Software engineering 2 project

PowerEnJoy

Requirements analysis and specification document (RASD)

Authors:
Madaffari Federico (matriculation number: 792873)(Computer Science)
Mandrioli Claudio (matriculation number: 849973)(Automation and Control)

Table of Contents

Introduction **5**
- **Revision history** **5**
  - Version 2.0 5
  - Version 3.0 5
- **Purpose** **5**
- **Scope** **5**
  - Goals 6
- **Definitions** **6**
- **Reference documents** **8**
- **Overview** **8**

Overall description **9**
- **Product Perspective** **9**
  - Hardware Interfaces 9
  - Software Interfaces 9
  - User Interfaces 9
- **Product functions** **9**
  - Managing users 9
  - Managing the cars 10
  - Managing rides 10
- **User Characteristics** **11**
- **Constraints** **11**
  - Regulatory Policies 11
  - Hardware Limitations 11
  - Interfaces to other applications 11
  - Parallel operation 11
  - Audit functions 11
  - Control Functions 12
  - High-order language requirements 12
  - Criticality of application 12
- **Domain assumptions** **12**

Specific requirements **14**
- **External interface requirements** **14**
  - User interfaces 14
  - Hardware interfaces 18
  - Software interfaces 18
  - Communication interfaces 18
- **Functional requirements** **19**
  - Scenarios 19
  - Analysis diagrams (static diagrams) 25
  - Use cases 29
- **Performance requirements** **48**
- **Logical database requirements** **48**
- **Design constraints** **48**
- **Standard compliance** **48**
- **Software system attributes** **49**
  - Reliability 49
  - Availability 49
  - Security 49
  - Maintainability 50
  - Portability 50

Alloy **51**
Alloy model **51**
Generated worlds **55**
World 1 55
World 2 56
World 3 57
World 4 58

Hours of work **60**
Madaffari Federico **60**
Mandrioli Claudio **60**

Introduction

Revision history

Version 2.0

In version 2.0 of this document the class diagram and the use cases diagram are updated in order to better represent the requirements of the project. No conceptual changes were applied.

Version 3.0

This is version 3.0 of this document. With respect to the first version of the document the following changes are made:

- Correction of some typos is made.
- The domain assumption [D5] is dropped as we were required to actually verify the users to be nearby the car when unlocking it: for this reason domain assumption [D15.3] was introduced. The idea is to verify the presence of the user nearby the car asking him to insert on the phone a numerical code shown by the car on board computer through a small display. Moreover also the definition of the functional requirement [FR14] is changed.

Purpose

This document contains the definition of the requirements for the developing of the power enjoy application, a electric-car sharing service.
It will provide definition of the context in which the system will act, constraints on how it can be realized and goals. Requirements will be divided between functional and non-functional ones. Those will be formally defined through UML diagrams. Its features are going to be verified using models implemented in the specification language alloy.
The document is written for those that will actually implement the system and also those that are going to manage the project development.

Scope

We are going to develop a digital management system for a car-sharing service that exclusively employs electric cars.
This system is supposed to interact with users by means of a mobile app and a web application, allowing them to use the cars provided by the car sharing and to pay trough on-line payment services (external to the system we are developing).
The sharing service is supposed to live in a city-perspective, this means that users are considered to be the citizens of a city (in our case Milan) and they will use the cars for in-city transportation.
Moreover the system will manage the car's battery state of charge, both making convenient for clients to provide to car recharge and also coordinating employees assigned to move cars that are out of charge to dedicated stations.

Goals

In this section we listed the system's goals sorted by the ones related to the user and others related to the system

USER GOALS

- [G1]:users can register to the system
- [G2]:the system must securely (from a privacy point of view) manage the personal info of users
- [G3]:the system must securely manage automated payment system (interfacing with some external payment on-line service)
- [G4]:users must be able to find cars near themselves (less than 1km)
- [G5]:users must be able to find cars in selected locations
- [G6]:users must be able to reserve a single available car
- [G7]:users must be able to tell the system that he reached the reserved car so the system can unlock it
- [G8]:users must pick up the car within 1h from the reservation

SYSTEM GOALS

- [G9]:the system must charge 1euro to a user that reserves a car without starting it within 1h
- [G10]:the system must set to available a car that has been reserved but not picked up within 1h
- [G11]:the system must start charging the payment when the user starts the engine
- [G12]:users must be notified about the battery state of charge
- [G13]:users must know how much currently is being charged on the payment in real time
- [G14]:payment stops when user exits the car and the car is left in a safe area
- [G15]:the system must lock the car once the user exited it
- [G16]:the system must be able to detect the number of passengers and to eventually apply a discount on the last ride of 10%
- [G17]:the system must apply a discount of 20% to the payment if the user leaves the car with more than 50% of charge
- [G18]:the system must apply a discount of 30% if the car is left in a charging area and he plugs the car into the power grid
- [G19];the system must take care of the state of charge of the batteries
- [G20];the system must charge a 30% extra to the payment when the user leaves the car at more than 3km for the nearest power grid station
- [G21]:the system must charge 30% extra to the payment when the user leaves the car with less than 20% of charge

Definitions

In this section we provide rigorous definitions for the terms used in the document.

- SYSTEM: when this word is used in the document it refers to the digital management system to project
- USER: is the costumer of the PowerEnJoy service, not necessarily already registered to the system
- REGISTERED USER: is the user that registered to the system providing his personal info, therefore in the system there must be stored those data
- EMPLOYEES: are those people that work for the PowerEnJoy company that will take care of cars out of charge
- CAR: in this document my car are intended only those electric cars related to the PowerEnJoy service (model of car: BMW i3)
- START THE ENGINE: when the driver activates the electric motor of a car
- RIDE: is a time span in which a certain user is using a certain car. This time span starts when the user starts the engine (ride start) and ends when all doors of the car are closed and no passengers are on the car (ride end)
- PICK UP THE CAR: this expression is used as a synonym of 'start the ride'
- LOCK THE CAR: when the system (as only the system can perform this action) locks a car at the end of a ride
- UNLOCK THECAR: when the system (as only the system can perform this action) unlocks the car after the user's request
- RESERVE: when a user tells the system that he is going to use a car and therefore car must be marked as unavailable
- RESERVED CAR: a car marked unavailable because some user reserved it
- CHARGE OF THE PAYMENT: is the total amount to be paid by the user after a ride
- PAYMENT SYSTEM: is the external service that provides the payment functionality to the PowerEnJoy service
- APPLY A DISCOUNT: when at the end of a ride the charge of a payment is reduced according to discount policy of the service provider
- CHARGE --% EXTRA: when at the end of a ride the charge of a payment is increased of some percentage according to fine policy of the service provider
- POSITION: coordinates that localize something (a car, a user, some area…)
- USER EXITS THE CAR: is used as a synonym of end of 'end of the ride'
- END OF THE RIDE: when during a ride all the doors are closed and no more passengers are in the car
- AVAILABLE CAR: a car is said to be available when it is not unavailable (see definition of 'unavailable') and moreover it is possible for the users to reserve it
- UNAVAILABLE CAR: a car is said to be unavailable when it is reserved, being used by some user or with the battery level under 20%
- SAFE AREA: predefined set of places where it is possible for the users to park the cars
- CHARGING AREAS: are the parking provided with plugs to recharge the cars of the PowerEnJoy service (these are a subset of safe area)
- POWER GRID STATION: is a synonym of charge area

- STATE OFCHARGE (Soc) : the state of charge of a car is the level in percentage of the energy stored in the battery
- PRE-DEFINED: when some information is supposed to have been previously provided to the database of the system ad therefore it is always available form the activation of the service
- WELL-DEFINED: when some information if defined precisely and without ambiguity
- SPECIAL SCREEN: by means of this name in this document is intended the screen that is supposed to show to the user the code necessary to unlock the car[1]

Reference documents

For the writing of this document the following where consulted:

- Assignments AA 2016-2017
- IEEE Recommended Practice for Software Requirements Specifications: *http://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=5841*

Overview

The document is structured in the paragraphs described below:

1. Introduction: gives an overview of the software product to be developed
2. Overall description: describes the context in which the software will work and the role that it will have in the PowerEnJoy service
3. Specific requirements: this section describes a bit more formally what the product is supposed to do both from the functional and non functional point of view
4. Alloy: this part presents an Alloy model for the system that describes and verifies some properties of it.

[1] This definition is introduced in version 3.0 of this document

Overall description

This section does not provide specific requirements, but simply illustrates the perspective and the background for specifying concrete requirements in the next section of this document.

Product Perspective

The software developed is completely self-contained and it does not rely on any other larger system or legacy system. However in the future it is possible to cooperate with other service providers, which can offer a much more pervasive experience for the users.

Hardware Interfaces

The software will work only on mobile phone devices and web application, therefore it will not provide any hardware interfaces

Software Interfaces

In order to use the web application, users are required to have installed any of the following web browsers: IE 6.0+, FF 10+ or Chrome 20+.
The back-end stores its data in a DBMS and can run on every platform that supports the JVM.
The software product will run in any mobile operating system like Android, iOS and Windows Phone, and it will provide the same functionality both on the web app and the mobile app. The cars built-in computer run a customized Linux-based kernel and provide to the developers the needed API to facilitate the communication between the cars software and the system developed.

User Interfaces

The user will be able to access the service both through a web app using his personal computer and a mobile app to grant flexibility and portability.
The software will provide an intuitive look and a user friendly interface to allow the user to be ready to interact with the app on his very first access. In order to provide a consistent experience the mobile app and the web app will present a similar interface as much as possible.

Product functions

We have identified 3 sets of main product functions that we are going to define:

- Managing users
- Managing cars
- Managing rides (car choice, reservation, ride info, payment)

Managing users

[FR1]: register through both mobile app and web browser
  [FR1.1]: define and manage a password to access the service
[FR2]: update personal info
[FR3]: update payment info
[FR4]: delete their profile

[FR5]: manage at the same time users with multiple roles (the active user and the employee)
[FR6]: user can cancel his reservation within 1h

[NFR1]: user personal data must be stored securely.
[NFR2]: user password must be at least 8 alphanumeric characters.
[NFR3]: the system must have an high reliability rate, the maximum response time must be lower than 2 seconds.
[NFR4]: the system must be able to manage a huge number of users (at least 150'000)

Managing the cars

[FR7]: know on real time of each car's state of charge
  [FR7.1]: the system must be able to communicate via web with the car
[FR8]: know on real time each car position
  [FR8.1]:cars must be provided with a geo-localization system sufficiently accurate
[FR9]: communicate to the employees to go and move an out of charge car to a power grid station
  [FR9.1]: the system must be able to communicate with the employees
[FR10]: the system must tell employees to bring back a car that has been left outside the city boundaries

[NFR5]: the system must have an high reliability rate, the maximum response time must be lower than 2 seconds.
[NFR6]: the system must be able to manage a huge number of cars (at least 300)

Managing rides

[FR11]: allow users to find available cars using current location
  [FR11.2]: determine whether a car is available or not
[FR12]: allow users to find available cars around a manually selected location
[FR13]: allow users to reserve a single car per time
  [FR13.1]: a reserved car must be marked as un-available
  [FR13.2]: allow user to cancel a reservation
~~[FR14]: allow users to unlock cars~~
[FR14]: verify that the user is nearby the car and if so allow him to unlock it[2]
[FR15]: make users know on real time the battery state of charge
[FR16]: make users know on real time the payment amount
  [FR16.1]: the system must compute on real time the cost of the ride
  [FR16.2]: the system has to automatically determine discounts and fines
   [FR16.2.1]: if multiple discounts are available, only one of them is going to be applied

[2] This functional requirement is updated in the third version of the document when the hypothesis that users unlock the car only after having reached it is dropped.

User Characteristics

User should meet the following characteristics:

- High school education level.
- Knowledge in using a browser.
- Knowledge in using a mobile device.

Constraints

Regulatory Policies

The user must accepts the ToS of the software in order to register to the service, in the ToS the user grants to take the responsibility to ensure that the use of the system complies with the local laws and policies
The system needs the permission of the user to acquire his current position and to manage, store and process sensible data, such as his driving license. The system needs also to permit to the user the possibility to delete all his personal data.

Hardware Limitations

The software has strict hardware limitations, because the application runs on mobile devices, which as limited resource and power computation.
The following are the specification that the hardware must satisfy:
Mobile app:

- 3G Connection
- Enough free space on the device for the app package (>= 30 MB)
- GPS connection enabled (Optional)
- Enough RAM to run the app (>= 1GB)

Web app:

- 8 Mb/s internet connection
- 800x600 screen resolution

Interfaces to other applications

The software relies on an external payment system, such as PayPal, and on the use of an external search-on-map service, such as Google Maps.

Parallel operation

The system developed has to provide, on his back-end, simultaneous operations from multiple users for every feature offered. The system must also be able to support the use of multiple cars.

Audit functions

The software product does not perform any audit.

Control Functions

The software has to interface with the built-in computer and sensors on the cars to be able to provide functions such as:

- Lock/Unlock of the doors
- Detect number of passengers
- Detect opening/closing of the doors
- Detect the battery state of charge
- Built-in GPS on cars

High-order language requirements

To develop the software for both the web and the mobile device the knowledge of different kind of programming language is required. In fact, the team has to know language such as HTML, JavaScript, PHP, SQL to develop the Web App and Java, Objective C/Swift, C# to develop the Mobile App.

Criticality of application

The system does not present any kind of criticality to the user or the employees.

Domain assumptions

ASSUMPTIONS ABOUT USERS

[D1]: users have access to the internet
[D2]: users are localizable through GPS and/or 3G/4G
[D3]: registered users have the driving license
[D4]: the person that reserves the car and that drives the car are the same
~~[D5]: users confirm to have reached the car only having actually reached it (they would have no interest in lying about this)~~
[D6]: users are provided with a smartphone and/or pc

ASSUMPTIONS ABOUT EMPLOYEES

[D7]: employees are provided with some external charging medium to move cars with low battery and far from charging stations
[D8]: employees are able to unlock and start unavailable cars
[D9]: employees are always able to reach cars that are required to
[D10]: emplyees do what they are supposed to (i.e. moving and recharging cars when they are communicate to)

ASSUMPTIONS ABOUT PAYMENTS

[D11]: payments are managed by paypal system.
[D11.1]: if a user runs out of credit to pay this is a problem between the user and paypal, not the PowerEnJoy service
[D12]: payment data provided by the user are valid

ASSUMPTIONS ABOUT CARS

[D13]: cars are provided with an internal computer

[D14]: cars are not damaged and/or wont present manufacturing defects that avoid they're proper working

[D15]: car are provided with internal sensors for detecting the number of passengers

[D15.1]:cars are provided with sensors for detecting whether doors are open or not

[D15.2]:cars are provided with sensors detecting whether the car is plugged or not

[D15.3]:cars are provided with a special screen that can show a numeric code to someone external to the car[3]

[D16]: there is always place in charging stations for parking

[D17]: maximum number of passengers in the car is never exceeded

[D18]: a car enters the low-battery state only at the end of a ride

[D19]: unavailable cars (ie: car with low battery that were put in charge) cannot be unplugged

[D20]: users wont start a ride on a plugged car

[D20.1]:users start a car only when they are on it

[D21]: car involved of the model BMW i3 (4 places)

ASSUMPTIONS ABOUT AREAS

[D22]: city boundaries are pre-defined and well-defined in the system

[D23]: safe areas and charging stations are pre-defined and well-defined i the system

[D24]: charging areas are a subset of safe areas

[3] This domain assumption is introduced in the third version of the document.

Specific requirements

In this section several tools (UML, mockup, scenarios) are use to present more specific requirements for the product to be developed.

External interface requirements

User interfaces

Here we present a mockup of the main features of the mobile app and the web app.

