



Politecnico di Milano

## RASD - Requirements And Specifications Document

November 11, 2016

### **Version 1.1**

Authors:

- Domenico FAVARO (Mat. 837995)
- Matheus FIM (Mat. 876069)
- Caio ZULIANI (Mat. 877266)

Prof. Elisabetta DI NITTO

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Problem Definition - PowerEnJoy . . . . .	3
1.2	Goals . . . . .	4
1.3	Glossary . . . . .	6
1.4	Domain Assumptions . . . . .	7
1.5	Constraints . . . . .	8
1.6	Stakeholders . . . . .	9
1.7	Reference Documents . . . . .	9
<b>2</b>	<b>Proposed System</b>	<b>10</b>
<b>3</b>	<b>Actors</b>	<b>11</b>
3.1	User . . . . .	11
3.2	CRM . . . . .	11
<b>4</b>	<b>Requirements</b>	<b>11</b>
4.1	Functional Requirements . . . . .	11
4.1.1	<b>User Requirements:</b> . . . . .	11
4.1.2	<b>CRM Requirements:</b> . . . . .	15
4.2	Non Functional Requirements . . . . .	16
4.2.1	User Interface . . . . .	16
4.2.2	Documentation . . . . .	20
4.2.3	System Architecture . . . . .	21
4.2.4	Hardware Consideration . . . . .	21
4.2.5	Incident Handling . . . . .	21
4.2.6	Security . . . . .	22
<b>5</b>	<b>Scenario Identifying</b>	<b>22</b>
<b>6</b>	<b>UML Modeling</b>	<b>30</b>
6.1	Use Case Diagram . . . . .	30
6.1.1	Use Cases Description . . . . .	31
6.2	Class Diagram . . . . .	34
6.3	Sequence Diagrams . . . . .	35
6.4	Statechart Diagram . . . . .	38
6.5	Activity Diagram . . . . .	39
<b>7</b>	<b>Alloy Modeling</b>	<b>40</b>
7.1	Model . . . . .	40
7.1.1	Signatures . . . . .	40
7.1.2	Facts . . . . .	42
7.1.3	Login Creation . . . . .	42

7.1.4	Making a new Reservation . . . . .	43
7.2	Alloy Result . . . . .	45
7.3	Generated World . . . . .	45
<b>8</b>	<b>Used Tools</b>	<b>48</b>
<b>9</b>	<b>Hours of Work</b>	<b>49</b>
<b>10</b>	<b>Changelog</b>	<b>50</b>

# **1 Introduction**

## **1.1 Problem Definition - PowerEnJoy**

The requested System denoted PowerEnJoy, is a car-sharing service that exclusively employs electric Cars.

As a car-sharing service it should allow the use of a Car via Reservation, taking it and bringing it back to determined parking areas and paying for the use made.

The System should allow Users to register so it can authorize them to use the car-sharing services, and allow them to reserve an available Car near their location or on a given address, so they can use it for a ride.

The Users will provide their credentials and payment info so all the transactions for the service use are made automatically by the System.

The System should incentivize Virtuous Behaviors of the Users by providing extra discounts, or fees in case of misconduct, on the current ride. These Virtuous Behavior rules will compensate service expenses for the Cars, specially re-charging costs.

The System should provide these functionalities based on a mobile and web application that can be used by Registered Users to use the PowerEnJoy services. The System should interface as well with the Car so it can obtain important information from it, like location and battery use, and so it can control its locking or unlocking for the use.

For any issue that the User may find using the service or also for any incident a User may cause to the service, we've designed a Customer Relationship Manager (CRM) who'll be allowed to check the status of the Cars and Users, and be able to solve any inconvenient presented during the use of the service.

## 1.2 Goals

User:

- G.1)** First-Time Users must be able to register to the System creating an Account.
- G.2)** Registered Users must be able to login to their Account at any time they want.
- G.3)** Users can save his/hers credentials in the System.
- G.4)** A Registered User will be able to make a Reservation of any available Car near his/her current location or from an address that she/he can specify.
- G.5)** Users that had made a Reservation must be able to notify the System when they are nearby the Reserved Car so the System can unlock it.
- G.6)** A User that has made a Reservation must be able to cancel it before 1 hour starting from the time when the Reservation was made.
- G.7)** In case an User hasn't started using the Reserved Car at 45 minutes after the Reservation was made, he/she will be notified that either if the Reservation is not canceled or the Car is not used in 15 minutes, the Reservation will be automatically canceled and a 1 Euro fee will be charged to her/his Account.
- G.8)** When Users that start using their Reserved Car, they must be able to see their current expenses on the service through a System screen inside the Car.
- G.9)** The User must be able to know where the safe parking areas are nearby his/her current location or any address that she/he can specify.
- G.10)** Users must be able to finish their use of the Car when leaving it in a Safe Parking Area and exiting the car. The User will then be charged for the use of the service. The used Car will be locked and freed for Users to be reserved.
- G.11)** The User will always be notified when any Transaction is made on his Account.
- G.12)** Notify the Users that are currently using a Car of any available discounts on their ride if they abide by the 'Virtuous Behaviour' rules and of the extra fee in case of not respecting the facilitation of

the re-charging of the Car on site. These extra discounts/charges will be applied on the Total fee at the end of the ride. These rules are:

- G.12.1)** Apply a 10% discount if the User takes at least 2 other passenger into the car.
  - G.12.2)** Apply a 20% discount if the User leaves the Car with the battery at least half-full.
  - G.12.3)** Apply a 30% discount if the User leaves and plugs the Car in a Re-Charging Station.
  - G.12.4)** Apply a 30% extra fee if the User leaves the Car at more than 3KM from the nearest Re-Charging Station.
  - G.12.5)** Apply a 30% extra fee if the User leaves the Car with the battery less than 20% full.
- G.13)** Users can activate the Money Saving option on their Account to be notified of any nearby Re-Charging station on their arrival destination. Leaving the car at the end of the ride at this station and plugging it will register as a 'Virtuous Behavior' and will apply an extra discount when charging the User.
- G.14)** Users must have the option to contact a CRM at any moment, the System must provide a Customer Service area that offers a Chat feature within the App or a phone number.
- CRM:**
- G.15)** CRM must be able to log in into the System with an Account and see a list of all the Cars available, reserved, in use or unavailable.
  - G.16)** CRM has to be able to receive User Reports via chat or outside the app via phone call and be able to register it to the System.
  - G.17)** CRM upon request from a User or any major cause can Reserve or cancel any active Reservation.
  - G.18)** CRM must be able to change the Status of a Car given a User Report and tag it, in case it's unavailable, if it's due to Re-Charge, Fix, or Removal.
  - G.19)** Upon the completion of a User Report CRM must be able to apply extra fees or refund the payment to any User in case the situation demands it.

### 1.3 Glossary

- **First-Time User:** A User that has not created an Account and thus has not yet been registered by the System. Can have a client version of the application, but he/she can't use the service until Registered.
- **Registered User/Users:** A User that has provided valid credentials and payment information to create an Account in the System.
- **Account:** Allows the Registered User or the CRM to authenticate to the System and access the car-sharing service.
- **Car:** An electric motored vehicle registered to the PowerEnjoy car-sharing service and thus interfaced to the System. It can be available for reservation, reserved, ready to use, on use and unavailable.
- **Reservation:** Option for the User to denote their future use of an available Car. Once activated it marks the Car as Reserved and is active until the User starts the use of the Car or until 1 hour after its activation, at which point it's cancelled and an user fee is applied.
- **Ride:** The actual use of the Car by the User. It starts when the User ignites the Car engine for the first time and it ends when the car is parked in a safe area and the User exits the Car.
- **Safe Parking Area:** Pre-defined areas by the System where a Car can be left by the User to be able to finish their ride.
- **Fee:** Amount of money that is due by the User for any of the System services or for any User misbehavior.
- **Transaction:** Any exchange of money made by the System on the User Account, every fee payment is a transaction.
- **Re-Charging Station:** Special stations where the Cars can be left and plugged in to have their battery re-charged. They count as Safe Parking Areas.
- **Virtuous Behavior Rule:** Any of the incentive rules that provide the user with a discount on their current ride if they follow it.
- **Status of the Car:** It is the name of a check list made by the car's System at the end of each ride. This check list consists in the status of battery, the number of passengers on the car, if the car is plugged on a recharge station and their localization.

- **CRM:** or Customer Relationship Manager will be an employee able to Log in to the System with special privileges to update reservation, car status, transactions, etc. in order fix any eventuality an User may find during his/her use of the service.
- **User Report:** A User that has contacted a CRM or a major cause that would create a problem in the System will register into the system as a User Report, made by a CRM. This Report can describe for example an incident, a defect during the ride or a Car that has been fined, and will include if necessary a fine or a refund to the User depending on the case.
- **App:** Smartphone application that does the interface between the User and the System.
- **Main Page:** A page within the app where a map is showed centered in the users position.

## 1.4 Domain Assumptions

As not all aspects of the given world are considered in the problem definition, the following domain assumptions will be made so they hold for the entirety of the project.

- Once Users have registered with valid payment information, all payments made with their Account will be valid and arrive successfully.
- GPS locations on Users and Cars will always be valid.
- Available Cars will be always in good working conditions.
- Users that have made a Reservation will always be able to follow its status until the end of the ride, meaning that the screen on the Car will stay functional or by defect their use of the mobile app will not be interrupted by missing connection or empty battery on their mobile device.
- Once a Car is Picked Up and Unlocked we assume the Users will follow to start its engine and proceed with the Ride.
- The Cars will be equipped with weight sensors on each seat so it can detect how many passengers they are on a ride as well as being able to issue seat belt warnings.
- All Cars on the service will be at least 5 seat Cars so the option to take 2 passengers or more is always available.



