R5]** To carry out successfully the registration, the guest has to insert driving license data matching the one inferred from the driving license photo as inferred by the image analysis software.
- **[R6]** To carry out successfully the registration, the guest has to insert data related to a driving license that a proper authority acknowledges as actually existing.

3.2.2 [G1] Allow a registered user to log in.

- **[R0]** The user must be already registered to log in.
- **[R1]** The user must insert a correct username-password couple.

3.2.3 [G2] Allow logged in user to check the location of nearby cars.

- **[R0]** At least one car must be located within a 1 km radius from the user location.
- **[D0]** Users always provide an address corresponding to their actual location.

3.2.4 [G3] Allow logged in user to reserve a car.

- [**R0**] The user has to have carried out a search for nearby cars.
- [**R1**] The user must not be blocked.

3.2.5 [G4]Allow logged in user to cancel its reservation until 15 minutes from the reservation time

- [**R0**] The user has to have reserved a car.
- [**R1**] No more than 15 minutes can have passed since the reservation was made.

3.2.6 [G5] Allow logged in user to open the car reserved

- [**R0**] The user must have reserved a car and received the pairing code.
- [**R1**] No more than an hour can have passed since the reservation was made.
- [**R2**] The car has to not have been damaged.
- [**D0**] A user who realizes the car reserved has been damaged always reports the damage, even if the car is driveable.
- [**D1**] Every user has 30 seconds to enter the car it has reserved once the lock has been lifted.

3.2.7 [G6] Carry out the transaction at the end of the ride, after the calculation of discounts and penalties.

- [**R0**] The user must have exited the car.
- [**R1**] Five minutes have to have gone by since the user did.
- [**D0**] Users do not seek to trick the sensors of the car to trigger the application of a discount (for example carrying around weight to mimic the presence of passengers).

3.2.8 [G7] Allow logged in user to ride the car.

- [**R0**] The user has to have opened the car.
- [**R1**] the user has to open the car door within 30 seconds from having unlocked the car.
- [**R2**] The user has to ignite the engine.

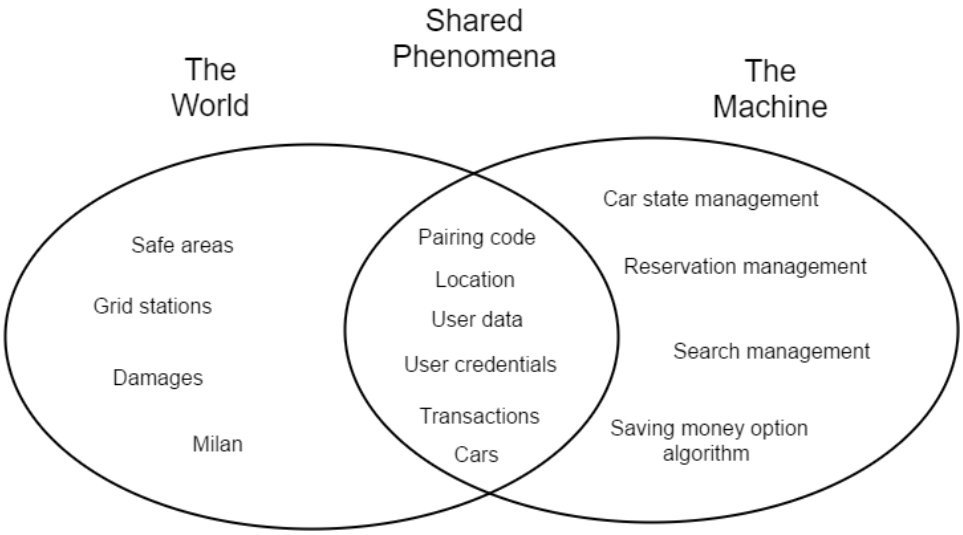
- **[R3]** The user has to park in a safe area, otherwise the ride will not be deemed concluded.
- **[D0]** A ride is deemed ended when the user parks in a safe area, the engine is turned off and the user has got out of the car. If the engine is turned on again before the user has got out of the car when that is parked in a safe area the counter restarts using the ride price previously calculated (that is the amount previously shown on the car display up to a second before parking) as zero value.
- **[D1]** It is thus not possible to stop the car, exiting and reentering the car and then keep driving under any circumstance.

3.2.9 [G8] Allow logged in user to enable money saving option when in the car

- **[R0]** The user has to have entered the car
- **[R1]** The user has to have closed the car doors.
- **[R2]** The user has to insert an existing destination.

3.3 The World & the Machine

In this section we present a schematic domain analysis of the application using the *The World & the Machine* model by M.Jackson & P.Zave. The world contains the physical elements to be considered within the development process, whereas the machine includes the software ones, the shared phenomena being the intersection between the two and containing consequently entities such as the cars, physical yet provided with software that enables them, for example, to calculate the shortest path to the destination within the MSO context.



3.4 Scenarios

3.4.1 Scenario 1

Murphy is in a peripheric area of Milan and decides to reserve an electric Car through his PowerEnjoy app to get to the center. Unfortunately the closest car owned by the company is parked further than 1 km from Murphy's current location, so Murphy walks around a bit searching repeatedly for an available car in his surroundings. After a bit the service finally locates a car in Via della Speranza, but being tired from his walking Murphy doesn't manage to reach a car within an hour, his reservation is cancelled and he is charged 1 euro.

3.4.2 Scenario 2

Kerouac is walking down the street and catches a glimpse of a Power Enjoy electric car; feeling in the mood for an ecofriendly car journey he makes a reservation for it from his smartphone and boards it. He would like to drive all the way to Rome, but when he approaches the city boundaries a warning is displayed on the car's LCD screen that the service is not supported outside of the city, yet Kerouac ignores it and keeps driving. When the battery falls to 5% a second warning is displayed, but Kerouac ignores this one as well. Suddenly the car slows down and comes to a stop near a sidewalk. Realizing his mistake, Kerouac gets out of the car and makes his way back to the city on foot, grimly checking the notification of the transaction on his phone: he's been charged 30% more than the normal for having left the car with a totally discharged battery.

3.4.3 Scenario 3

Faust makes a bet with his friend Mephisto: he will drive a PowerEnjoy car and enjoy all the discounts that can be applied to the ride. Having previously joined the service he makes a reservation for a car in the surroundings. Once he, Mephisto and his girlfriend Gretchen have entered the car, fastened their belts and closed the car doors Faust enables the money saving option from the screen inputting their final destination and ignites the engine. After driving for a short bit he reaches the station pointed by the system: the car has still 51% of its battery left. He and his two passengers leave the car and he takes care of connecting the car to the power grid within 5 minutes from leaving the car. After a bit, the payment notification appears on Faust's smartphone: all the discounts have actually been applied. Mephisto is enraged since he lost the bet, so he burns down the car. Murphy, having just reserved the same car, arrives some seconds later, realizes the damage and reports it to the system, heading now towards the next, nearest car pointed to him by the service.

3.4.4 Scenario 4

Sisyphus is deadly bent on registering to PowerEnjoy, so he downloads the app from the App Store. When asked for his info, he provides a fake credit card number since he wants to try to trick the system, but his attempt ends up in a failure since he is asked to repeat the process filling the related field with a valid one. This time he tries to leave the field blank, but he is newly asked to repeat the process filling the space. Sysiphus finally complies and provides correct data, being notified of the success of the registration the following day.

3.4.5 Scenario 5

Kierkegaard, longtime user of the service, does not want to be automatically localized by the system whenever he reserves a car, since he prefers to be presented with the choice of being localized through GPS or address each time he makes a reservation. This time he has chosen to give his address and ended up choosing a car quite far from his present location. Being unsure he would make it in time, he ended up deleting the reservation after 10 minutes from making it.

3.5 UML Models

We decided to represent the actor "System" in the UCDs in spite of it being unnecessary to allow for an easier comprehension of the content.

3.5.1 Use Case: Guest registration

Table 2: Guest registration

Participating actors	Guest
Entry condition	The guest is not registered into the system
Flow of events	1. The guest downloads the app from the app store on his device, gives permission to access Bluetooth, location info and camera and opens the app. 2. The guest clicks on the register button and enters the registration page. 3. The guest provides his credentials, a photo of his driving license, his payment information, accepts terms of use and confirms. 4. The system checks the driving license and payment information. 5. The system sends the password to the guest and informs the guest its registration has ended successfully.
Exit condition	The guest is now a registered user.

