Politecnico di Milano

A.A 2016-2017

Requirements Analysis and Specifications Document

Version 1.1

PowerEnjoy

Instructor : Prof. Di Nitto

Authors:
Amico Simone
Chianella Claudia Beatrice
Giovanakis Yannick

CONTENTS

1. Revision History

1.1 Version 1.1 -Updated on 07/01/2017

The following changes were made to the previous version 1.0:

- Class diagram was removed as it is not considered essential in this stage of development.
- Specific requirements were improved and report action added.
- New mockups added

2. INTRODUCTION

2.1 Purpose

The aim of this document is to give an overview of the requirements and specifications of the system to be developed. The goal of the document is to describe in detail all functional and non-functional requirements of the system, analysing the needs of the customer and explaining common use case scenarios. It will set a baseline for project planning and cost estimation, giving a detailed insight to all stakeholders which include the PowerEnjoy Board , Investors and finally engineers (present and future) involved in development, testing and maintenance.

2.2 System

The to be developed software system provides a complete support to the new PowerEnjoy car-sharing service. It will allow users to register ,log-in and use the car sharing service within the limits imposed by the below specifications.

2.3 Scope

The new PowerEnjoy software system will allow visitors to register and retrieve all necessary information of common domain (ToS , contact information...).
Once the visitor has successfully registered into the system and his/her driving license has been approved, he/she will be provided with a unique username-password combination. This enables a user to log into the system at any time , from any mobile device.
Once logged in, the user can search within a certain distance for vehicles based on his/her current location or based on an input address. Eventually the user chooses to reserve an available vehicle, which can be picked-up within a time span of one hour.
If the user doesn't pick-up the car within the one hour availability limit, then the system will tag the vehicle as available and charge the user a fee

of 1 Euro.
If the user reach the car within the limit,he/she must be able to interact with the system in order to unlock the vehicle and grant the user access to the car. As soon as the engine ignites,the system starts charging the user for a given amount of money per minute. The current charge will be notified through a display inside the car.
The system stops charging the user as soon as the car is parked in a safe area and the user exits the car. At this point the system will automatically lock the car. If the user stops in a non-safe area he/she won't be granted to end the transaction and will be charged until he/she parks in a safe area. The set of safe area parking spots is defined inside the system and must be available all the time to the user's knowledge through the on-board display.
In addition to the functionality above, the system should incentivize the virtuous behaviours of the users with some bonuses:[1]

- If the user shares a ride with at least two other passengers , the system will detect them and will apply a discount of 10% the last ride.
- If the user parks the car in a safe area and the battery charge level is more than 50% , the system will apply a discount of 20% on the last ride.
- If the user parks the car in a safe area provided with a PowerEnjoy charging station and takes care of the recharing ,the system will apply a discount of 30 % on the last ride.
- If the user parks the car in a safe area with a battery charge level less than 20% or the distance to the next power grid is greater than 3 KM, the system will add a fee of 30% of the last ride to compensate for the cost required to recharge the car on-site.
- If the user enables money saving option,he/ she can input the final destination and the system provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of the cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

[1]Bonuses are not combinabale. Only the highest bonus will be awarded

2.3.1 Goals

The PowerEnjoy software system needs to be:

- **[G1] : Practical**
  It wants to give the users a better connectivity among places in the participant cities that common public transport often can't offer.
- **[G2] : Convenient**
  The service needs to be convenient: it's less expensive than taking a taxi. It's an eco-friendly solution : car-sharing reduces the number of vehicles circulating in the cities and doesn't pollute thanks to electric engines.
- **[G3] : User friendly**
  The system needs to be as user friendly as possible in order to be used easily by everyone through their own intuitive knowledge.
- **[G4] : Fast**
  The system needs to be fast and responsive even during high peaks of activity.
- **[G5] : Available**
  The system needs to be available at every time and on the largest possible number of devices (mobile and not) with an internet connection.

2.3.2 Applications

To achieve the above mentioned goals the following software tools need to be developed:

- **Mobile Application**: to be developed for all major operating systems (iOS, Android , Windows Phone). The mobile software must allow the user to perform all the tasks in the *Scope section (1.3)*.
- **Web Application**: must be compatible with all major browser applications (Chrome, Firefox, Safari, Edge, Opera). The user must be allowed to perform all tasks in the *Scope section (1.3)*.

- **On board car application**: this application needs to handle user interaction such as displaying general car information (charge, fee, tire pressure...), letting the user validate his/her identity through a pin code and report about the status of the vehicle. Furthermore it must control if the vehicle is parked in a safe area.

- **Back-End Application**: this application handles all search and reservation requests, payment transactions and car communications. It is also an essential tool for customer support operators.

2.4 Reference Documents

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications

- Project description: Assignments AA 2016-2017.pdf

- Alloy Language Reference : `http://alloy.mit.edu/alloy/documentation/book-chapters/alloy-language-reference.pdf`

- UML Language Reference : `https://www.utdallas.edu/œchung/Fujitsu/UML_2.0/Rumbaugh--UML_2.0_Reference_CD.pdf`

2.5 Glossary

2.5.1 Actors

- *Visitor*: a visitor is defined as a person who is not logged into the system.

- *Registered User*: is defined as customer who is logged into the system.

- *Car*: is defined as the mean of transport to be used by a registered user.

- *Support operator*: The support operator is an employee who is trained and has a full grasp of the different software application.

2.5.2 Other definitions

- *Valid Payment*: as payment we accept only Credit Cards (Visa, MasterCard and American Express).
  The payment method must be valid : the card must not be expired nor blocked. The validity will be checked immediately during registration phase.
- *Valid Drivers License*: the user must be provide a valid european drivers license. The license must be valid: it must not be expired or suspended. The validity will be checked immediately during registration phase.
- *Safe Area* : a safe area is any part of the city where the user is allowed to park. Safe areas are pre-defined and installed into the cars' on board systems. If a user parks the car in an area which is *not* safe , he/she will not be able to end the ride and will be charged until he/she parks in a safe area.
- *Money Saving Option*: when a user starts a ride he/she can enable this mode on the onboard display: he/ she can input the final destination and the system provides information about the station where to leave the car to get a discount.
- *Reserved Car*: is a vehicle currently used by a user. Reserved cars are not visible as results of a search query.
- *Time left to unlock*: as soon as a user reserves a car, he/she has one hour to unlock it. The time left is displayed on top of the user's mobile/web application.
- *Charging Station*: all PowerEnjoy's cars are provided with an electric engine. A charging station is a location where the user can park and recharge the car. All charging stations are listed in the on- board system.
- *Battery level* : the current state of the battery charge is indicated by the battery level.
- *Passenger* : is a person (not necessarily registred) who shares a ride with a registred driver. A passenger cannot drive but his presence will be detected by the system.

- *On-board application*: is the software installed on every car. The user can interact with it through a touch screen display.
- *Mobile application*: is the downloadable software that can be installed on major mobile devices.
- *Web application*: is the online software that is accessible through all common web browsers.
- *Back-End application*: is the part of the system that manages queries ,data and user-system interaction.
- *Ride*: a ride is a time span starting from the moment the user turns the car on and ends when the user parks it in a safe area after the car has automatically locked the doors.
- *Payment History* : a chronological list of all payments and rides made by a given user.
- *Available car* : a car that is not reserved or charging. These cars are shown as result of search queries and can be reserved by users.

2.6 Document overview

1. **Introduction**: this section gives a general description of the software and its characteristics
2. **Overall description**: this section gives a detailed overview of the document focusing on domain assumptions, requirements (functional and not) and the characteristics of different types of interacting users.
3. **Specifications**: this section gives a deep insight about the system's main functionality, analyzing scenarios with their relative use-cases and includes an extensive alloy model. Event flows are model using sequence and state diagrams . An Alloy model will be provided and discussed and finally the non functional requirements will be discussed.
4. **Appendices** : this section talks about the tools used for the creation of this documents and the time spent working.

3. Overall Description

The overview section highlights all the main functionality and constraints of the PowerEnjoy software.

3.1 Software overview

The system-to-be is going to be composed of 4 main applications:

- Web Application
- Mobile Application
- On-board Application
- Back-End Application

3.2 User characteristics

Specifies the characteristics of the different actors mentioned in *Actors section (1.5.1)*

- **Visitor**: A visitor can see only the log-in page with a registration form and some common information ,such as the company's customer office contact information and terms of service. There are no constraints about visitors : every person can access the main site and try to register.
- **User**: a want-to-be user must match the following requirements:
  - the user has to be **at least 18** years old
  - the user must have a **valid** driving license [2]
  - the user must enter a **valid** payment method [3]

[2] For more check out the *Glossary*
[3] For more check out the *Glossary*

- **Car**: is property of the PowerEnjoy Company. A car must be able to communicate to with the system all the time. If a car is available (i.e currently not reserved by a registered user) then its current location must be stored in the system so to be shown on all users' devices. As soon as the engine ignition starts , the car's system must time the vehicle's usage in order to charge the user the right amount of money. Once the user has parked the car inside a safe area and has exited, the car informs the system about the fee to charge ,locks the doors automatically and finally informs the system that it is available again.

- **Support Operator**: his job is to offer costumer support concerning software issues and help out in case of emergency.
  Contrary to a user, he has access to the back-end software which enables him to overview the system status.

3.3 Domain assumptions

3.3.1 General Domain Assumptions

We hold the assumption that the system will be deployed in large metropolitan cities with a maximum of 3M inhabitants.
We assume that the mean highest peak of activity is around dinner time on Fridays and Saturdays. We assume that during the peak the system will be used by around 50k people , 80% of which gain access to the system through the Mobile Application, while the remaining 20% through the Web Application.
We assume that the the fleet of vehicles will be proportional to the number of inhabitants, reaching a peek of 300 vehicles in the biggest area.

3.3.2 User Assumptions

We assume that all users are provided with an enabled internet connection equipped with GPS tracking which is the only way to search, reserve and unlock a car. We assume that all active users carry their driving license

with them and drive following the traffic code[4].

3.4 Constraints

3.4.1 Software Limitations

The Mobile and Web Application can not exceed 50MB of RAM usage.
The On-Board application can not exceed 75MB of RAM usage.
The Back-End application cannot exceed 100GB of RAM usage.

3.4.2 Hardware limitations

The Mobile Application must be available for the following mobile operating systems:

- iOS version 7.0 or higher.
- Android version 4.0 or higher.
- Windows Phone 8 or higher.

The applications must interface with Google Maps API [5] and follow the design guidelines provided by each platform [6] [7].

The Web Application must be developed in order to be correctly used by the following web browsers [8]:

- Mozilla Firefox
- Google Chrome
- Safari
- Microsoft Edge
- Opera

[4] http://www.mit.gov.it/mit/site.php?p=normativa&o=vd&id=1&id_cat=&id_dett=0

[5] https://developers.google.com/maps/documentation/javascript/

[6] https://developer.android.com/design/index.html

[7] https://developer.apple.com/design/

[8] http://www.w3schools.com/browsers/default.asp

The on-board software system must be equipped with a GPS Module and interface Google Maps API. The following safety regulations must be followed:

- **Night mode**: an automatic night mode must be implemented. This will avoid that drivers get blinded by excessive brightness of the display.
- **Safe - area parking recognition**: the system must recognize whether the vehicle is parked in a safe-area or not.
- **Additional passenger recognition**: the system must recognize passenger. This feature is crucial for bonus credit distribution to the driver.

3.4.3 Concurrency

The back-end application must be able to handle correctly multiple user requests without creating conflicts : if two users desire two reserve the same car simultaneously ,only one can be granted a reservation.

3.4.4 Regulatory Policies

- **Privacy Policy**: Data should be collected and stored following the privacy policy guidelines provided by Mozilla Foundation `https://developer.mozilla.org/en-US/Marketplace/PublishingPolicies_and_Guidelines/Privacy_policies`
- Network connections must be encrypted.
- Data storage on PowerEnjoy databases must be encrypted

3.5 Main functions

A more detailed view about functions and requirements can be found in section 3. *Functional requirements(3.2)*

1. *Web and Mobile Application*: both user applications must provide form for user registration and log - in. A user must be able to review his payment history and if necessary update his personal information.

The main functionality are available car research and car reservation. If a car has been reserved it must be unlockable from the application.

2. *On-board Application*: must be installed on all cars. It must allow a driver to chose his/her destination and eventually enable the money saving option. The on-board display provides all necessary data to inform the user about the current fee and if the car is located in a safe area. Lastly ,the on-board system must allow the user to end his ride via on-board display.

3. *Back-End application* : must provide interfaces for registration and login processes checking also if the user input is valid. It must handle correctly search requests and handle the automatic car locking process. Lastly, it must correctly calculate and charge fees based on discounts or penalites.

3.6 Future implementation

The system must be as scalable as possible: future implementations can require a deployment in very large metropolitan cities that exceed by far the maximum amount of planned users.
An important thing to keep in mind is that the nature of vehicles could change in the distant future : not only cars , but also electric powered motorcycles could be added to the PowerEnjoy vehicle fleet.This means that the on- board software should be as expandable as possible using for instance Hierarchy Design Patterns in the vehicle type definition.

4. Specific Requirements

4.1 External Interface Requirements

This section is dedicated to illustrate the mockups of the user interfaces.