07:53

Reserve a Car

Via Giovanni Celoria, 2 - Milano

46%

CONFIRM

07:53

Pending Reservation

Via Celoria, 2 - Milano

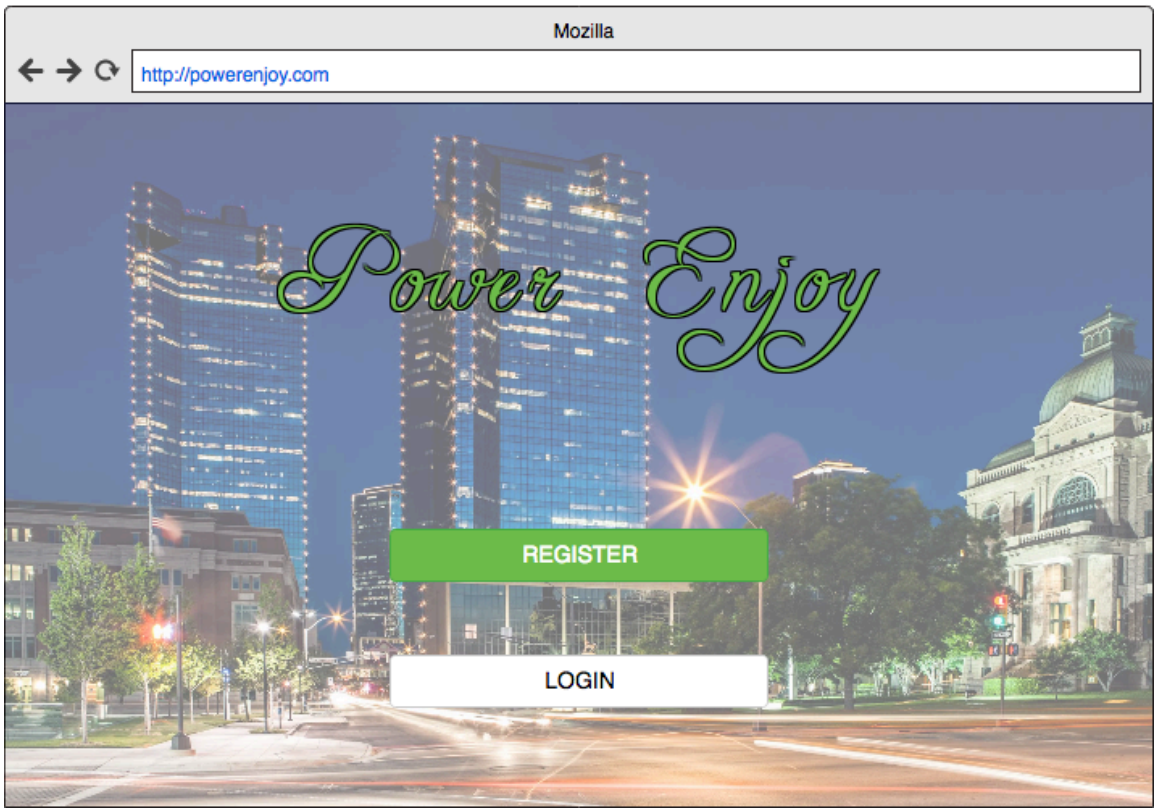
46%

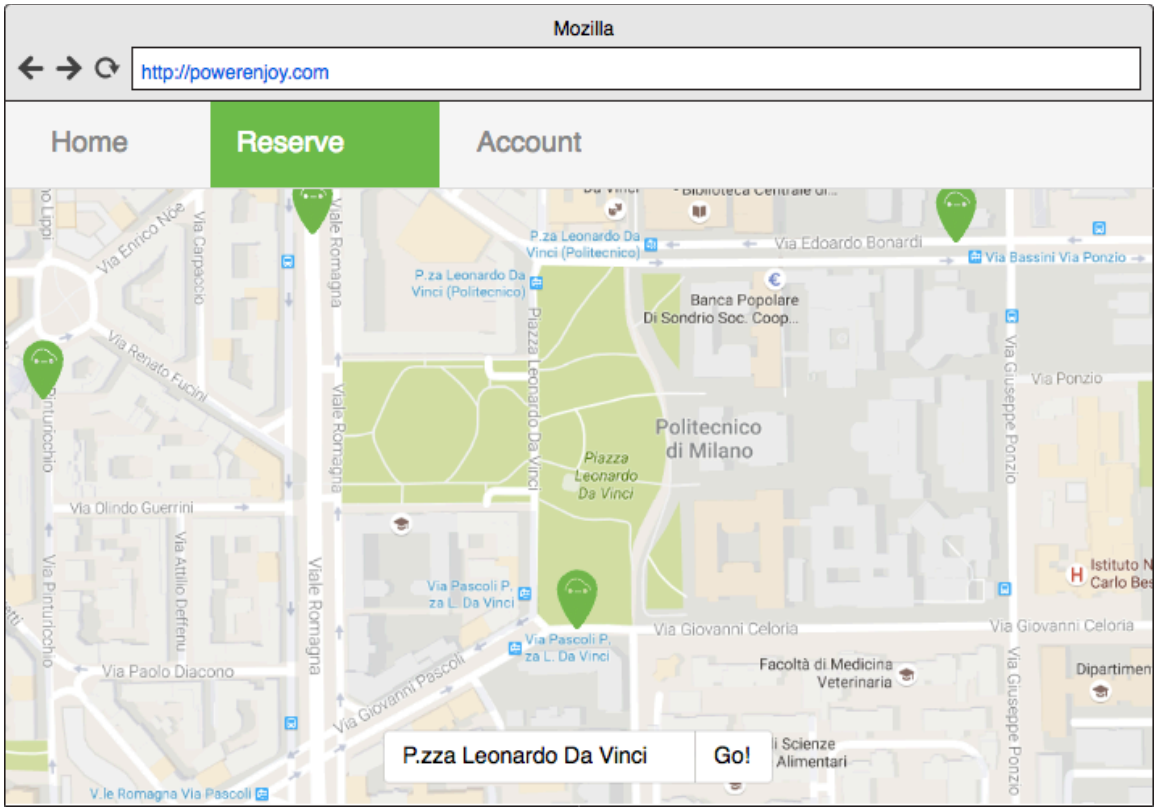
Reserved on: 14/11/2016, 7.53

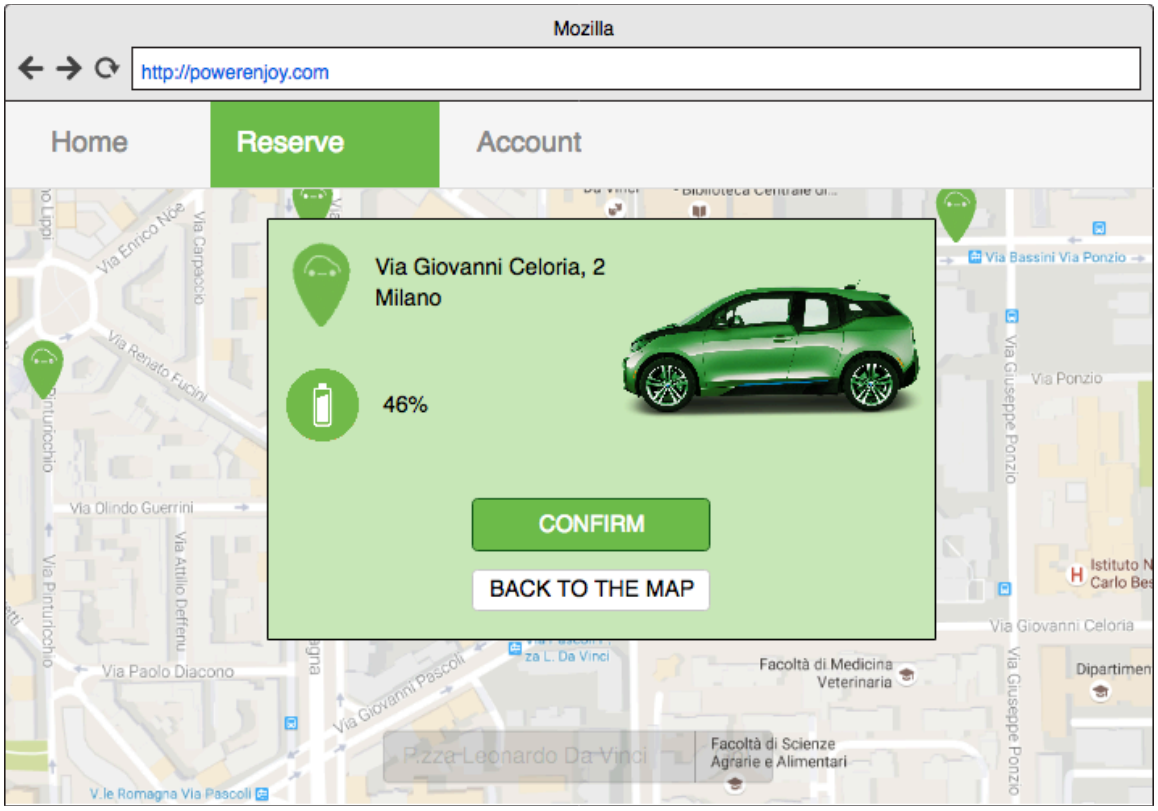
Expires in: 59 minutes 50 seconds

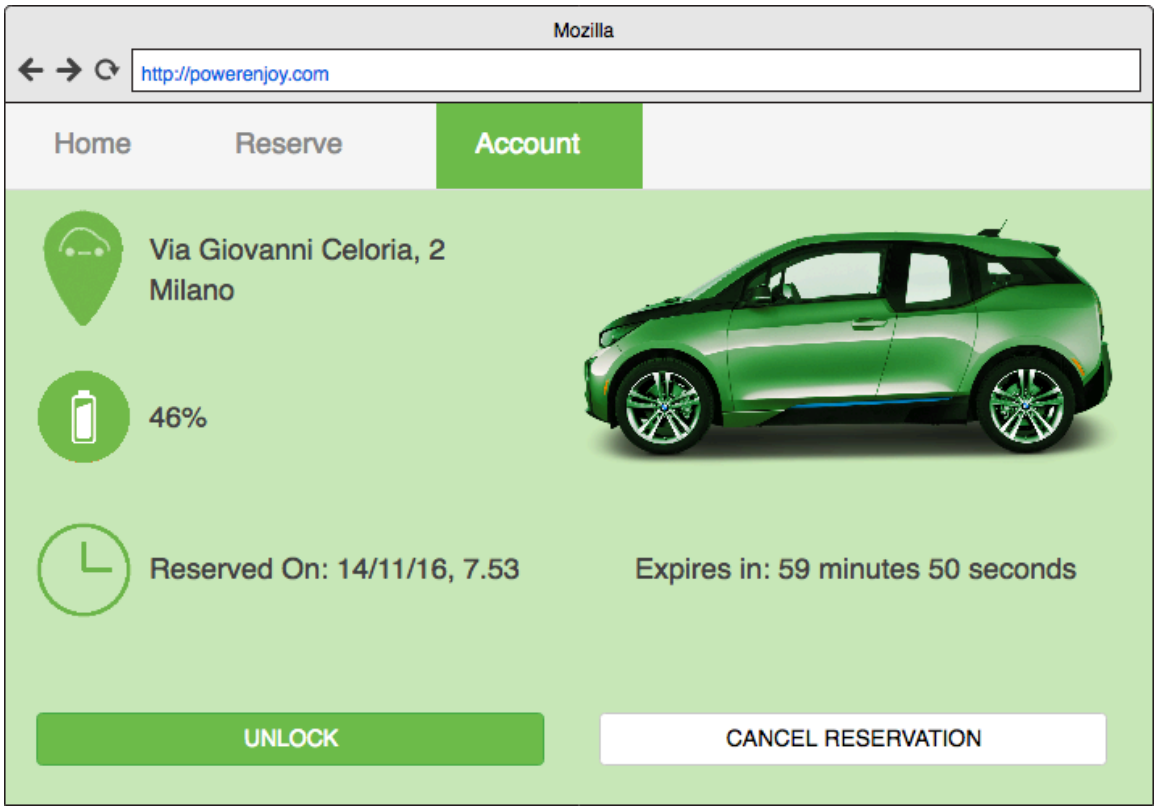
UNLOCK

CANCEL RESERVATION


Mozilla
http://powerenjoy.com
Power Enjoy
REGISTER
LOGIN


Mozilla
http://powerenjoy.com
Home
Reserve
Account
P.zza Leonardo Da Vinci
Go!


Mozilla
http://powerenjoy.com
Home
Reserve
Account
Via Giovanni Celoria, 2
Milano
46%
CONFIRM
BACK TO THE MAP
Via Bassini Via Ponzio
Via Ponzio
Via Giuseppe Ponzio
Via Giovanni Celoria
Via Olindo Guerrini
Via Paolo Diacono
Facoltà di Medicina Veterinaria
Facoltà di Scienze Agrarie e Alimentari
P.zza Leonardo Da Vinci
V.le Romagna Via Pascoli


Mozilla
http://powerenjoy.com
Home
Reserve
Account
Via Giovanni Celoria, 2
Milano
46%
Reserved On: 14/11/16, 7.53
Expires in: 59 minutes 50 seconds
UNLOCK
CANCEL RESERVATION

Hardware interfaces

The software will work only on mobile phone devices and web application, therefore it will not provide any hardware interfaces

Software interfaces

The back-end of the system developed requires the following software products:

Database Managements System	
Name	MySQL
Mnemonic	MySQL
Specification number	Community edition
Version number	5.7.15
Source	http://www.mysql.com/downloads

Application Server Java EE	
Name	Java Enterprise Edition
Mnemonic	Java EE
Specification number	
Version number	7
Source	http://www.oracle.com/technetwork/java/javaee/overview/index.html

Communication interfaces

Protocol	Port	Service
TCP	80	HTTP/HTML Connection
TCP	3306	MySQL Service

For the first release of the project, we will assume that the DBMS and the Application Server run both on the same physical server. All the mobile devices, to work with the entire system, must be connected to internet network, through high-speed connection (3G/4G).

The connection must be encrypted via HTTPS or SSL to ensure the total privacy of account.

Functional requirements

Scenarios

In this section we present some scenarios: those informal descriptions should provide an idea of how the system will interact with the user. For this reason these are provided with real world details as locations and timing to make them s realistic as possible.

Name of the scenario	User registration
Requirements	[FR1][FR1.1]
Related goals	[G1][G2]
Assumptions	[D1][D3][D6][D12]
Scenario description	
Carlo is afraid about pollution in Milan and thinks that the sharing-economy is a good solution to improve air quality in his city. So he discovers about PowerEnJoy and decides to subscribe. He downloads the app from the on line store and installs it. Carlo starts the application and the application asks him whether he is already registered or not. He selects no and the app starts the registration procedure. Carlo during the procedure inserts his personal info, the data of his paypal account and agrees to the treatment of his personal info. Moreover he defines a password to log in securely. The system accepts his registration and the user may now access all the functionalities provided by the service.	

Name of the scenario	Info update
Requirement	[F2][F3]
Related goals	
Assumptions	[D1][D6]
Scenario description	

Carlo is already a user of the PowerEnJoy service but he notices that when he registered he inserted the wrong telephone number. Therefore from the homepage (he was already logged in) of the app he opens the section related to the settings and selects the option “update personal info”. The app shows Carlo the present personal info and allows him to change them. Carlo corrects his telephone number and then confirms the updating. At this point the app verifies coherence of the info (ie the telephone number must be a numeric sequence) and only after the successful check accepts the changes.

Name of the scenario	Delete of an account
Requirement	[FR2]
Related goals	[G2]
Assumptions	[D1][D6]
Scenario description	
Carlo has been registered to the PowerEnJoy service for some time but now, as he bought a scooter, is no more interested in it and wants to delete his account. He starts the app on his smartphone and opens the settings section. He selects the option “delete account” and after the system asks in he is sure to delete his account and alerts him that he will no more be able to rent the electric cars of the service without the account he confirms. The system elaborate the request deleting all the info related to Carlo’s account, both personal ones and payment ones. When it has finished the app shows a window on the smartphone that confirms that the procedure was completed successfully. Then Carlo exits and uninstalls the app.	

Name of the scenario	Car reservation
Requirement	[FR7][FR8][FR11][FR11.2][FR13][FR13.1]
Related goals	[G4][G6]
Assumptions	[D1][D2][D6][D13]
Scenario description	

Carlo wants to hang out with some friends tonight and wants to reach the pub renting a car of the PowerEnJoy service. The appointment is at 10pm and it takes 15 minutes to reach the pub by car from Carlo's neighborhood.
At 9pm he opens the PowerEnwJoy app on his smartphone and selects the "rent a car" option. When starting this procedure the app shows to the user a map of the city centered the phone position (acquired from the GPS) with all available cars highlighted. Among those Carlo selects the nearest to him and confirms the reservation. After this the car is no more available for other users.
Now the app gives the possibility to Carlo of unlocking the car. Moreover he has to reach the reserved car and start the engine within 1h otherwise the system will automatically charge him of 1euro

Name of the scenario	Regular Ride
Requirement	[FR7.1][FR8][FR14][FR16][FR16.1] [FR16.2]
Related goals	[G7][G11][G14][G15]
Assumptions	[D1][D2][D4][~~D5~~][D11][D12][D15.1] [D15.3]
Scenario description	

Carlo has reserved a car at 5pm and he reaches it at 5:30pm.
Once he is nearby the car he picks the smartphone and requires to unlock the car. The car displays the code on the special screen and the user now inserts it in the app. At this point the system unlocks the car's doors, Carlo enters and sits at the driver's seat. Once seated Carlo starts the engine at 5:32pm (as it is an electric car the engine is actually only activated, not really started).
As consequence of this action the system does two things: stops the countdown started at the moment of the reservation (the timing is of 32 minutes and therefore no fine is charged to Carlo) and starts the timer of the ride.
Carlo drives safely until he reaches his destination at 6:07pm and parks the car in parking with white stripes (that in Milano means free parking). After this Carlo exits gets off the car and closes the car's door.
The system detects that there are no more people on the car and the doors are closed: as consequence of this it locks the car and stops the timing of the ride at 35 minutes.
The system evaluates the amount of the payment, as no discounts or fines are detected it simply charges a value proportional to the timing of the ride to Carlo.