- All Available Cars will have at least 20% battery left, all Cars not available because of low battery will be alerted by a CRM and will be moved by the service personnel to a Re-Charge station by the end of the day.
- Reservations take immediate effect so no conflict of concurrency can happen for a Car, it cannot belong to more than one Reservation at once.
- CRM will be employees of the service that will be registered directly into the database, the System does not need to allow the option for them to Register.
- CRM will be working in an office able to login to the web application of the System, with an office phone available and we assume there will be at least one available at every hour the Service is working.

## 1.5 Constraints

As the problem description has not explained every aspect of the System in detail, we'll impose the following constraints that'll ensure the consistency and good functioning of our System.

- Users will be able to have at most one active Reservation at a time.
- Cars can have at most one Reservation active at a time and only one active Ride that originates from that Reservation.
- Once a Car is Picked Up a Ride is created by the Reservation and the base fee is set for that Ride, even if the User does not move the Car.
- If a Car is left parked Unlocked in an active Ride for any reason (not confirming the end of the Ride) after one hour the System should Lock it and end the current Ride.
- Fee Calculation will have a base Fee, that is applied as soon as the Ride is created and will have a fixed fee by minute of use from the moment the User ignites the Engine. When the Ride is finished, the discounts (or extra fees) will deduce from these subtotal to get the Total Fee of the Ride.
- To avoid misuse of the Reservations and cancelation a CRM will be notified when an User has made and canceled 3 reservations in a row and will be able to block the User Account for an ammount of time that will deny him from any Reservation during that time.

## 1.6 Stakeholders

Our Main Stakeholder is the PowerEnJoy Car-Sharing Service, owned by Prof. DiNitto, that wants a management System for the main functionalities of its service.

## 1.7 Reference Documents

- Specification Document: Assignments AA 2016-2017.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications (IEEEExplore-SRS-template.pdf)
- Scenario-based Requirements Engineering (ScenarioPaper.pdf)
- Example Documents:
  - RASD sample from Oct. 20 lecture.pdf - myTaxiService
  - RASD esempio.pdf - Management Project Homework

## 2 Proposed System

Our Proposed System consist in a 3-tier Client-Server Architecture, with web clients for Users and CRM, mobile App clients for Users and the Car interface that will be implemented as a Client as it's external to our Server System. The Server side will consist on a Java EE Server that will implement a Web Tier for Web Clients and a Business Tier that'll communicate directly to the App Clients. The Server will then connect to the Company's Database that'll be located in our EIS Tier.

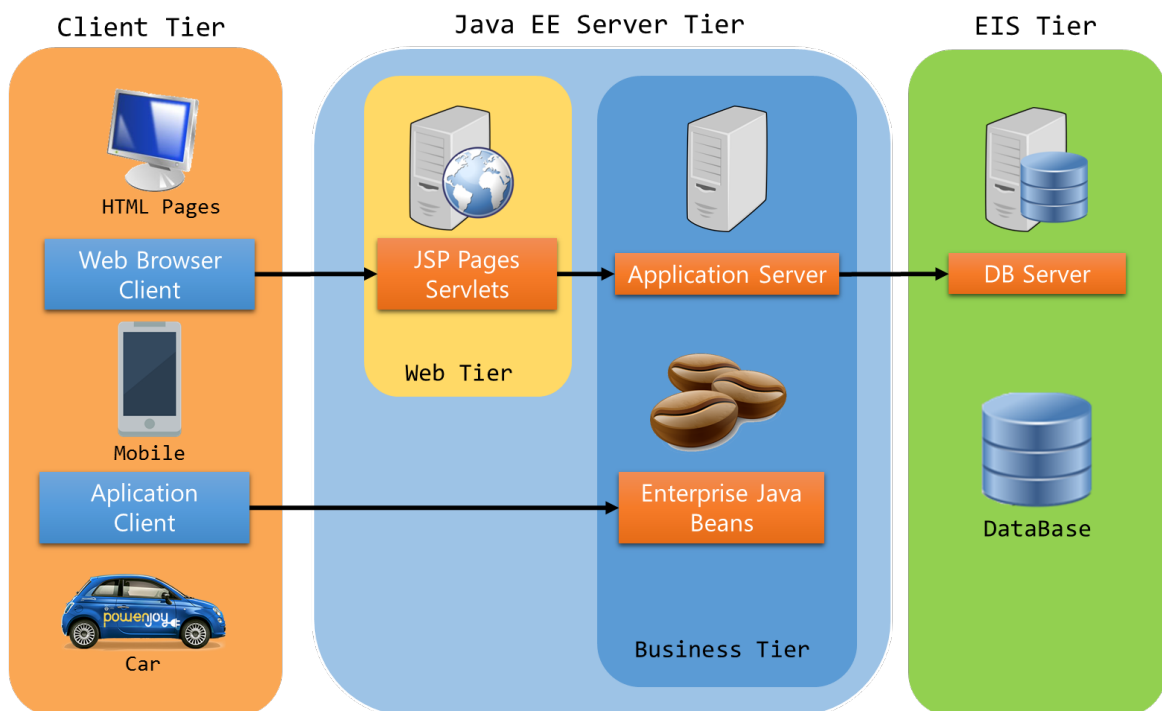


Figure 1: Proposed System Architecture

## **3 Actors**

### **3.1 User**

Is the Main Actor of our service. Any person that once Registered can take advantage of the features of the PowerEnJoy service.

### **3.2 CRM**

Customer Relationship Manager, will interface with the User in case of any problem that may arise and can intervene in the System to fix any issues. His permissions in the System include but are not limited to check on every Car of the service, perform or cancel any Reservation, modify the fee of an User's ride in case of a confirmed major cause, change the Status of a Car and execute a Transaction to a User post-ride.

## **4 Requirements**

### **4.1 Functional Requirements**

#### **4.1.1 User Requirements:**

**G.1)** First-Time Users must be able to register to the System creating an Account.

- The System must allow an user that has not created an Account to register by providing valid credentials, an e-mail and payment information.
- The System must sent a password to the Registered User Email that can be used to Log into the Account.

**G.2)** Registered Users must be able to login to their Account at any time they want.

- The System must allow Registered Users to log into their Account only if they provide their correct e-mail and password.

**G.3)** Users can save his/hers credentials in the System.

- The System must be able save Users credentials upon their request.
- When the user choose to save his/her credential they don't need to inform it in every time they open the System.

**G.4)** A Registered User will be able to make a Reservation of any available Car near his/her current location or from an address that she/he can specify.

- The System must be able to locate the User via GPS.
- The System must allow the Users to enter a determined address, in case they don't want to use their current location to locate the cars in the area.
- The System must be able to locate all available Cars via GPS.
- The System must show the User all the available Cars near the user given location.
- The System must allow the Users to select one of the Cars showed to them to create a Reservation, the selected Car will be marked as reserved.

**G.5)** Users that have made a Reservation must be able to notify the System when they are nearby the Reserved Car so the System can unlock it.

- While a Reservation is active, the System must allow the User to notify that they are nearby their Reserved Car.
- The System must confirm the notification by checking that the User current location and the Car location coincide.
- The System must unlock the Reserved Car and set it to Ready to Use.

**G.6)** A User that has made a Reservation must be able to cancel it before 1 hour starting from the time when the Reservation was made.

- The System must check how long has passed from the reservation time, if it less then one hour the user will be able to open the reservation and cancel it without being charged for it.
- The System must confirm the user request and set the car as available again.

**G.7)** In case an User hasn't started using the Reserved Car at 45 minutes after the Reservation was made, he/she will be notified that either if the Reservation is not canceled or the Car is not used in 15 minutes, the Reservation will be automatically canceled and a 1 Euro fee will be charged to her/his Account.

- After 45 minutes from the reservation time has passed, the System must send a notification to the user's registered smart-phone warning that he/she has 15 minutes left to pick up the car or to cancel the reservation.
- After the full hour has passed and the reservation is still active, the System must cancel the reservation, block this user account for the next hour and charge the fee of 1 Euro for not picking up the car.

**G.8)** When Users start using their Reserved Car, they must be able to see their current expenses on the service through a System screen inside the Car.

- The car System must keep track of the gross fee to be paid by the user.
- As soon as the car ignites, the car System must start showing the fee on the computer monitor and keep it this way until the user is finished using the car.

**G.9)** The User must be able to know where the safe parking areas are nearby his/her current location or any address that she/he can specify.

- The System must be able to show, when requested, a map with the safe parking zones and points with electricity plugs .

**G.10)** Users must be able to finish their use of the Car when leaving it in a Safe Parking Area and exiting the car. The User will then be charged for the use of the service. The used Car will be locked and freed for Users to be reserved.

- The System must verify if the car is parked in a Safe Parking Area. If the car isn't located in one, the System must notify the user on the screen of the car that it's not possible to finish the ride.
- The System must show a message on the car's screen asking if the user want to finish the ride or if want to continuing the use of the car.
- The user must answer to the request. If the user want to keep using it, the System will set the car as reserved for one hour. Nevertheless in terms of payment the System will charge the same amount of money in the final Transaction as if the car has been in a ride. If the user does not want to use the car the System will finish ride.
- Upon the user leaving of the car the System must lock the car and consider the user's answer to whether set the car as available or reserved.

**G.11)** The User will always be notified when any Transaction is made on his Account.

- The System must notify the user of every Transaction made on his Account by email. Further information related to the ride as the total time, the driven route and the charged Fees must compose the email.