Exceptions	• The guest doesnt give the permission to access Bluetooth, location info and camera. The app is not downloaded. • The credentials are not correct. • The driving license is not accepted. • The payment information are not correct. • To be filled fields are not filled. All the exceptions are handled requesting the guest to continue the registration process from the point where incorrect info were provided or critical fields were left blank.

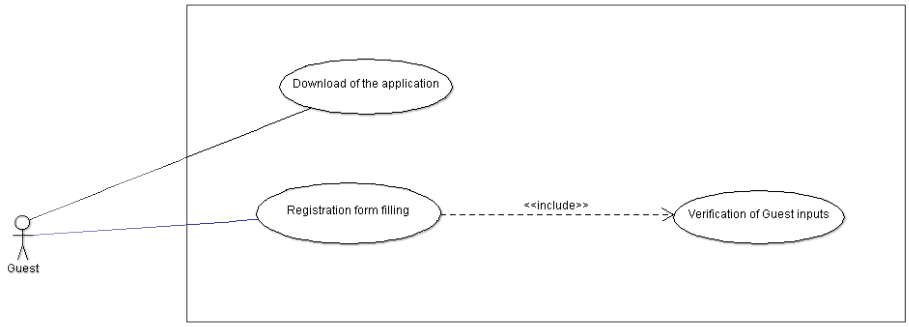


Figure 26: Guest registration use case diagram

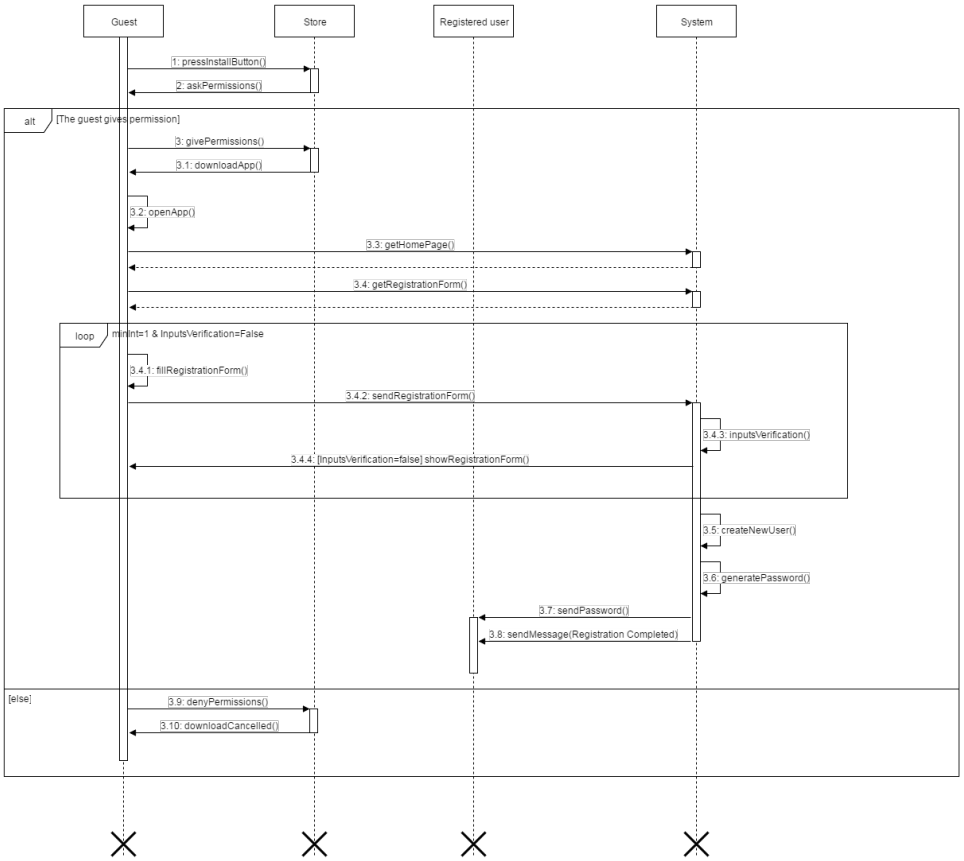


Figure 27: Guest registration sequence diagram

3.5.2 Use Case: Login

Table 3: Login

Participating actors	• Guest • Registered user • System
Entry condition	The visitor has already joined the community
Flow of events	1. The guest inserts the username-password couple associated to him. 2. The system checks the couple username and password and allows the guest to log in as registered user if the username-password couple is found in the database.
Exit condition	The user is logged in and redirected to his personal page.
Exceptions	• The couple username and password is not found in the database and the login fails. The app notifies the guest and redirects him to the login page.

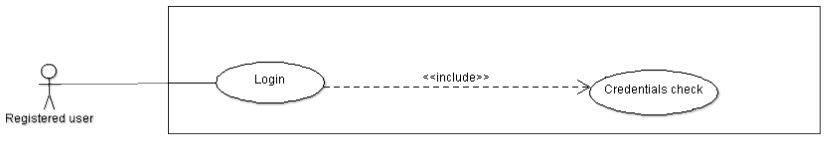


Figure 28: Login use case diagram

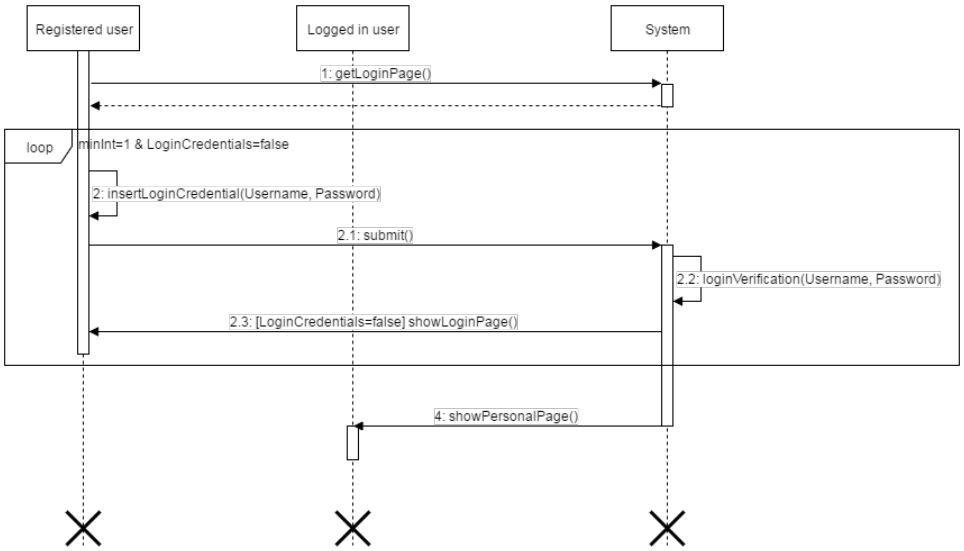


Figure 29: Login sequence diagram

3.5.3 Use Case: Find available car

Table 4: Find available car

Participating actors	• Logged in user • System
Entry condition	The user is logged in
Flow of events	1. The user selects the option Find car. 2. The system searches for available cars within 1 km from the location derived by GPS localization or specified explicitly by the user.
Exit condition	The cars closest to the user are visualized on the map.
Exceptions	• The user has not yet given his consent to be automatically localized. Both the options to give the address corresponding to his current location or be localized through GPS are displayed. • There are not available cars within 1km. The system informs the user that no available car is reasonably close.