Name of the scenario	Reservation cancelling
Requirement	[FR13.2]
Related goals	
Assumptions	[D1][D4][D6]
Scenario description	
Carlo reserved a car at 8:30pm to use it for hanging out with some friends. At 9pm he and his friends decide not to meet anymore so Carlo isn't going to need the car. In order not to have the 1 euro fine charged Carlo decides to delete the reservation he made. He opens his laptop and starts the web browser. At this point he opens the PowerEnJoy web page and logs in to his account. The computer shows him that there is an activated reservation on his account. Among the other options Carlo selects the one related to cancelling the reservation. The system takes into account his request and successfully executes it showing some 'operation executed' message. After this the web browser is redirected to the website's homepage. Carlo logs out and closes the laptop.	

Name of the scenario	User receives the 1h delay fine
Requirement	[FR16.2]
Related goals	[G8][G9][G10]
Assumptions	[D11]
Scenario description	
Carlo regularly reserved a car at 9am using the function to look for available cars nearby a inserted position instead of the GPS. Unluckily to Carlo it took more than 1h our to reach the car due to a failure of the lift of his building. Exactly at 10am the system detects that the car was reserved and not started within 1h, therefore it cancels the reservation and, although it wasn't Carlo's fault, charges 1euro of fine to him. Carlo is then notified by means of the mobile app that the car is no more reserved and a fine was charged on his account.	

Name of the scenario	Ride with discount
Requirement	[FR7][FR8][FR16.2][FR16.2]
Related goals	[G16]
Assumptions	[D11][D12][D13][D15][D23]
Scenario description	
Carlo had reserved a car for going to university today at 9am. At 9:30am he reaches the car with 2 room mates of him in order to go to the campus together. The ride proceeds up to the end without any problem. After the ride ends (ie all the passengers left the car and all the doors of the car are closed) the system that has detected a total of 3 passengers on the car during the ride computes the bill accounting a 10% of discount on the final amount.	

Name of the scenario	Ride with fine
Requirement	[G19][G21]
Related goals	[FR7][FR16.1][FR16.2]
Assumptions	[D11][D12][D18]

Scenario description
Carlo lives in porta Venezia and reserves a car near his house to go to the neighborhood named Lorenteggio. He reserves and uses a car provided by the PowerEnJoy service. At the beginning the car and 80% of the battery. As it is a quite long trip ant the end the ride when Carlo leaves the car it has less than 20% of the battery available. For this reason, according to the fine policy of the service at the moment of computing the total amount of the payment the system increases it of 30%.

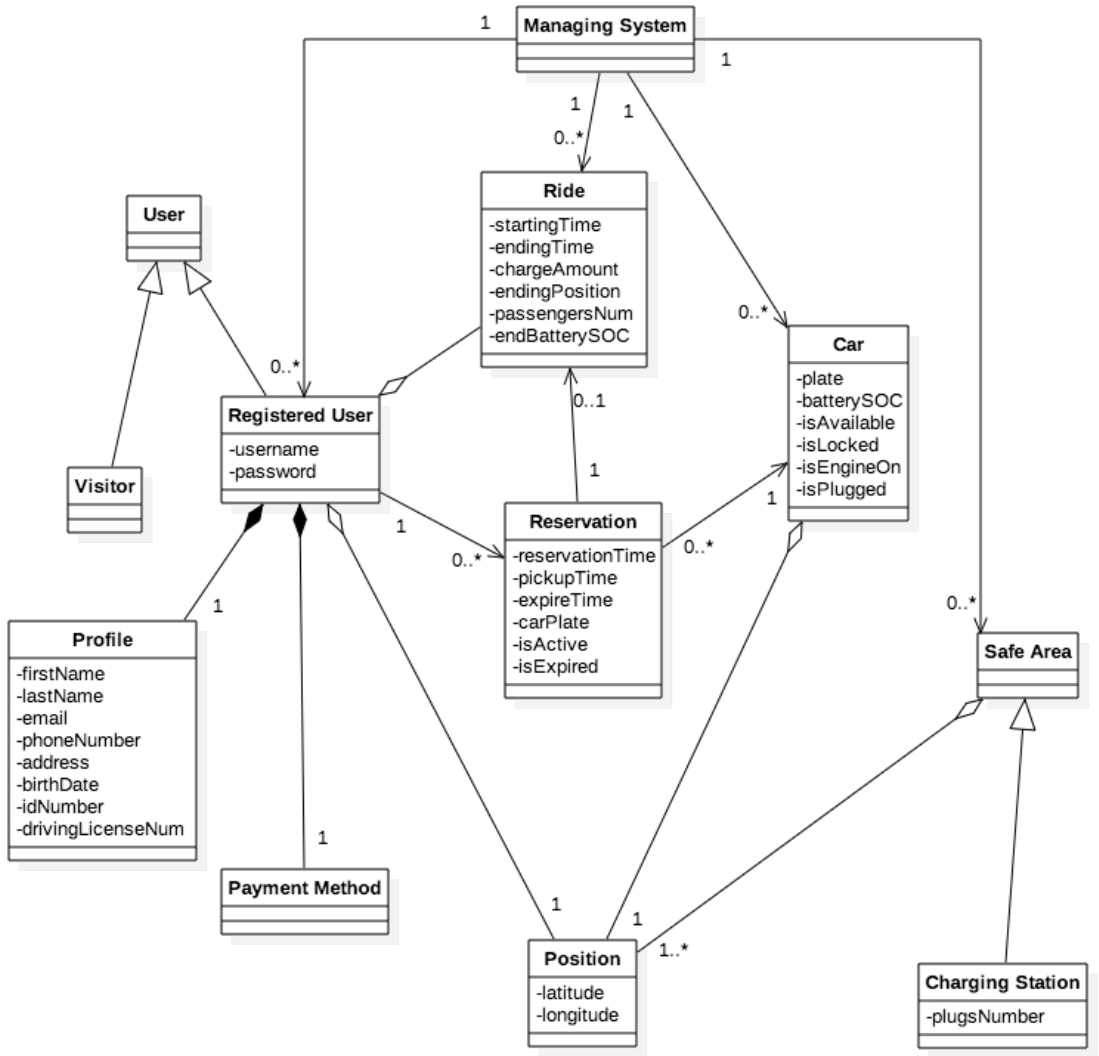
Name of the scenario	Low battery car
Requirement	[G19]
Related goals	[FR7][FR9][FR9.1]
Assumptions	[D7][D8][D9][D10][D16][D18][D23]
Scenario description	
The system detects car with the battery state of charge of 13% parked in Celoria street. Hence it automatically contacts the employee Daniele providing him the address at which is left the car. The employee reaches the car and moves it to the nearest charging station in Leonardo da Vinci square and plugs it in.	

Name of the scenario	Car out of city boundaries
Requirement	
Related goals	[FR10]
Assumptions	[D7][D8][D9][D10][D22]
Scenario description	
The system detects a car parked in Rozzano, a neighborhood out of the boundaries provided by the PowerEnJoy service. The system therefore contacts the employee Daniele providing him the exact address of the car. Daniele reaches the car and then drives it to a safe area in the city downtown.	

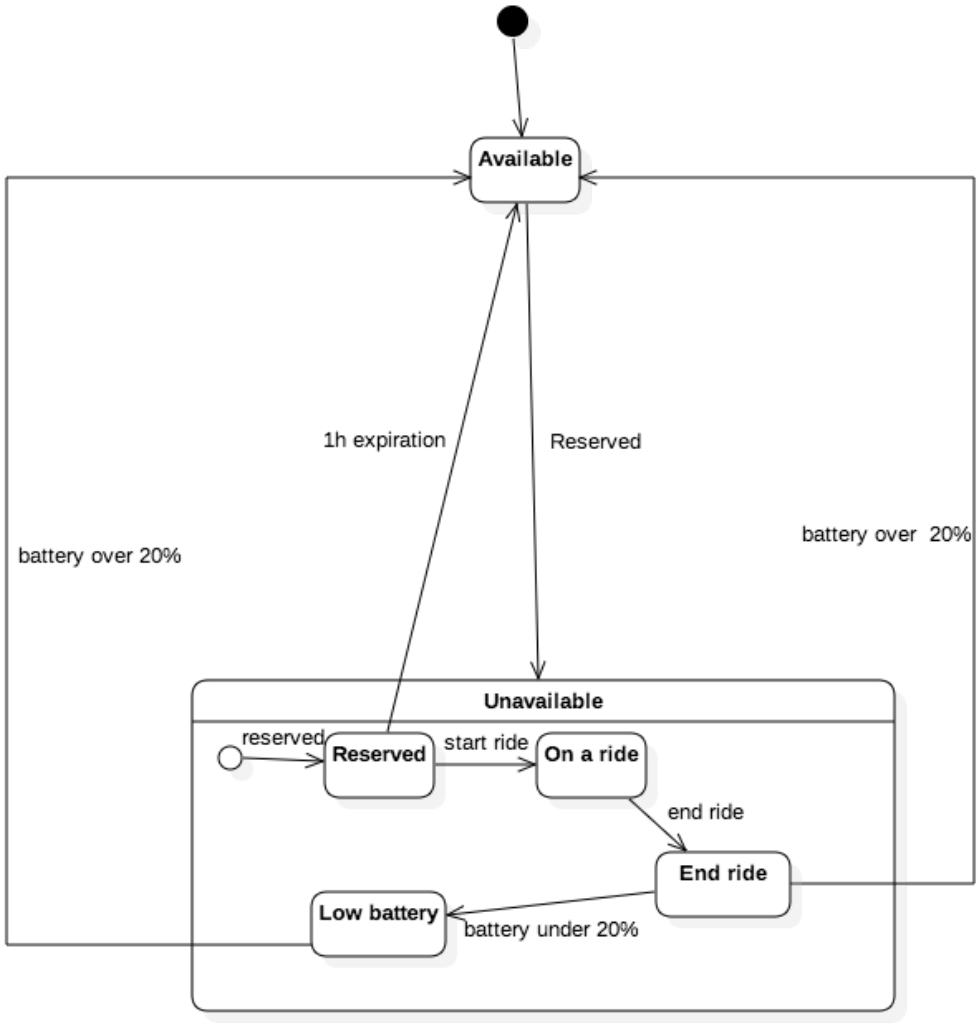
Analysis diagrams (static diagrams)

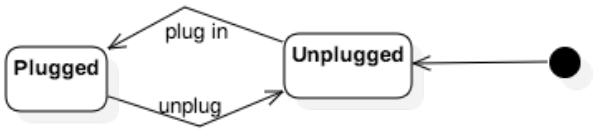
Before studying in detail each use case in the next section, it is useful to introduce the analysis diagram, which points out all the main characteristics of the project.
The class diagram describes the main entities that will be present in some way in the software system and how these will interact.
State charts instead describe the states and actions in which cars, users and rides will be involved.
Note that those are purely descriptive diagram and present only the qualitative behavior that the system will present.