4.1.1 Mobile Application

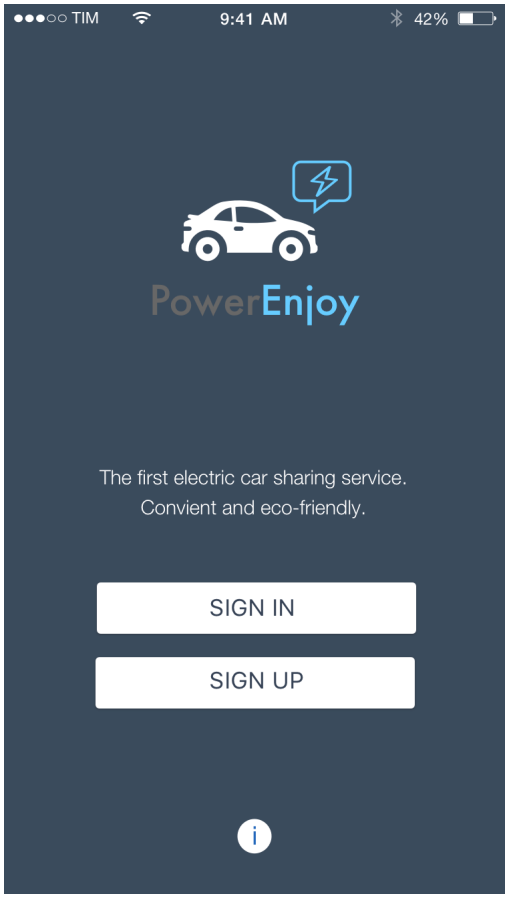


Figure 1: Confirm selection

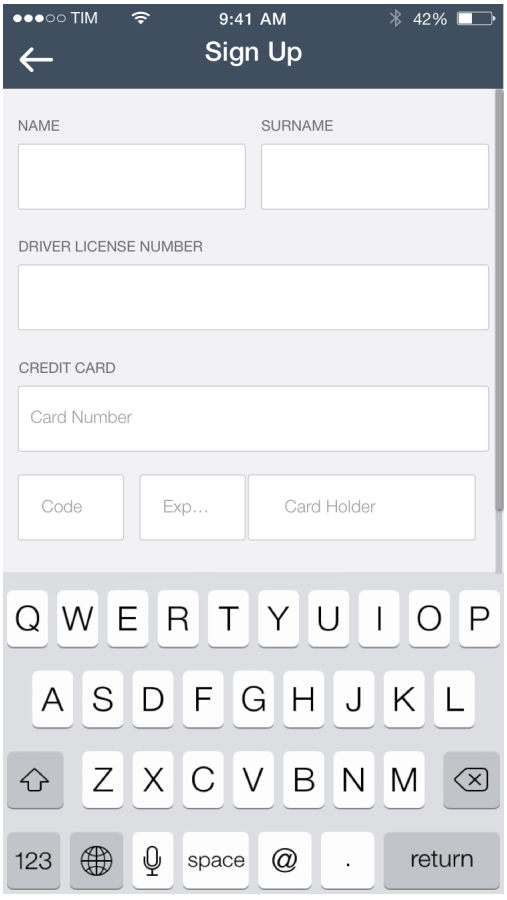


Figure 2: User sign in

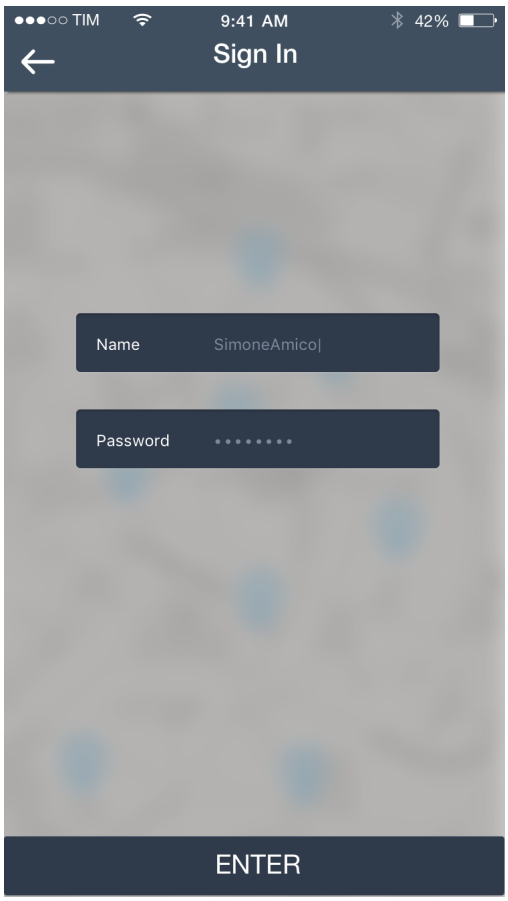


Figure 3: Sign In

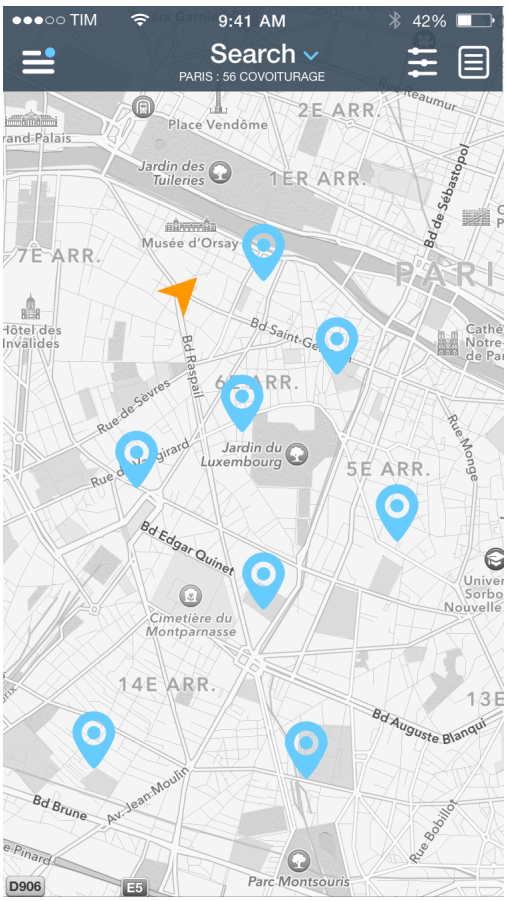


Figure 4: Search

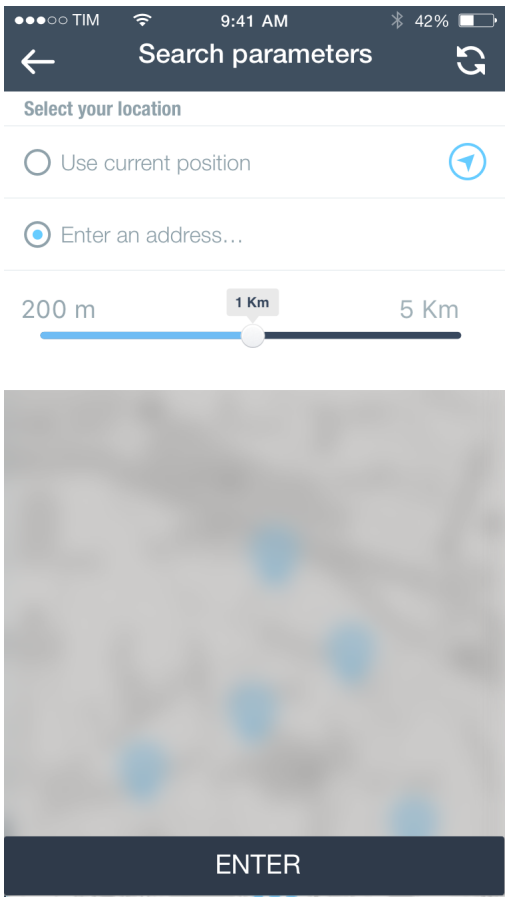


Figure 5: Address input

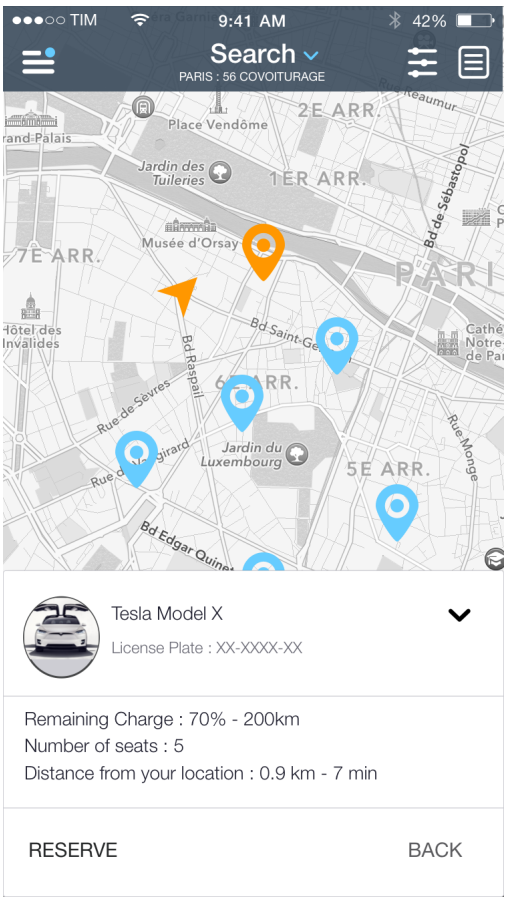


Figure 6: Car Selection

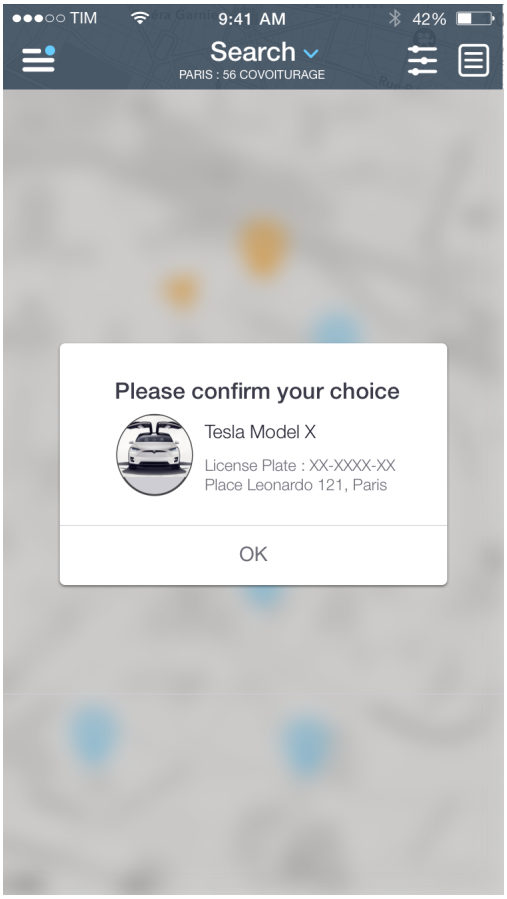


Figure 7: Confirm selection

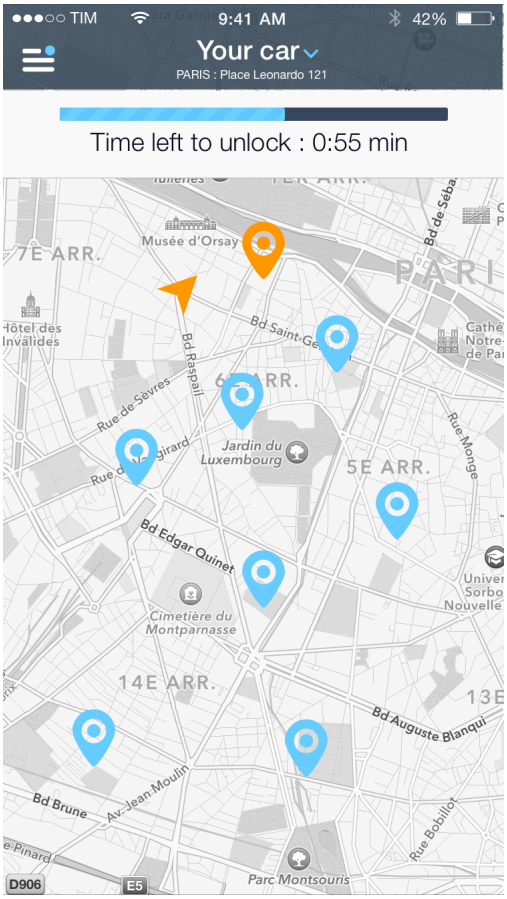


Figure 8: Time left to unlock

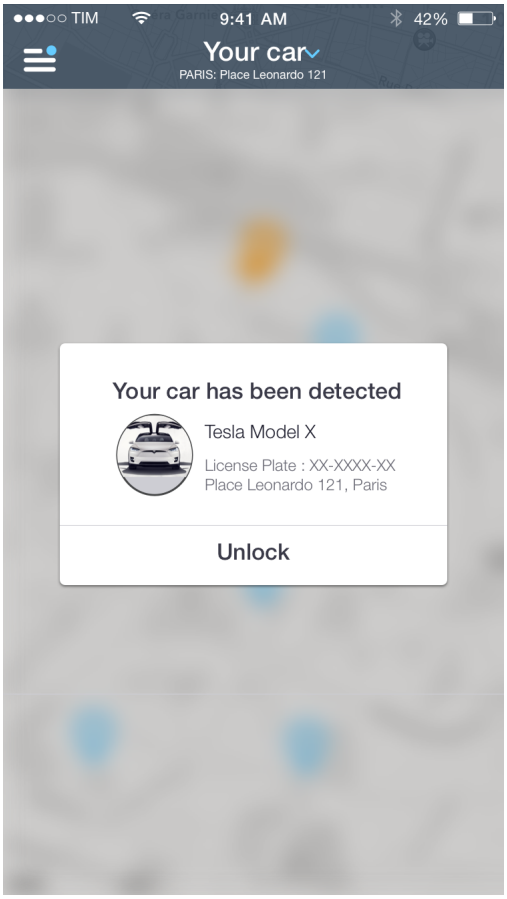


Figure 9: Unlock car

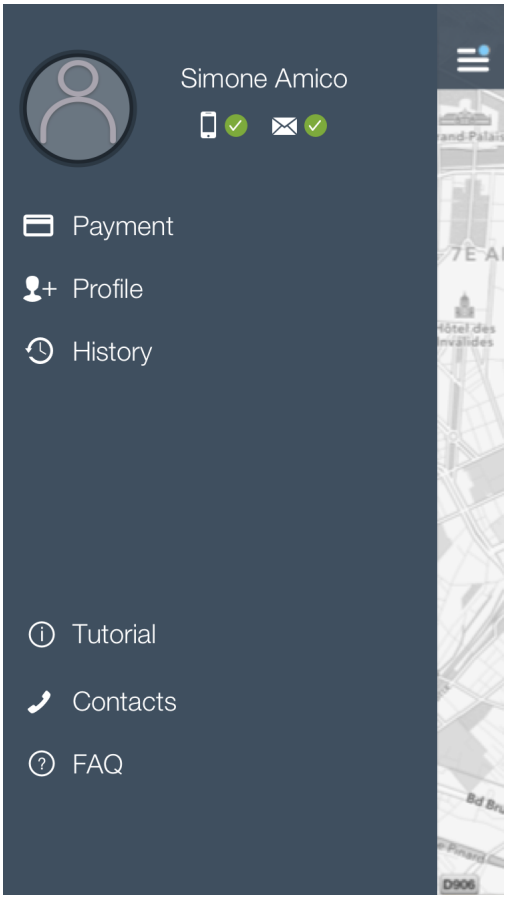


Figure 10: Menu

4.1.2 Web Application

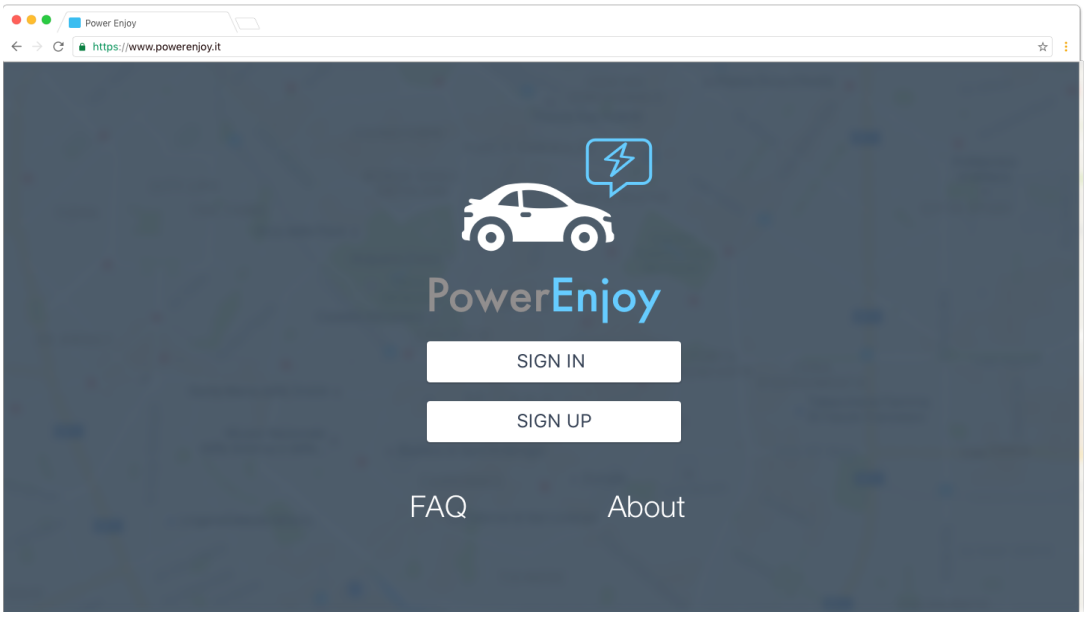


Figure 11: Homepage

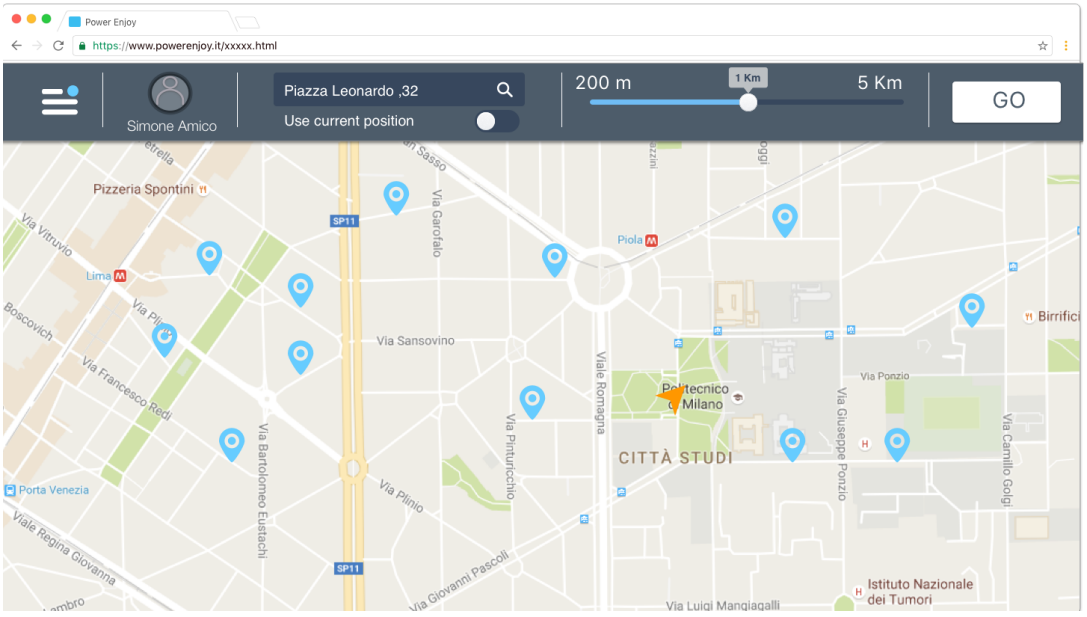


Figure 12: Main page with map an car locations

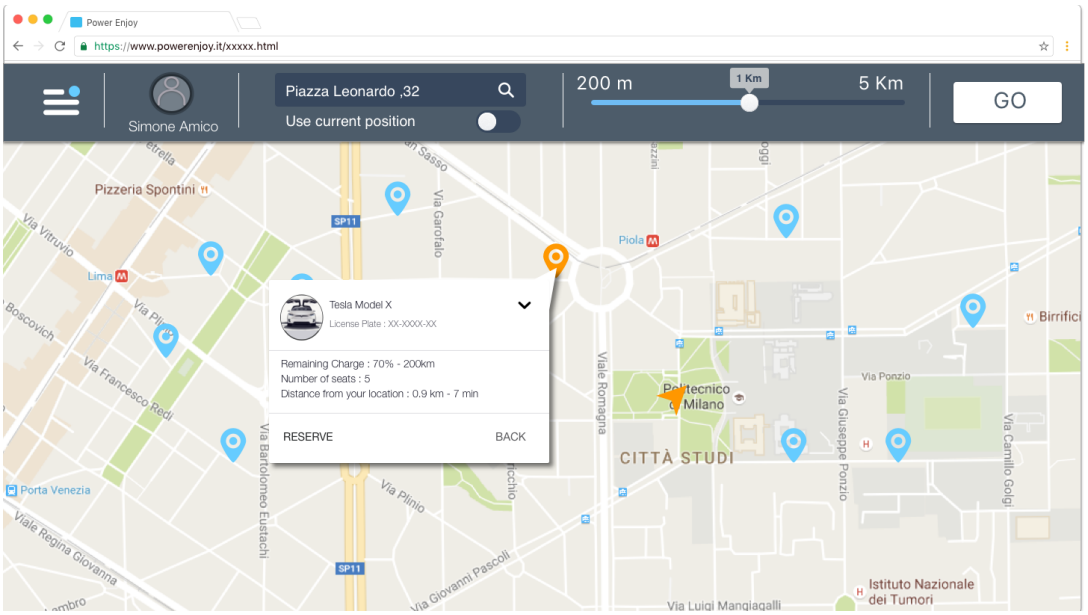


Figure 13: Main page with car selection

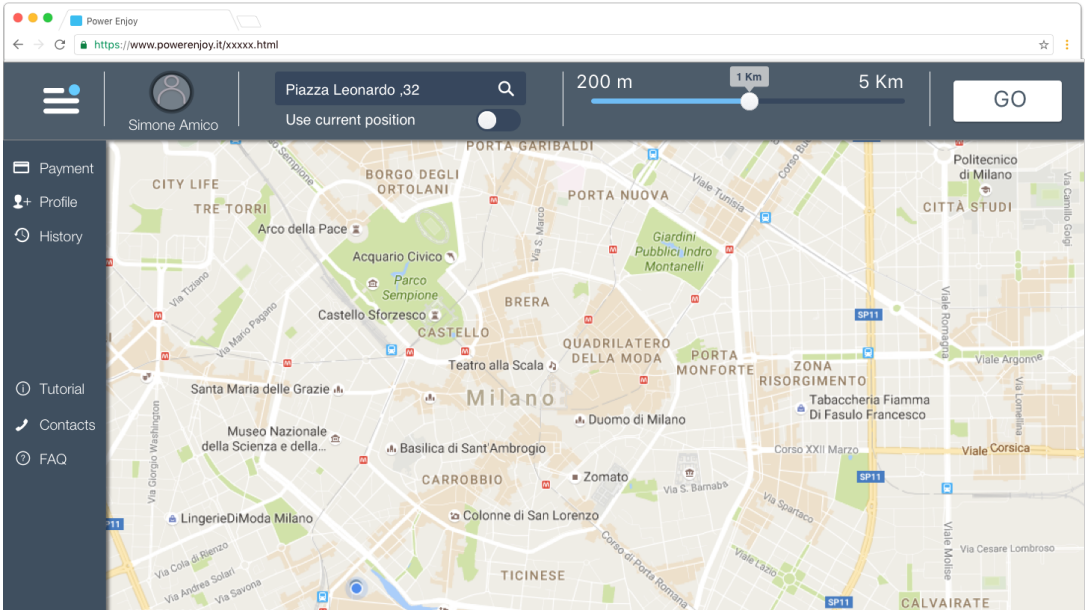


Figure 14: Main page with account info bar

4.1.3 On-Board Application (Touch)

Figure 15: Welcome screen

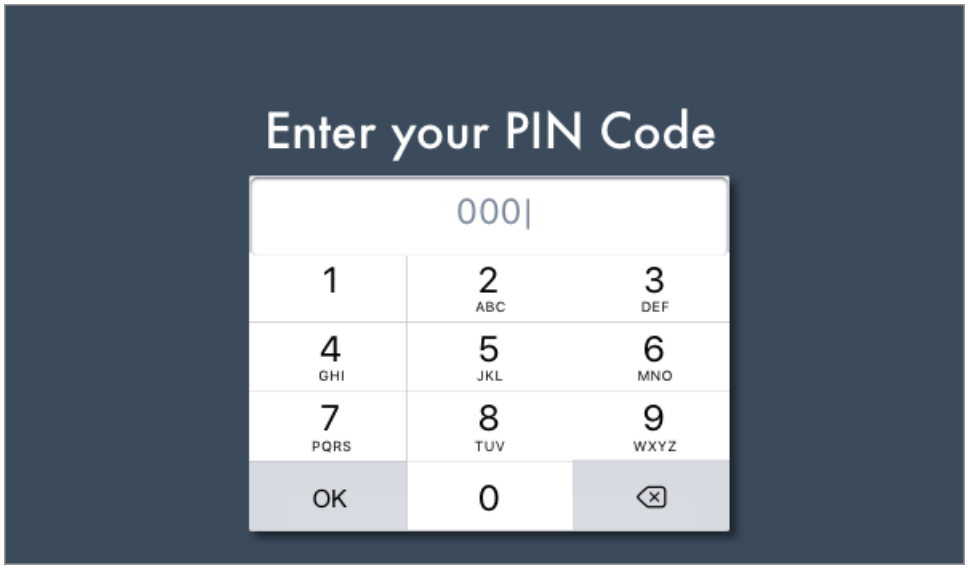


Figure 16: Pin request

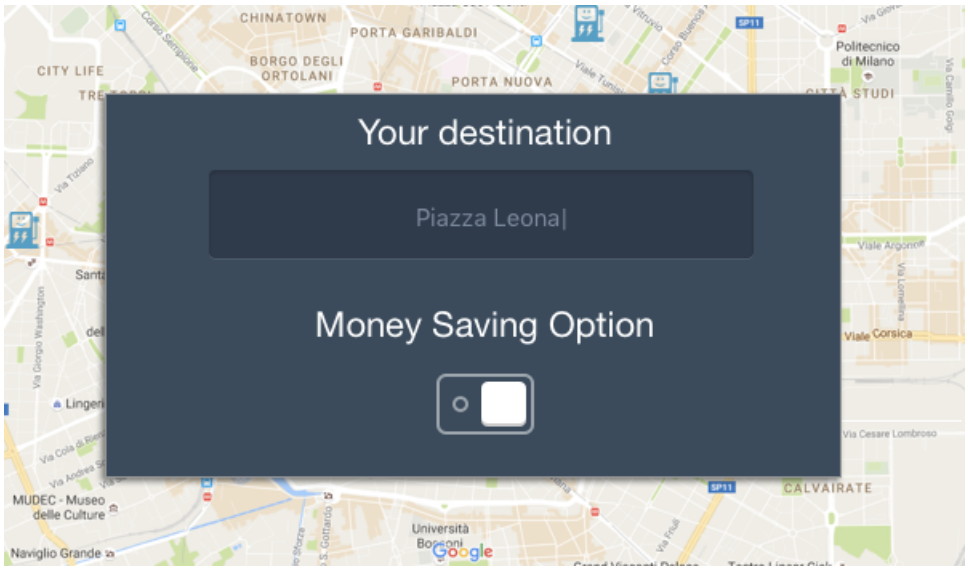


Figure 17: Destination input screen

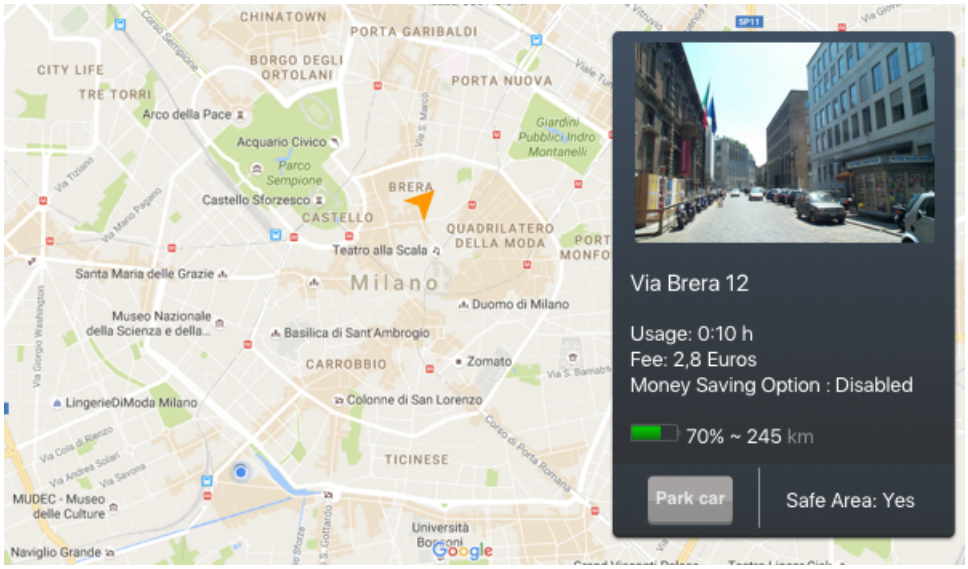


Figure 18: Main screen with disabled park car button

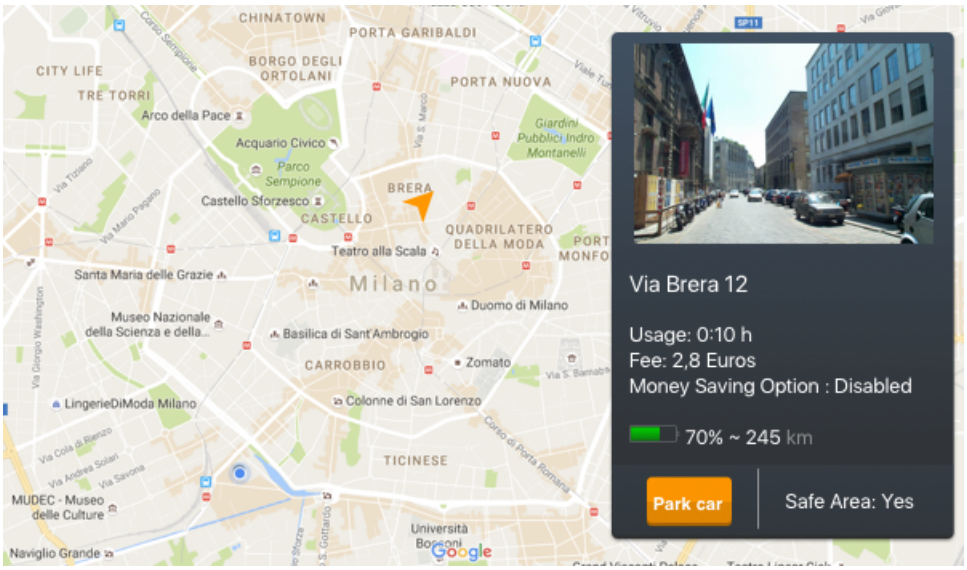


Figure 19: Main screen with enabled park car button

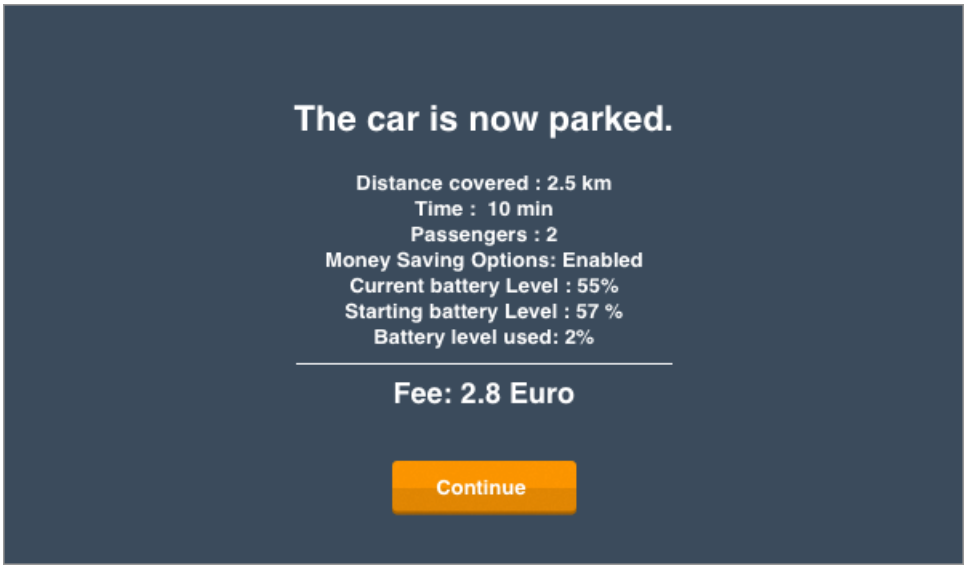


Figure 20: Car parked main screen

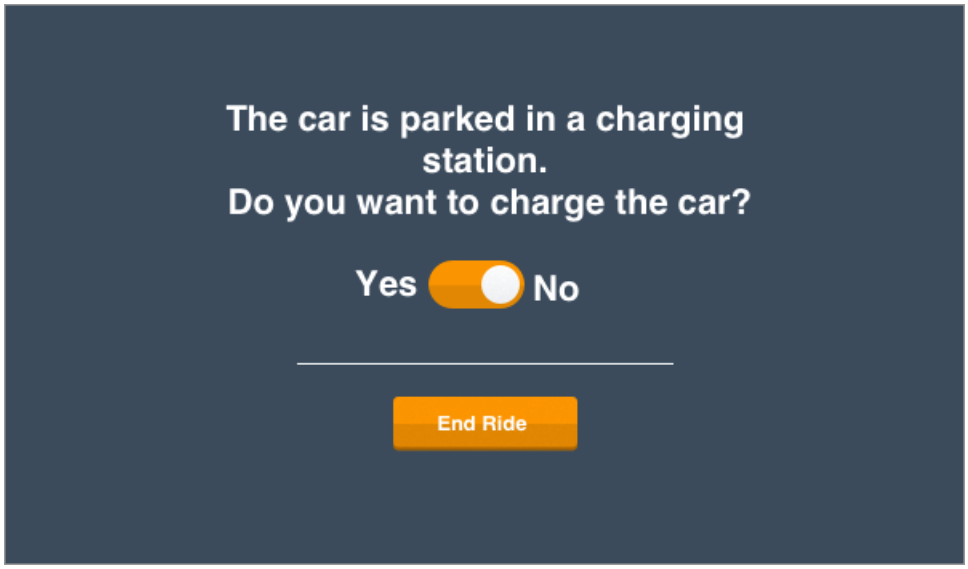


Figure 21: Charging option screen

4.2 Specific Requirements

1.Mobile and Web Application