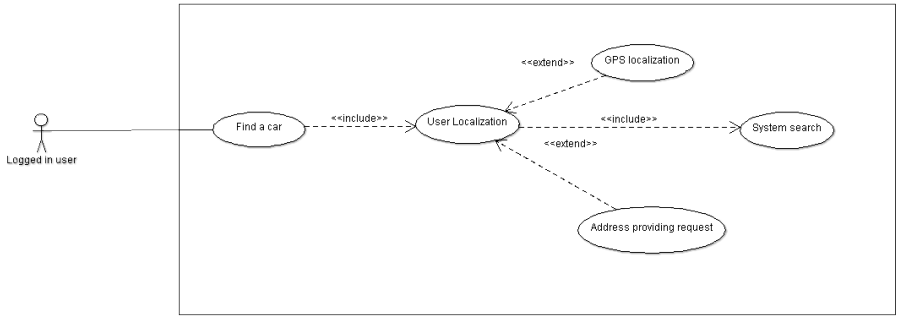


Figure 30: Find available car use case diagram

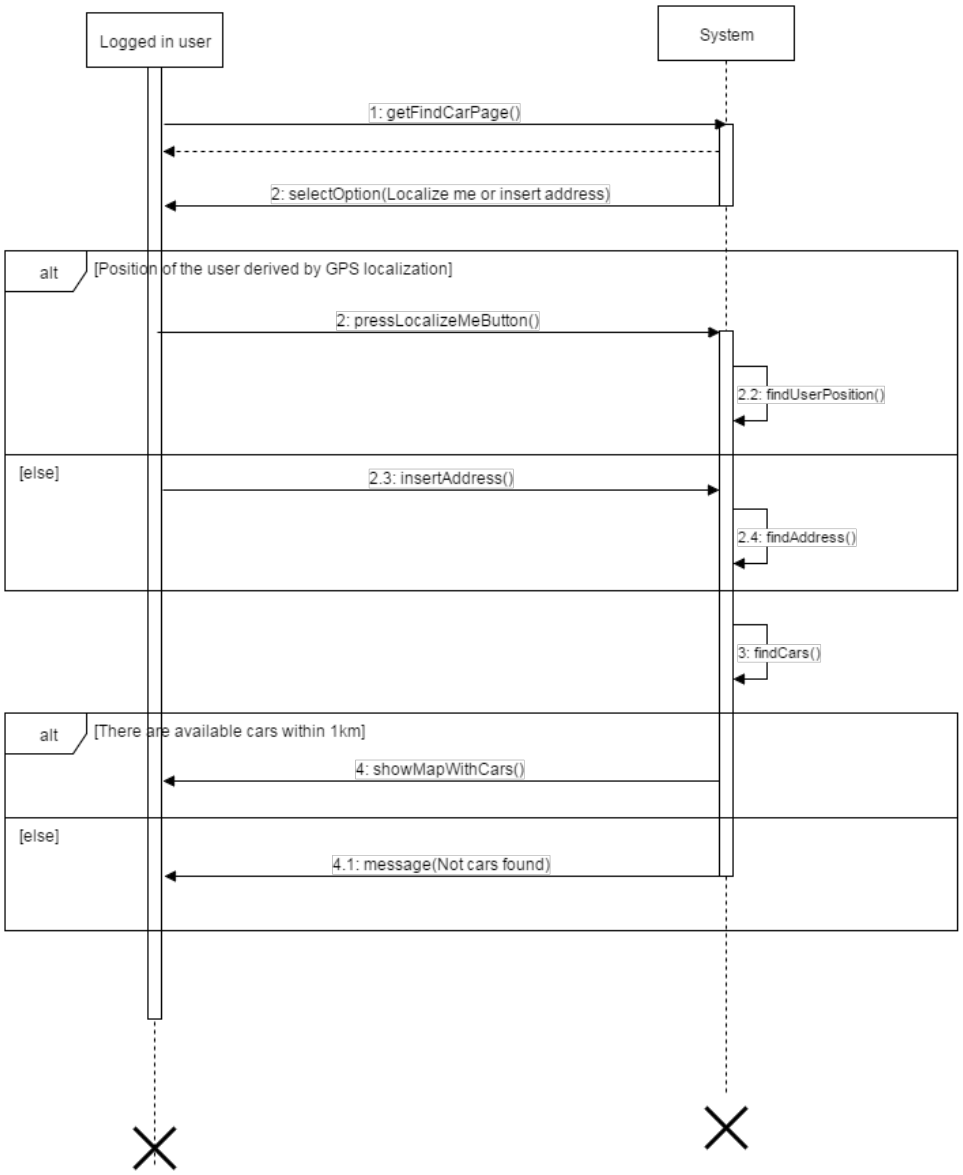


Figure 31: Find available car sequence diagram

3.5.4 Use Case: Reserving a car

Table 5: Reserving a car

Participating actors	• Logged in user • System
Entry condition	A search for available cars has just been conducted.
Flow of events	1. The user chooses a car from the map. 2. The system reserves the car for the user, who is given a pairing code matching with the one set up at the same time for the car. 3. Two timers are initialized by the system: the user has now 15 minutes to cancel the reservation without suffering charges and 1 hour to pick up the car.
Exit condition	The user has now a reservation.
Exceptions	• The car selected is reserved by another user in the interval between the visualization and the selection. The user is asked to select another car.

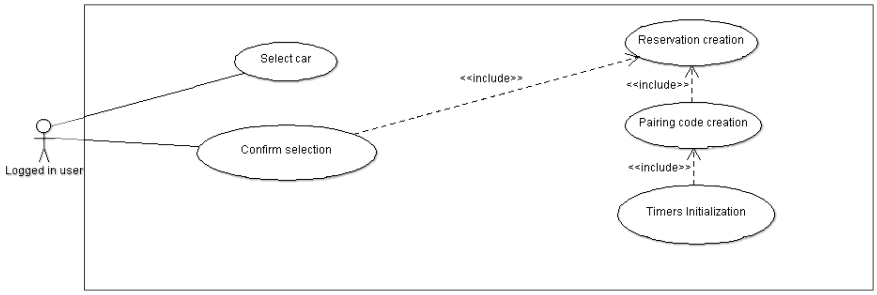


Figure 32: Reserving a car use case diagram

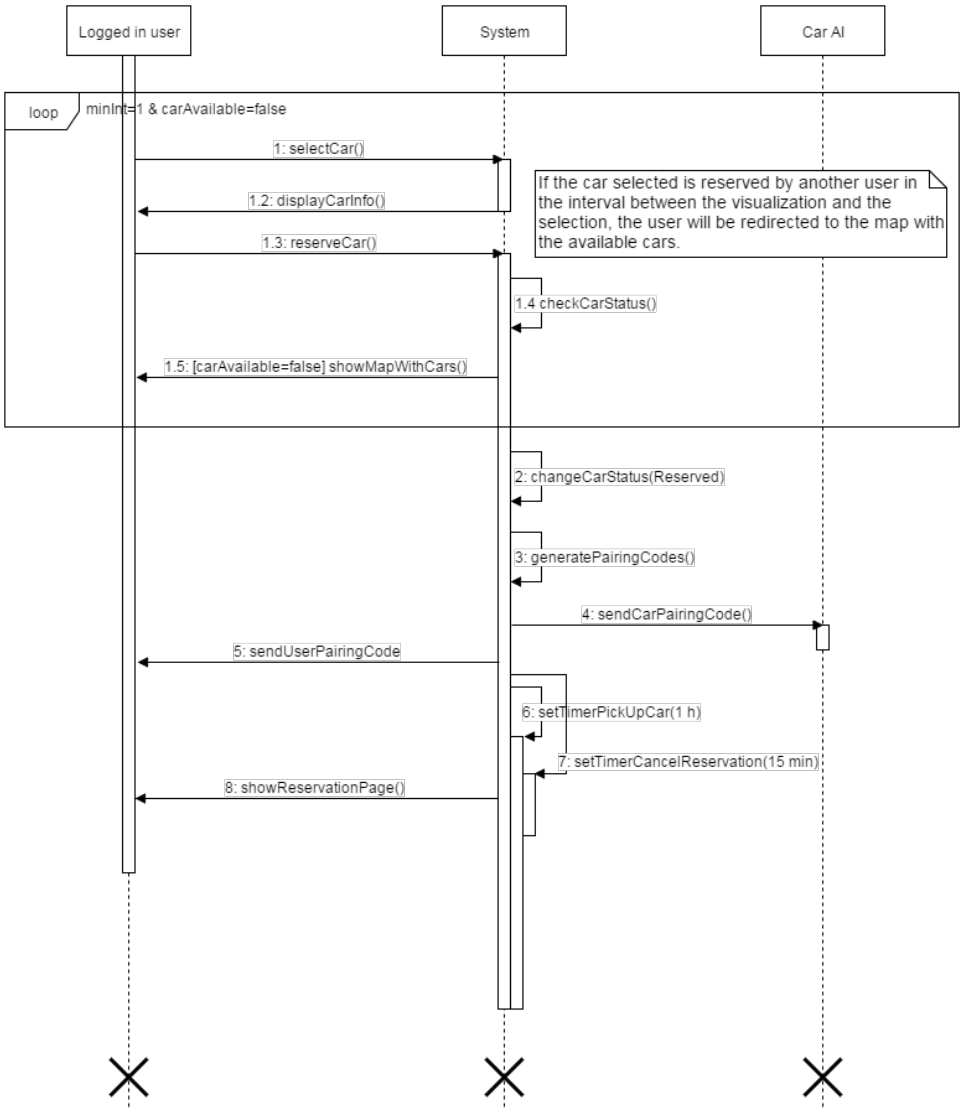


Figure 33: Reserving a car sequence diagram

3.5.5 Use Case: Deleting a reservation

Table 6: Deleting a reservation

Participating actors	• Logged in user • System
Entry condition	The user reserved a car not longer than 15 minutes ago.
Flow of events	1. The user cancels the reservation. 2. The system redirects the user to the homepage and changes the car status as available again.
Exit condition	The cancellation is carried out successfully.
Exceptions	• The timer expires during the cancellation process. The cancellation is carried out successfully.