Class diagram

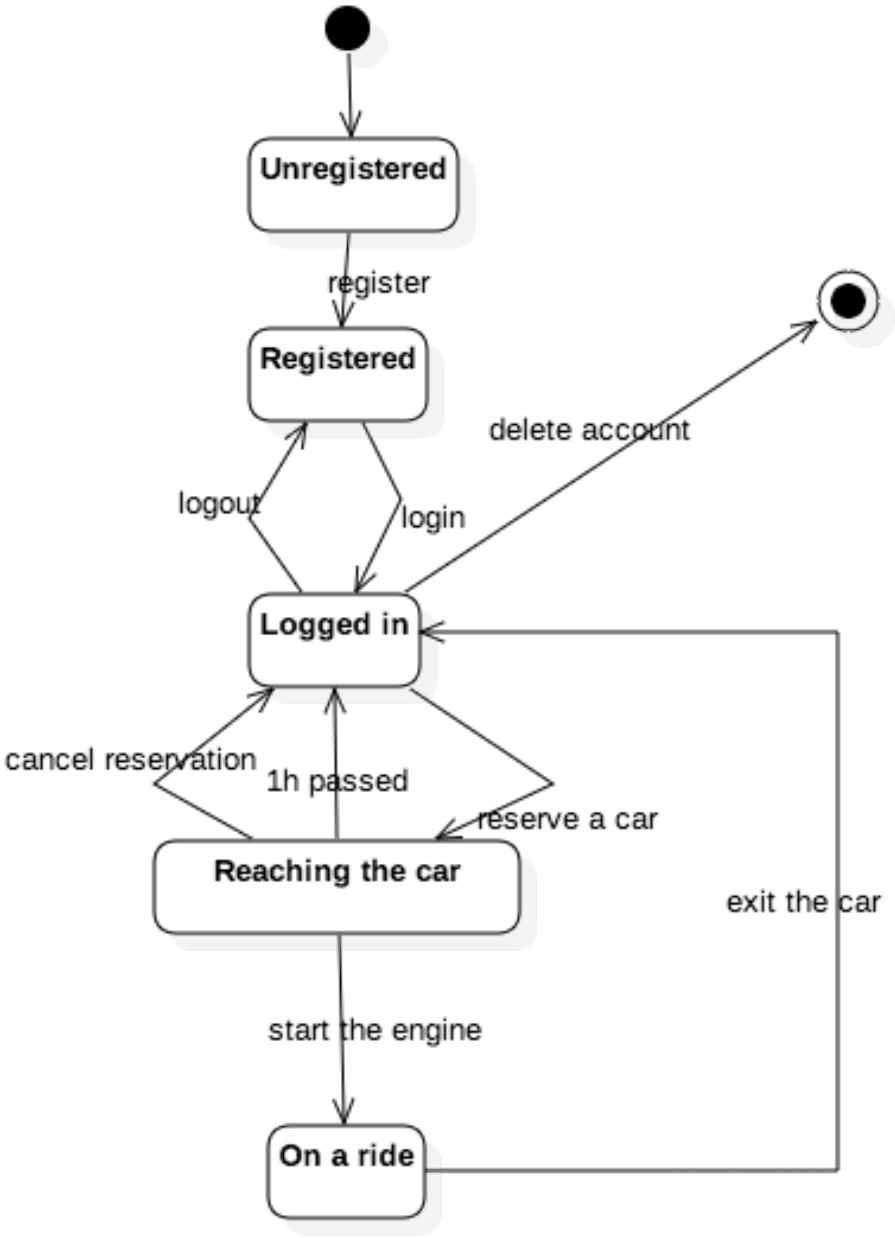


Car state chart

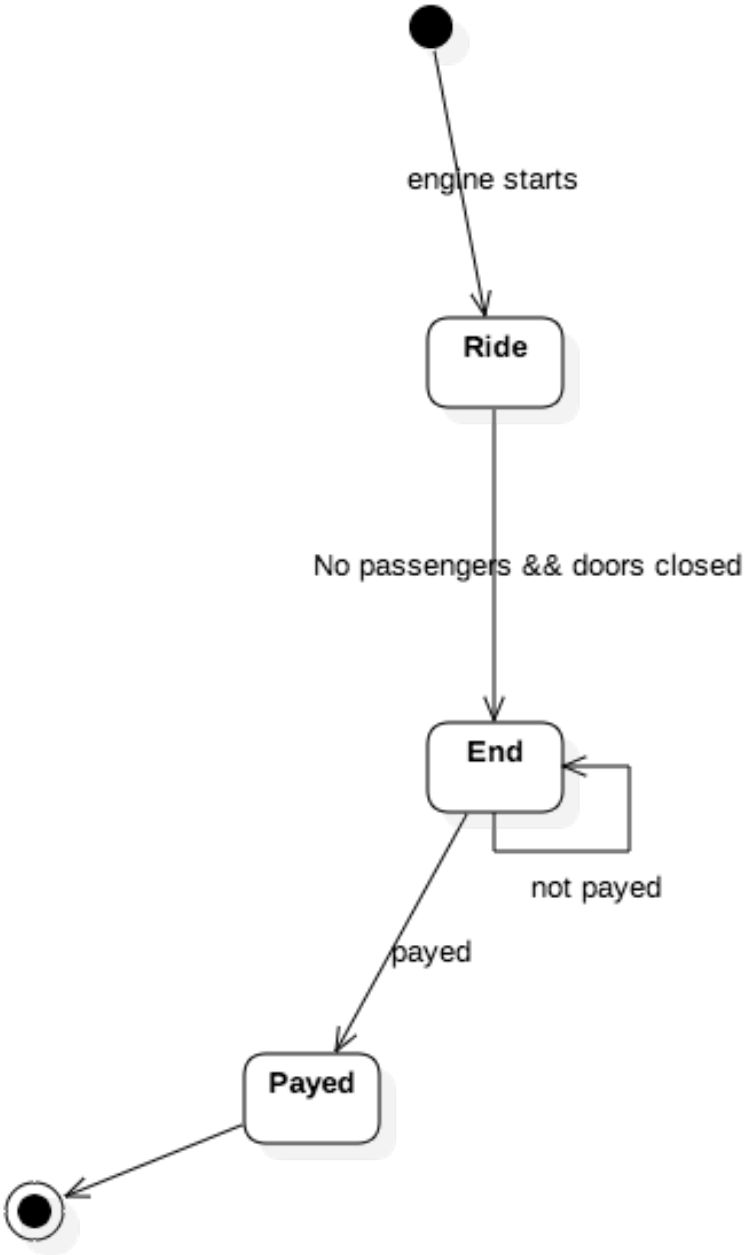




User state chart



Ride state chart

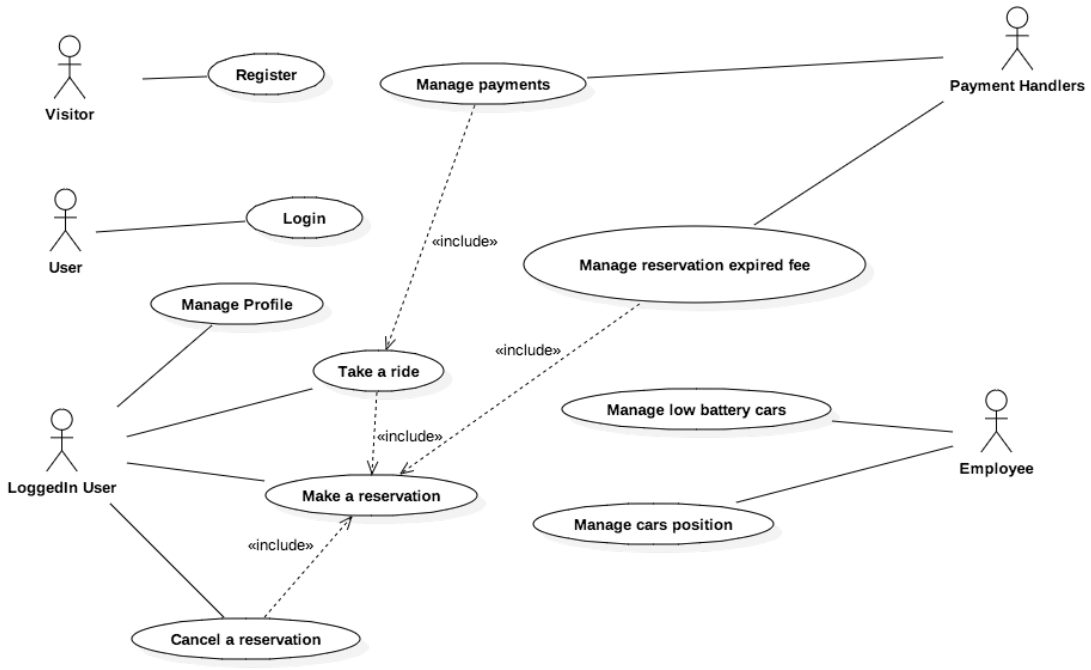


Use cases

During this phase, we identified the following actors:

- User
- System

The following diagram shows the general use cases for each of these actors, in the next subsections we explained them in detail:



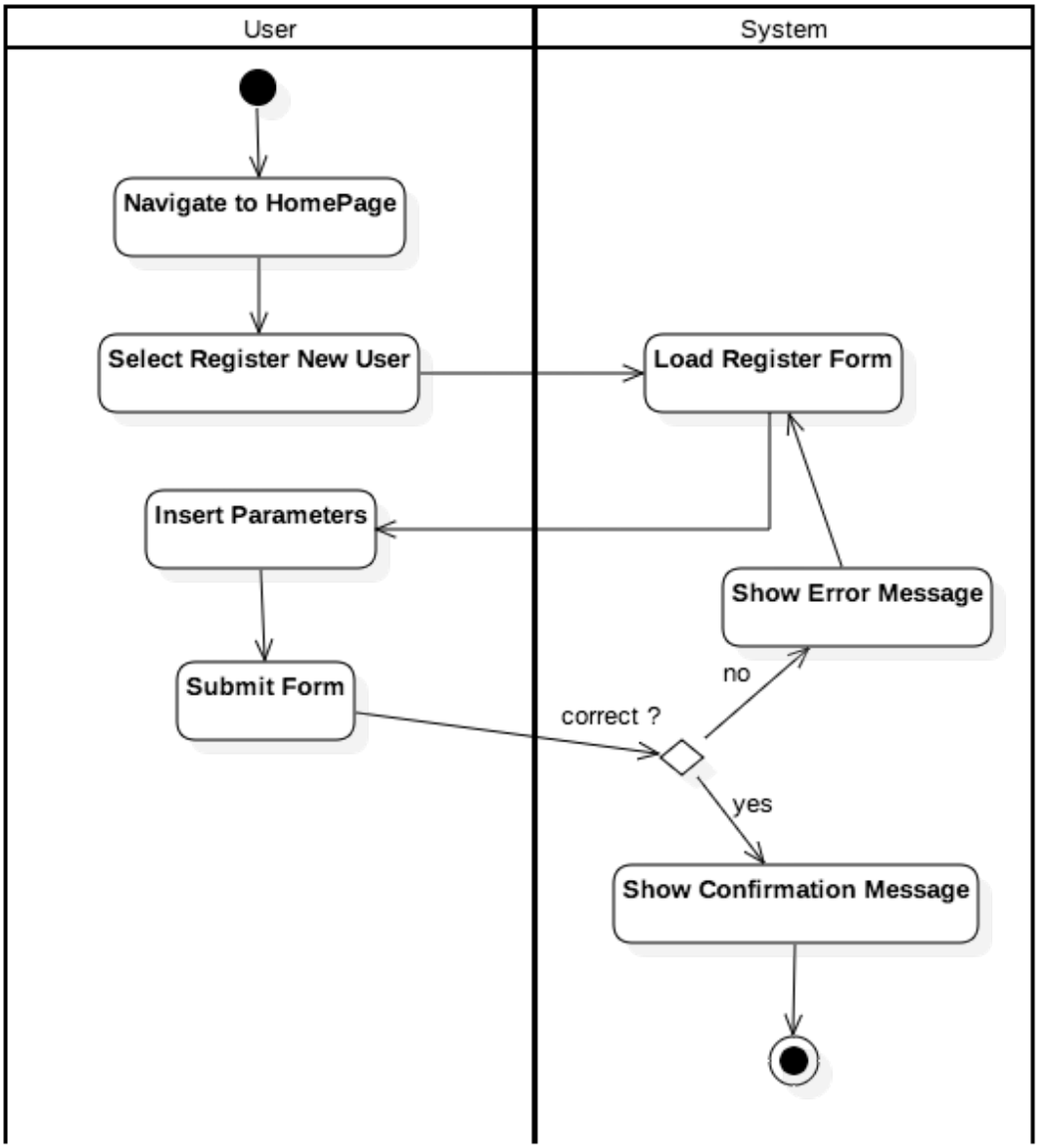
USER:

We have identified the following use cases, which belongs to the user:

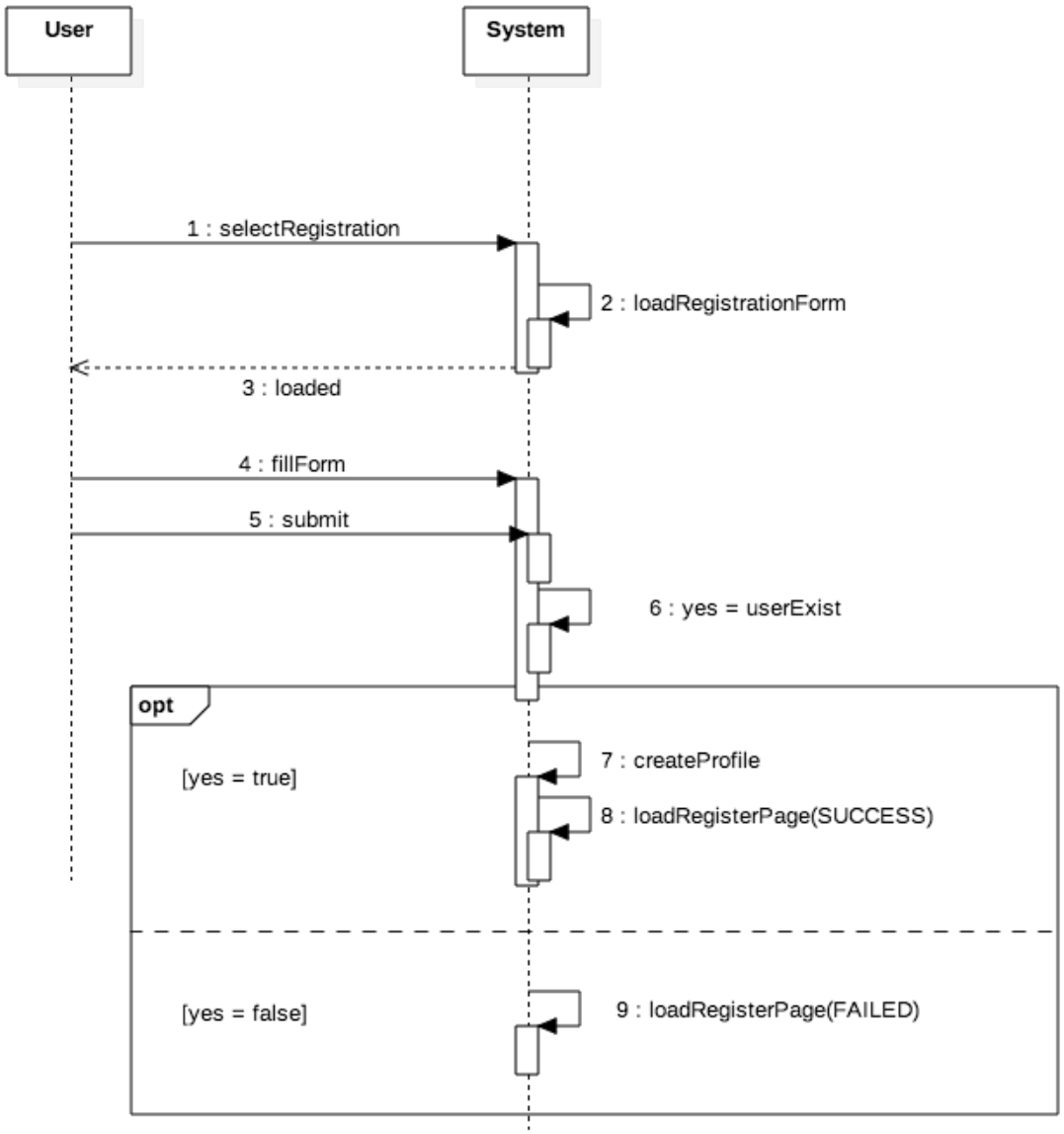
- Registration
- Login
- Manage profile
- Make a reservation
- Cancel a reservation
- Take a ride

Registration	
Description	The guest can register to the service
Goal	[G1] [G2]
Assumptions	[D1][D6]

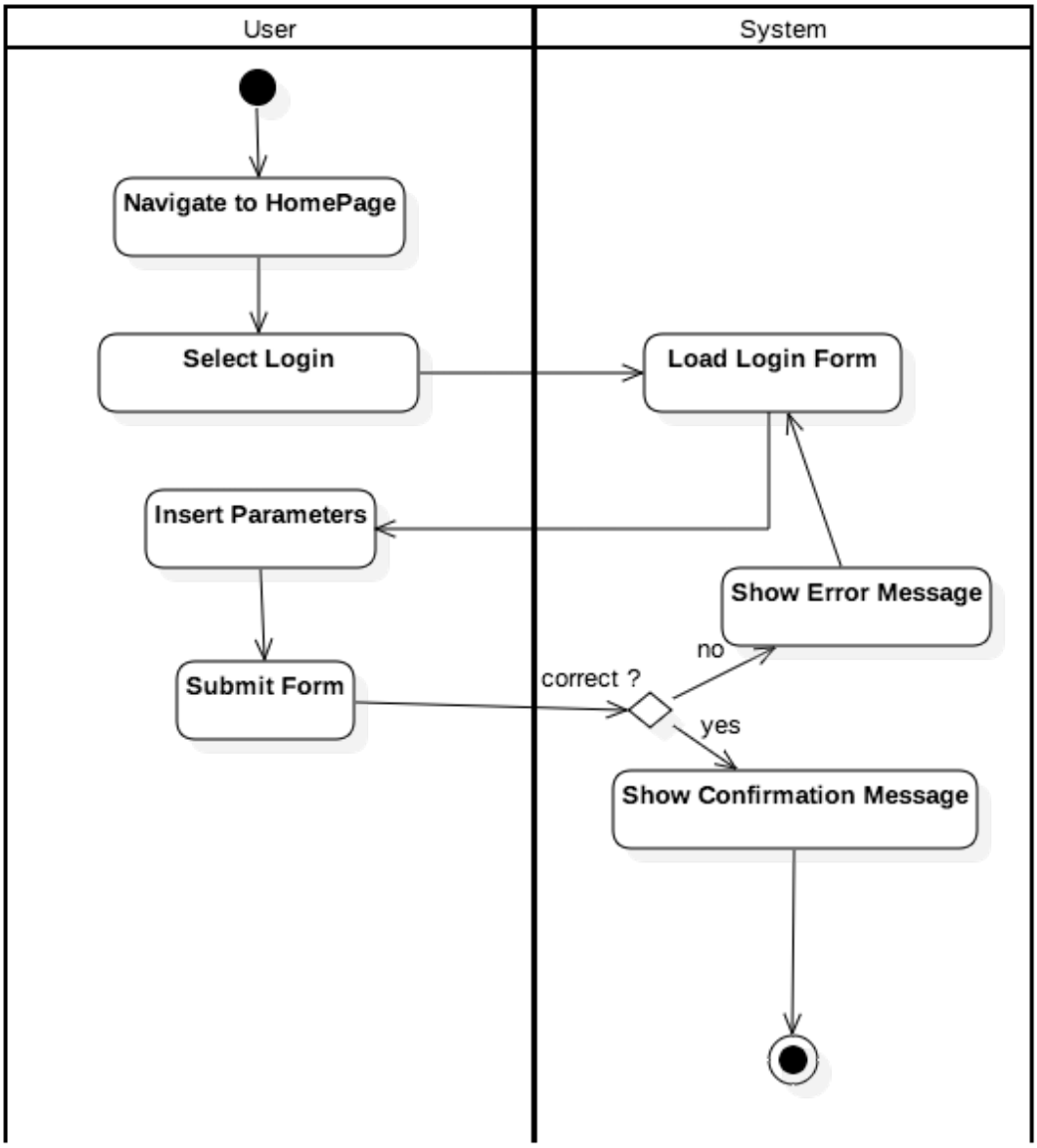
Actors	User
Pre-condition	The guest is not already registered to the service
Post-condition	The guest is a registered user
Flow of event	