- G.12)** Notify the Users that are currently using a Car of any available discounts on their ride if they abide by the 'Virtuous Behaviour Rules' and of the extra fee in case of not respecting the facilitation of the re-charging of the Car on site. These extra discounts/charges will be applied on the Total fee at the end of the ride.
- The System must notify the user on the Car's screen of the any information related with a discount or extra fee.
  - The System must calculate at the end of the ride if any discount or extra fee is applicable. These fees are:
- G.12.1) Apply a 10% discount if the User takes at least 2 other passenger into the car.
- The System must read from the status of the Car the number of passengers and if it's 2 or more, apply the 10% discount to the ride fee.
- G.12.2) Apply a 20% discount if the User leaves the Car with the battery at least half-full.
- G.12.5) Apply a 30% extra fee if the User leaves the Car with the battery less than 20% full.
- The System must read from the status of the Car the amount of battery left and if it's 50% or more calculate the 20% discount else if it's less than 20% calculate the 30% extra fee.
- G.12.3) Apply a 30% discount if the User leaves and plugs the Car in a Re-Charging Station.
- The System must detect from the status of the Car if it's plugged in to a Re-Charging Station and if it is, calculate the 30% discount.
- G.12.4) Apply a 30% extra fee if the User leaves the Car at more than 3KM from the nearest Re-Charging Station.
- The System must read from the status of the Car its current location and the distance to the nearest Re-Charging Station, if it is more than 3KM then calculate the 30% extra fee.
  - The System must apply these discounts or extra fees at the end of the ride over the total fee.
- G.13)** Users can activate the 'Money Saving'a option on their Account to be notified of any nearby Re-Charging station on their arrival destination. Leaving the car at the end of the ride at this station and plugging it will register as a 'Virtuous Behavior' and will apply and extra discount when charging the User.

- Users must be allowed to enable the 'Money Saving' option on their accounts at any time they choose.
- The System must allow the user to input his/her destination at any time of the ride whether on the app or on the car's screen.
- Upon the final destination of the user the System must direct the end of the ride considering a uniform distribution of parked cars of the System, the nearest charging station in relation with the address informed and the availability of these charging stations.

**G.14)** Users must have the option to contact a CRM at any moment, the System must provide a Customer Service area that offers a Chat feature within the App or a phone number.

- The System must show a Customer Service number in its Customer Service Area.
- The System must have a Chat feature that will allow the User contact directly a CRM.

#### **4.1.2 CRM Requirements:**

**G.15)** CRM must be able to log in into the System and see a list of all the Cars available, reserved, in use or unavailable.

- The System must allow log in to CRM if they have provided valid username and password.
- The System must show to CRM a list of all the Cars that can be sorted by Status.

**G.16)** CRM has to be able to receive User Reports via chat or outside the app via phone call and be able to register it to the System.

- When a User uses the Chat feature to contact a CRM the System will redirect the chat to the first available CRM and notify him/her that a User is contacting her/him.
- The System must allow the CRM to create a User Report that will link a User with a Car and if necessary allow the option to add a Transaction.

**G.17)** CRM upon request form an User or any major cause can Reserve or cancel any active Reservation.

- The System must allow from the list of all Cars showed to the CRM, the option to Reserve any available Car for a User or cancel a Reservation of a Reserved Car. This action does not include a User Report.



**G.18)** CRM must be able to change the Status of a Car given a User Report and tag it, in case it's unavailable, if it's due to Re-Charge, Fix, or Removal.

- The System must allow from the list of all Cars showed to the CRM, the option to Reserve any available Car for a User or cancel a Reservation of a Reserved Car. This action does not include a User Report.

**G.19)** Upon the completion of an User Report CRM must be able to apply extra fees or refund the payment to any User in case the situation demands it.

- The System at the end of an User Report must ask for confirmation of the CRM in case the Report includes a Transaction. This can be an extra fee that the User must pay or a refund the service owns the User.

## 4.2 Non Functional Requirements

### 4.2.1 User Interface

For our User Interface (UI) we'll offer a mobile App for Users and a desktop web App for Users and CRM. They will offer:

- **LogIn Page:** First screen of the app with the LogIn and SignIn options.



Figure 2: UI LogIn Page

- **Main Page:** Main page of the app where the user sees the map his location and selects the available cars. In Case of CRM, it can see all Cars.

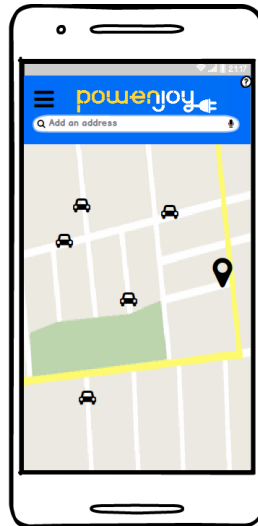


Figure 3: UI Main Page

- **Car Details Page:** Once a Car is selected this page allows to reserve a car and contains informations for the decision.

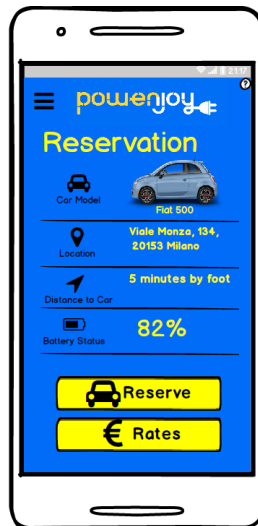


Figure 4: UI Car Details Page

- **Reservation Page:** Once the Reservation is made, the User can see the details of her/his Reservation and have the option to cancel it.

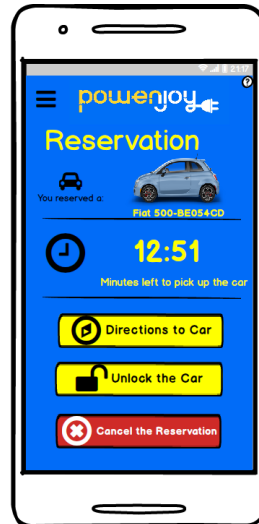


Figure 5: UI Reservation Page

- **Ride Page:** Once a Ride started this page shows the details of the current ride, it's present also in the Car screen App.

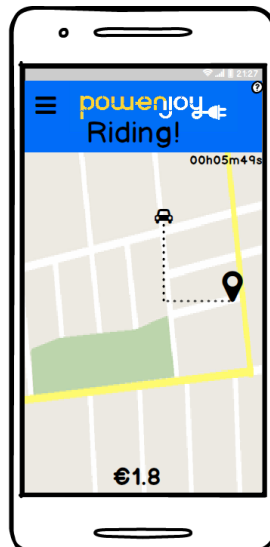


Figure 6: UI Mobile Ride Page

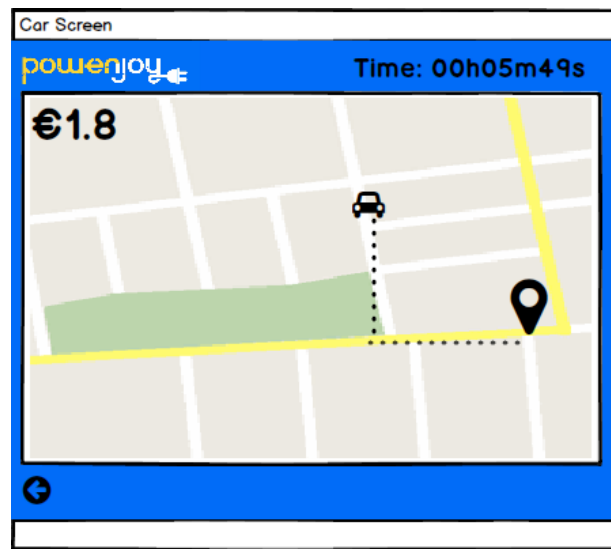


Figure 7: UI Car Ride Page.

- **Email Page:** While not part of the App UI, the System should respond with an email to the User at the end of the Ride showing the final details for it.

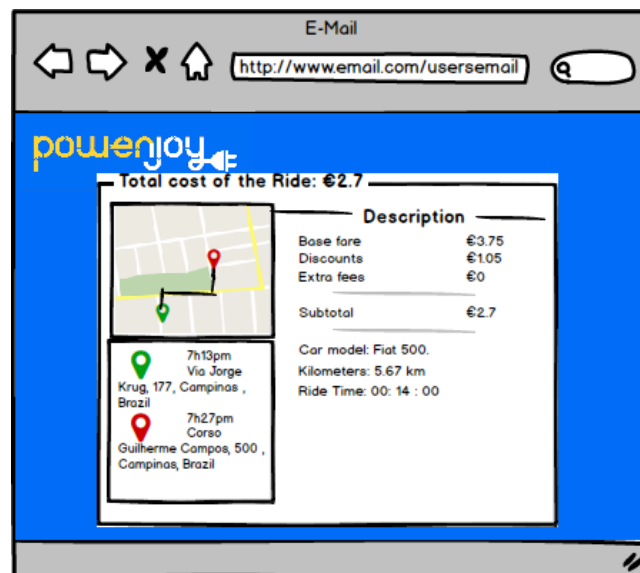


Figure 8: UI Email Page

- **Contact Page:** Page where the User can contact the CRM operator.

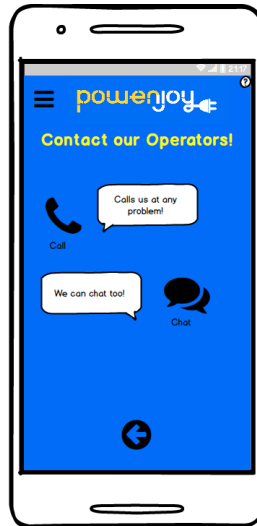


Figure 9: UI Contact Page

#### 4.2.2 Documentation

To organize this project workflow and development the following Documents will be drafted:

- **Requirements And Specifications Document (RASD):** Describes the Goals of the project and how to achieve them by specifying the Requirements.
- **Design Document (DD):** Defines the real structure of the System. Architecture, Components and design decisions will be explained here.
- **Project Testing Document:** Will detail the Test process for the system before deployment. Test case specifications, procedures, incident, reports will be considered Documentation Testing.
- **Project Management Documentation:** Specific assignment of tasks, estimations and goals for each of the development phases.