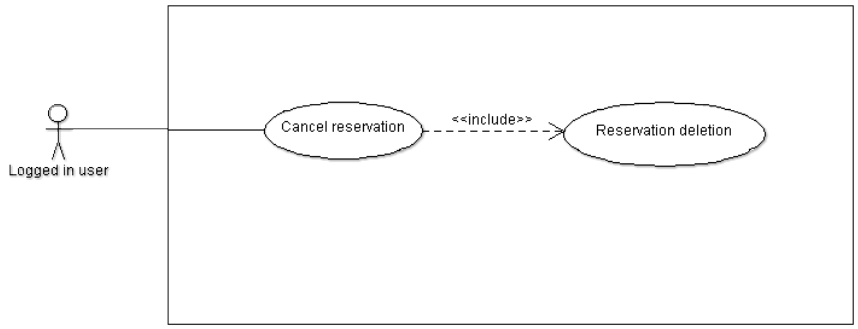


Figure 34: Deleting a reservation use case diagram

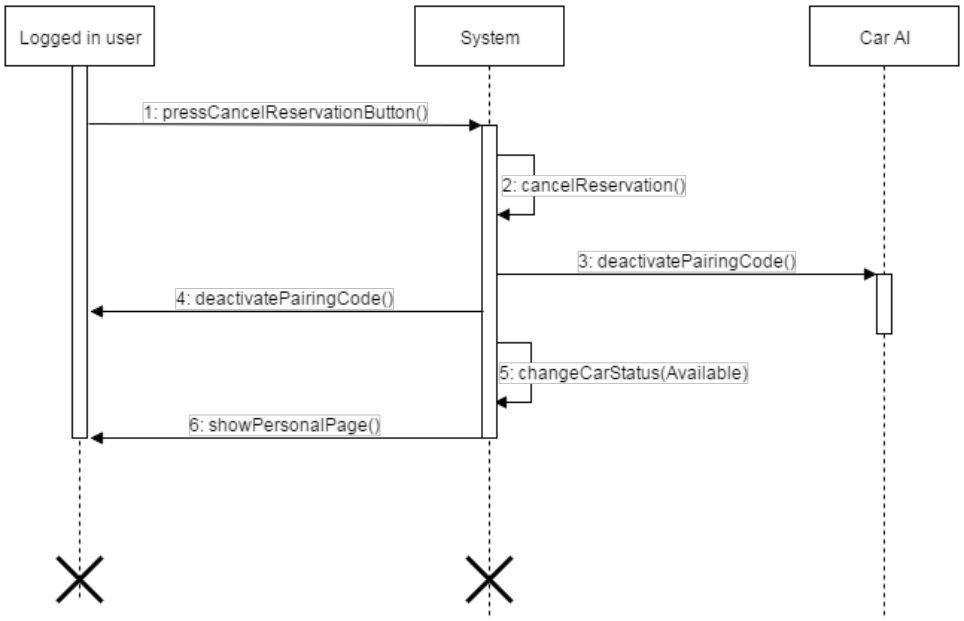


Figure 35: Deleting a reservation sequence diagram

3.5.6 Use Case: Picking up a car

Table 7: Picking up a car

Participating actors	• Logged In User • Car AI
Entry condition	The user has reserved a car and has reached it within the allotted time.
Flow of events	1. The car detects the presence of a smartphone connected to Power Enjoy whose Bluetooth function is activated, and starts the pairing procedure. 2. If the pairing is successful the app asks to the user if the car is damaged. If there is no damage is reported, the user can unlock the car through the app.
Exit condition	The car is unlocked.
Exceptions	• The timer expires when the user does not reach the car. The reservation expires, the system changes the car status to available again and charges 1 euro to the user. • The car is damaged. The app asks the user to take a picture of the damages and the system changes the car status to not available.The user will be redirected to the closest available car. • The user does not enter the car after he has requested its opening. After 30 seconds from the end of the pairing procedure the car newly locks itself and the reservation is cancelled.