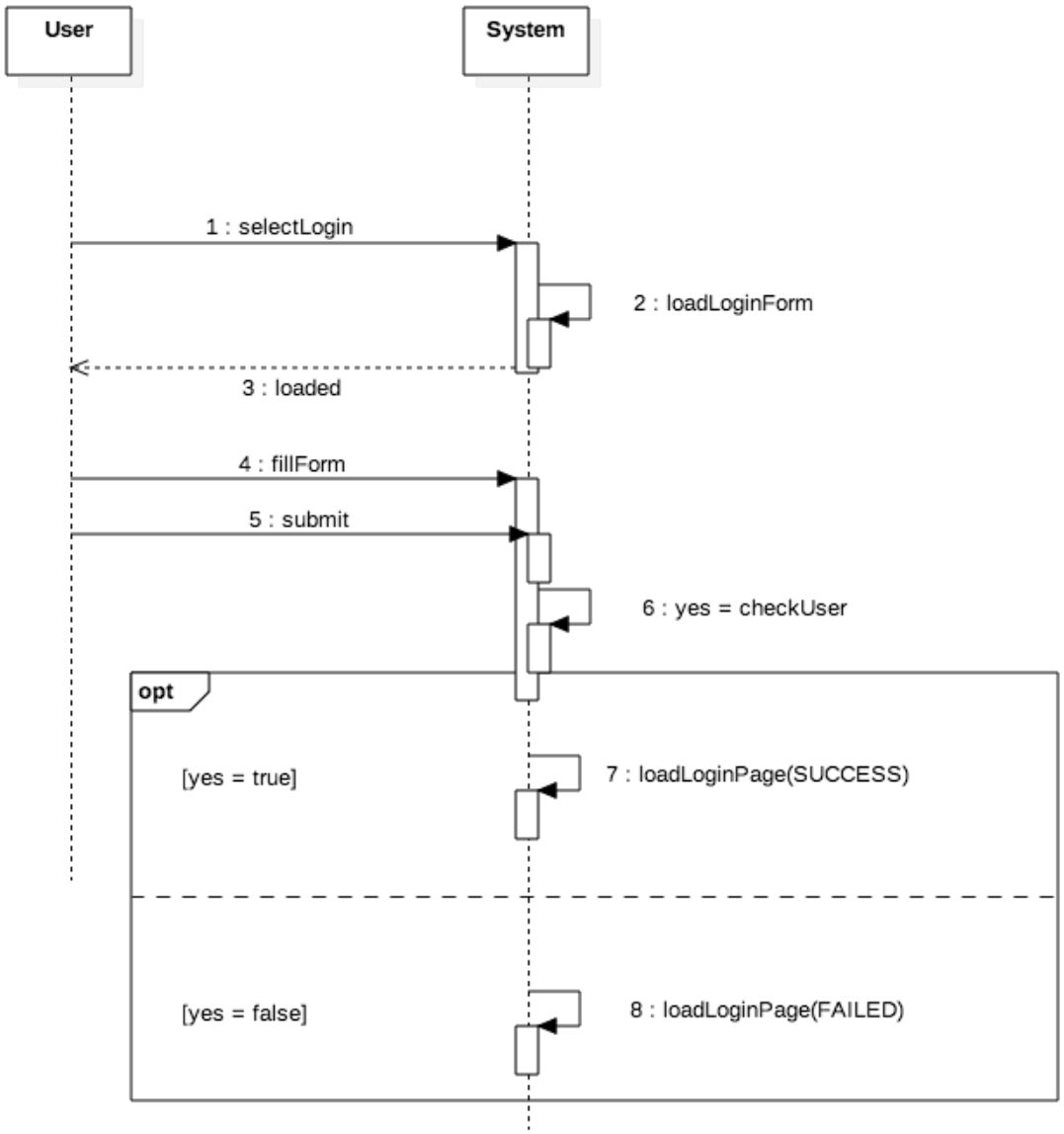


Exception	When some exceptions occur, the registration form is completely restored by the system. If the user is already registered or if the user leave some field empty an exception occurs.

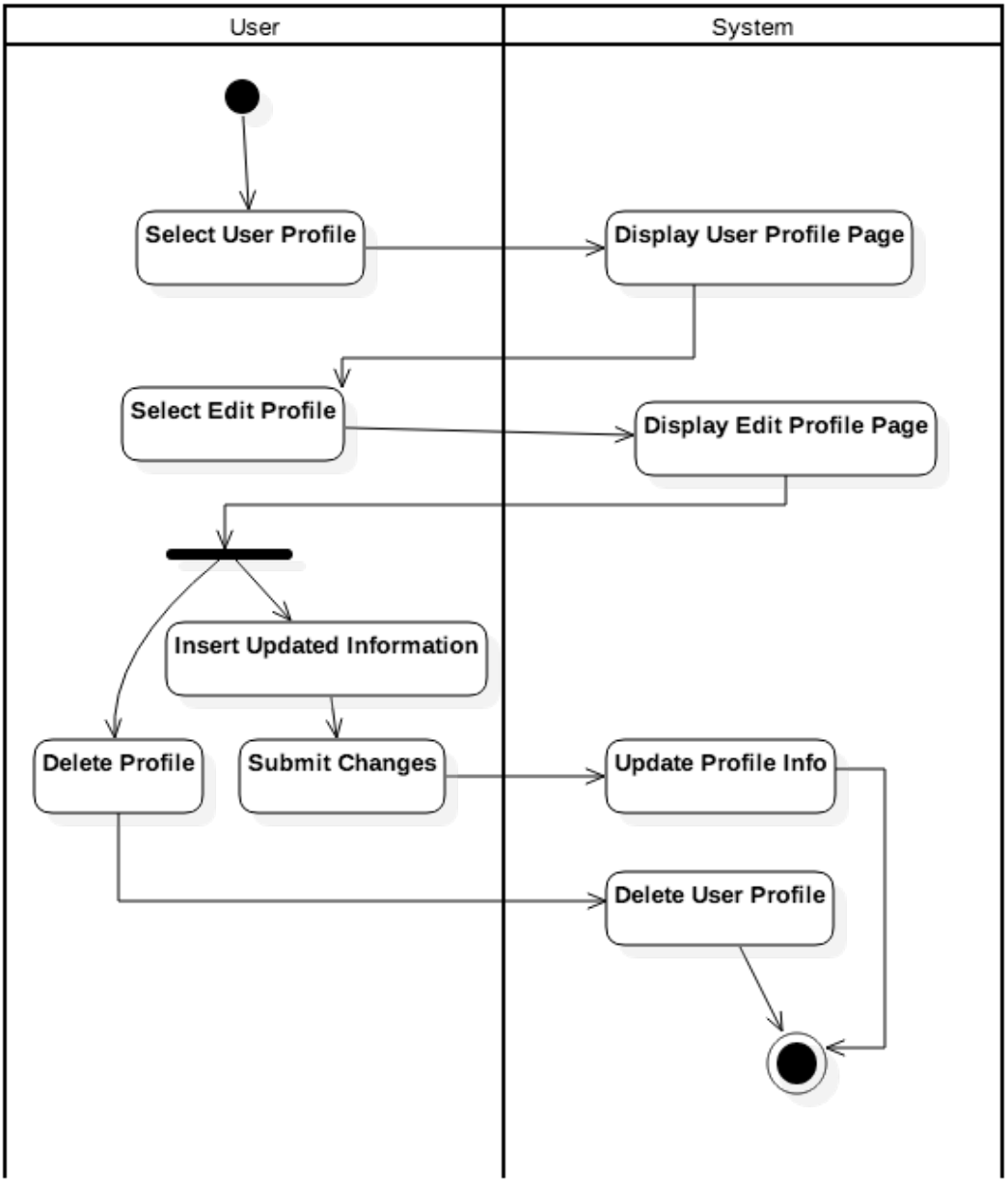


Login	
Description	The registered user can login to the service
Goal	[G1] [G2]
Assumptions	[D1][D6]
Actors	User
Pre-condition	The user is registered to the service
Post-condition	The user is now logged in

Flow of event (see next page)	
	
Exception	If the user is not already registered or if the password does not correspond to the email provided than an exception occurs and the system end the login process. If the guest enters wrong credentials three times in a row, the system will prevent any attempt for the following minute.

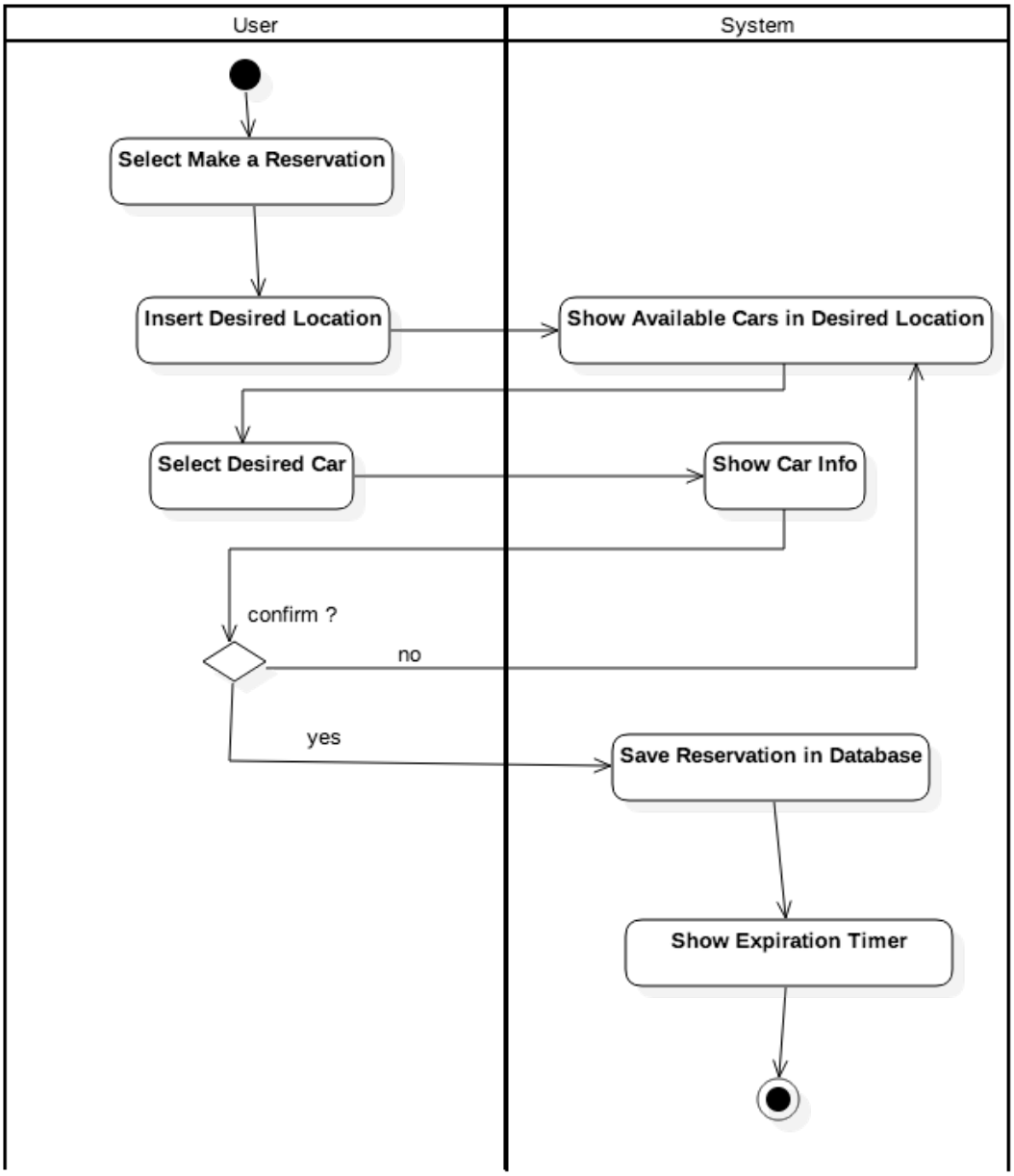


Manage Profile	
Description	The user can insert and modify his credentials
Goal	[G2]
Assumptions	[D1] [D6]
Actors	User
Pre-condition	The user is logged into the service
Post-condition	The personal informations are updated
Flow of event (see next page)	

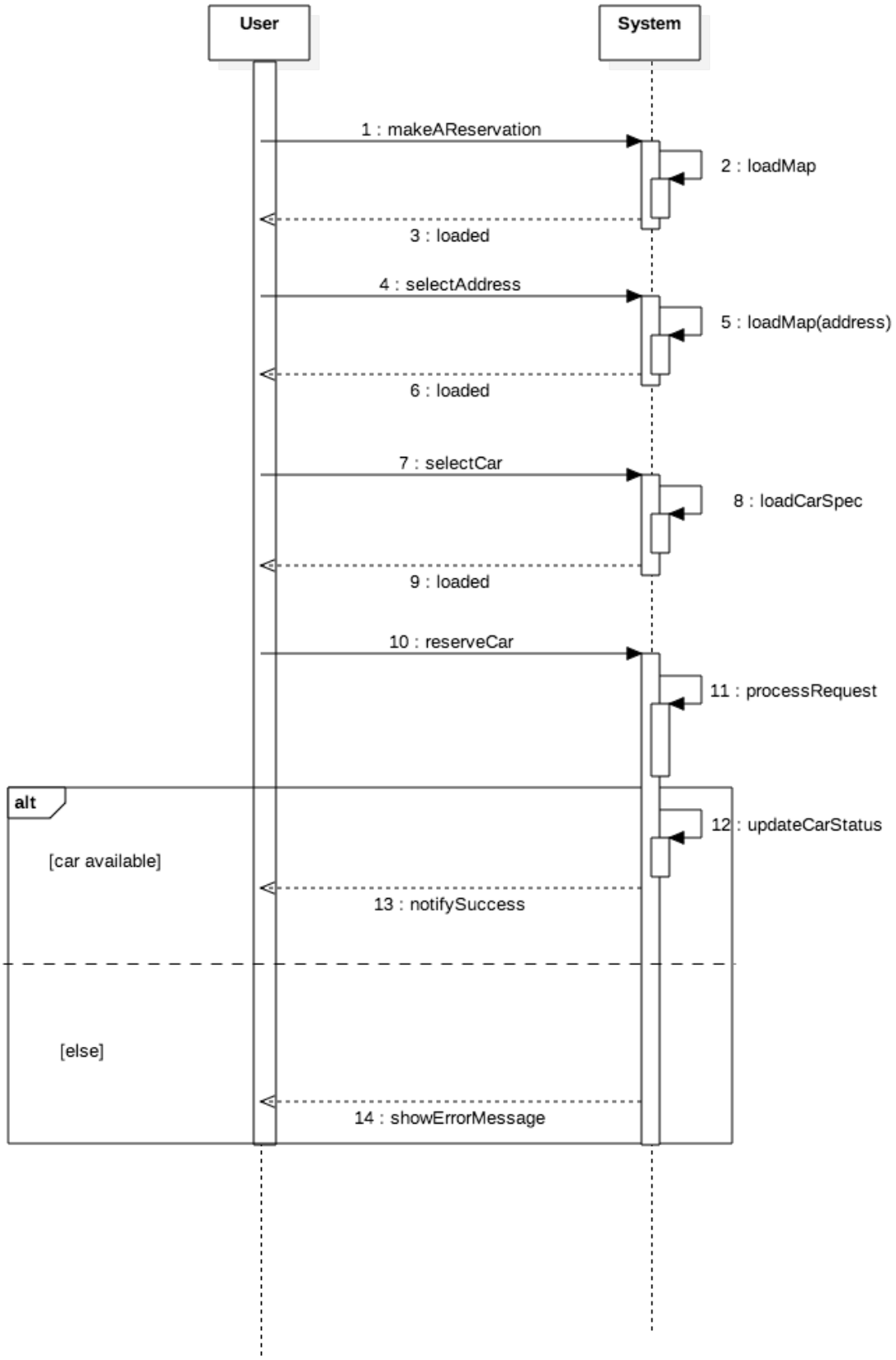


Exception	If the user decides to not confirm his operation than the old profile is restored and the deletion/updating process is aborted.