- **[R1.1]**: Allow a guest to register entering all relevant personal data and a valid driving license whose validity will be check on the spot
- **[R1.2]**: Allow a registered user to log-in
- **[R1.3]**: Allow a registered user to update his personal information
- **[R1.4]**: Allow a registered user to search for an available vehicle given his position within a selected distance
- **[R1.5]**: Allow a registered user to search for an available vehicle given an input position within a selected distance
- **[R1.6]**: Allow a registered user to reserve a selected vehicle.
- **[R1.7]**: Allow a registered user to check his payment history
- **[R1.8]**: Allow a registered user to recover his account log-in information in a secure way
- **[R1.9]**: Allow a registered user to search for an available vehicle given an input position and a radius.
- **[R1.10]**: Allow a registered user to unlock *his/her* reserved car if the user is nearby.
- **[R1.11]**: Allow a registered user to report an issue.

2.On-Board Application

- **[R2.1]**: Enable a registered user to keep track of the distance covered, time spent using the car ,the current owed fee and the remaining battery charge.
- **[R2.3]**: Provide a method to enable the money saving option.
- **[R2.3]**: Enable the driver to end the ride if the car is parked in a safe area.
- **[R2.4]**: Enable the driver to charge the car.

3.Back-End System

- **[R3.1]**:The back-end must provide an interface for the registration of users.
- **[R3.2]**:The back-end must be able to check payment methods and driver's licenes on the spot.
- **[R3.3]**:The back-end must provide a password recovery system.
- **[R3.4]**:The back-end must handle all search requests correctly.