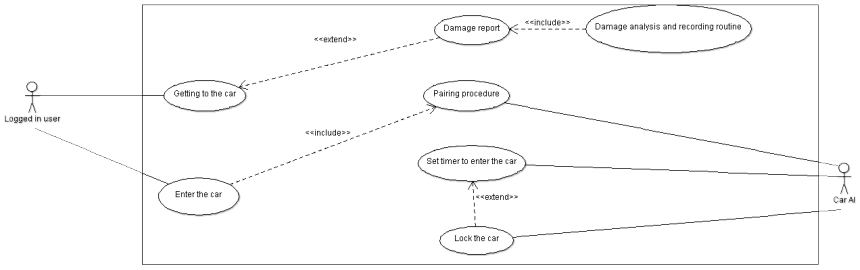


Figure 36: Picking up a car use case diagram

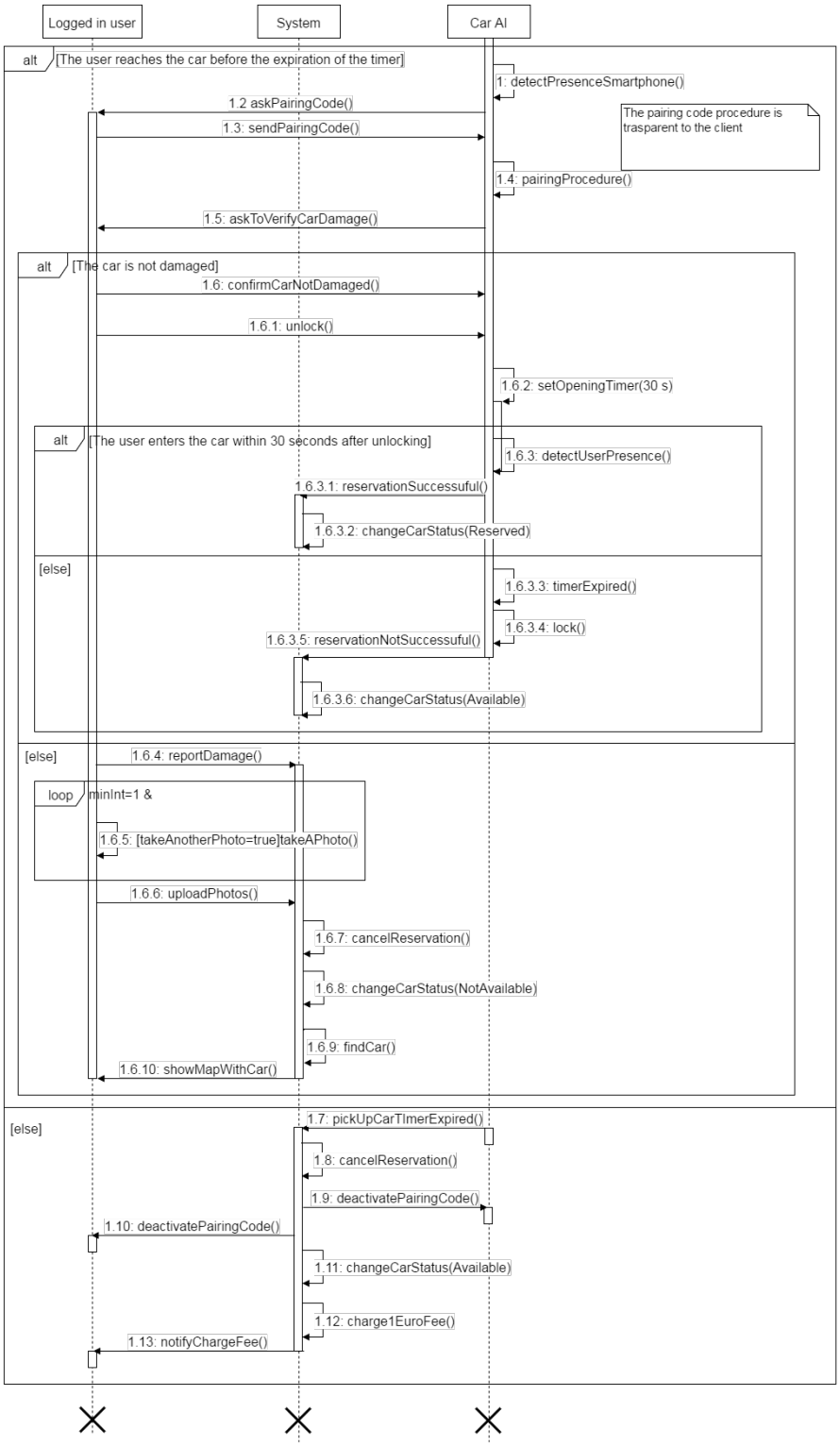


Figure 37: Picking up a car sequence diagram

3.5.7 Use Case: Managing the car being driven

Table 8: Managing the car being driven

Participating actors	• Car AI • Logged In User
Entry condition	The user is in the car that he reserved.
Flow of events	1. When the engine ignites, the system starts charging the user and does not stop until the car is parked in a safe area. 2. The system displays through a screen the amount of money spent by the user, updating the said amount every minute, as well as the safe areas in the vicinity of the car in real time. 3. When the user parks the car in a safe area and turns off the engine, the ride is deemed concluded for the moment.
Exit condition	The user parks the car in a safe area.

Exceptions	• The engine is ignited after the user has already stopped in a safe area. The ride restarts, the base price is set as the price of the previous ride. • The user tries to get out of Milan. A warning is issued to the system and to the user shortly before reaching the boundaries. • The car battery falls to 0% while the car is being driven and the car stops outside of a safe area. The ride is deemed concluded, but the user will be held responsible for any accident or disturbance caused.
Special requirements	• The system detects that the user takes at least two passengers onto the car. The system is made aware by the car that the user will be entitled to a discount of 10% when the final price of the transaction will be calculated.

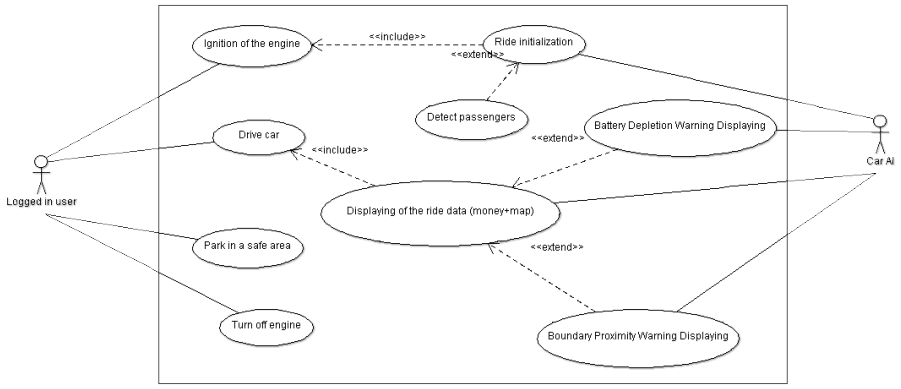


Figure 38: Managing the car being driven use case diagram

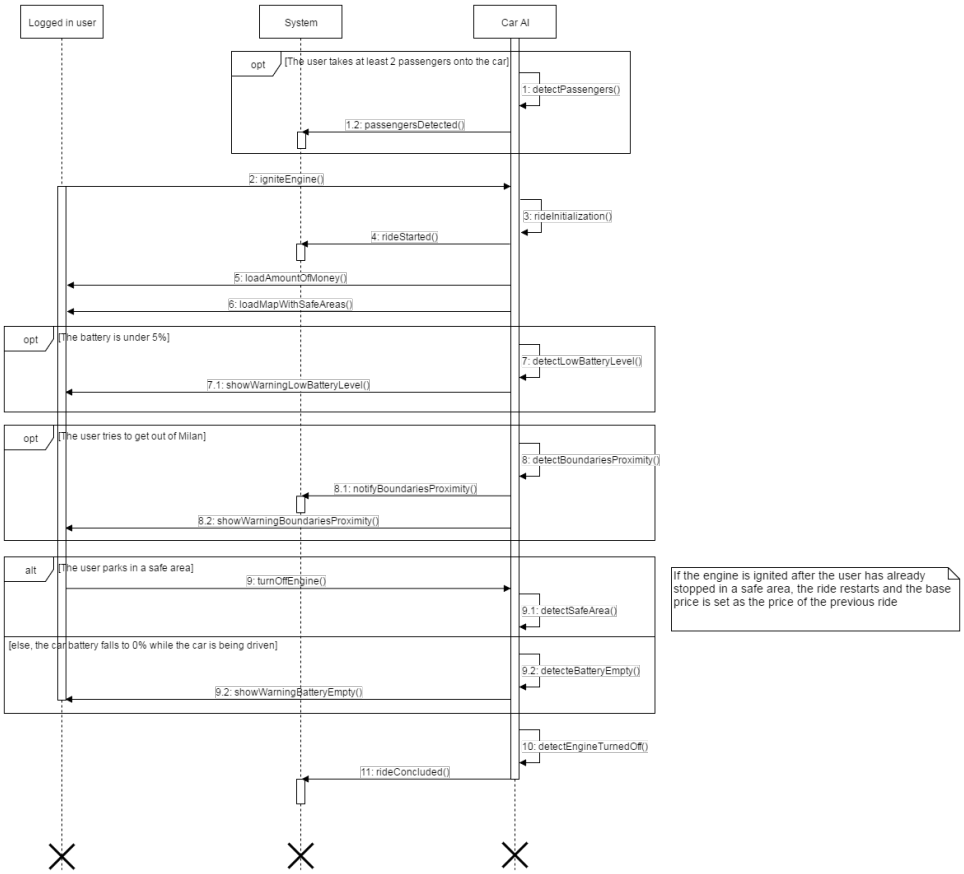


Figure 39: Managing the car being driven sequence diagram

3.5.8 Use Case: Concluding/ending the ride

Table 9: Concluding/ending the ride

Participating actors	• System • Car AI
Entry condition	The user has parked the car in a safe area, turned off its engine, got out of the car and closed the car door.

Flow of events	1. The car locks itself. 2. After 5 minutes the payment procedure starts. 3. The system first calculates the amount of money that the user has to pay based on the time spent in the car. 4. If the user has selected the money saving option, actually parking in the end in the station marked on the screen, a x% discount on the total price is applied. 5. If the car is left with more than half the battery full, a 20% discount on the total price is applied. 6. If the car was parked in a grid station and plugged in within 5 minutes from the car locking, a 30% discount is applied on the original fee. 7. If at the beginning of the ride the Car AI notified the system of the driver taking in at least two passengers a 10% discount is applied. 8. If the car is left with less than 20% of the battery a 30% penalty on the total price is applied. 9. If the car is left further than 3 km from the nearest grid station a 30% penalty on the total price is applied.
Exit condition	The calculation ends and the user is charged the amount calculated.
Exceptions	• The money available to the service through the payment method is not sufficient to cover the price of the transaction. The user is marked as blocked user.