### 4.2.3 System Architecture

Project will be developed using a JEE Platform based in a 2-Tier Client/Server Architecture. First idea consists in Remote Presentation structured layers where the Client Apps will serve just to interface with the Server that will store the logic and the Data of the System. The architecture as explained briefly in the proposed System will consist of:

- Client Tier:
  - Web Client on a browser with abilitated JavaScript for the Web App use.
  - Application Client for the Mobile App (Android and iOS), with GPS functionalities. We will consider the screen present in the Car as a Client App of the System that'll implement some logic to interface to the Car Software to Lock, Unlock, read the Battery level and other functions. This can be seen to as some level of logic distribution.
- Server Tier:
  - Web Server that'll answer connections for Web Clients. Implemented as an Apache Server, will interface with the Application Server.
  - Java Application Server, that'll implement the logic of the System. Application Clients will connect directly to it. Will implement JavaBeans to handle request from multiple Clients at a time.
  - DataBase Server, accessed by the Application Server to store all the data refering to the System.

### 4.2.4 Hardware Consideration

As stated in the Architecture Considerations we'll need a server able to store our Application and DataBase and be accessible by multiple Clients at a time. We'll consider the Server to be always accesible through an stable connection to Internet. As an extension of the System we have the Clients running in the Cars, they'll need to be accessible via 3G and have GPS feature for localization.

### 4.2.5 Incident Handling

We'll consider our data to be persistent. If by any case any data is lost or corrupted we expect not to create big losses to the System, all transactions of data will be considered atomic we expect no loss during any kind of transaction.

#### 4.2.6 Security

Data Backups will be considered daily for data security. Accounts will be protected by hashed passwords. As one of the requirements of the project asked to send Users a password via mail we assume that it'll be handled by secure servers (IMAPS) so we're assured just the User can see it. Firewalls will be considered for our Servers connections. In case of breach, User Accounts can be blocked by CRMs. CRM Accounts are introduced directly into the DataBase by a separate system that register the Company employees, we assume it safely as well.

### 5 Scenario Identifying

<b>Scenario</b>	Carol is a First-Time User
<b>Actor</b>	User
<b>Background conditions</b>	Carol do not have an Account but she wants to create one.
<b>Flux of events</b>	<ul style="list-style-type: none"><li>•Carol selects the sign-up option on the App;</li><li>•The System shows a form where Carol can insert her data;</li><li>•Carol inserts her credentials, email and the payment information and finishes the process;</li><li>•The System checks the consistence of the data;</li><li>•The System informs Carol of the success of the registration;</li></ul>
<b>Exit conditions</b>	System confirms the creation of a new user and send an email to the user's email address informing the password and confirming the registration.
<b>Exception</b>	Carol inserts data that the System can't validate. System can not finish operation until Carol inserts the correct data.

<b>Scenario</b>	Alex wants to reserve the nearest car
<b>Actor</b>	User
<b>Background conditions</b>	Alex is Registered User. He is a heavy user of the service, and to save time he had saved his credentials in the app to open it without the need of a login.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Alex open the App;</li> <li>•The System redirects to the Main Page;</li> <li>•By experience Alex manually search in the map for the nearest car;</li> <li>•Alex find one car select and reserves it;</li> <li>•The System reserve the car;</li> <li>•Alex walks in the direction of the location of the reserved car;</li> </ul>
<b>Exit conditions</b>	The System confirms the reservation of the car that Alex had selected.
<b>Exception</b>	<p>The saved credentials are not valid and the login fail.</p> <p>The car that Alex selects is already reserved and the System can not effectuate the reservation.</p>
<b>Scenario</b>	Francesco wants to login to see available cars without doing a reservation
<b>Actor</b>	User
<b>Background conditions</b>	<p>Francesco is a Registered user.</p> <p>He is a new user and want to explore the functions of the System.</p>
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Francesco open the app and login;</li> <li>•Francesco try to find a car manually searching in the map, but he does not like of searching by himself;</li> <li>•Francesco open the side menu and find the option 'Search for a car', he selects it;</li> <li>•The System automatically search for the nearest car based on Francesco's location;</li> <li>•The System find the finds a car and trace the route in the map of the App's map;</li> <li>•Francesco satisfied to know this functionality of the app and closes it without reserving a car;</li> </ul>
<b>Exit conditions</b>	The System allows the user to search for the available cars without the need of a reservation.
<b>Exception</b>	<p>The login fail.</p> <p>The System does not find available cars and can't return results for the search.</p>



<b>Scenario</b>	Mario is doing overtime. He knows that the metro stops at 0h so he wants to reserve a car near the metro station. He'll exit at 0h20.
<b>Actor</b>	User
<b>Background conditions</b>	Mario is a Registered user. He uses the System occasionally.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•It's 23h50. Mario open the App and login;</li> <li>•The System redirects to the Main Page where there is an option to inform an address to find a car;</li> <li>•Mario inform the address of the metro;</li> <li>•The System return the metro station location;</li> <li>•Mario see the available cars selects one and reserve it;</li> </ul>
<b>Exit conditions</b>	The System reserves the car selected by the user and assure reservation for a period of one hour.
<b>Exception</b>	The login fail. There is no available car in the address that Mario and he does not reserve a car.
<b>Scenario</b>	Angela is walking back home when she sees a car parked by her side. She wants to unlock it to drive back home.
<b>Actor</b>	User
<b>Background conditions</b>	Angela is a Registered user. She did not make a reservation of the car that she wants to drive.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Angela open the App and do the login;</li> <li>•The System redirects to the Main Page;</li> <li>•Angela immediately finds the car, that is beside her location;</li> <li>•She selects the car and request a reservation;</li> <li>•The System reserves the car;</li> <li>•Angela proceeds to unlock the car in the App;</li> <li>•The System checks the reserve and the current position of the User and the Car;</li> <li>•The System unlocks the car and set it for Ready to Use mode;</li> <li>•Angela get in the car and drives home;</li> </ul>
<b>Exit conditions</b>	System reserves the car and unlock it upon User location.
<b>Exception</b>	The login fail. The car that is besides Angela is already reserved so she can't reserve the car for her use in the moment.

<b>Scenario</b>	Sarah wants to cancel a reservation
<b>Actor</b>	User
<b>Background conditions</b>	Sarah is a Registered User. She does not need to get the car anymore so she wants to cancel the reservation that she made 30 minutes ago.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Sarah open the App and do the login;</li> <li>•Sarah selects in the menu of the App the option 'My reservation';</li> <li>•The System shows Sarah's reservation;</li> <li>•Sarah see the status of the reservation;</li> <li>•Sarah selects the option of canceling the reservation;</li> <li>•The System checks the time that Sarah did the reservation, verifies that is possible to cancel the ride and proceeds to do this operation;</li> </ul>
<b>Exit conditions</b>	The System verifies if the reservation's time is under 1 hour if so the user can cancel the reservation.
<b>Exception</b>	The login fails. Sarah tries to cancel the reservation after one hour period of reservation and System does not allow Sarah cancel the reserve.
<b>Scenario</b>	Paulo is late to pick up the reserved car.
<b>Actor</b>	User
<b>Background conditions</b>	Paulo is a Registered User. He had reserved a car. It has passed 45 minutes.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Paulo have 15 minutes left to pick the reserved car;</li> <li>•The System emits an alert in the App informing that Paulo is running out of time;</li> <li>•Paulo sees the alert and rushes into the location where the car is parked;</li> </ul>
<b>Exit conditions</b>	The System emits an alert when the period of the car's reserve is ending.
<b>Exception</b>	The alert fails. Paulo don't see the alert. Paulo couldn't make to the car in 15 minutes and the System automatically cancel the reservation, charges 1 euro of Paulo's account and will not allow Paulo to reserve a car for 1 hour.

<b>Scenario</b>	Hugo is about to finish a ride.
<b>Actor</b>	User
<b>Background conditions</b>	Hugo is a Registered User. He was driving into a unknow direction.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•At some point Hugo decides to stop the car;</li> <li>•He finds parking spot and proceeds to park the car;</li> <li>•Hugo finishes the maneuver and turn off the car;</li> <li>•The System verifies if the parking spot is in some Safe Park Area and proceeds to finish the ride;</li> <li>•The System shows an interface on the car's screen asking if Hugo will finish the ride;</li> <li>•Hugo selects that he will finish the ride;</li> <li>•Hugo left the car;</li> <li>•The System finish the ride and lock the car;</li> </ul>
<b>Exit conditions</b>	The System verifies the Status of the car and proceeds to finish Hugo's ride sending to his email the information of the ride as the route and the transaction.
<b>Exception</b>	Hugo park the car in a no Safe Park Area, and the System can not finish the ride.

<b>Scenario</b>	Marina wants to use the System in the most economical way.
<b>Actor</b>	User
<b>Background conditions</b>	Marina is Registered User and she had saved her credentials on the app. She had just unlocked the car and is about to begin the ride.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Marina has a previous knowledge of the 'Money Saving' option so when she got in the car she informs in car's screen map her destination;</li> <li>•The System calculates the optimal destination considering the distribution of the cars in the city, the available plugs and the destination address;</li> <li>•The System return options of parking spots to Marina;</li> <li>•She considers these options and selects one. She starts the ride following the GPS route of the car's screen;</li> <li>•Mariana drives to the destination;</li> <li>•Marina turn of the car;</li> <li>•The System show a question on the car's screen asking if Marina will finish the ride;</li> <li>•Marina selects that she will finish the ride;</li> <li>•The System show a message on the saying that Marina is besides the plug and advice to connect the car to the plug;</li> <li>•Marina left the car and proceeds to connect the plug into the car;</li> <li>•The System finish the ride and lock the car;</li> </ul>
<b>Exit conditions</b>	The System verifies the Status of the car and proceeds to finish Marina's ride sending to his email the information of the ride as the route and the transaction.
<b>Exception</b>	The System can not return a destination due the lack of the available Re-Charging Stations in relation with the distribution of the cars.