- **[R3.5]**:The back-end must calculate the current amount to pay.
- **[R3.6]**:The back-end must handle discounts/penalties and recalculate the fee accordingly.
- **[R3.7]**:The back-end must handle car status changes.
- **[R3.8]**:The back-end must provide an interface to allow a support operator to perform all basics CRUD operations
- **[R3.9]**:The back-end must handle all ride requests.
- **[R3.10]**:The back-end must handle all charging requests.

4.3 Scenarios

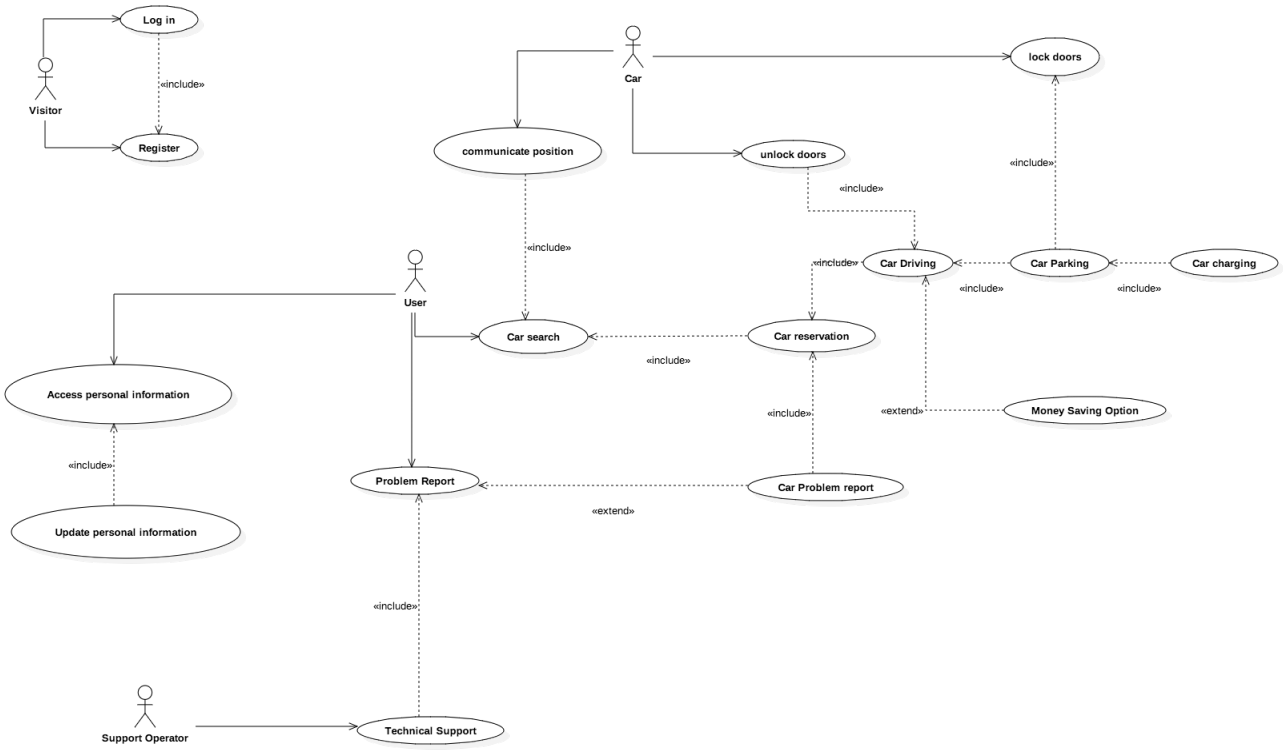


Figure 22: General use case scenrio

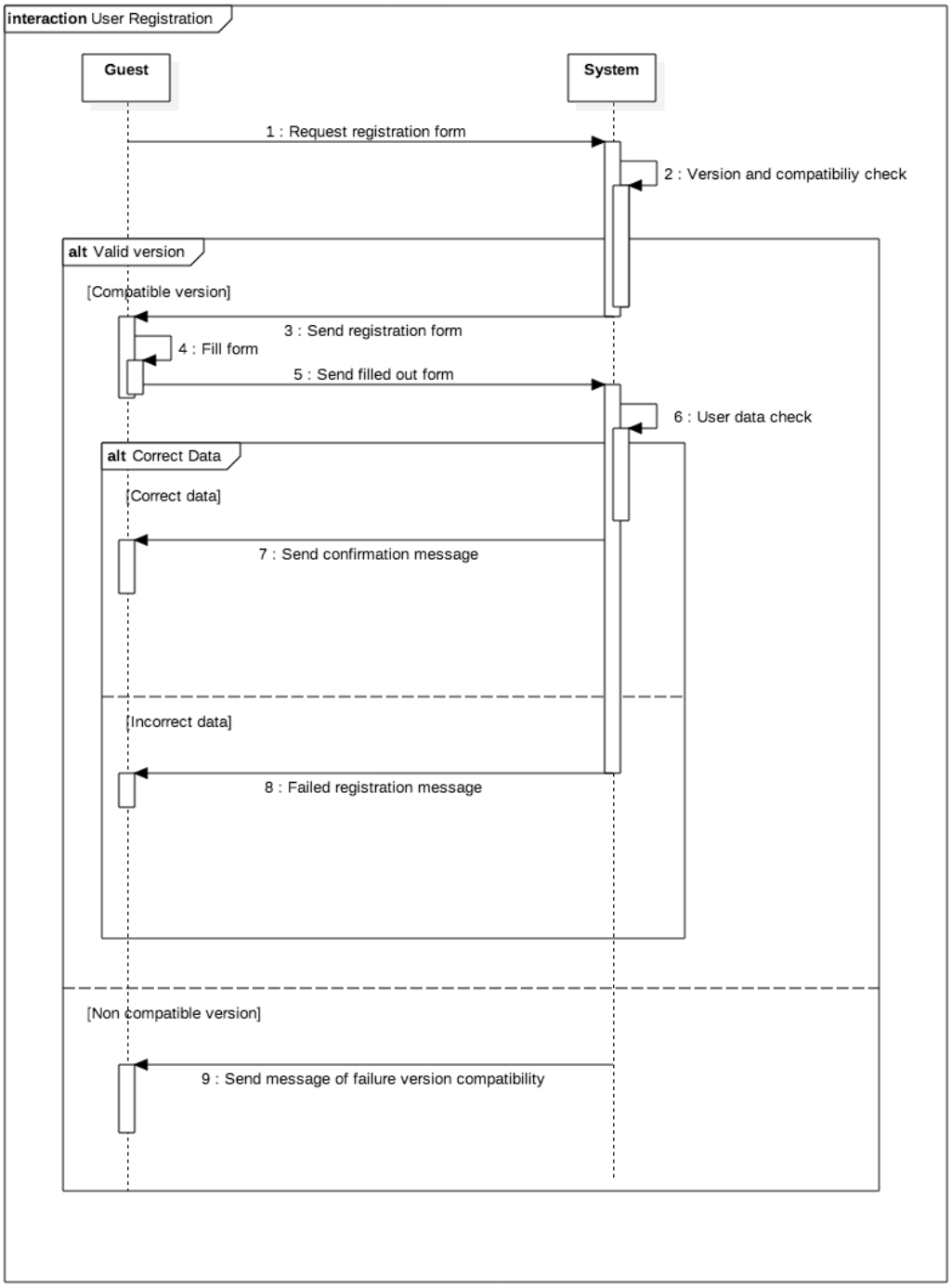
4.3.1 Scenario 1:User sign-up

Maria is going out with friends for dinner .She doesn't own a car but has to travel to the other side of the city. She knows that a taxi ride would be expensive and other forms of public transport are packed with people during rush hour. She decides to download the PowerEnjoy application. Once the download is completed she fill out the necessary registration informations providing a valid license and payment method and is ready to go and enjoy the evening with her friends.

Actor	Guest
Goal	User registration
Input Condition	NULL
Event Flow	1.Open the app/webpage 2.Click on the "sign up" button. 3.Fill out the required fields. 4.Press confirmation button 5.The system checks the input and sends out a confirmation message.
Output Condition	Guest end the registration process and is now a user so he/she can use his/her credentials to log into the system and use the service.
Exception	The user is already registered. The guest data are invalid (invalid payment method or driver's license) No internet connection or server malfunction.