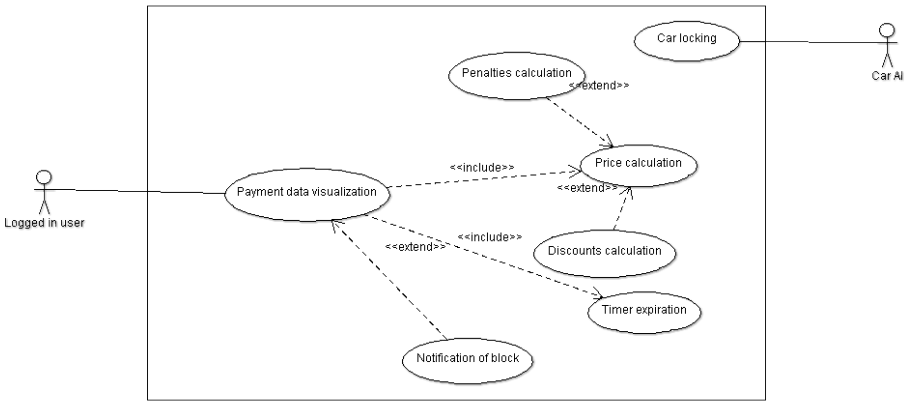


Figure 40: Concluding/ending the ride use case diagram

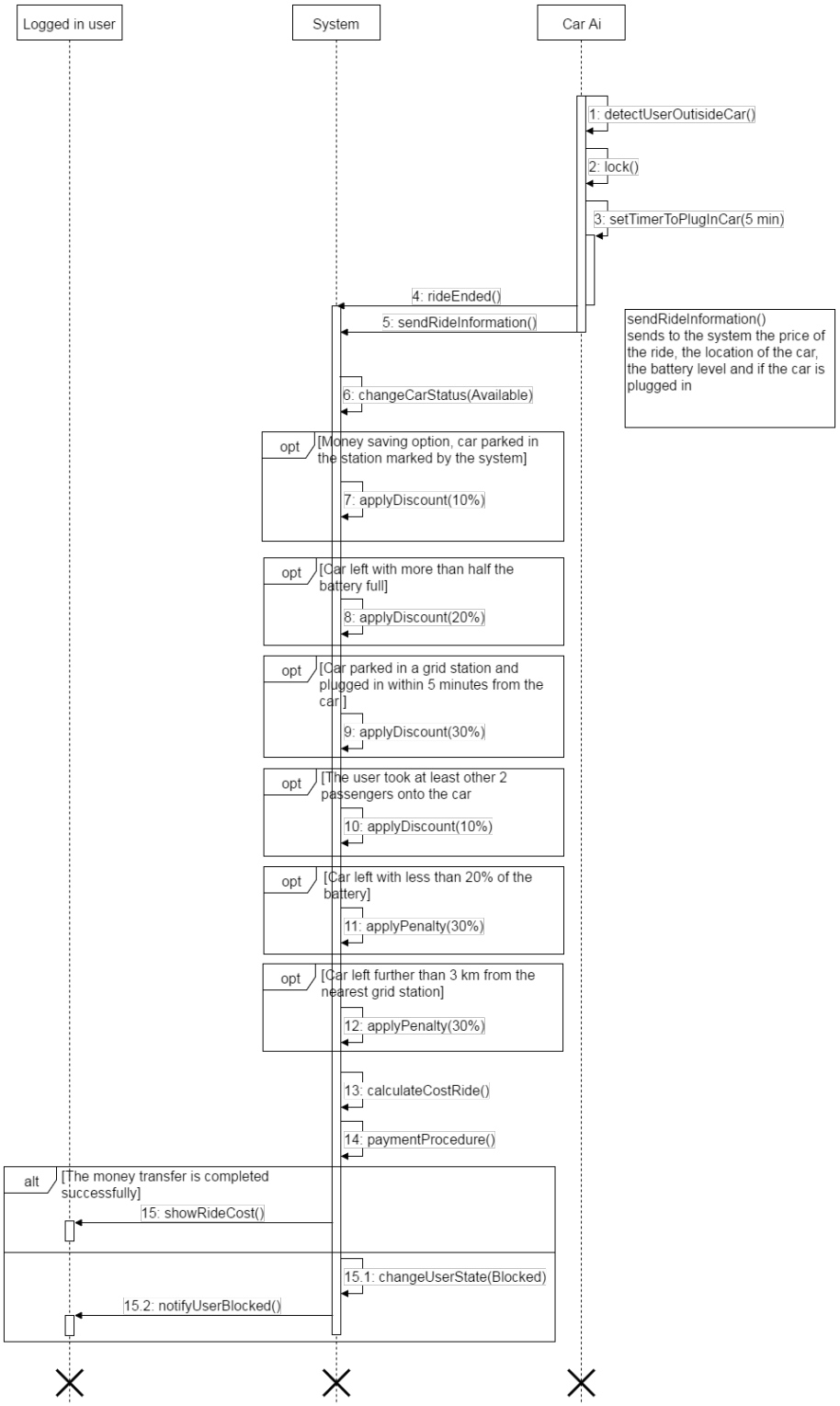


Figure 41: Concluding/ending the ride sequence diagram

3.5.9 Use Case: Managing of money saving option

Table 10: Managing of money saving option

Participating actors	• System • Car AI • Logged in user
Entry condition	A user has selected the money saving option on its vehicle screen menu
Flow of events	1. The car AI asks for the desired destination. 2. The user inserts the destination. 3. The car AI finds the location and sends it to the system. 4. The system chooses the station where the user is meant lo leave the car in a way that ensures an efficient distribution of the vehicles. 5. The car AI marks the station location on the map.
Exit condition	The station location is made known to the user.