<b>Scenario</b>	The Car that Julia was driving had mechanical problems during the Ride.
<b>Actor</b>	User, CRM operator
<b>Background conditions</b>	Julia is Registered User and she had saved her credentials on the app. She was driving a Car when it suddenly stops.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>• Julia open the App;</li> <li>• She selects the option 'Problems with the Car';</li> <li>• The System immediately provides an option whether to chat or call to an CRM operator;</li> <li>• Julia chooses to call and the System redirects to the dialing process;</li> <li>• The calling established a connection of Julia and the CRM operator;</li> <li>• Julia describe the situation to the CRM operator;</li> <li>• The CRM operator finishes the Julia's ride, check if there is any extra charging in the Transaction of the riding and set the car as unavailable;</li> <li>• Julia finishes the call and left the car;</li> </ul>
<b>Exit conditions</b>	The CRM operator directs the relation when between the User and the System in such events and is enable to finish the ride and to set the car as unavailable.
<b>Exception</b>	<p>The saved credentials are not valid and the login fail.</p> <p>The connection between the CRM operator and the User fails.</p> <p>There is no CRM operator available;</p> <p>The CRM operator could not identity the problem with the Car.</p>

<b>Scenario</b>	Frederico had a medical condition and could not pick up the reserved car
<b>Actor</b>	User, CRM operator
<b>Background conditions</b>	Frederico is a Registered user and had made and missed a reservation.He had an accident and was unable to get the car.
<b>Flux of events</b>	<ul style="list-style-type: none"> <li>•Frederico open the app and do the login;</li> <li>•He chooses the option 'Talk with us' and call for the CRM operator</li> <li>•The calling established a connection of Frederico and the CRM operator;</li> <li>•Frederico explains the case to the CRM operator;</li> <li>•The CRM operator analyses the case and apply some alterations on the system such as canceling the charged fee in one Transaction of Francesco's Account;</li> <li>•Frederico finishes the call;</li> </ul>
<b>Exit conditions</b>	CRM operator analyses Frederico's request and apply or not changes in the system following these requests
<b>Exception</b>	<p>The login fail.</p> <p>The connection between the CRM operator and the User fails.</p> <p>There is no CRM operator available;</p> <p>The CRM operator could not apply the changes upon the user's request.</p>

## 6 UML Modeling

### 6.1 Use Case Diagram

Use case Diagram represents all the situations available to Users explained in the Scenarios.

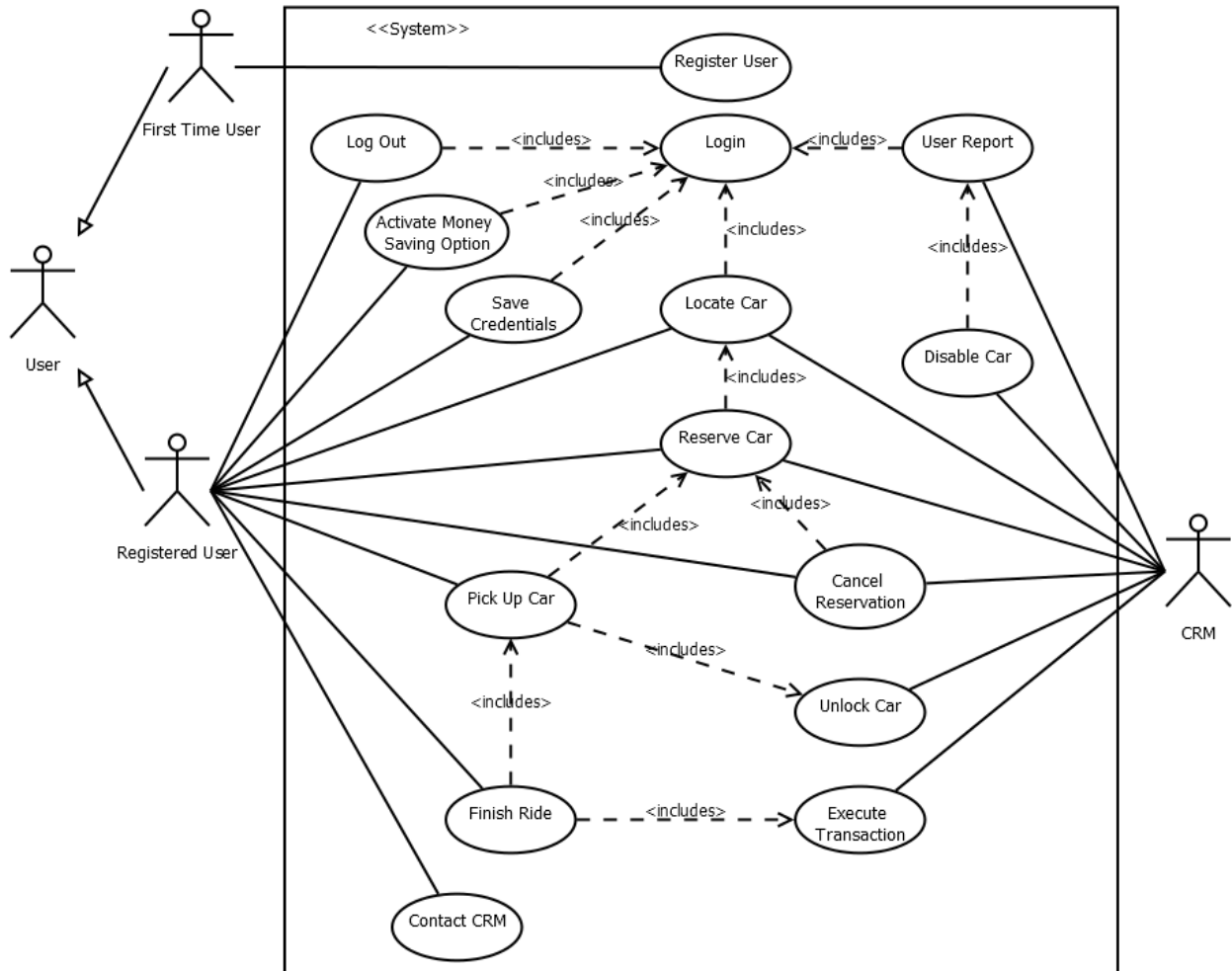


Figure 10: Use Case Diagram

### 6.1.1 Use Cases Description

Use Cases derived from the Scenarios previously detailed and based on the Use Case Diagram.

---

**Use Case:** First Time User Registers.

**Actors:** User

**Flow of Events:**

- User selects the Sign-Up option on the App.
- A form is presented to the User where he/she can fill her/his information.
- The User inserts all credentials and valid payment options and proceeds with the operation.
- The System informs the User the success of the Registration.

**Exit Conditions:** The System confirms the creation of the new User and adds its Account to the Database. An email with the User's Password is sent for her/him to Log in.

**Exceptions:** The User inserts invalid data so the System can't confirm the Registration.

---

**Use Case:** User Reserves and Rides a Car.

**Actors:** User

**Flow of Events:**

- User Logs In to the App in a mobile device.
- User sees in the Main Page the map with the available Cars near the User current Location.
- User selects a Car and reserves it.
- The System creates a Reservation with the User and the Car, maked now as reserved.
- User moves toward the Location of the Reserved Car and ask the System to unlock it.
- The System confirms that the User and the Car location are the same and unlocks the Car.



- As soon as the User ignites the Car the System starts the registration of the Ride.
- The User arrives to the desired destination and parks the Car in a Safe Parking Area.
- The System asks the User if he/she desires to finish the Ride.
- When the User confirms, the System ends the Ride and charges its fee to the User Account.

**Exit Conditions:** The System verifies the Status of the Car and sends an email to the User with the information of the ride as the route and the Transaction.

**Exceptions:** The User cannot find an available Car, the User cancels the Reservation or the User does not park in a Safe Parking Area, thus the System cannot end the Ride.

---

**Use Case:** User asks to cancel a Reservation.

**Actors:** User, CRM

**Flow of Events:**

- User makes a Reservation for a Car.
- After some time before the one hour limit has passed the User decides to cancel the Reservation
- Instead of canceling the Reservation with the App the User decides to call the Customer Service Number.
- A CRM answers the User and after confirming his/her information is correct, looks for the Reservation in the System.
- The CRM cancels the Reservation manually on the System.

**Exit Conditions:** The Reservation is confirmed canceled, the Reserved Car is marked as available, the User is able to make another Reservation.

**Exceptions:** The User cannot provide the CRM valid information of the Reservation to cancel it.

---

**Use Case:** User Activates and uses the Money Saving Option.

**Actors:** User

**Flow of Events:**

- A Logged in User sees the Money Saving Option in the App and activates it for her/his Ride.
- The System asked the User for a final destination of the Ride.
- The System finds the best Re-Charging station near the Location given by the User. This station is determined optimally by the System considering Car distribution in the city and availability of power plugs.
- The User finish his/her Ride by parking in the suggested Re-Charging Station and plugging in the Car.
- The System ends the Ride and charges the User with a 30% discount for Virtuous Behavior.

**Exit Conditions:** The System ends the Ride confirming that the Car is plugged in to the Power Station.

**Exceptions:** The System cannot find a suitable Re-Charging station near the User preferred Location. The User does not plug in the Car by the end of the Ride.