Table 1: User Registration

Figure 1: User registration sequence diagram



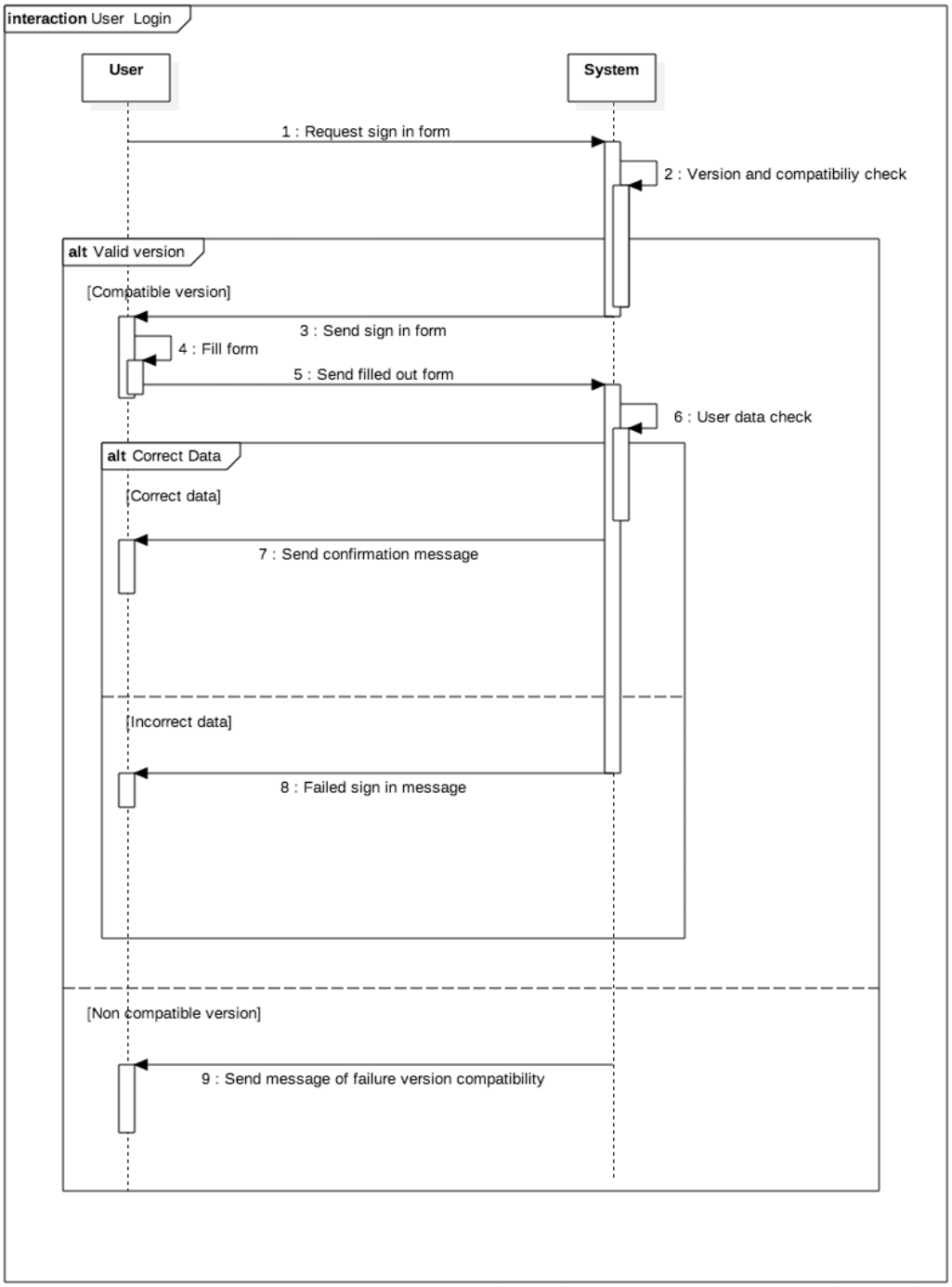
4.3.2 Scenario 2: Log in

Angelo woke up late for his exam and has to hurry up. The bus stop is too far away from his University so he has to find an alternative. So he logs into the system to look immediately for a car.

Actor	User
Goal	User Login
Input Condition	User is registered and connected to the internet Back-end system is working
Event Flow	1.Open the app/webpage 2.Fill out fields with credentials 3.Click on the ”sign in” button. 4.Back-end applications checks input data.
Output Condition	User is successfully logged in and is able to use the application.
Exception	The credentials are invalid No internet connection or server malfunction.

Table 2: User log in

Figure 2: User login sequence diagram



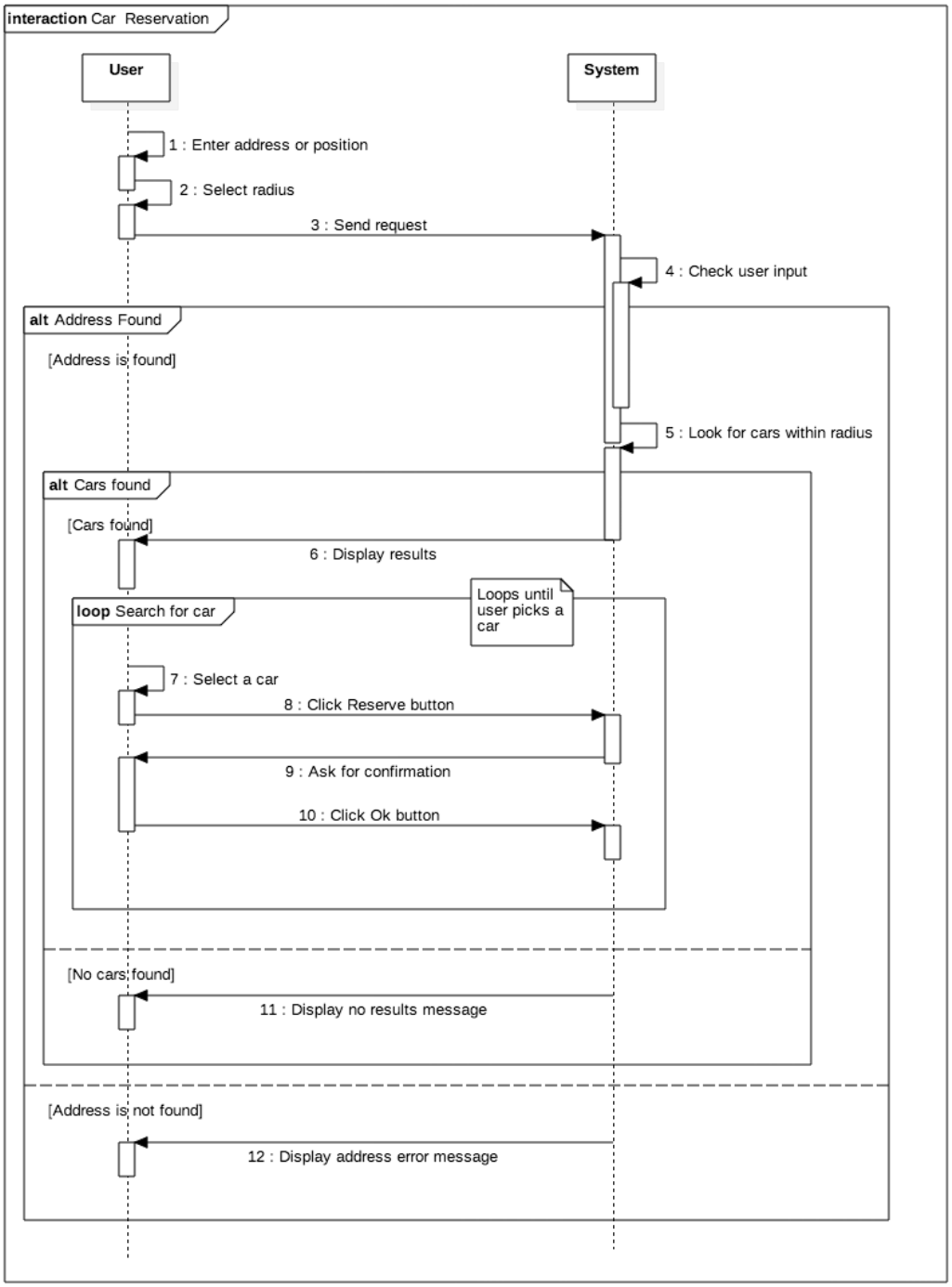
4.3.3 Scenario 3: Reserve a car

Francesca has to take the train. The station is far away so she has to drive but she doesn't want to leave her car parked in an expensive car lot. She decides to search and reserve the nearest PowerEnjoy.

Actor	User
Goal	Car reservation
Input Condition	User is logged in.
Event Flow	1.Enter an address (or current position) in the 'Search' bar 2.Select desired radius 3.Click on the "Go" button. 4.Back-end applications checks for matching cars in the area. 5.Select among displayed cars available. 6.Click "Reserve" button to reserve a car. 7.Click on 'OK' when asked for confirmation
Output Condition	User has successfully booked a car in and is able to unlock it.
Exception	Invalid address No available cars within radius

Table 3: Car reservation

Figure 3: Car reservation sequence diagram



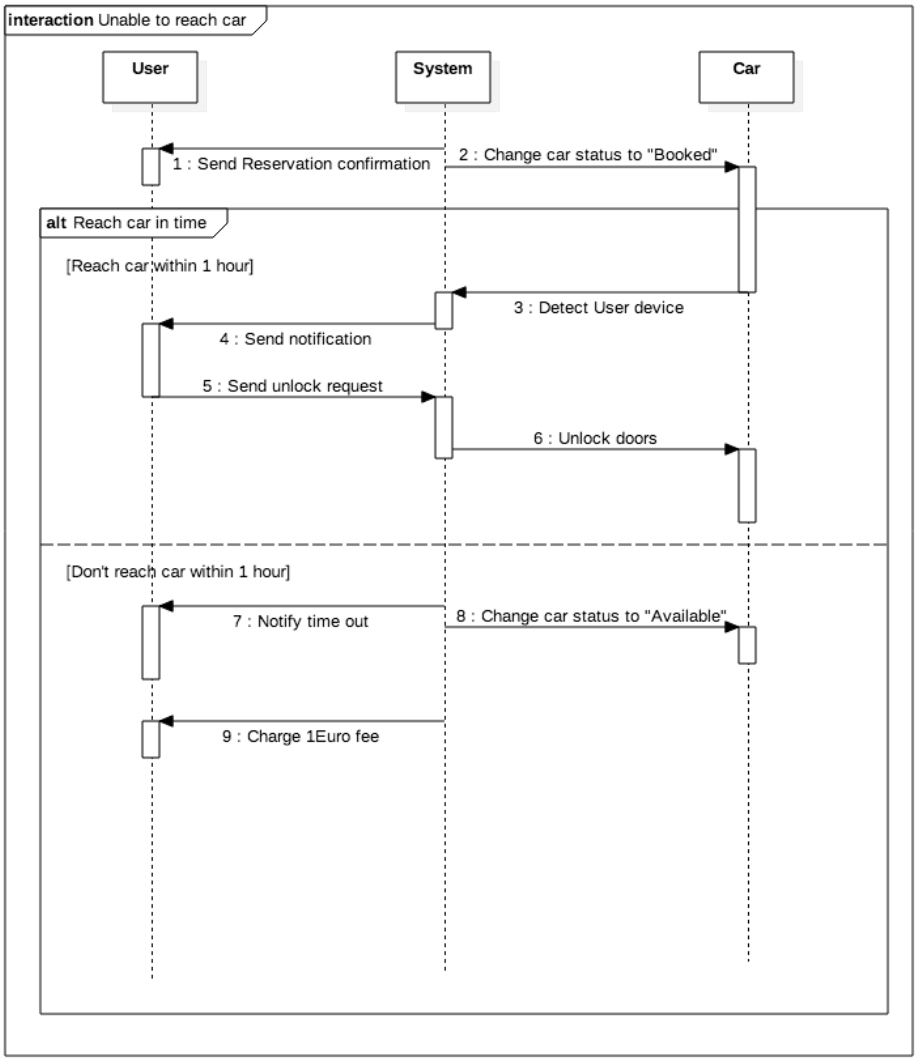
4.3.4 Scenario 4: Unable to reach car in time

Giacomo wants to meet his friends at the cinema. He reserves a car near him but as he steps out of his front door he meets an old friend who offers him a lift.Giacomo never reaches the car he reserved so he has to pay a fee of 1 Euro for not unlocking the car within one hour.

Actor	User
Goal	Pay a 1Euro fee and list car as available again
Input Condition	User has booked a car. User has not unlocked the car yet.
Event Flow	1.User has reserved a car 2.The user doesn't unlock the car within 1 Hour 3.Back-end charges user 1Euro fee. 4.Back-end lists car status as available.
Output Condition	The car is now available again and the user has been charged 1 Euro.
Exception	-

Table 4: Unable to reach car in time

Figure 4: Unable to reach car in time sequence diagram



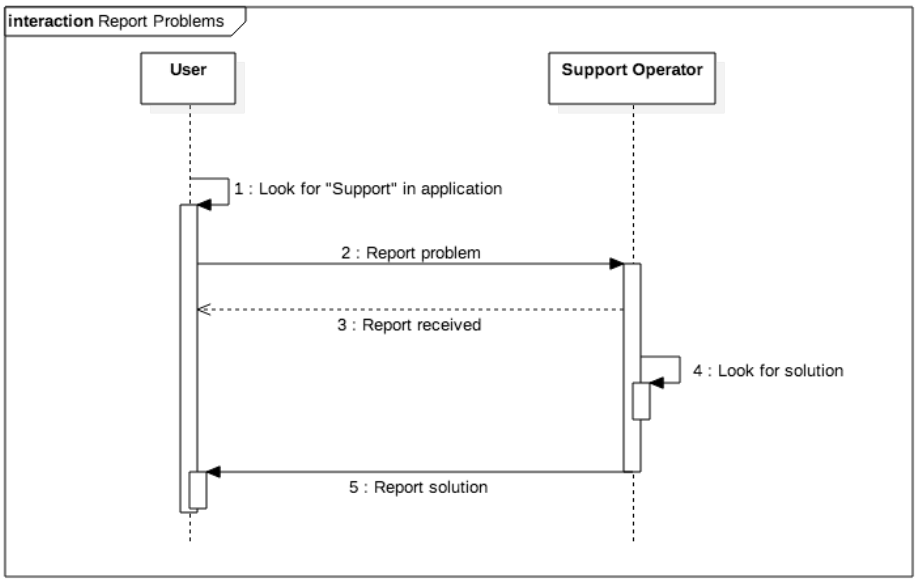
4.3.5 Scenario 5: Report a problem

Gianni reserved a car but once he reached it he notices that the car has a flat tire. So he calls the Support Operator for help.