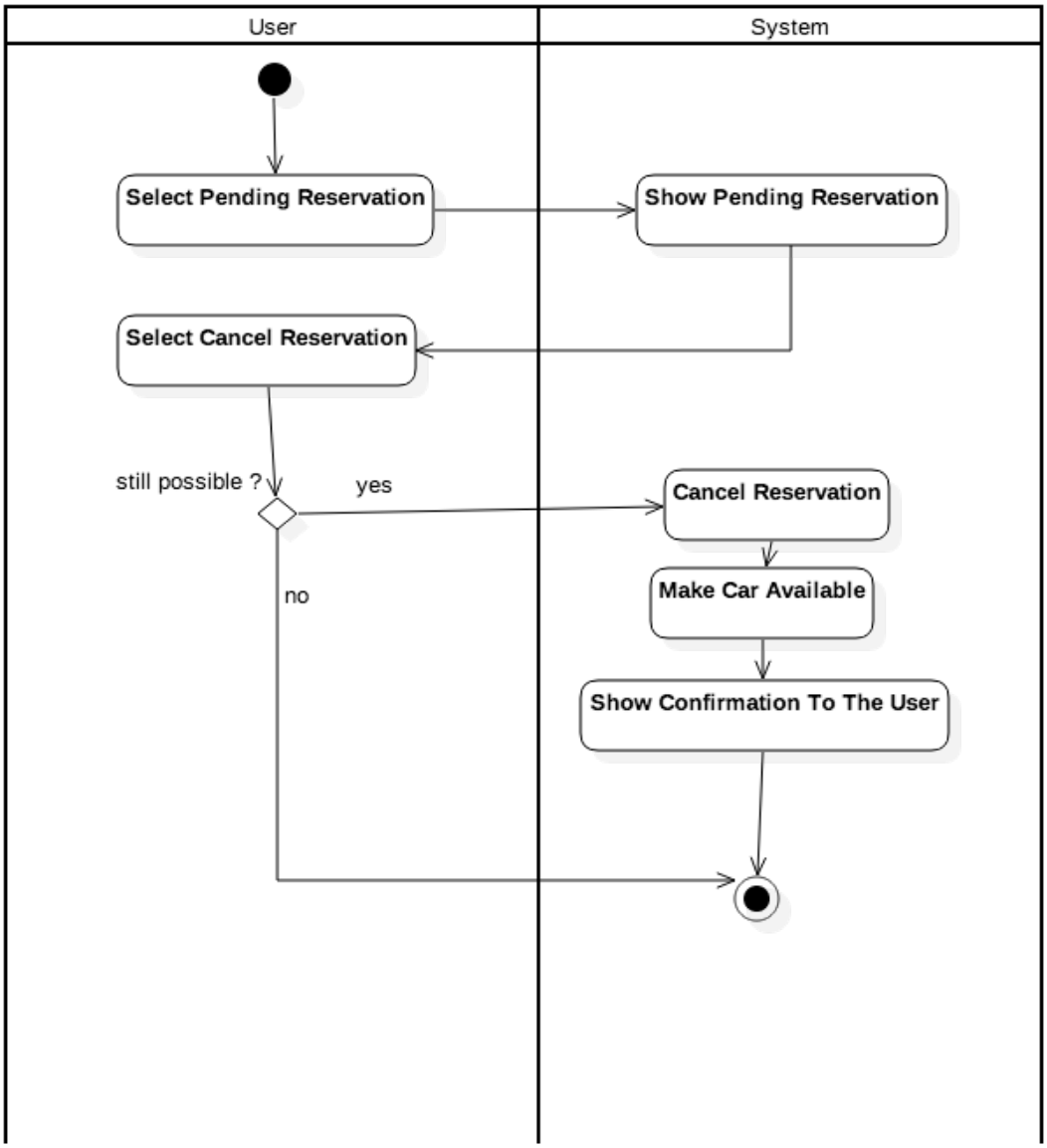
Make a reservation	
Description	The user can reserve a car
Goal	[G4] [G5] [G6]
Assumptions	[D1] [D2] [D3] [D4] [D6]
Actors	User
Pre-condition	The user is logged into the service and the car is available
Post-condition	The car chosen by the user is reserved
Flow of event (see next page)	



Exception	If the user tries to reserve more than one car the system notifies that it is forbidden.


User
System
1 : makeAReservation
2 : loadMap
3 : loaded
4 : selectAddress
5 : loadMap(address)
6 : loaded
7 : selectCar
8 : loadCarSpec
9 : loaded
10 : reserveCar
11 : processRequest
alt
[car available]
12 : updateCarStatus
13 : notifySuccess
[else]
14 : showErrorMessage

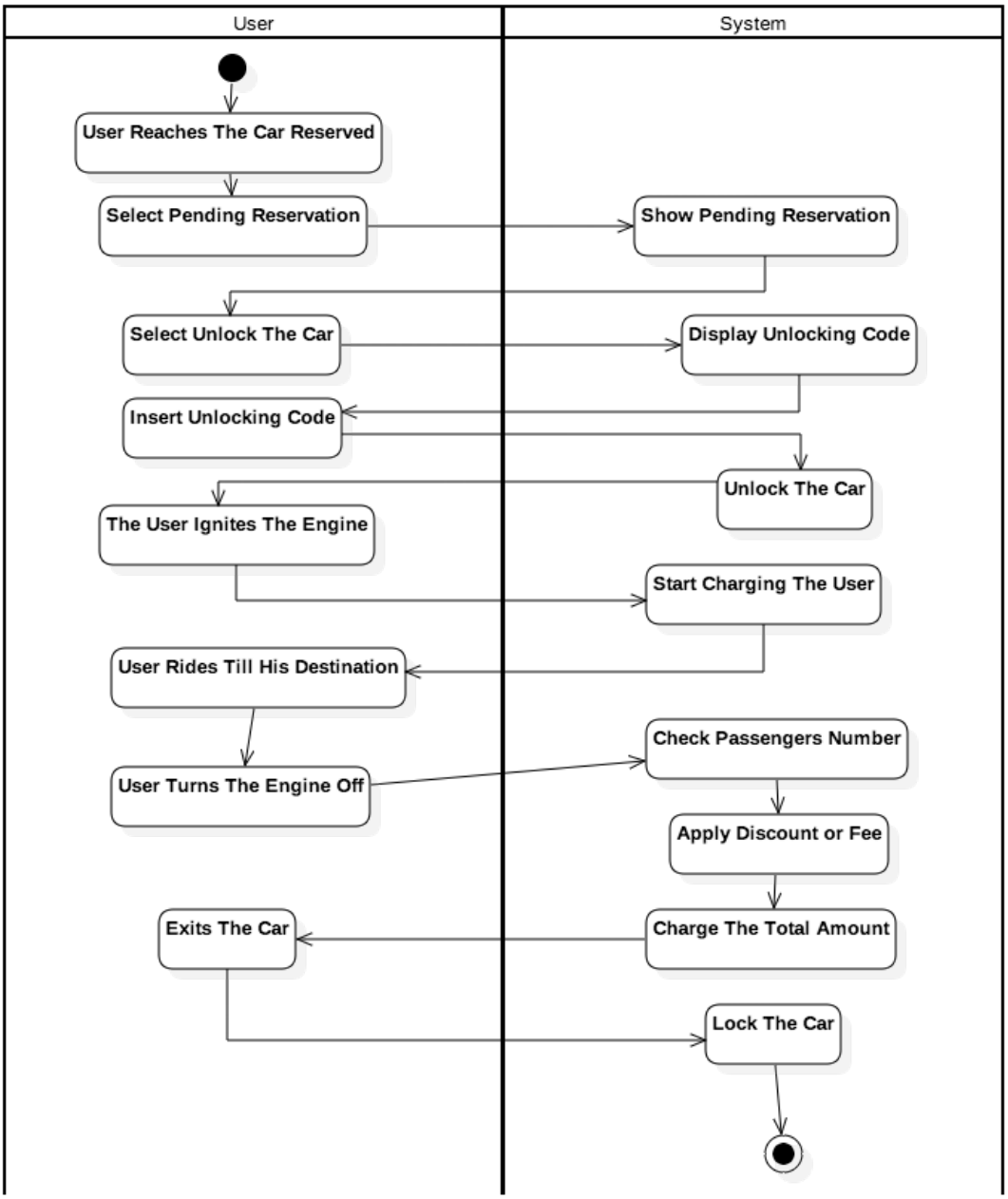
Cancel a reservation	
Description	The user can cancel a previously made reservation
Goal	
Assumptions	[D1] [D6]
Actors	User
Pre-condition	The user has previously made a reservation
Post-condition	The car previously reserved by the user is now available
Flow of event (see next page)	



Exception	If the user changes his mind and does not want to cancel anymore his reservation than the process is aborted and the old reservation is still active.

Take a ride	
Description	The user enters the car until he reaches the desired destination
Goal	[G1] [G2]

Assumptions	[D1] [D2] [D3] [D4] ~~[D5]~~ [D6] [D15.3] [D20]
Actors	User
Pre-condition	The user has reserved and reached an available car
Post-condition	The user exits the car and pays the total amount
Flow of event (see next page)	

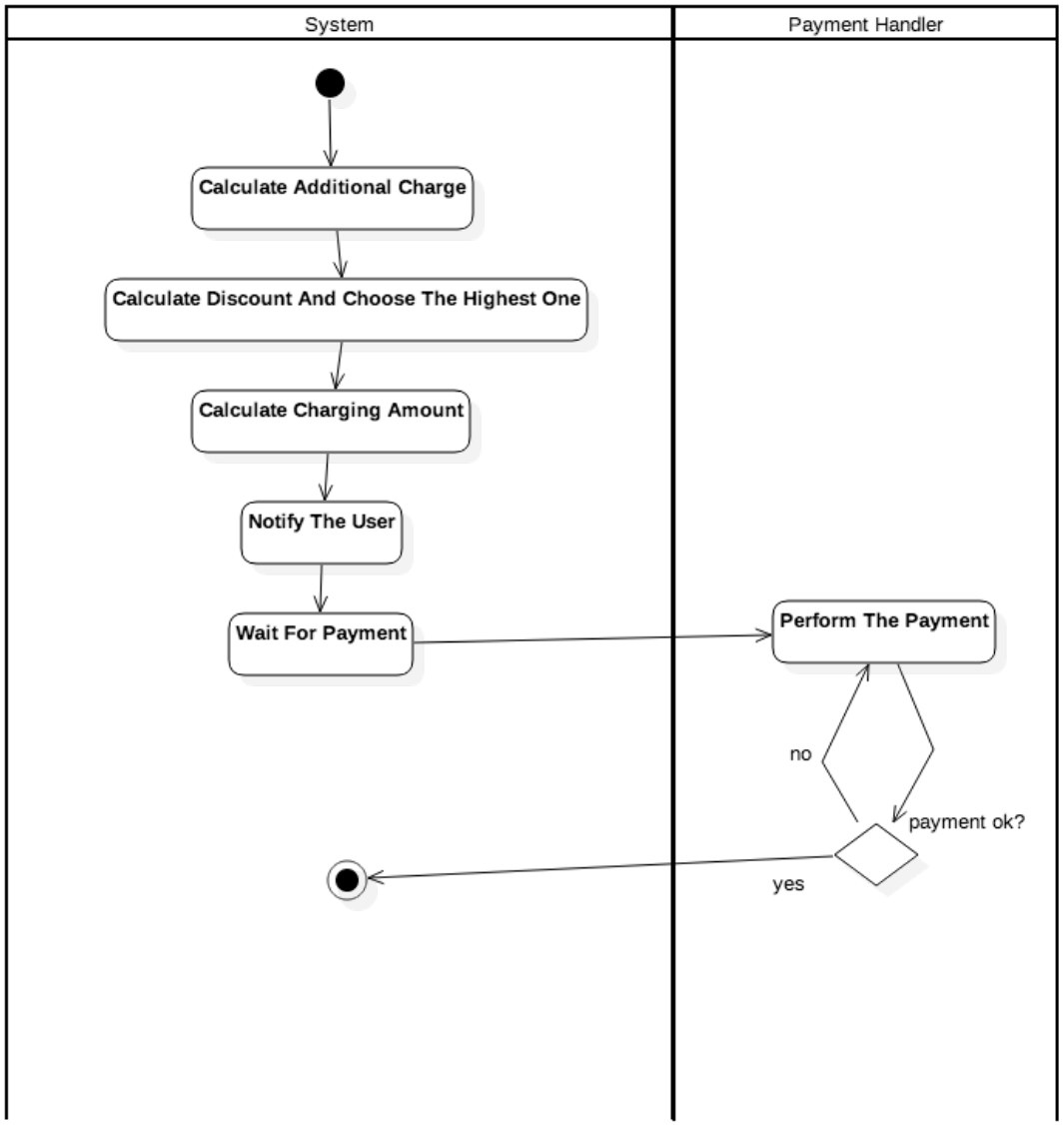


Exception	If the user wants to unlock the car but he is further than 3 m away from it, the unlock car button on the reservation screen will not be active.

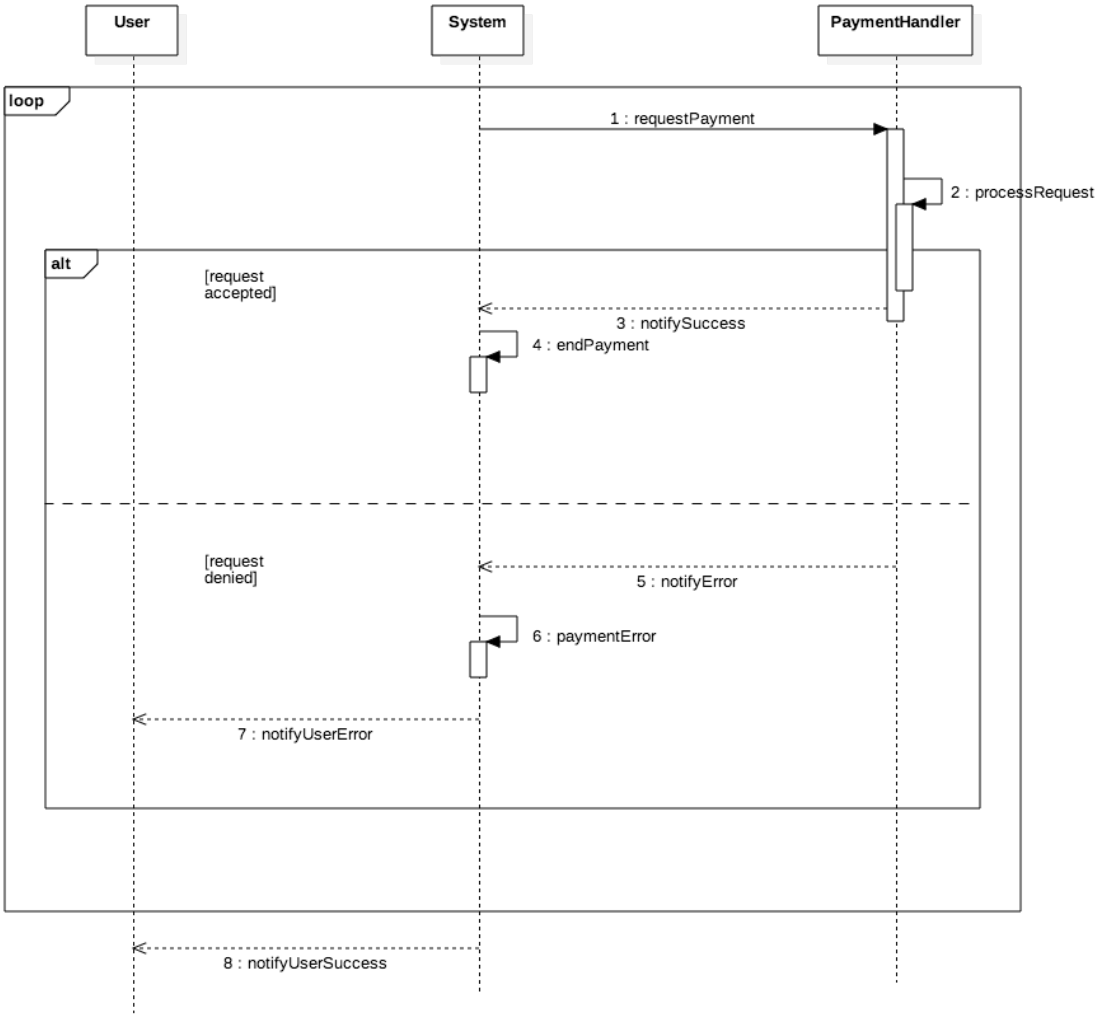
SYSTEM:
We have identified the following use cases, which belongs to the system:

- Manage payments
- Manage reservation expired fee
- Manage low battery cars
- Manage cars position