---

**Use Case:** User cannot pick up a Reserved Car.

**Actors:** User, CRM

**Flow of Events:**

- The User has made a Reservation and goes to the Reserved Car.
- The User tries to open up the Car with the App but it does not work.
- The User contacts a CRM via Customer Service and explains his/her problem.
- The CRM confirms the User information and locates the given Reserved Car.
- The CRM tries to Unlock the Car using the System but it doesn't work. Ex. Connection issue.
- The CRM then proceeds to cancel the Reservation, mark the Car as unavailable and reserve the next nearest available Car for the User to Ride.

- The CRM creates an User Report linking the User that reported it and the Car explaining the reason why the Car has been mark as Unavailable and due for inspection.
- The CRM can decide to apply a 10% discount to the User's Ride for the inconvenience.

**Exit Conditions:** The User Reservation is cancelled and a new one is created, the User then proceeds with the Ride normally.

**Exceptions:** The CRM override to unlock the Car works and allows the User proceed normally. The CRM cannot find an available Car near the User.

## 6.2 Class Diagram

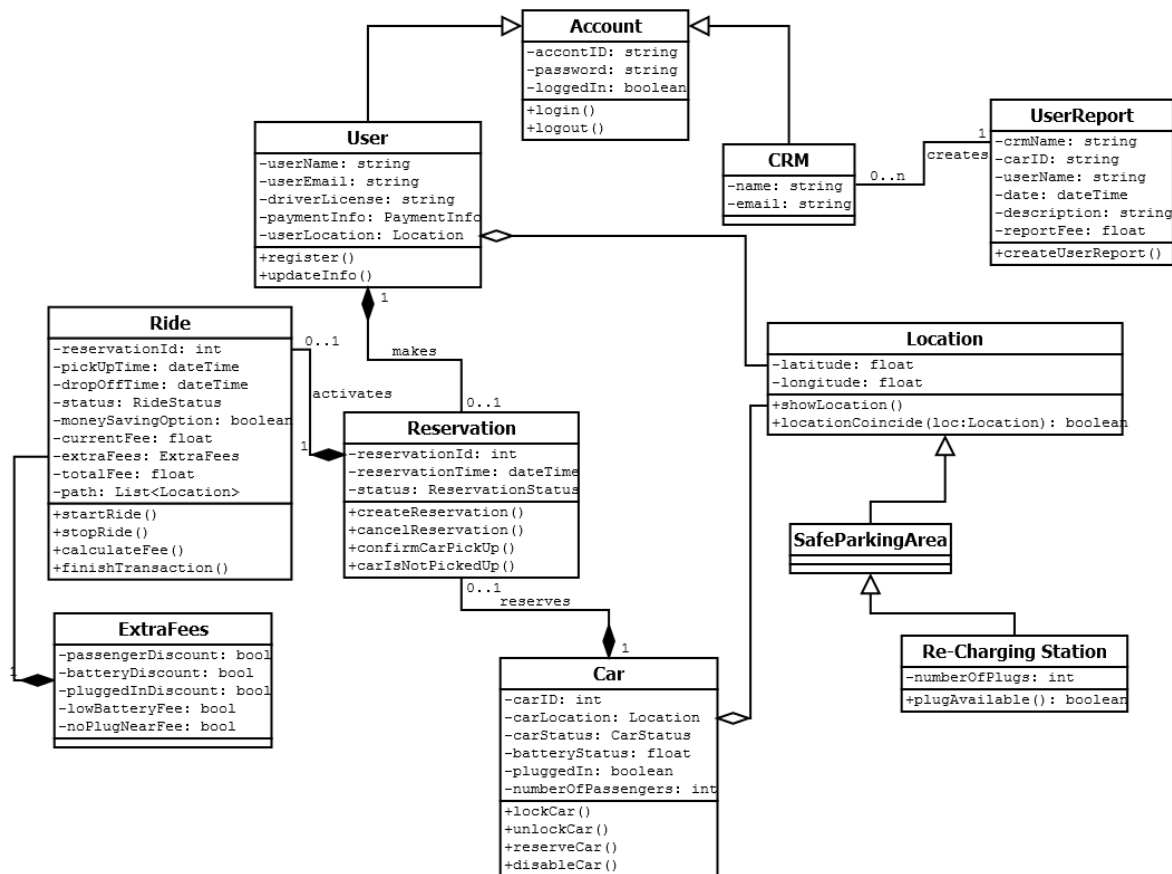


Figure 11: Class Diagram

## 6.3 Sequence Diagrams

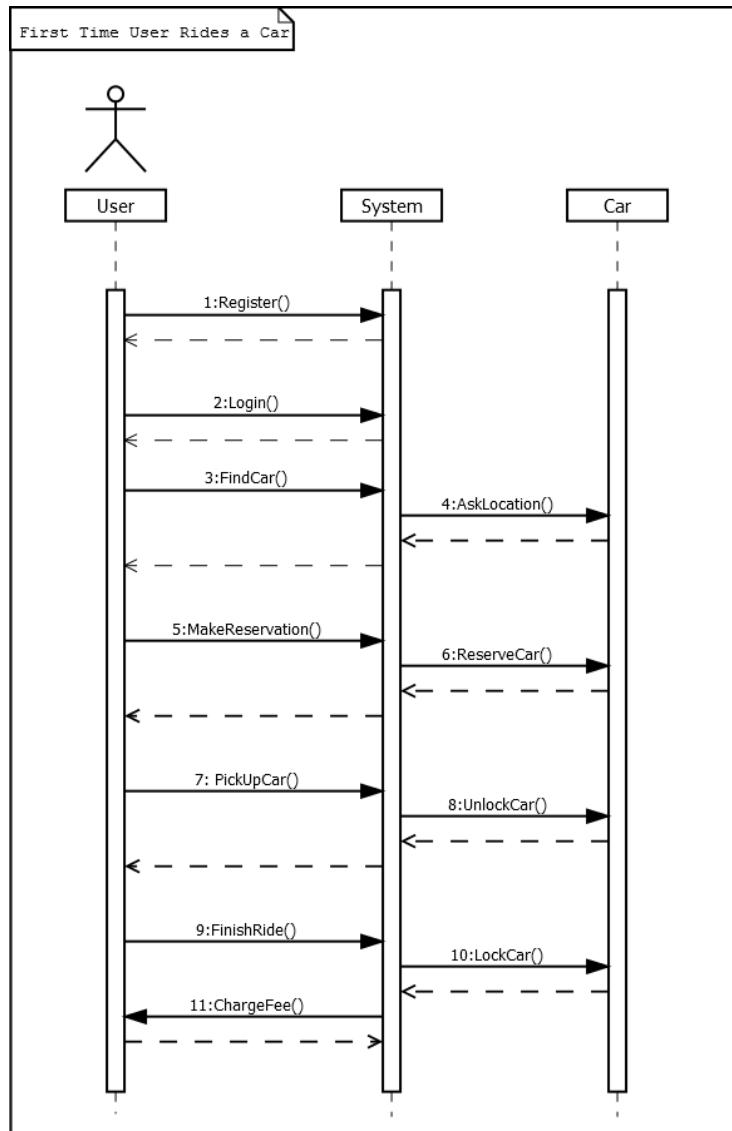


Figure 12: Sequence Diagram 1: User takes a Ride

In the first Sequence Diagram (fig. 3) we abstract the elements present inside the System and show the flow of interaction with the external classes of the System, that is showing the System as a mean to interface the User and the Car. This is presented in a normal case of User asking for a Ride.