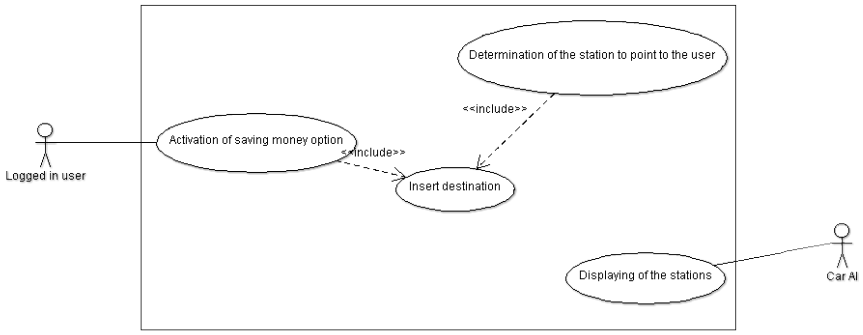


Figure 42: Managing of money saving option use case diagram

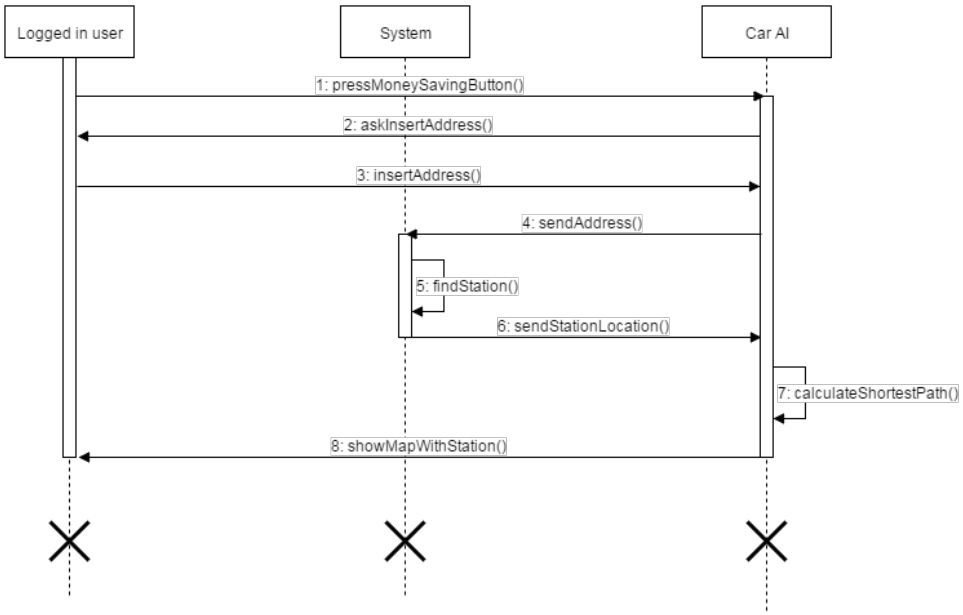


Figure 43: Managing of money saving option sequence diagram

3.5.10 Class Diagram

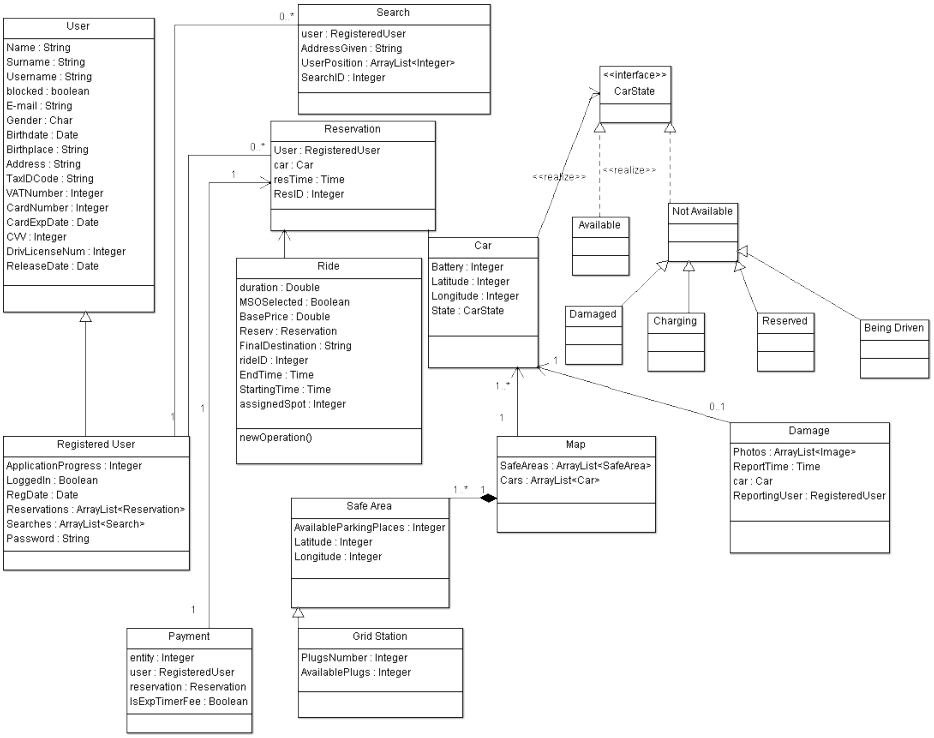


Figure 44: Class Diagram

3.5.11 State Machine Diagram

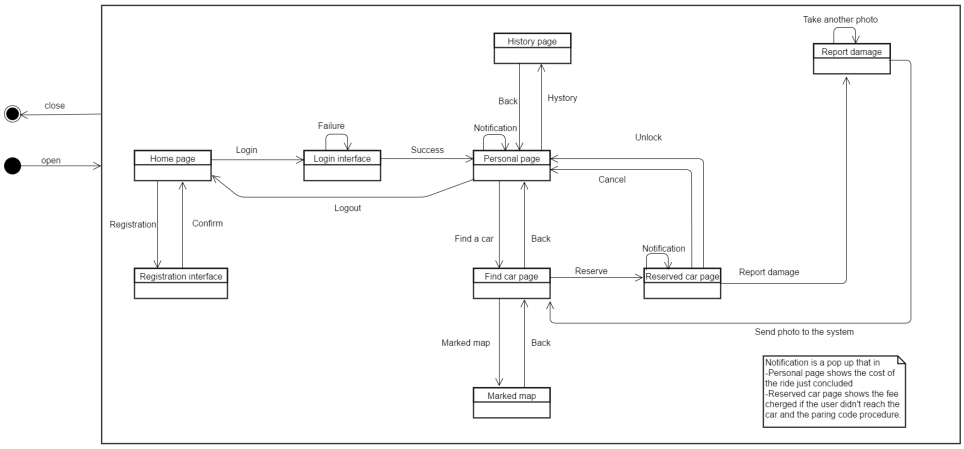


Figure 45: State Machine Diagram

3.6 Non Functional Requirements

3.6.1 Performance Requirements

Performance must be acceptable to guarantee a great user experience. The response time of the app depends mostly on the internet connection between the server and the smartphone. We assume that the connection time between cars and server is close to zero.

3.6.2 Design Constraints

The mobile app for Android operating system will be developed with Java EE. The iOS version will be developed using Swift, the latest programming language created by Apple and considered as the main standard.

3.6.3 Software System Attributes

3.6.3.1 Availability The application will be accessible online *24 hours a day/ 7 days a week/ 365 days a year.*

All the system will be hosted on a cloud platform like Amazon EC2, in Frankfurt.
We assume that failures will occur on average every 1000 days and that no more than 24 hours will be needed to repair the system in each case. So from these data we can infer that the mean time between failure (MTBF) will roughly correspond to 1001.

3.6.3.2 Maintainability The application doesn't provide any specific API, but the whole application code will be documented to inform developers about how the app works and how it has been developed. On a monthly basis software maintenance will be carried out, forcing the use of the backup servers.

3.6.3.3 Portability PowerEnjoy application can be installed and used on any mobile device with Bluetooth and GPS sensors that supports JVM. The Swift version ensures total compatibility with the iPhone.

3.6.4 Security

3.6.4.1 External Interface Side **PowerEnjoy app** implements a login authentication to protect user informations. In particular, each user password is saved using hashing mechanism. This methodology provide good level of security, even if the system doesn't require anything about the password strongness. So, it could be developed a system that requires

a strong password with 8 or more characters mixing numbers, uppercase and lowercase letters and special symbols. The password is static and the user isn't involved in changing the password. The system will ask to user to change the password every 3 months.

Among the future possible implementations we took into account figures a login procedure including captcha, to prevent potential attack from botnets, and other multi-factor authentication system.

Biometric authentication: through fingerprint, or retina scan.

3.6.4.2 Application Side On the application side, the register and login procedures implement a filtering system. However, malicious users could fill the form with SQL code to have access to hidden information, using SQL injection methodology.

PowerEnjoy implements https connection to guarantee communication confidentiality and integrity and also mutual authentication. SSL is resistant to man in the middle attack(MITM) but need a server certificate signed by a Certification Authority(CA).

Chapter 4

4 Appendix

4.1 Alloy

The following alloy model presented is created using the class diagram. We divided the code in some parts. In the last part there is the generated word. During the modelling phase we took into account just the entities and requirements we believed to be the most relevant.

4.1.1 Data Type

This is the definition of data type.

```
open util/boolean

sig string{}
sig Integer{}

abstract sig User{
username: string,
name : string,
surname: string,
TaxIDcode: string,
}

sig UserBeingRegistered extends User{}

//search: contains all the user searches
abstract sig RegisteredUser extends User{
password: string,
reservations: set Reservation,
searches : set Search,
loggedIn : Bool,
carcurrentlyreserved: lone Car
}
```

```
sig BlockedUser extends RegisteredUser{}

sig UnblockedUser extends RegisteredUser{}

sig Car{
CarID : Integer,
BatteryLevel: Integer,
status: CarStatus
}

sig CarStatus{
available : Bool,
beingchargedbutbelow75 : Bool,
beingdriven : Bool,
damaged : Bool,
reserved : Bool}

sig Available extends CarStatus{}

abstract sig Unavailable extends CarStatus{}

one sig BeingChargedAndBelow75, BeingDriven, Damaged, Reserved extends Unavailable{}
```

4.1.2 Abstract Entity Implementation and Signature

This is the implementation of some abstract entity and other signature.

```
//user: is the user who carried out the resevation
//car: is the car reserved
//search: is the research through wich the user was able to reserve the car

sig Reservation{
reservationID : Integer,
user : lone RegisteredUser,
car : lone Car,
resTime: string,
search: Search}

//reservations: contains all the resevations excluding those that have been deleted
//reservations contiene tutte le prenotazioni effettuate escluse quelle che sono state cancellate
one sig ReservationList{
identifier: Integer,
reservations : set Reservation
}

sig Payment{
amount: Integer,
batterydiscountapplied : Bool,
batterypenaltyapplied : Bool}

sig Ride {
duration : Integer,
MSOSelected: Bool,
Price : Integer,
reservation : Reservation,
FinalDestination : string,
active : Bool,
payment : lone Payment,
finaldistancefromsafestationgreaterthan3km : lone Bool,
BatteryLevelAtTheEndOfTheRidegreaterthan50 : lone Bool,
BatteryLevelAtTheEndOfTheRidelowerthan20: lone Bool
}
```