Actor	User and Operator
Goal	Solve a user or car related problem
Input Condition	NULL
Event Flow	1.Open mobile/web application. 2.Search for 'Support Operator' in the 'Contact' section. 3.Call.
Output Condition	NULL
Exception	User is unable to call the Support Operator

Table 5: User support

Figure 5: Report a problem



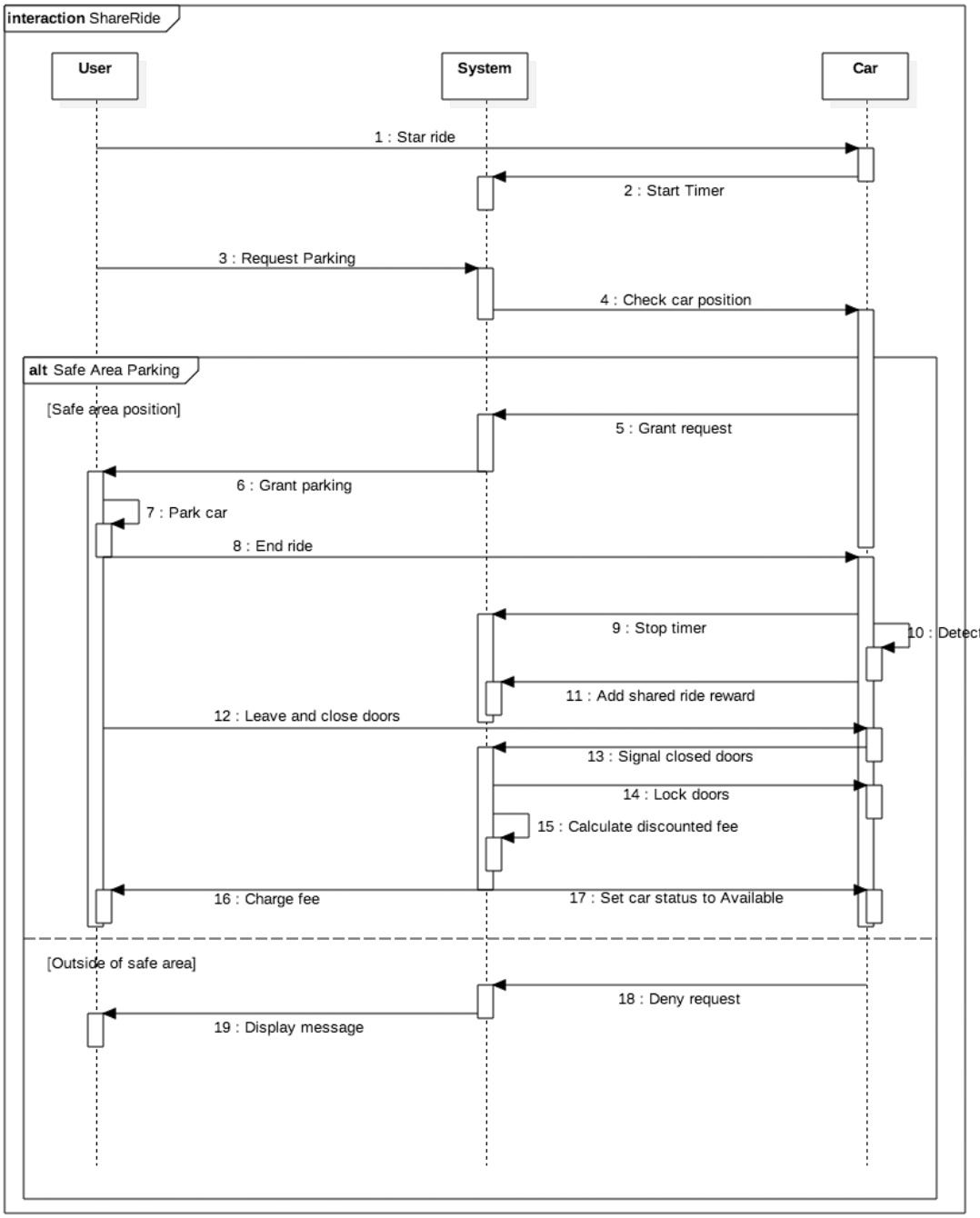
4.3.6 Scenario 6: Share ride with passengers

Lucia's parents are in town and she has to pick them up a the airport. She reserves a PowerEnjoy car as she knows that she will get a discount of 10% on the ride's fee if she takes two passengers with her.

Actor	User
Goal	Reward user with 10% discount
Input Condition	User has reserved a car.
Event Flow	1.User enters the car with a least two other passengers. 2.System recognizes the passengers through sensors. 3.Back-end charges 10% less on total fee.
Output Condition	NULL
Exception	Passengers not recognized (too light)

Table 6: Share ride

Figure 6: Shared ride sequence diagram



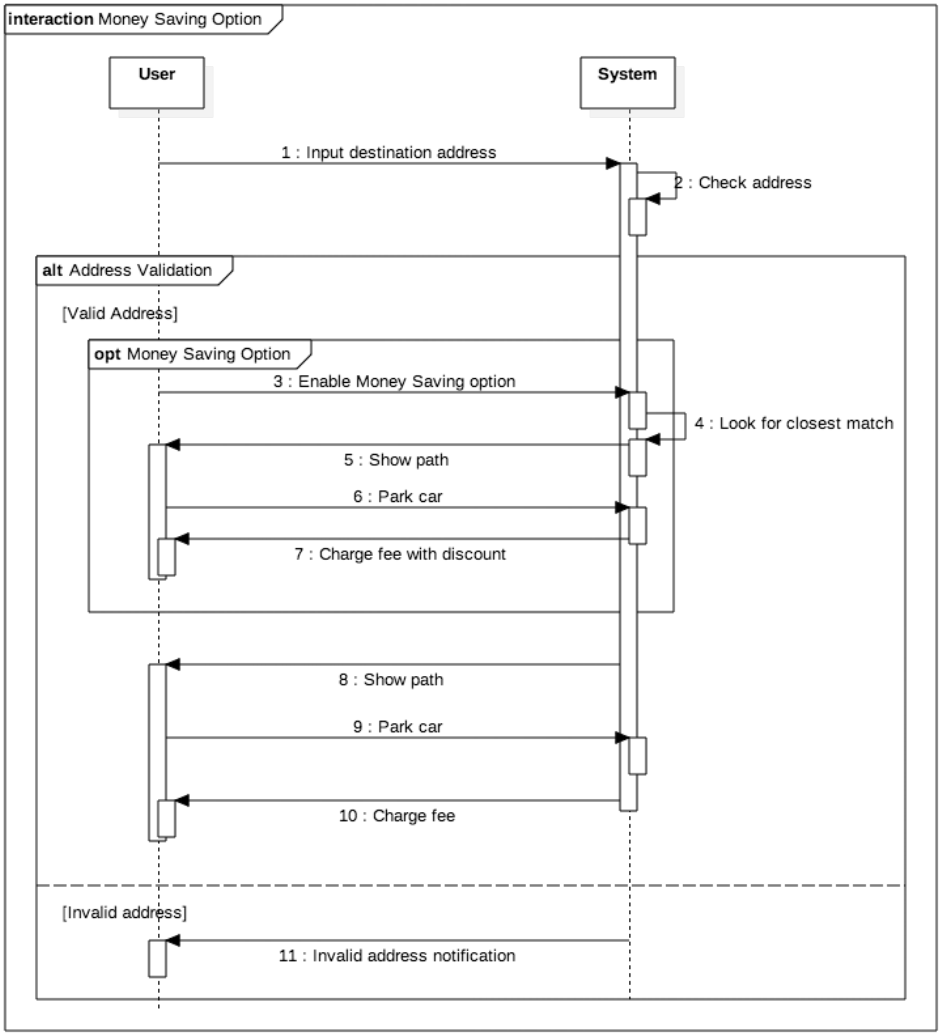
4.3.7 Scenario 7: Drive with Money Saving Option

Flavio wants to attend his brother football match. He decides to use the money saving option which automatically shows him the optimal parking solution closest to his destination. In return he gets a discount on his ride's fee.

Actor	User
Goal	Reward customer with a discount.
Input Condition	User has reserved a car. User has unlocked the car.
Event Flow	1.User picks up car. 2.User inputs the desired destination into the on-board system. 3.User toggles the money saving option. 4.User parks car in recommended area 5.Back-end system applies discount on the fee
Output Condition	-
Exception	User parks outside of recommended area No free parking spots in recommended area

Table 7: Money Saving Option

Figure 7: Money saving option sequence diagram



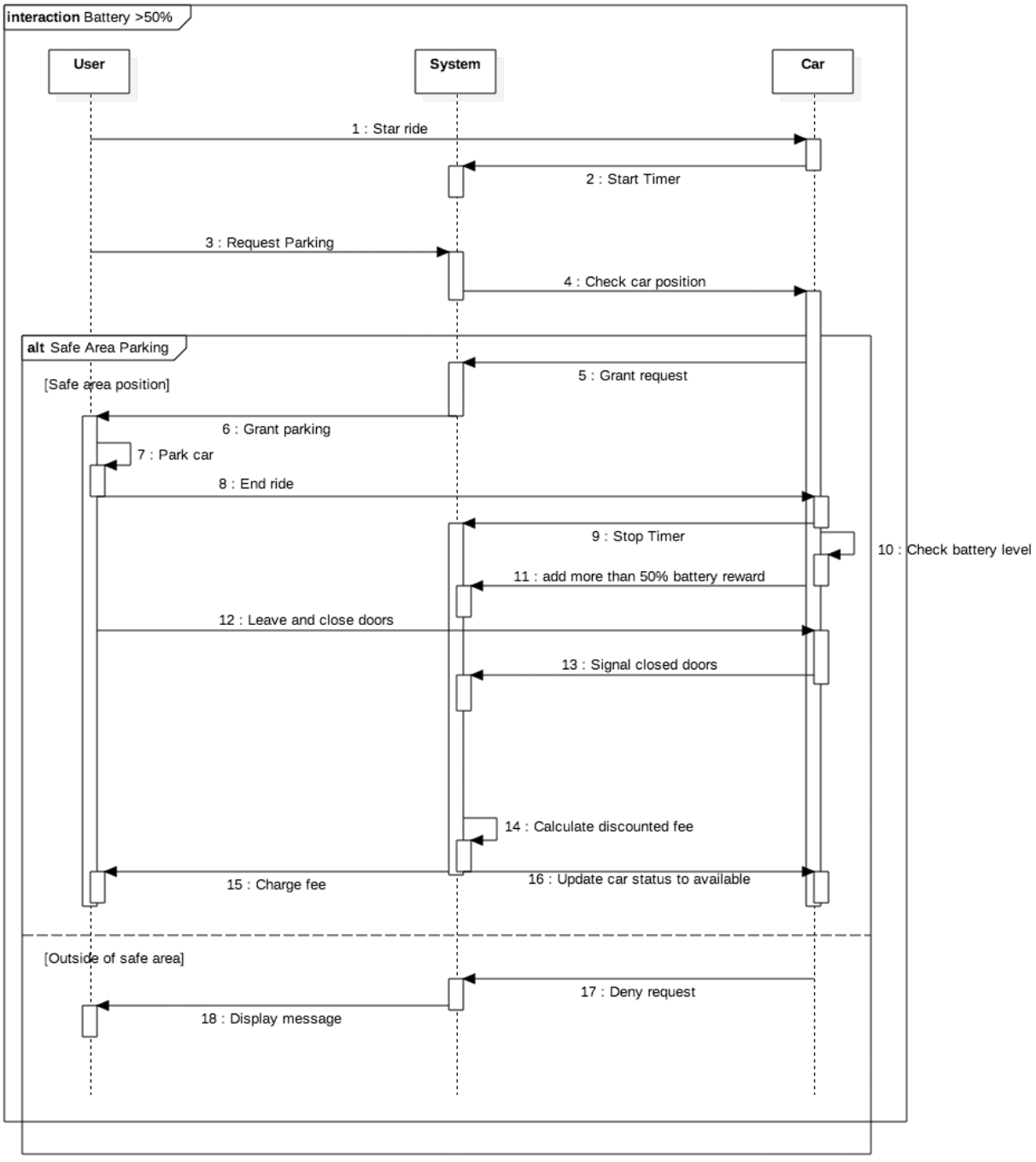
4.3.8 Scenario 8: Leaving a car with battery level at >50%

After a long day working , Edoardo is tired but has to visit his ill mother who lives not too far away. He reserves a car and takes a short ride to his destination. He leaves the car with a battery level over 50% so he's entitled to get a discount of 20% on his fee.

Actor	User , Car
Goal	Reward customer with 20% discount.
Input Condition	User has reserved a car. Car has an initial battery level >50%
Event Flow	1.User picks up car. 2.User drives to his desired destination. 3.User correctly parks car. 4.System detects a remaining battery level of a least 50%. 5.System applies a 20% discount on the user's fee
Output Condition	Car has a battery level of at least 50%
Exception	Car is involved in an accident/problem

Table 8: 20% discount ride

Figure 8: Park car with battery level >50% sequence diagram



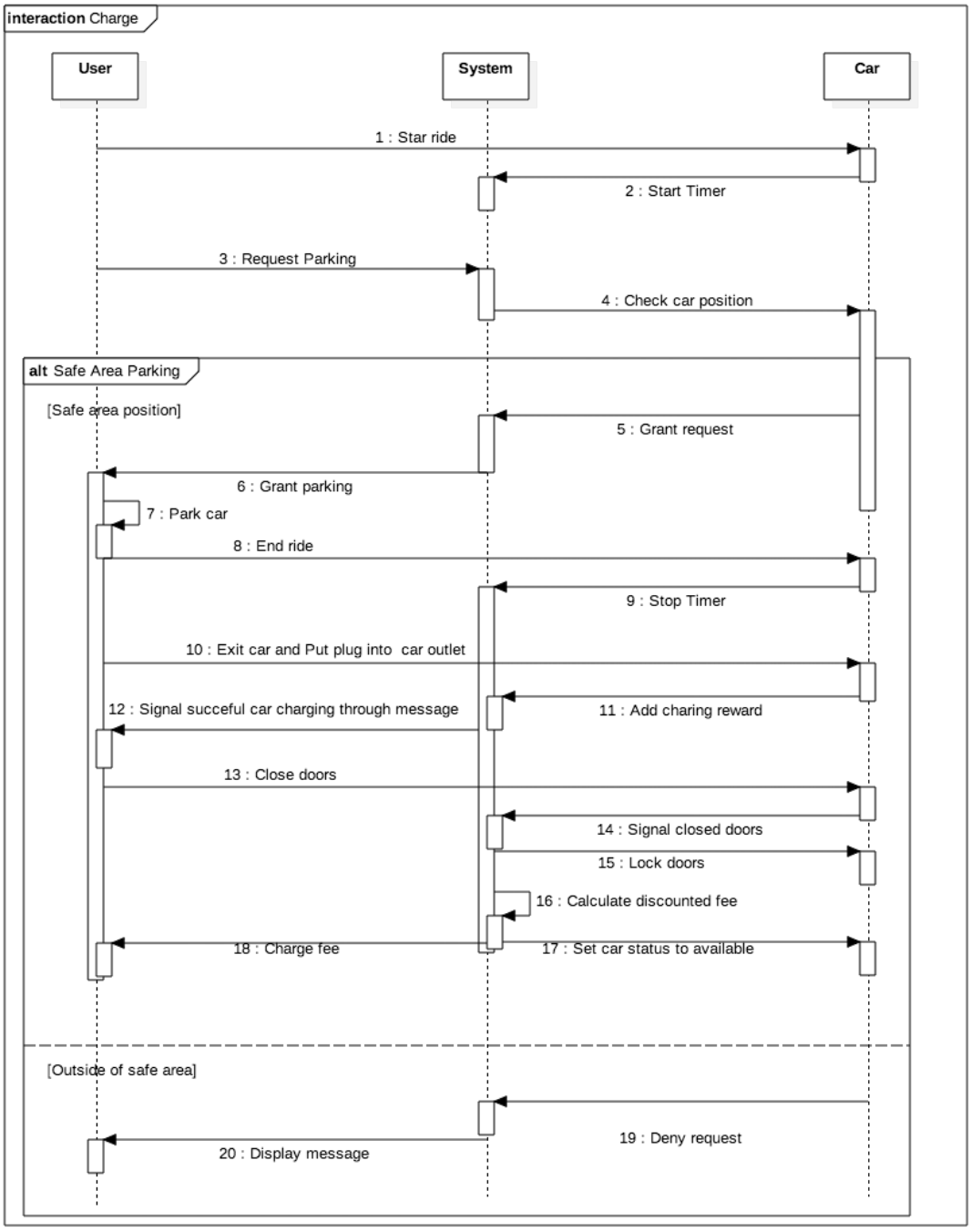
4.3.9 Scenario 9: Charge car

Daniele is driving a PowerEnjoy car to get to the football stadium. Near the stadium there is a PowerEnjoy charging station. He decides to park the car there and to plug it into the charging station : as a reward PowerEnjoy applies a discount of 30% on his fee.