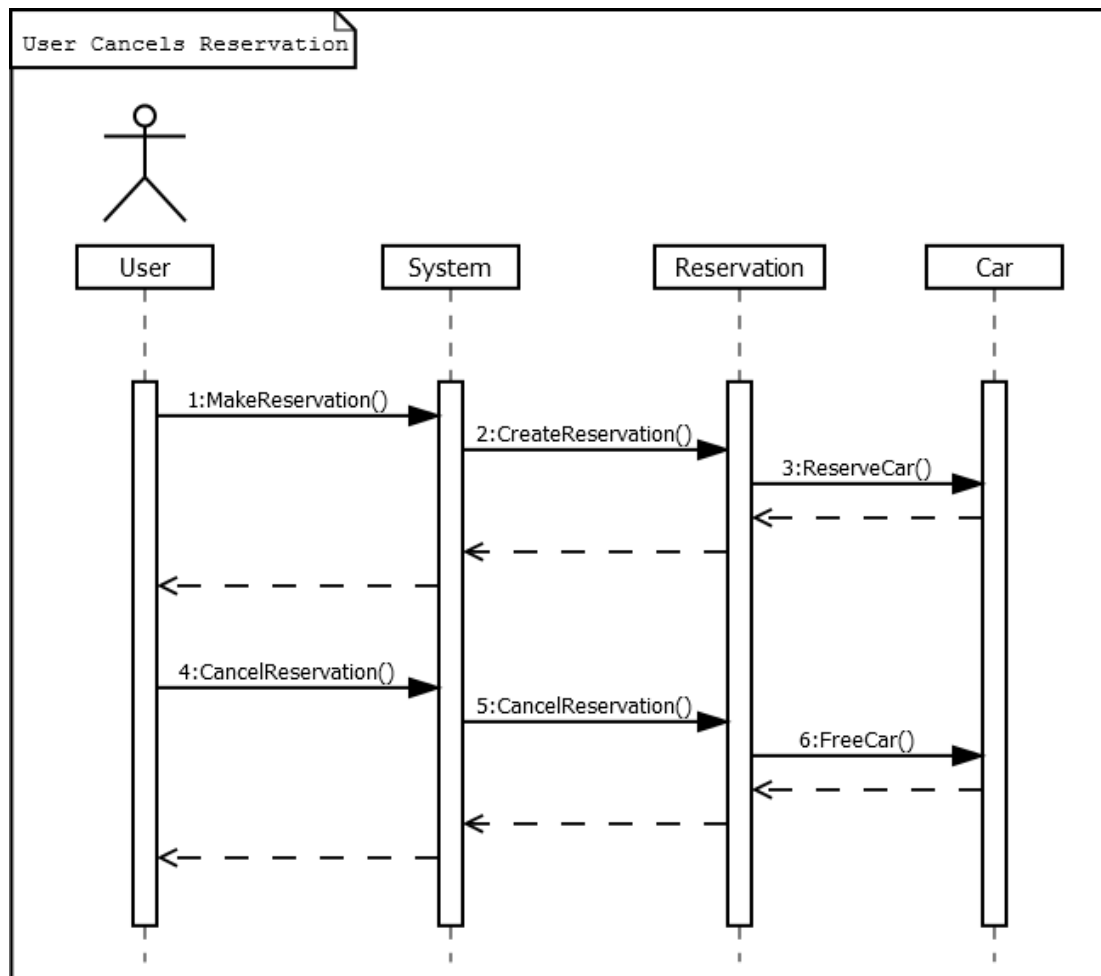


Figure 13: Sequence Diagram 2: User cancels a Reservation

In this Sequence Diagram (fig. 4) we represent a sequence within the lifecycle of a Reservation, that is a class which functionality exists inside the System and serves as the mean for the System to communicate with the Car.

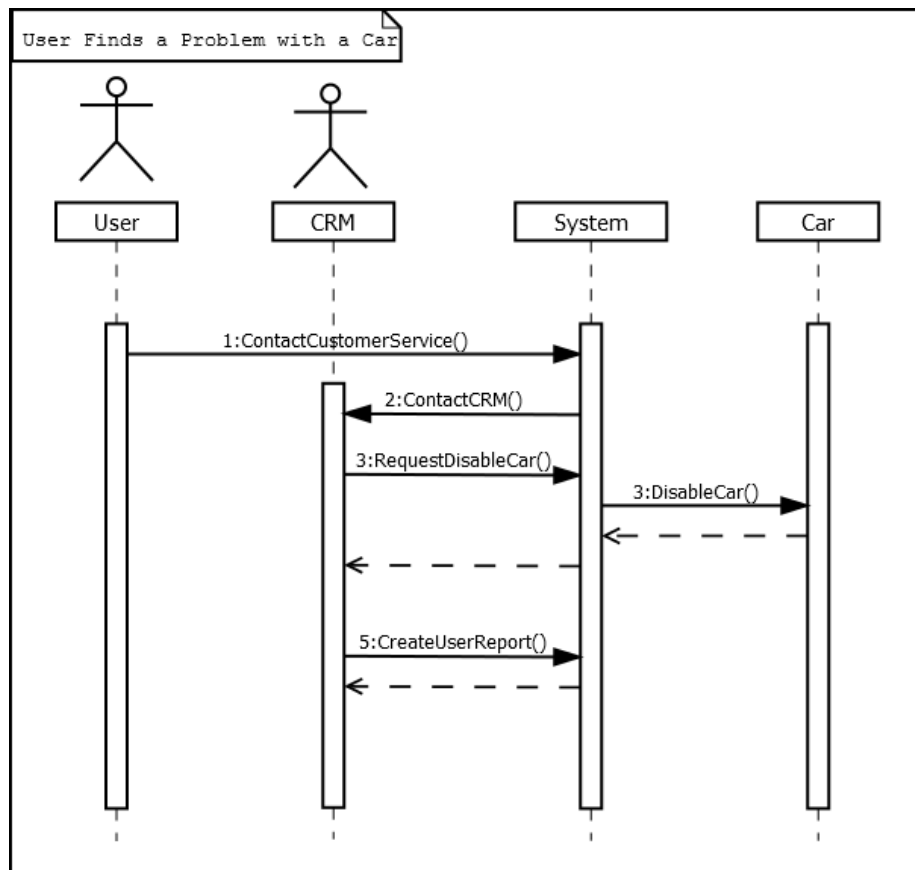


Figure 14: Sequence Diagram 3: User finds a problem with a Car

In this last example of Sequence Diagram (fig. 5) we present a situation in which the User uses the System to interface with our other Actor, the CRM, which in turn uses the System again to interface with the Car.

## 6.4 Statechart Diagram

## 6.5 Activity Diagram

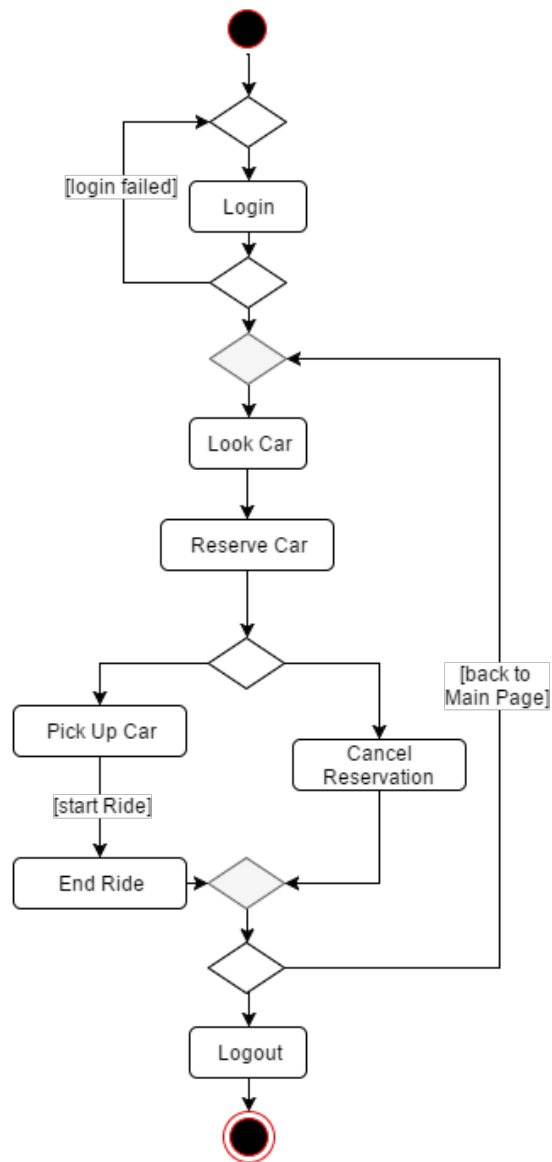


Figure 15: Activity Diagram



## 7 Alloy Modeling

### 7.1 Model

The whole system model was implemented in Alloy and the functionalities of "Login creation" and "Making a new reservation" were tested by the Alloy Analysing tool. These specific functionalities were chosen to be implemented in a formal language for being the key points of the system.

#### 7.1.1 Signatures

```
sig Date{}

sig PayInfo{}

sig Location{
    latitude: Int,
    longitude: Int
}

sig System{
    listOfRentedCars: set Car, // all reserved cars
    listOfAvailableCars: set Car, // cars that are available for pick up
    listOfReservs: set Reservation, // all active reservations
    listOfAccounts: set Account, // accounts createad
    listofInvoice: set Ride, // Archive of old rides and reservations
    listOfReports: set UserReport //CRM operations
}

// User will be considered as a first time user, so it has no attibutes
sig User{}

sig Account{
    id: one Int,
    credentials: one User,
    bankData: one PayInfo,
    password: one Int
}

sig Reservation{
    id: one Int,
    madeBy: one Account,
    selectedCar: one Car,
```

```

    time: one Date, // charging for not picking up the car
}

sig Car{
    id: one Int,
    loc: one Location,
    numOfPassengers: one Int,
    battery: one Int,
}{
    numOfPassengers ≤ 5
    battery ≤ 100
    battery ≥ 0
}

sig Ride{
    reservInQuestion : one Reservation,
    timeofPickUp: lone Date,
    timeofDropOff: lone Date,
    NotPickUp: one Int, //will be charged fee for not picking up the car
    PassengerDisc: one Int,
    batteryDisc: one Int,
    ReChargingStationDisc: one Int,
    HarshConditionsFee: one Int // more then 3km or less the 20% bat-
tery
}

sig SafeParkingArea{
    zone: set Location
}

sig ReChargingStation extends SafeParkingArea{
    spots: set Location -> one Int, //this Int is the number of available
plugs
}

sig Crm{
    name: String,
    email: String
}

sig UserReport{
    crmName: Crm,
    car: Car,
    userName: Account,

```

```

    reportFee: Int
}

```

### 7.1.2 Facts

```

//every car, reservation and account is managed by the system
fact systemManagesAll{
    all c:Car — one s: System — c in (s.listOfRentedCars + s.listOfAvailableCars)
    all r:Reservation — one s: System — r in s.listOfReservs
    all a:Account — one s:System — a in s.listOfAccounts
}

fact noCarIntwoLists{
    all s:System —
    s.listOfRentedCars & s.listOfAvailableCars = none
}

fact noFareIn2List{
    all s:System —
    s.listOfReservs & s.listofInvoice.reservInQuestion = none
}

fact OneFareOneRide{
    no disj f1,f2:Ride, r:Reservation — f1.reservInQuestion=r and f2.reservInQuestion=r
}

fact noDuplicateId{
    no disj a1, a2: Account — a1.id = a2.id
    no disj r1, r2: Reservation — r1.id = r2.id
    no disj c1, c2: Car — c1.id = c2.id
}

```

### 7.1.3 Login Creation

```

// the same user cannot create two accounts
fact noUserHas2Accounts{
    no disj a1, a2:Account, u: User — a1 != a2 and a1.credentials = u
    and a2.credentials = u
}

// returns a list of accounts that has "a" in it. will be used to assure no
duplicates
fun lookup [a: Account, s: System] : set Account {