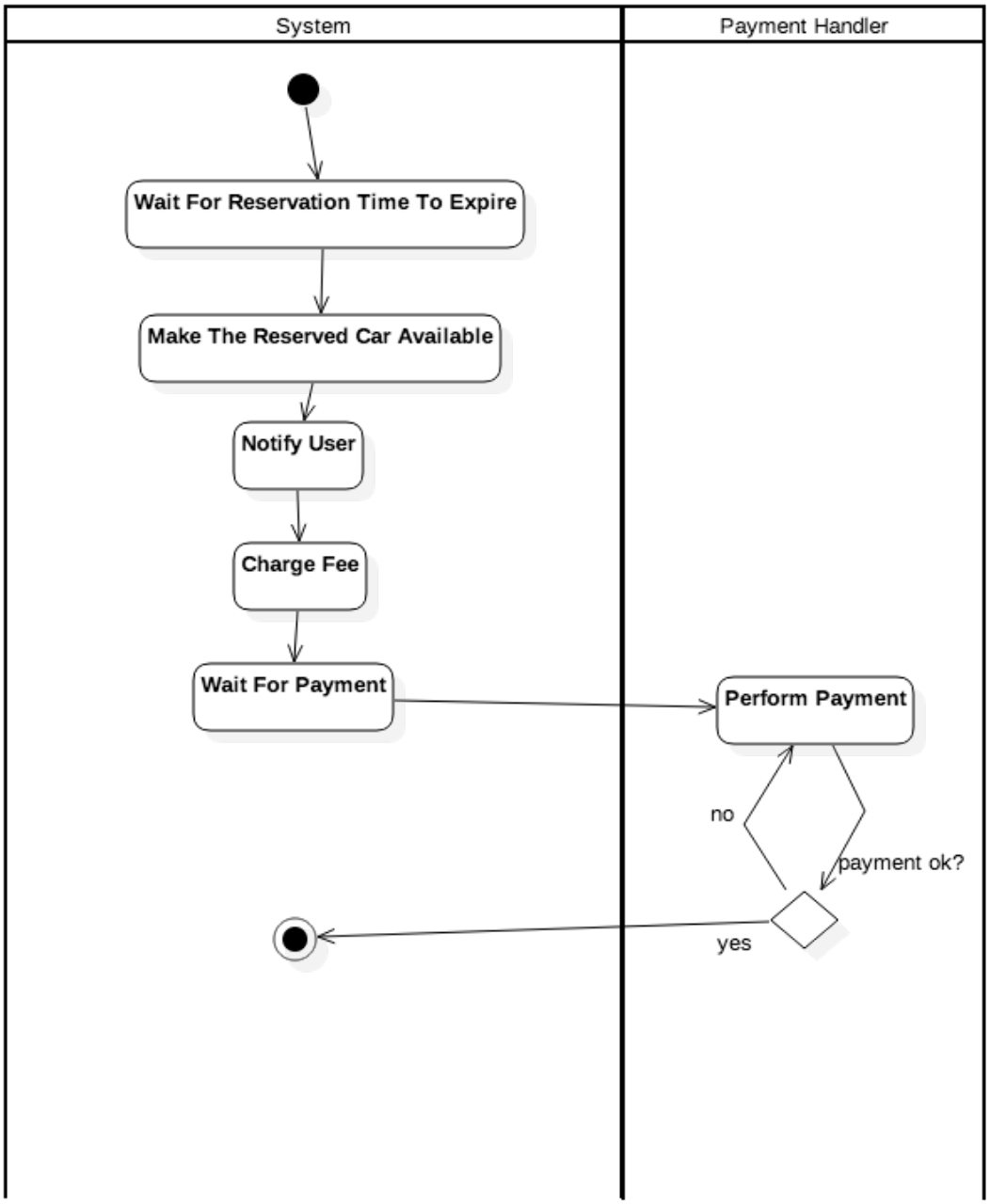
Manage Payments	
Description	The system takes care of the charge of the ride
Goal	[G11] [G13] [G14] [G16] [G17] [G18] [G20] [G21]
Assumptions	[D11] [D12] [D13] [D15] [D21]
Actors	System
Pre-condition	The user has exited the car
Post-condition	The ride total amount has been correctly paid
Flow of event (see next page)	



Exception	If the payment handler blocks the payment request, the system notifies the user and restart the process. If the process does not end well more than three times the user account gets locked and prevented from taking any other reservation in the future.

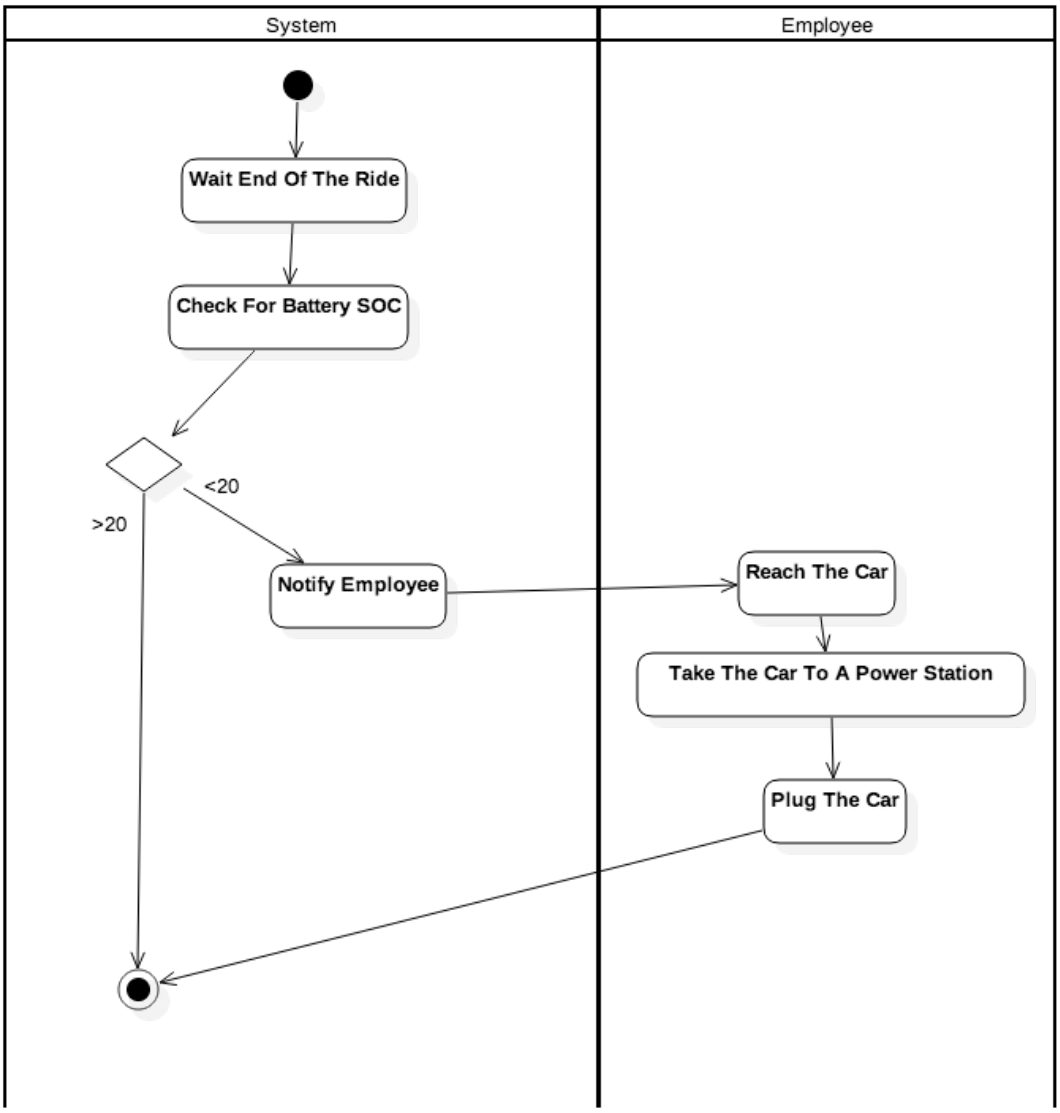


Manage reservation expired fee	
Description	The system charges the user of a fee if he does not pick up the car within 1h
Goal	[G9]
Assumptions	[D11] [D12]
Actors	System
Pre-condition	The user has reserved a car without using it within 1h
Post-condition	The user has been correctly charged of the fee
Flow of event (see next page)	



Exception	If the payment handler blocks the payment request, the system notifies the user and restart the process. If the process does not end well more than three times the user account gets locked and prevented from taking any other reservation in the future.

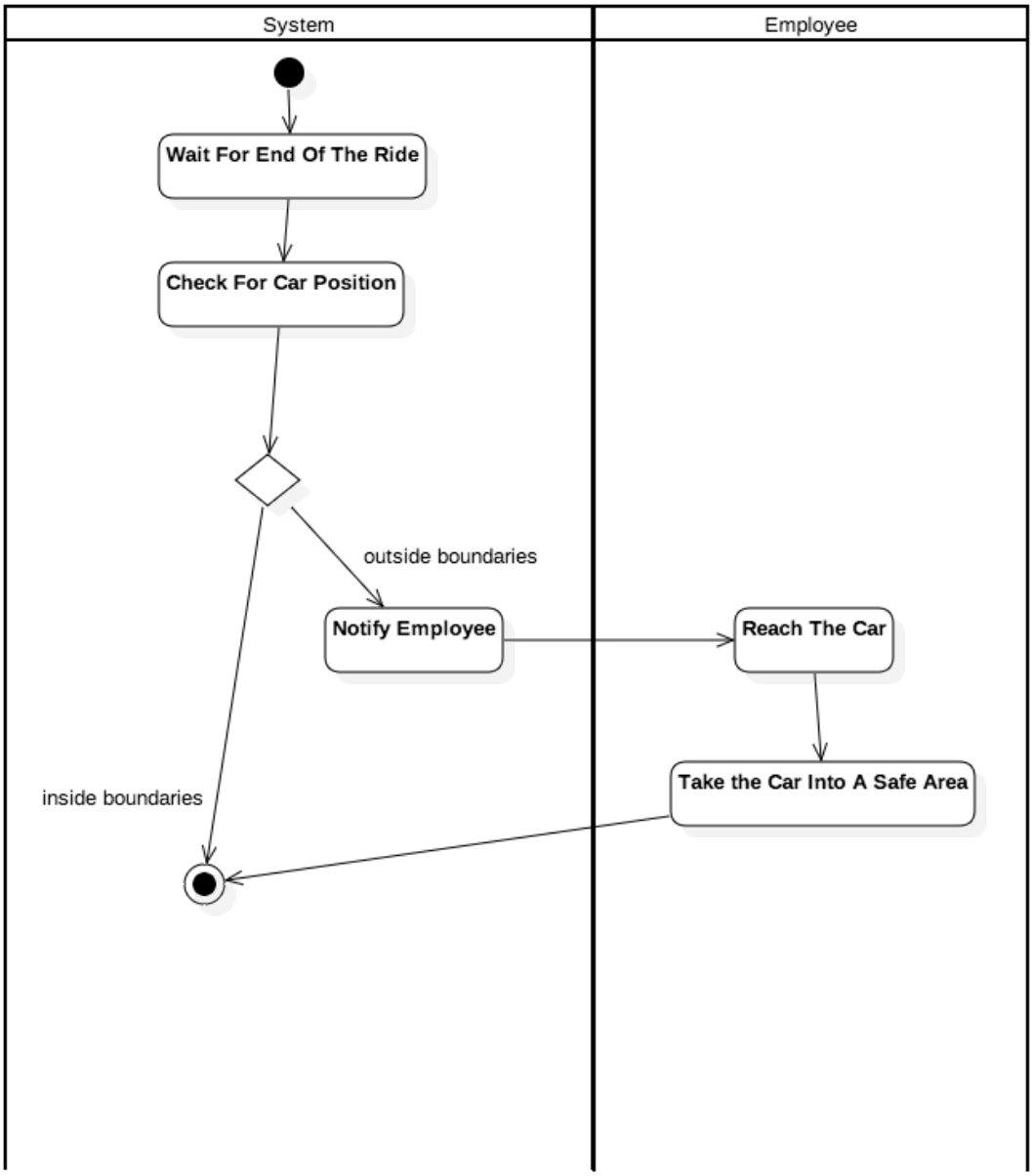
Manage low battery cars	
Description	The system instructs the employees of the low battery cars so they can plug them in a power grid station
Goal	[G19]
Assumptions	[D7] [D8] [D9] [D10] [D16] [D18]
Actors	System
Pre-condition	The car has less than 20% of battery and it is not plugged in
Post-condition	The car is plugged into a power station
Flow of event (see next page)	



Exception	none

Manage cars position	
Description	The system takes care of the cars that are out of the city boundaries
Goal	
Assumptions	[D8] [D9] [D10] [D22]
Actors	System

Pre-condition	The user leaved a car out of the city boundaries
Post-condition	The car is now available inside the city boundaries
Flow of event (see next page)	



Exception	none

Performance requirements

The system must meet high standards of performance with regard to the computation of the algorithms of car reservation and to the bandwidth management, because of the many requests, which the system has to manage in real time.
As we mention before it is necessary to support a high level of concurrent users and concurrent car reservation.
These following are example of important requirements, which the system should satisfy:

a. The system shall be able to process 1000 reservation transactions per minute in peak load.

b. The system must not have an upper-bound as far as the total number of registered users is concerned.

c. In standard workload, the CPU usage shall be less than 50%, leaving 50% for background jobs.

d. Production of a reservation shall take less than 5 seconds for 95% of the cases.

e. The general navigation of the mobile application shall take at most 1 second.

f. The car unlock requests shall take less than 2 seconds for 100% of the cases.

Logical database requirements

A relational database is a perfect solution for this project. It must save the entities identified in the analysis diagram. It will manage users, ride and reservations. This server will be subjected to high stress, as numerous request and reservations can be sent at the same time. In order to satisfy the performance requirements, it will be necessary to manage buffer memories to improve the read/write operations.

Design constraints

The system must be designed and implements in JEE technology. The service is a work-in-progress application so that it needs to be implemented in the best way to make new changes in the future.

Standard compliance

The system should be compliant with national standards of privacy and security, especially linked with traffic laws and the use of mobile devices. The service must satisfy all the UNI-ISO and European regulations.

Software system attributes

Reliability

The system must assure the integrity and the durability of car reservation and unlocking, nevertheless it must assure the integrity of personal information. The reliability of the cars built-in computer must also be taken into account.
Here some reliability measure the system should satisfy:

a. The precision of car localization shall be at least 98%.

b. The system defect rate shall be less than one failure per 1000 hours of operation.

c. No more than one per 1000000 database transactions shall result in a failure requiring a system restart.

d. The estimated loss of data in case of a disk crash shall be less than 0.01%.

e. The system shall be able to handle up to 10000 concurrent users when satisfying all their requirements and up to 25000 concurrent users with browsing capabilities.

Availability

- The system must be online 24/7 during all the year.
- The system PowerEnjoy shall have an availability of 999/1000 or 99%.
- The system shall not be unavailable more than 1 hour per 1000 hours of operation.
- Less than 20 seconds shall be needed to restart the system after a failure 95% of the time. (This is a MTTR (Mean-Time to Repair) requirement)

Security

The software must protect both users' email and password as their travels in the city. These requirements can be achieved using high quality secure protocols and effective cryptography algorithm, such as AES or RSA.
Other security requirements are:

a. Unauthorized access to the system and its data is not allowed.

b. Ensure the integrity of the system from accidental or malicious damage.

c. The system's data administrator may only change the access permissions for system data.

d. All system data must be backed up every 24 hours and the backup copies stored in a secure location, which is not in the same building as the system.

e. All external communications between the system's data server and clients must be encrypted

f. When an account is deleted, all its information, rides and reservations are removed.

g. At least 99% of intrusions shall be detected within 10 seconds.

h. The system must stop providing service to legitimate users while under denial of service attack (resistance to DoS attacks)

Maintainability

The system needs maintainability in:

a. Active/inactive users

b. Servers files system backup and restoring

c. Every program module must be assessed for maintainability. 70% must obtain "highly maintainable" and none "poor".

d. The cyclomatic complexity of code must not exceed 7. No method in any object may exceed 200 lines of code.

e. Installation of a new version shall leave all database contents and all personal settings unchanged.

f. The product shall provide facilities for tracing any database field to places where it is used.

g. The delivered system shall include unit tests that ensure 100% branch coverage.

h. Development must use regression tests allowing for full retesting in 12 hours.

Portability

The system can be installed in every mobile phones, which can be localized and can connect to Internet, but also in every computer, which supports a Java Virtual Machine and has an internet browser.
The portability is the key element of the entire project. People can make a request from everywhere through the device they prefer.

Alloy

In this section we present a model for the system that represents and verifies some properties of the described software product. Mainly it refers to the synchronization mechanism among rides, reservations, users and cars, giving less emphasis to other aspects as the battery state of charge, positions and payments.