Actor	User,Car
Goal	Reward customer with 30% discount.
Input Condition	User has reserved a car. User has unlocked the car.
Event Flow	1.User drives car to a Charging Station. 2.User correctly parks car and gets off. 3.User plugs the car into the power grid 4.The car's battery level starts to charge 5.Back-end applies 30% discount on the fee.
Output Condition	Car is charging.
Exception	Car is plugged in incorrectly. Plug or power grid is out of service.

Table 9: Car charging

Figure 9: Car charging sequence diagram



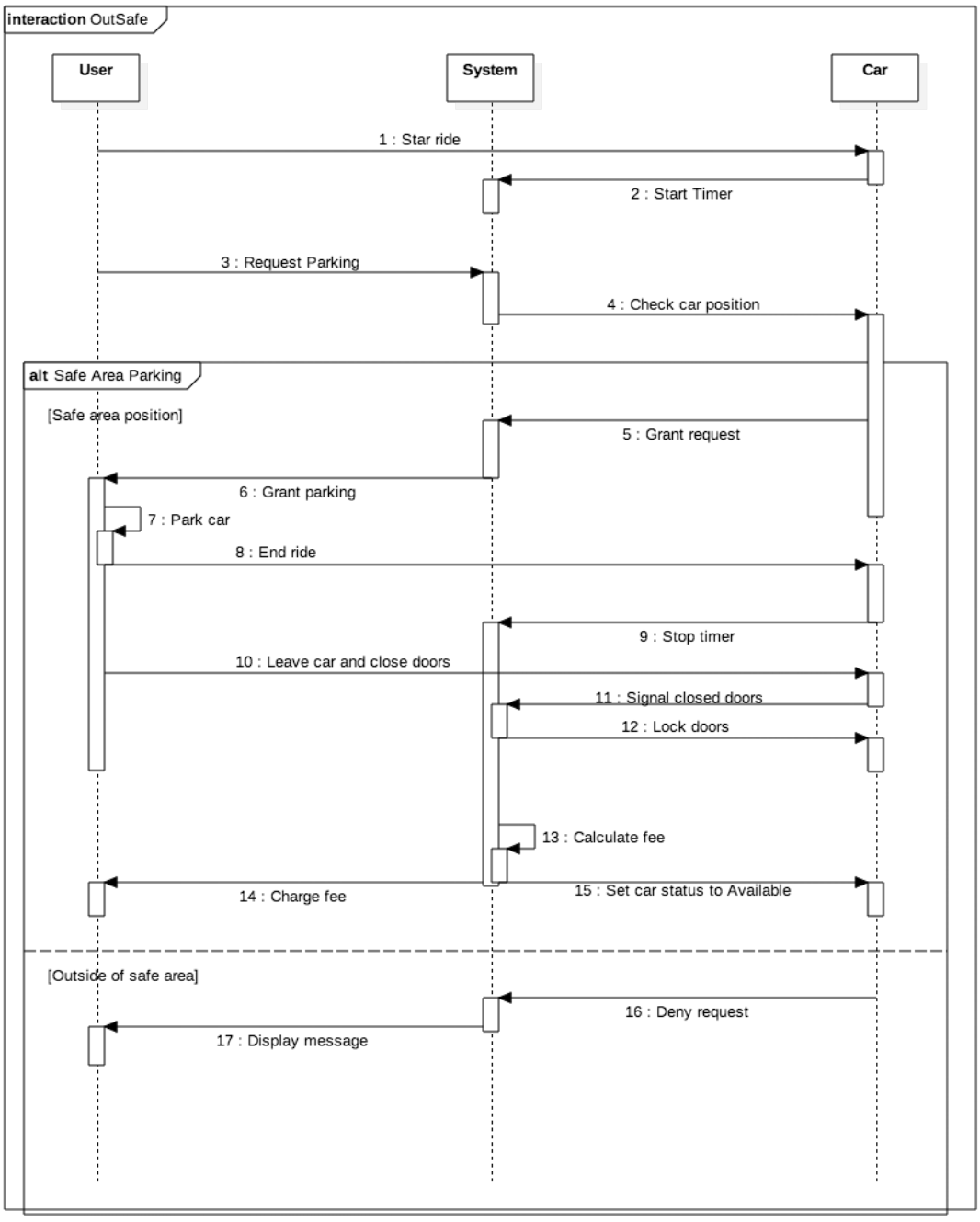
4.3.10 Scenario 10: Out of safe area parking

Andrea has to meet his girlfriend's parents in the suburbs. He decides to take a PowerEnjoy and park on a private driveway. Unfortunately the parking spot is marked as *unsafe* so the onboard display notifies him about the situation. This way Andrea parks the car in an accessible position for everyone.

Actor	User
Goal	Notify incorrect parking position
Input Condition	User has reserved a car.
Event Flow	1.User drives car 2.User decides to park car out of safe area 3.On-board system detects wrong parking position. 4.On-board display notifies user about bad parking position 5.User is now aware
Output Condition	User is notified of incorrect parking position through onboard display.
Exception	-

Table 10: Out of safe area parking

Figure 10: Out of safe area parking sequence diagram



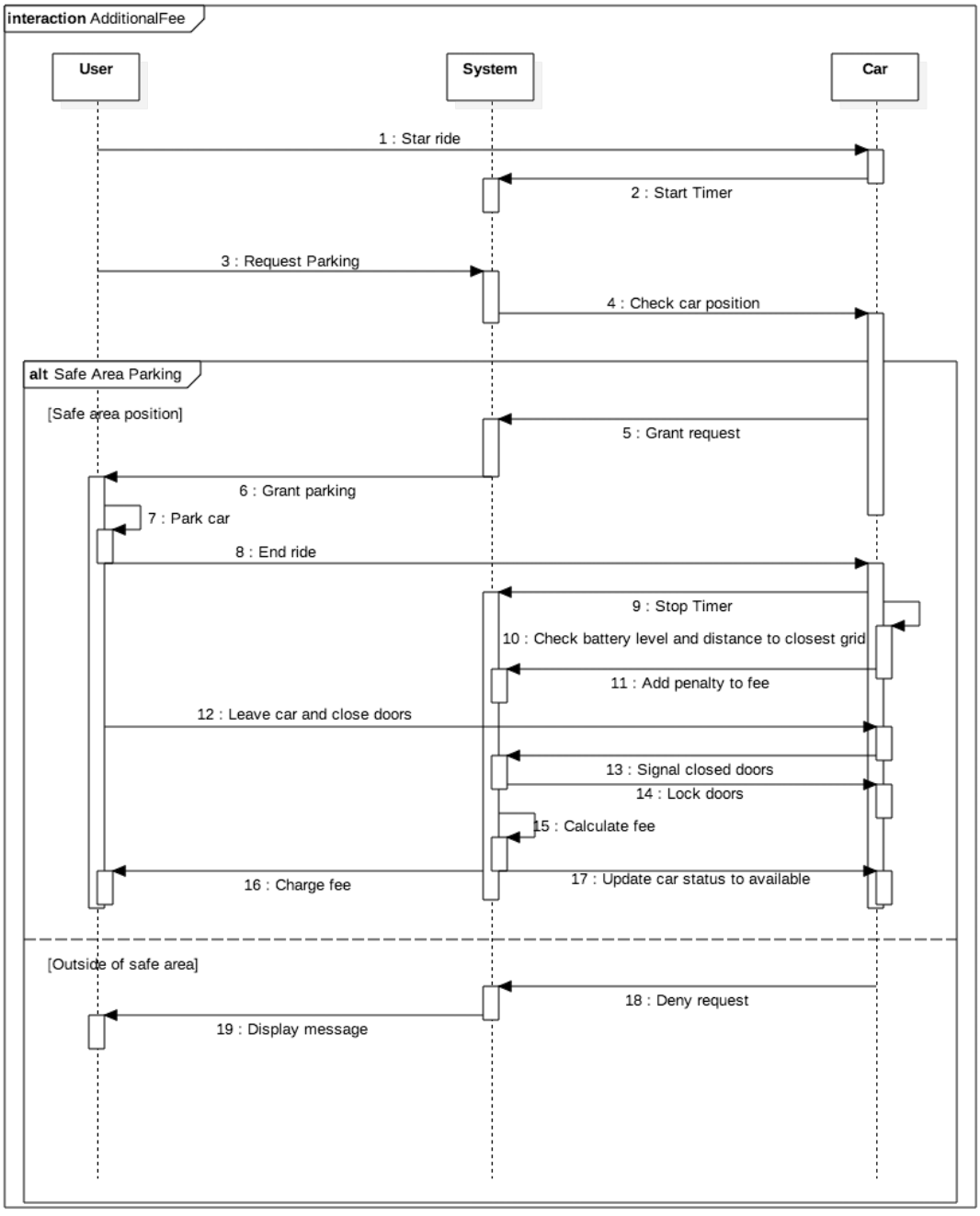
4.3.11 Scenario 11: Additional fee charged

Alessandro is running late for his date and decides the park his reserved car as close as possible. Unfortunately he parks the car far away from the nearest power grid : PowerEnjoy has to take care of this situation , so Alessandro is charged with an additional 30% on his ride's fee.

Actor	User
Goal	Add 30% on user's last ride
Input Condition	User has reserved a car.
Event Flow	1.User drives car 2.User parks car in a safe area. 3.Back-end system checks if car has a battery level of more than 20% 4.Back-end checks if the car's distance to the nearest power grid is greater than 3 Km. 5. If 3. or 4. are true, than the system adds 30% on the user's fee.
Output Condition	-
Exception	Car is involved in an accident/problem

Table 11: Bad parking position

Figure 11: Parking with additional fee sequence diagram



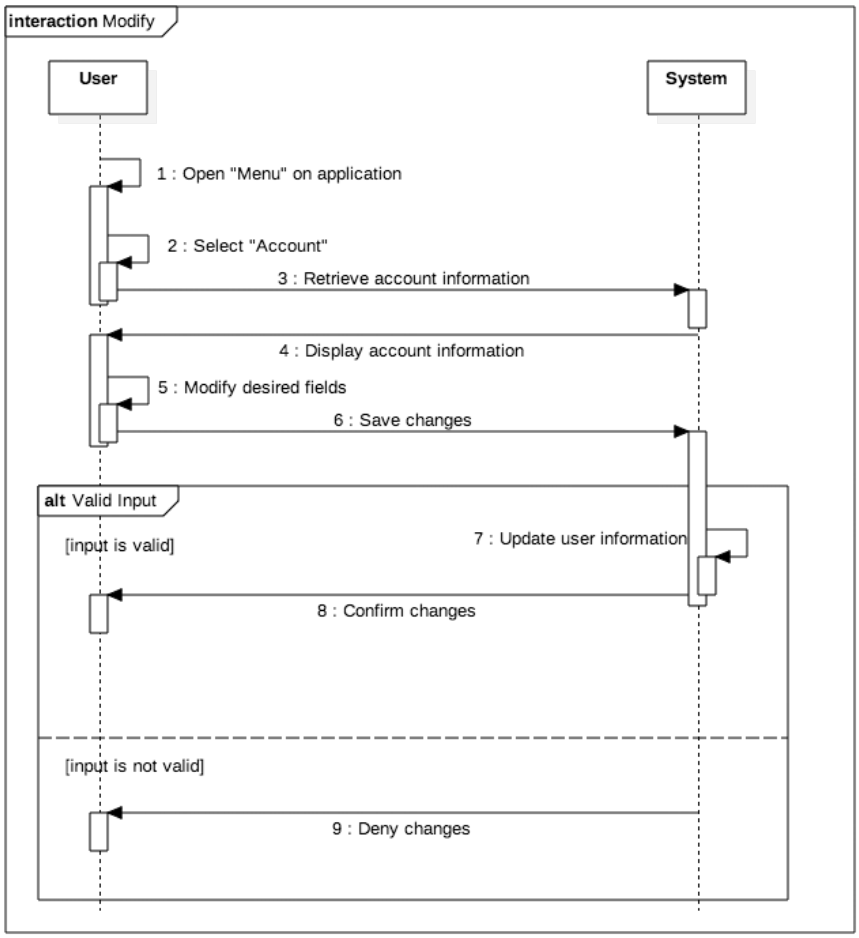
4.3.12 Scenario 12: User information update

Leonardo's credit card is expiring next week. As PowerEnjoy requires a valid payment method , Leonardo decides to update his payment method.

Actor	User
Goal	Modify user information
Input Condition	User is signed in.
Event Flow	1.User is logged into the system (web/mobile). 2.User clicks on the menu button to open the 'Account' menu. 3. User selects desired option he wants to changes 4.User clicks on confirmation button 5.Back-end checks user input and saves all changes.
Output Condition	User data is up to date.
Exception	Incorrect data input. New data same as previous. Back-end server problems.

Table 12: Modify user information

Figure 12: Modify user data sequence diagram



4.4 State charts

This section shows the user action event flow and the car ride event flow with the help of two state charts.