4.1.3 Fact

This is the fact part that defines the constraint of the class.

```
//There can not be two different users with the same username and the same tax code

fact noduplicateUser{
no disj u1, u2: User | u1.username=u2.username or u1.TaxIDcode=u2.TaxIDcode
}

//There cannot be two different cars with the same ID

fact noduplicateCarID{
no disj c1,c2: Car |c1.CarID=c2.CarID}

//There cannot be two different searches with the same ID

fact noduplicateSearch{
no disj s1, s2: Search | s1.searchId=s2.searchId
}

//There cannot be two different reservations with the same ID
fact noduplicateReservationId{
no disj r1,r2: Reservation | r1.reservationID=r2.reservationID}

//Each reservation is associated with a search.
//Reservation and search associated are related to the same user and the same car.

fact ASearchForEachRes{
all  r: Reservation| r.search.user=r.user and r.car in r.search.cars
}

//Each search allows to see only "available" cars

fact OnlyAvailableCarsAreDisplayedInTheSearchInterface{
all c: Car, s: Search|  c in s.cars  implies  c.status in Available}
```

```
//Each reservation in the reservation list must have an associated user and a car.

fact ResInResList{
all r: Reservation| some rl: ReservationList | r in rl.reservations implies (#r.user=1 && #r.car=1)
}

//Two distinct users can not have the same car reserved at the same time

fact  noCarReservedBy2User{
all c1, c2: Car, disj u1, u2: RegisteredUser| u1.carcurrentlyreserved=c1 and u2.carcurrentlyreserved=c2 implies c1!=c2
}


//The reservations can be carried out only by "unblocked" users

fact onlyUnblockedUsersCanReserveCars{
all  u: RegisteredUser |  #u.carcurrentlyreserved= 1 implies u in UnblockedUser}


//If at the end of the ride the car battery is greaterthat 50%, can not be lower than 20%
//Alla fine della corsa se la batteria della macchina è sopra il 50% non può essere sotto il 20% e viceversa
fact batteryLevelOrGraterThan50OrLowerThan20{
all ri: Ride | isTrue[ri.BatteryLevelAtTheEndOfTheRidegreaterthan50] implies isFalse[ri.BatteryLevelAtTheEndOfTheRidelowerthan20]
all ri: Ride | isTrue[ri.BatteryLevelAtTheEndOfTheRidelowerthan20] implies  isFalse[ri.BatteryLevelAtTheEndOfTheRidegreaterthan50]
}

//If at the end of the ride is applied a discount then the car battery is grater that 50%
//Alla fine della corsa se è applicato uno sconto allora la batteria della macchina sarà sopra il 50%
fact BatteryDiscountApplication{
all  ri: Ride  |isTrue[ri.payment.batterydiscountapplied] implies isTrue[ri.BatteryLevelAtTheEndOfTheRidegreaterthan50]}


//If at the end of the ride is applied a penalty then the car is parked at more than 3 km from the nearest station or the car battery is lower than 20%
//Alla fine della corsa se è applicato una penalizzazione allora la macchina è parcheggiata a più di 3km dalla stazione più vicina o la batteria della macchina è sotto il 20%
fact BatteryDistancePenaltyApplication{
all  ri: Ride  |isTrue[ri.payment.batterypenaltyapplied] implies ( isFalse[ri.BatteryLevelAtTheEndOfTheRidegreaterthan50] or
isTrue[ri.finaldistancefromsafestationgreaterthan3km])}

//Only if the ride is concluded then "battery level at the end of the ride" and "car position at more than 3 km from the nearest station" will be activated
//Solo se la ride si è conclusa allora i parametri "batteria alla fine della corsa" e "posizione della macchina a più di 3 km dalla stazione più vicina" saranno attivi
fact RelationBetweenActiveandParameters{all ri: Ride| isFalse[ri.active] implies #ri.finaldistancefromsafestationgreaterthan3km=1 and #ri.BatteryLevelAtTheEndOfTheRidegreaterthan50=1
and #ri.BatteryLevelAtTheEndOfTheRidelowerthan20=1}
```

4.1.4 Assert

In this last code part is presented the assert used to verify the model.

```
//Check if two distict users have the same reservation related to the same car at the same time

assert No2UserReserveTheSameCarAtTheSameTime{
all disj u1, u2: RegisteredUser| u1.carcurrentlyreserved!=none and u2.carcurrentlyreserved!=none implies u1.carcurrentlyreserved!=u2.carcurrentlyreserved
}

check No2UserReserveTheSameCarAtTheSameTime for 3 expect 0

//Check that if at the end of the ride are applied both the discout and the penalty,
// then the penalty is referred to the fact that the car is parked at more than 3 km away from the nearest station

assert NoBatteryDiscountAndBatteryPenaltyAtTheSameTime{
all ri: Ride| isFalse[ri.active] and isTrue[ri.payment.batterydiscountapplied] and isTrue[ri.payment.batterypenaltyapplied]
 implies isTrue[ri.finaldistancefromsafestationgreaterthan3km]}

check NoBatteryDiscountAndBatteryPenaltyAtTheSameTime for 3 expect 0

//Check that there is no "active" ride in where the car used is "damaged"

assert DamagedCarsNotRideable{all c:Car, ri:Ride |
c.status in Damaged and isTrue[ri.active] implies ri.reservation.car!=c}

check DamagedCarsNotRideable for 3 expect 0

//Check that a research doesn't display "unavailable" cars

assert SearchDoesNotLeadToVisualizationOfNotAvailableCars{
all s: Search, c: Car | c.status in Unavailable  implies c not in s.cars}

check SearchDoesNotLeadToVisualizationOfNotAvailableCars for 3 expect 0

//Check that the user making a reservation is not "blocked"

assert BlockedUsersCanNotReserveCars{
all c: Car, r: Reservation, u: RegisteredUser|  r.user=u  and u.carcurrentlyreserved=c  implies u not in BlockedUser }

check BlockedUsersCanNotReserveCars for 3 expect 0
```

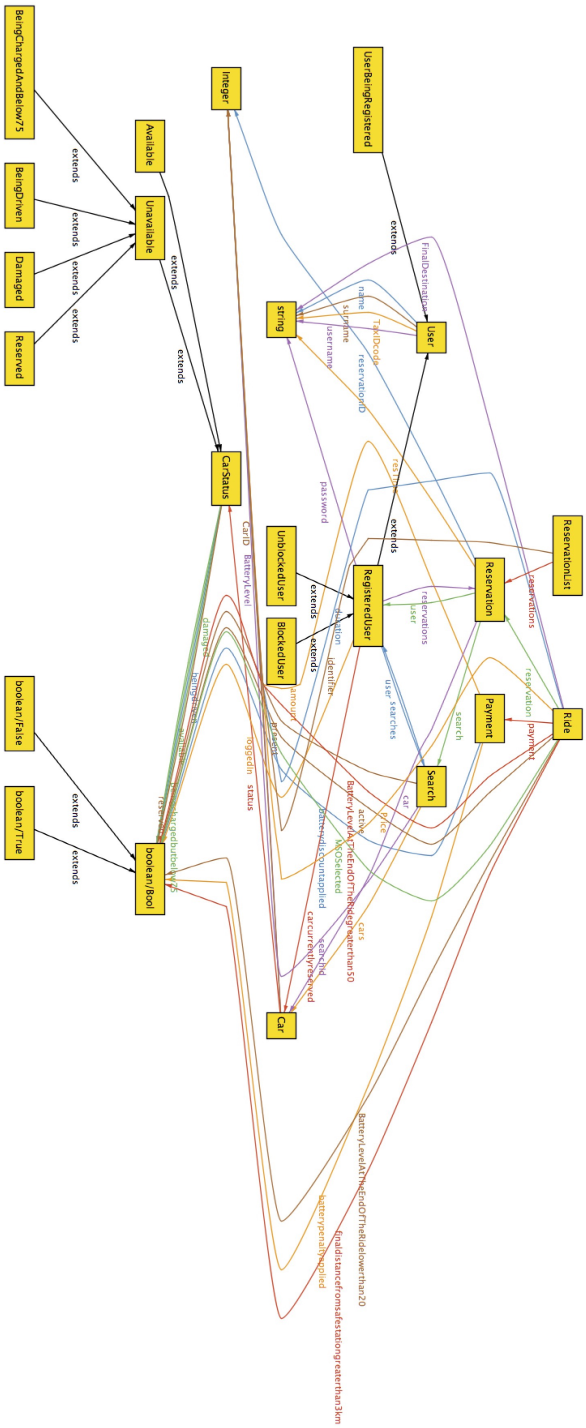
4.1.5 Predicate

This is the predicate used.

```
// A deleted reservation doesn't have an associated user and car

pred DeletedRes( r : Reservation) {
#r.user=0 and #r.car=0
}
```

4.1.6 Generated world



4.2 Software and tool used

- Lyx (http://www.lyx.org/)
- Alloy Analyzer (http://alloy.mit.edu/alloy/)
- draw.io Diagrams (https://www.draw.io)
- Argo UML (http://argouml.tigris.org)
- Balsamiq Mockups (https://balsamiq.com)

4.3 Hours of work

- Marini Alberto: 30 hours
- Marrone Matteo: 30 hours
- Sabatelli Antonella: 30 hours

4.4 RASD Modifications

- Added an assumption
- Updated Use Case Diagrams
- Updated Class Diagram
- Modified part of non functional requirements