Alloy model

```
open util/boolean

sig RegisteredUser {
	position: lone Position,
	ride: set Ride,
	reservation: set Reservation
}{}

sig Position {
	longitude: one Int,
	latitude: one Int
} {
	longitude >= 0
	latitude >= 0
}

sig Ride{
	startingTime: one Int,
	endingTime: one Int,
	chargeAmount: one Int,
	endingPosition: one Position,
	passengersNum: one Int,
	endBatterySOC: one Int
} {
	startingTime > 0
	endingTime > 0
	endingTime > startingTime
	chargeAmount > 0
	passengersNum >= 1
	passengersNum <= 4
	endBatterySOC >= 0
	endBatterySOC <= 100
}

sig Reservation {
```

```
        reservationTime: one Int,
        pickupTime: one Int,
        isActive: Bool,
        isExpired: Bool,
        ride: lone Ride,
        car: one Car
} {
        reservationTime > 0
        pickupTime > 0
        pickupTime > reservationTime
        isExpired = True implies isActive = False
        isActive = True implies car.isAvailable = False
}

sig Car {
        plate: one Int,
        batterySOC: one Int,
        isAvailable: Bool,
        isLocked: Bool,
        isEngineOn: Bool,
        isPlugged: Bool,
        position: one Position
} {
        plate > 0
        batterySOC >= 0
        batterySOC <= 100
        batterySOC < 20 implies isAvailable = False
        (position.longitude > 500 or position.latitude > 500) implies isAvailable =
False  //car out of city boundaries
        isLocked = False implies isAvailable = False
}

// a ride cannot happen without a previous reservation
fact RideHasReservation{
   all r: Ride | (one res: Reservation | r in res.ride)
}

// a ride is performed by one user
fact RideHasUser{
   all r: Ride | (one u: RegisteredUser | r in u.ride)
}

// a reservation must be performed by only one user
```

```
fact ReservationHasUser {
  all res: Reservation | one u: RegisteredUser | res in u.reservation
}


//a ride is performed by the same user that reserved it
fact RideReservationUser {
 all r: Ride , res: Reservation | r = res.ride implies (one u:  RegisteredUser | r in u.ride
and res in u.reservation)
}

//a user cannot have simoultaneous rides
fact userRidesTimeConcistency{
  all disj res1,res2: Reservation | (one u: RegisteredUser | res1 in u.reservation and res2 in
u.reservation) implies (res2.reservationTime > res1.ride.endingTime or
res1.reservationTime > res2.ride.endingTime)
}

//time consistency between ride and reservation
fact timeConsistecyRideRes{
  all res: Reservation , r: Ride |  r = res.ride implies ( r.startingTime = res.pickupTime )
}

//Unavailable car conditions
fact UnavailableCar {
        all c: Car | (c.isAvailable = False) implies (c.isEngineOn = False and (no res:
Reservation | res.isActive = True and res.car = c))
}

//Available car conditions
fact AvailableCar {
        all c: Car | (c.isAvailable = True) implies (c. isLocked = True and c.isEngineOn =
False  and (no r: Reservation | r.car = c and r.isActive = True))
}

//Uniqueness of the car
fact UniqueCar {
        no disj c1, c2: Car | (c1.plate = c2.plate)
}

//a car cannot have simoultaneous rides (or rides while is reserved)
fact carRidesTimeConsistency{
```

```
   all disj res1,res2: Reservation | res1.car = res2.car implies (res2.reservationTime >
res1.ride.endingTime or res1.reservationTime > res2.ride.endingTime)
}

//No cars in same position
fact EachCarHasItsOwnPosition{
        no disj c1, c2: Car | c1.position = c2.position
}

//We show only meaningful position
fact OnlySpecifiedPosition{
        all p: Position | ((some u: RegisteredUser | u.position = p) or (one c: Car |
c.position = p))
}

pred show {
}

run show for 10
```

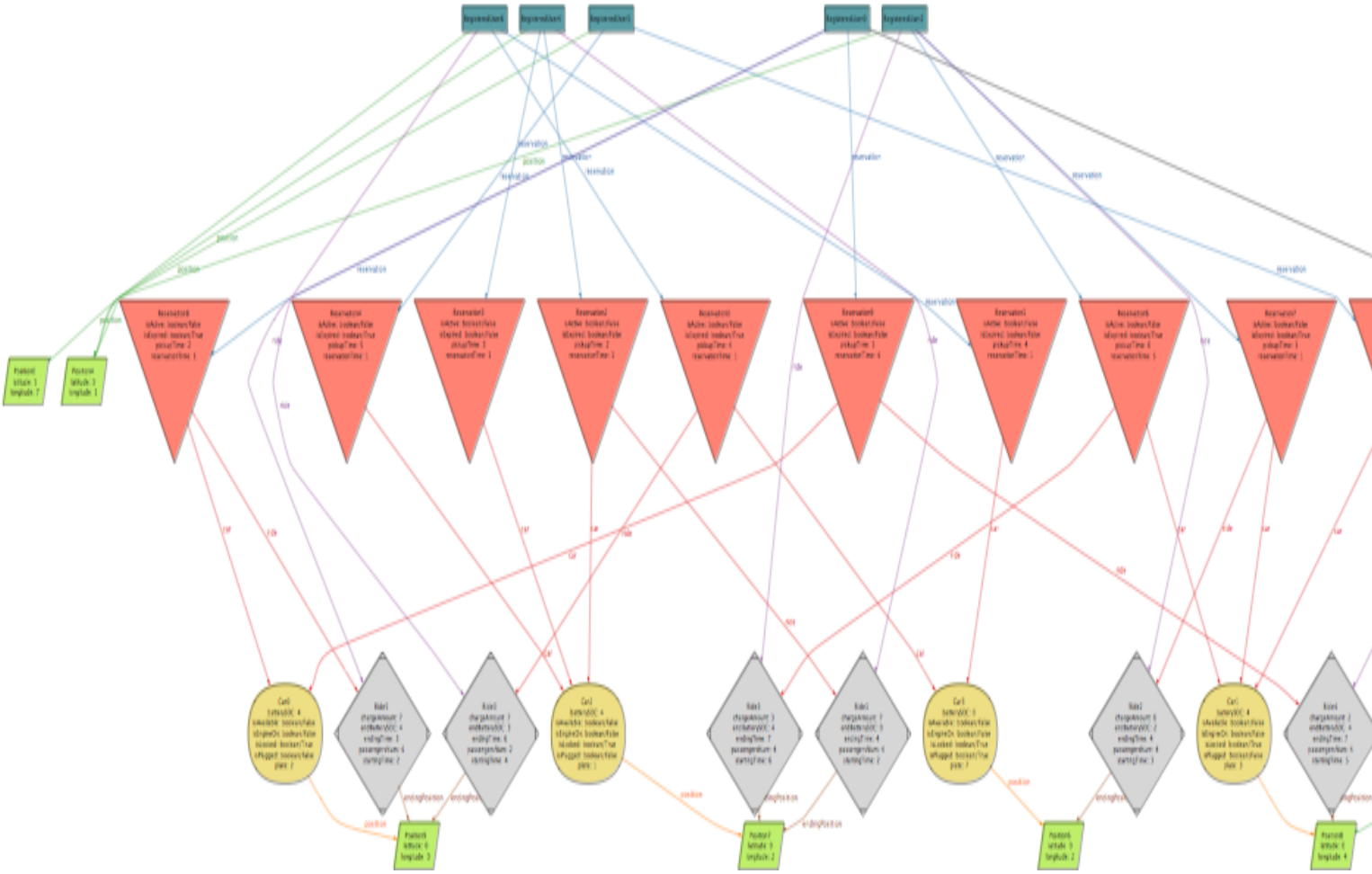
Generated worlds

Here there are some examples of the worlds generated with the presented model.

World 1

This world presents:

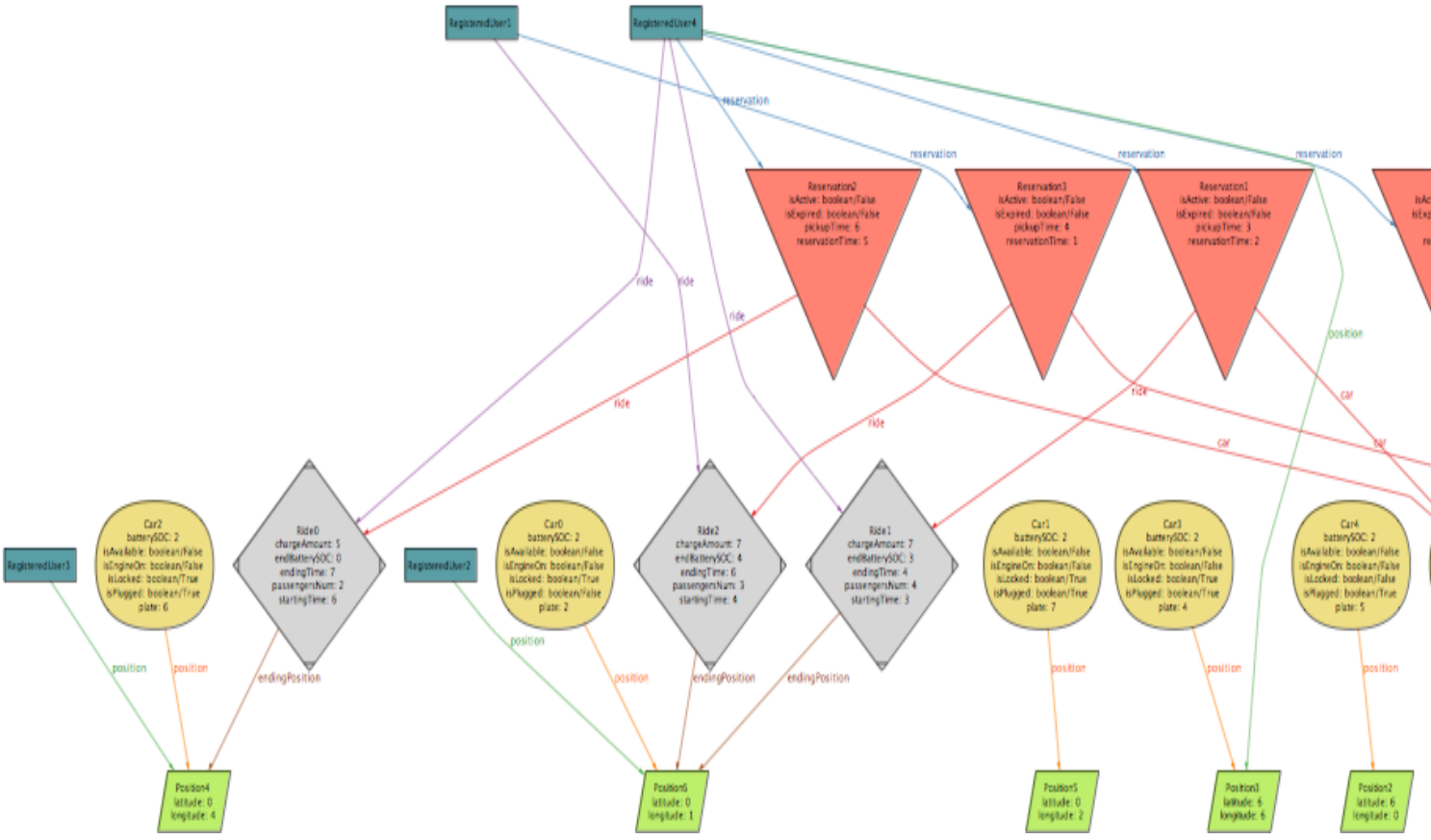
- 5 active registered users
- 10 reservations
- 6 rides
- 4 cars



World 2

This world presents:

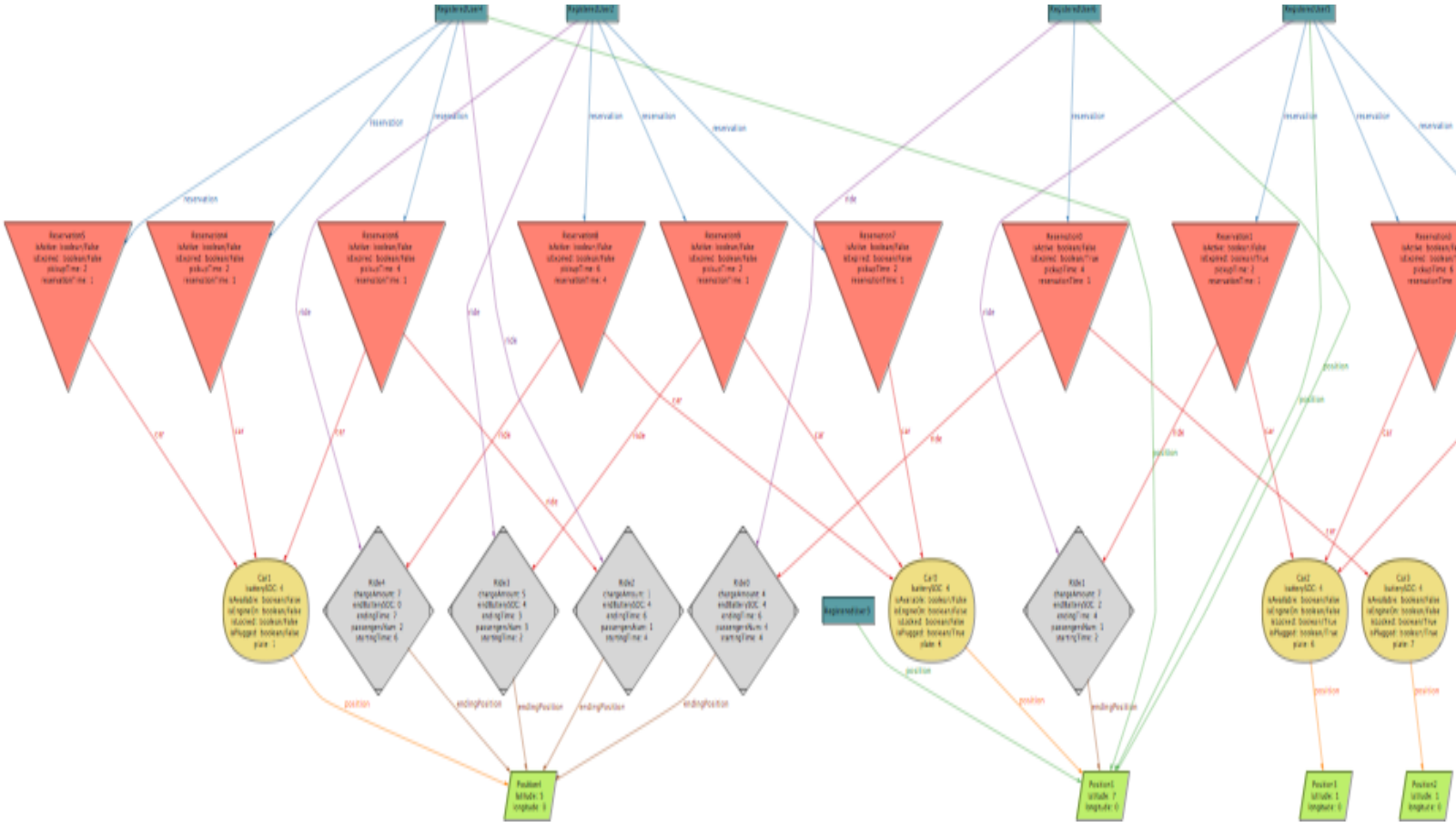
- 2 active registered users
- 4 reservations
- 3 rides
- 7 cars of which only 2 are actually used



World 3

This world presents:

- 4 active users
- 10 reservations
- 5 rides
- 4 cars and each of them is use this time



World 4

This world presents:

- 6 active users
- 10 reservations
- 5 rides
- 6 cars (only one not used)

Hours of work

Madaffari Federico

- 2/11/16 (1h 30m)
- 3/11/16 (30m)
- 4/11/16 (3h) group work
- 7/11/16 (3h)
- 8/11/16 (4h)
- 9/11/16 (3h)
- 10/11/16 (2h)
- 11/11/16 (3h) group work
- 12/11/16 (3+6+5) (morning-day-night) group working
- 13/11/16 (7h)

Mandrioli Claudio

- 31/10/16 (2h)
- 2/11/16 (30m)
- 4/11/16 (3h) group work
- 5/11/16 (3h)
- 6/11/16 (3h)
- 7/11/16 (4h)
- 8/11/16 (2h)
- 10/11/16 (2h)
- 11/11/16 (3h) group work
- 12/11/16 (3+6+5) (morning-day-night) group working
- 13/11/16 (8h)