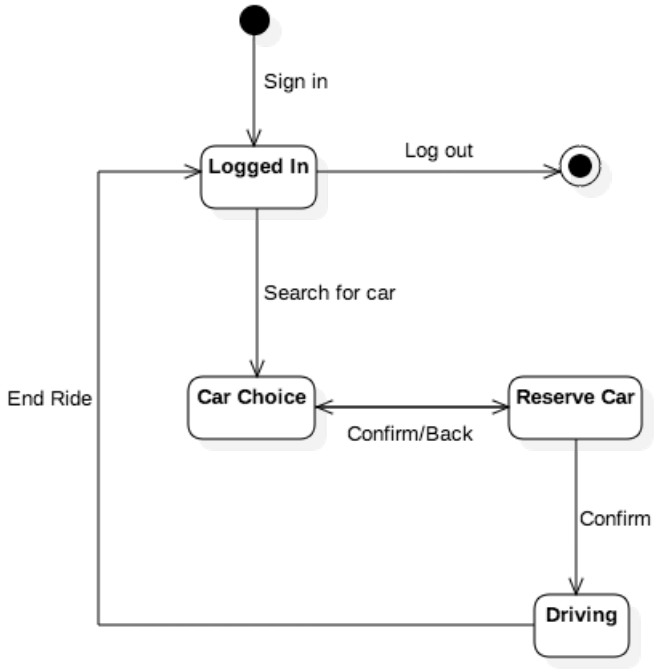


Figure 13: User state chart

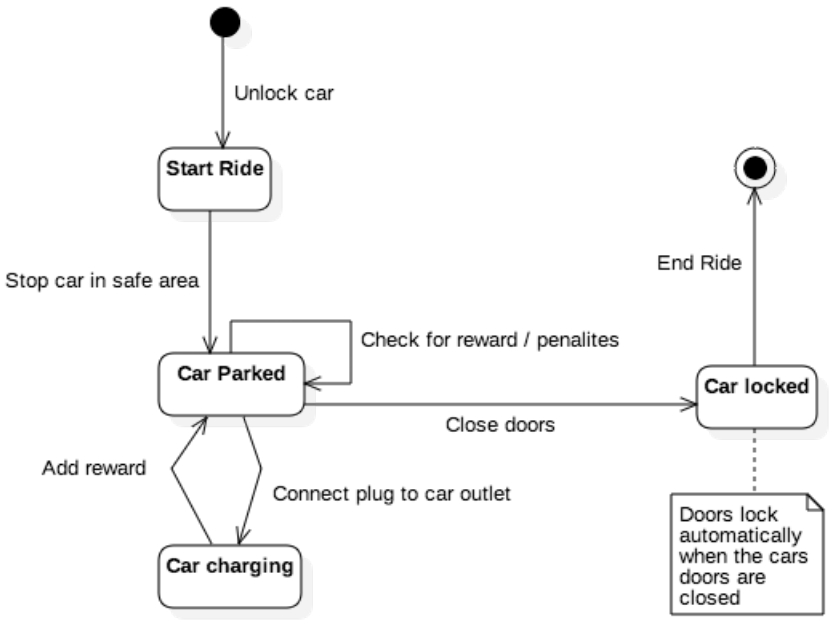


Figure 14: Ride state chart

4.5 Alloy

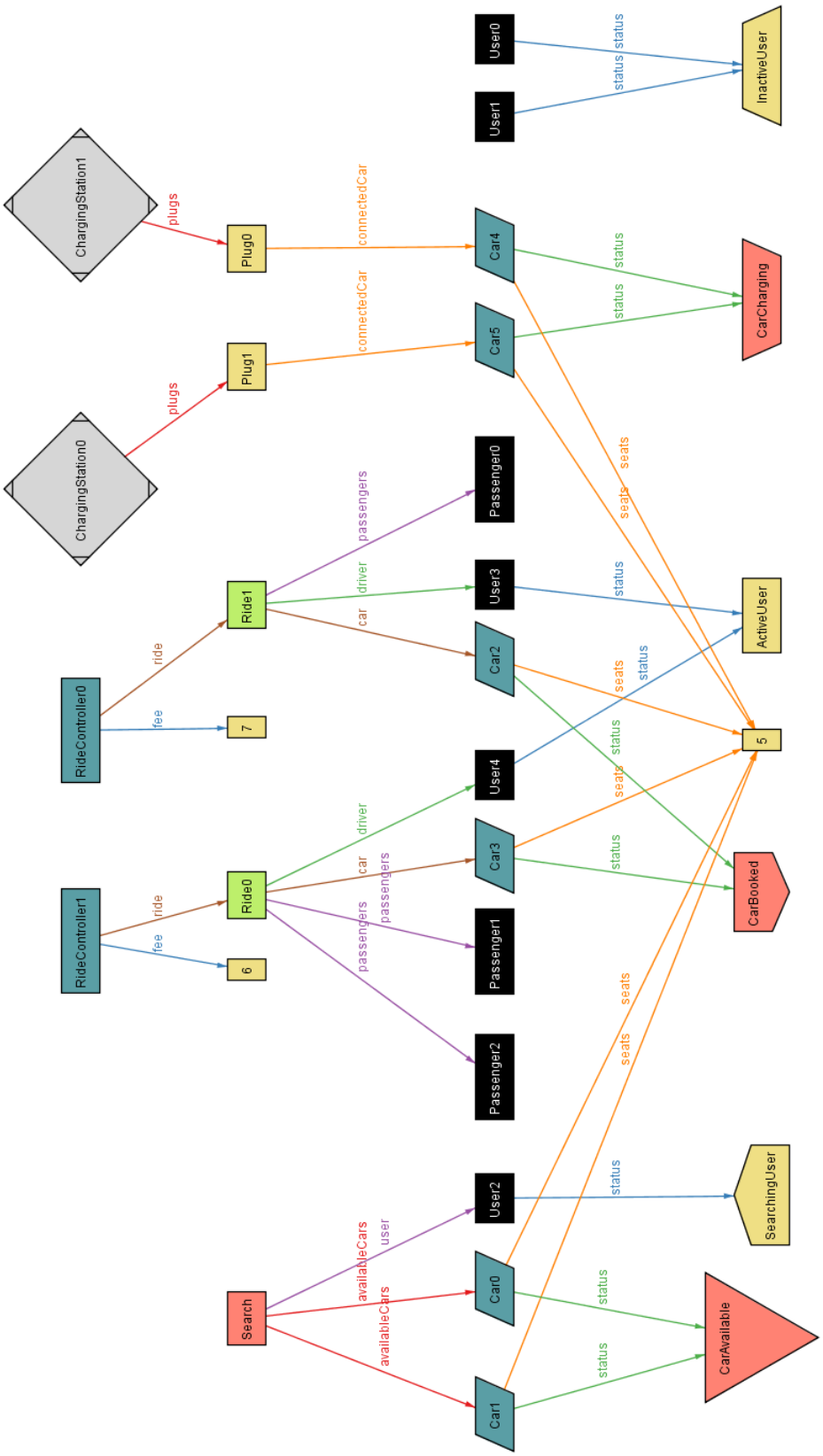
```
module PowerEnjoy

/* ************************ SIGNATURES ************************ */

/** PEOPLE **/
abstract sig Person{}
        sig Passenger extends Person{}
        sig User extends Person{
                status : one UserStatus,
        }

abstract sig UserStatus{}
        lone sig ActiveUser extends UserStatus{}
        lone sig InactiveUser extends UserStatus{}
        lone sig SearchingUser extends UserStatus{}


/** COMPANY **/
sig Car{
        status:one CarStatus,
        seats : 5
}

abstract sig CarStatus{}
        lone sig CarAvailable extends CarStatus{}
        lone sig CarBooked extends CarStatus{}
        lone sig CarCharging extends CarStatus{}


sig ChargingStation{
        plugs : set Plug
}{#plugs ≥1}

sig Plug{
        connectedCar : lone Car
}

/**RIDE**/
sig Ride{
        driver: one User,
        car : one Car,
        passengers: set Passenger
}{#passengers ≥0}


sig RideController{
        ride : one Ride,
        fee : one Int,
        discount: lone DiscountType,
}{ fee ≥1}

abstract sig DiscountType{}
        lone sig MSODiscount extends DiscountType{}
```

```
/** SEARCH **/

sig Search{
        user : one User,
        availableCars : set Car
}


/* ********************** FACTS ********************** */


//An inactive user cannot search nor be active
fact userInactive{
        all u:User | isUserInactive[u] iff not(isUserActive[u] or
            ↪ isUserSearching[u])
}

// A car is involved in maximum one ride
fact carHasOneRide{
        all c:Car | lone r:Ride | c in r.car
}

//Enough seats for all users
fact enoughSpace{
        all r:Ride | peopleInCar[r] =< r.car.seats
}

// User can drive max one car at the time
fact userHasOneRide{
        all u:User | lone r:Ride | u in r.driver
}
//An active user is in a Ride
fact userInRideIsActive{
        all u:User | isUserActive[u] iff isUserInRide[u]
}

// If an user has searching status he is doing a search
fact searchingUserIsInSomeSearch{
        all u:User | isUserSearching[u] iff isUserInSearch[u]
}

//A user can do only one search
fact oneSearchPerUser{
        no disjoint s1,s2 :Search | s1.user = s2.user
}

// All cars that are available must be displayed in searches
fact carsInSearchAreAvailable{
        all c:Car | all s:Search | isCarAvailable[c] iff c in s.
            ↪ availableCars
}

// A booked car in a Ride must have booked status
fact carInRideIsBooked{
        all c:Car | isCarUsed[c] iff isCarInRide[c]
}
```

```
//A chargingCar is in some ChargingStation
fact chargingCarInStation{
        all c:Car | isCarCharging[c] iff isCarInStation[c]
}


// A charging car is plugged into one plug
fact chargingCarIsPluggedIn{
        all c:Car | isCarCharging[c] iff isPlugConnected[c]
}

//A plug belongs to one station
fact plugHasOneStation{
        all p:Plug | one s:ChargingStation | p in s.plugs
}

// A car cannot be plugged into two plugs
fact carHasOnePlug{
        no disjoint p1,p2:Plug | p1.connectedCar = p2.connectedCar
}

//A passenger is in only one car
fact passengerIsAtMostInOneCar{
        no disjoint r1,r2: Ride |some p:Passenger | p in r1.passengers and p
            ↪  in r2.passengers
}

//All passenger belong to one car
fact allPassenger{
        all p:Passenger | one r:Ride | p in r.passengers
}

//Every ride has exactly one ride controller
fact rideHasController{
        all r:Ride | one c:RideController | r in c.ride
}

//If MSO instance exists than it must be part of at least one ride
    ↪ controller
fact MSOMustBePartOfRideController{
        no m:MSODiscount | some r:RideController | m not in r.discount
}

/* ************************ PREDICATES ************************ */

// Check if the user has active status
pred isUserActive[u:User]{
        some s:ActiveUser | u.status in s
}
// Check if the user has inactive status
pred isUserInactive[u:User]{
        some  s:InactiveUser | u.status in s
}
// Check if the user has searching status
pred isUserSearching[u:User]{
        some s:SearchingUser | u.status in s
```

```
}
// Check if the user is the drive of a ride
pred isUserInRide[u:User]{
        some r:Ride | u in r.driver
}
// Check if the user is the one doing a search
pred isUserInSearch[u:User]{
        some s:Search | u in s.user
}
//Checks if the car has booked status
pred isCarUsed [ c:Car]{
        some s:CarBooked | c.status in s
}
//Checks if the car has available status
pred isCarAvailable[c:Car]{
        some s:CarAvailable | c.status in s
}
//Checks if the car has charging status
pred isCarCharging[c:Car]{
        some s:CarCharging | c.status in s
}

//Checks if the car is involved in a ride
pred isCarInRide[c:Car]{
        some r:Ride | c in r.car
}
//Checks if the car is in a charging station
pred isCarInStation[c:Car]{
        some s:ChargingStation | c in s.plugs.connectedCar
}
//Checks if the car is conncted to a plug
pred isPlugConnected[c:Car]{
        some p:Plug | c in p.connectedCar
}

/* ************************ FUNCTIONS ************************ */
//Return the number of people in the car
fun peopleInCar[r:Ride]: Int {
        #r.passengers + 1
}

//Return the number of cars in charge
fun carInCharge[c:ChargingStation]:Int{
        #c.plugs.connectedCar
}

pred showGeneral{

        #ChargingStation =2
        #Car = 6
        #Ride =2
        #User =5
        #Search = 1
        #Passenger=3
}
run showGeneral for 10
```

The alloy model simulates a common scenario : some user's are involved in rides which means that they are drivers of a car that has to be booked. Some other user is doing a search which lists only available cars. A user doing in a search can't be inactive nor active (driving). Every ride has an associated fee . We chose to implement a Money Saving Option discount as example of the different discounts that can gained by users. Finally some cars are located in charging station , which are composed of plugs. Obviously a car can only be connected to only one plug at the time and if in a charging station it must be in 'charging' status.


ChargingStation1
ChargingStation0
plugs
plugs
Plug0
Plug1
connectedCar
connectedCar
Car4
Car5
Car2
Car3
Car0
Car1
User0
User1
User3
User4
User2
InactiveUser
ActiveUser
SearchingUser
CarCharging
CarBooked
CarAvailable
Passenger0
Passenger1
Passenger2
RideController0
RideController1
Ride1
Ride0
ride
ride
fee
fee
7
6
driver
car
passengers
driver
car
passengers
passengers
passengers
Search
availableCars
availableCars
user
status
seats
5

4.6 Non functional Requirements

Some non functional requirements have been to be necessary in the to be developed system in order to provider a correct operation flow and an optimized UX.

- *User friendliness*: the Mobile/Web as well as the On-Board Application must be easily handled by the average web user with basic web knowledge.
- *Performance*: to supply a suitable service the system must be responsive even at peak activity times. To achieve this client-server transaction must be reduced to a minimum.
- *Reliability*: the PowerEnjoy service must be active 24h/24h . Maintenance must be performed without compromising functionality. At all the time , a user must be able to be charged the owed fee.
- *Data integrity and availability*: Data have to be always available and accessible.Data availability can be achieved by duplicating data on back-up servers. For instance the use of RAID, mirroring and backup could present a valid solution.
- *Maintainability*: the software should be implemented so as to be as modular as possible to allow future expansions and easy modifications
  Google API are required but can be easily updated from time to time on all systems.
- *Security*: user accounts are provided with two factor authentication with a code sent to the user's mobile phone to add a secure connection method to the system. Sensible user information are stored using a hashing mechanism .
  User data input should be filtered to avoid malicious modifications (i.e SQL Injections).
  All connections should be encrypted and run on a secure HTTPS protocol with valid certificates that uses the secure TSL protocol to avoid Man in the Middle Attacks.

5. APPENDICES

5.1 Tools

The following tools where used in the creation of this document:

- *TexMaker 4.5* as Editor
- *Alloy 4.2* for Alloy Modelling
- *Git* for version control
- *Sketch & Photoshop* for Mockups
- *StarUML* for UML Diagrams

5.2 Hours of work

- Simone Amico 35 h
- Chianella Beatrice 35 h
- Giovanakis Yannick 35 h