```

```

    a & s.listOfAccounts // & = intersection
}

// the list cannot have more than one item
fact noDuplicateUser{
    all a: Account, s:System —
        #lookup[a, s] ≤ 2
}

// Creating a new account
pred addAccountToSystem(a:Account, s, s':System){
    (s'.listOfAccounts = s.listOfAccounts + a)
}
run addAccountToSystem

// if two accounts have the same data, they are equal
assert addEqualAccount{
    all a, a':Account, s, s':System —
        addAccountToSystem[a,s,s'] and lookup[a, s'] = lookup[a', s']
        implies
        a = a'
}
check addEqualAccount for 4 but 1 System

```

#### 7.1.4 Making a new Reservation

```

fact carOnlyInOneReserv{
    all c:Car — no disj r1, r2: Reservation —
        r1.selectedCar = c and r2.selectedCar = c
}

fact UserOnlyInOneReserv{
    all a:Account — no disj r1, r2: Reservation —
        r1.madeBy = a and r2.madeBy = a
}

fact noReservedCars{
    all c:Car, s:System —
        c in s.listOfReservs.selectedCar
        implies
        c in s.listOfRentedCars
}

```

```

fact noReservedCars2{
  all c:Car, s:System —
    c not in s.listOfReservs.selectedCar
  implies
    c in s.listOfAvailableCars
}

pred createReservation[a: Account, c:Car, r,r': Reservation]{
  r'.id = r.id + 1
  r'.madeBy = a
  r'.selectedCar = c
  r'.time = Date
}
run createReservation

pred makeReservation[a: Account, c:Car, s,s',s'': System, r, r':Reservation]{
  // create a reservation
  createReservation[a, c, r,r']
  // add reservation the the list
  s'.listOfReservs = s.listOfReservs + r'
  // change car list
  s'.listOfAvailableCars = s.listOfAvailableCars - c
  s''.listOfRentedCars = s'.listOfRentedCars + c
}
run makeReservation for 5 but 3 Reservation, 1 System

assert sameCar2Reserv{
  no disj a1,a2:Account, s1,s2,s3:System, c:Car, r1,r2, r3:Reservation
  —
    makeReservation[a1, c, s1,s2,s3, r1,r2] and makeReservation[a2, c,
s1,s2,s3, r2,r3]
}
check sameCar2Reserv

assert sameUser2Reserv{
  no disj c1,c2:Car, a:Account, s1,s2,s3:System, r1,r2, r3:Reservation—
    makeReservation[a, c1, s1,s2,s3, r1,r2] and makeReservation[a, c2,
s1,s2,s3, r2,r3]
}
check sameUser2Reserv

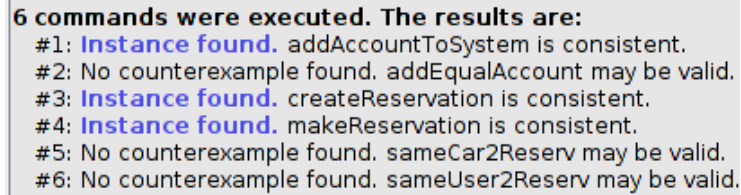
```

## 7.2 Alloy Result

As stated in the previous Alloy code, the following checks were made:

- Adding an account to the system
- Trying to add two equal accounts
- Creating a new reservation
- Creating a new reservation and integrating it with the rest of the system
- Trying to reserve the same car twice
- The same user tries to make two reservations

The Alloy Analyser tool gave the following feedback:



```
6 commands were executed. The results are:  
#1: Instance found. addAccountToSystem is consistent.  
#2: No counterexample found. addEqualAccount may be valid.  
#3: Instance found. createReservation is consistent.  
#4: Instance found. makeReservation is consistent.  
#5: No counterexample found. sameCar2Reserv may be valid.  
#6: No counterexample found. sameUser2Reserv may be valid.
```

Figure 16: Alloy result

This result proves our model is mathematically consistent for the implemented cases.

## 7.3 Generated World

Each of the worlds generated with the Alloy Analysing tool are consistent with the proposed system. Special attention was given for the relations between entities to respect especific rules that could potentially cause inconsistency in the system with its use, as not allowing duplicates or always making sure that cars that appears in reservations are placed only in the rented cars list.

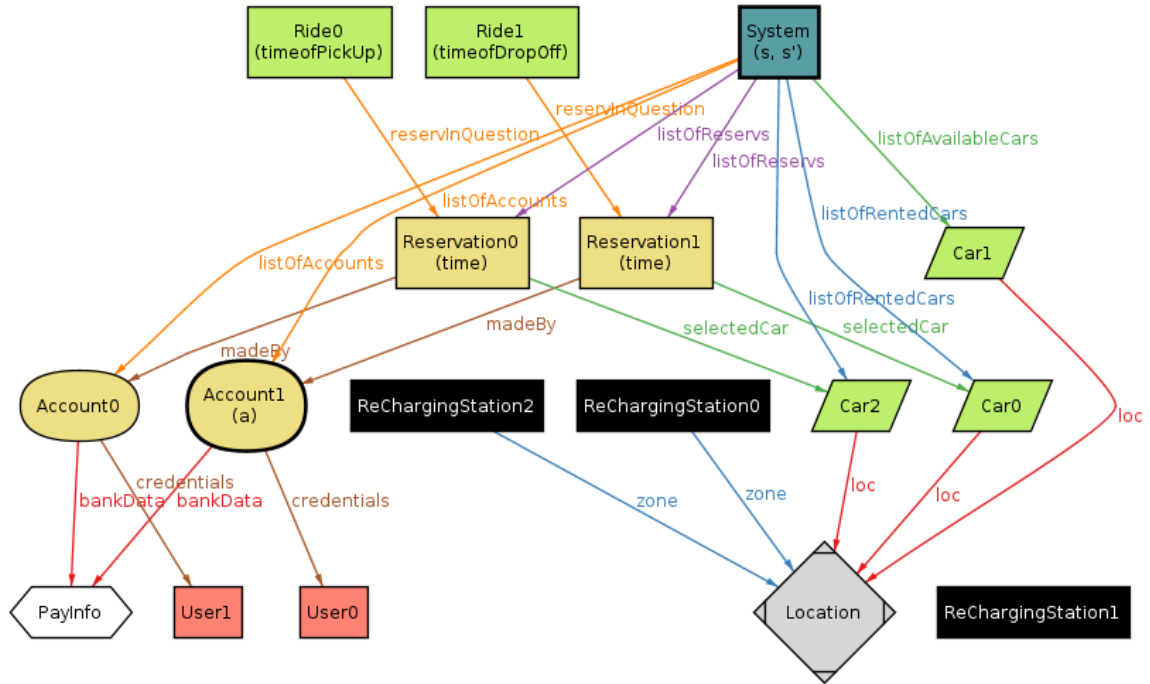


Figure 17: First generated instance

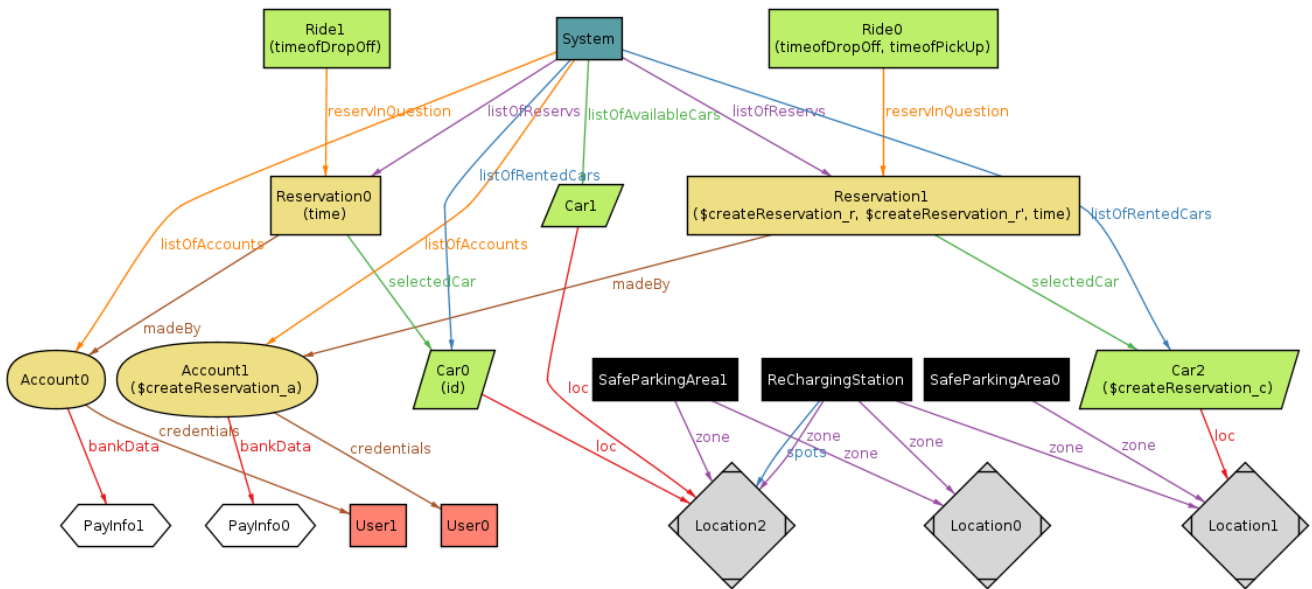


Figure 18: Second generated instance





## 8 Used Tools

The Tools used to develop this RASD document were:

- **GitHub:** for Version Control
- **Dia Diagram Editor:** for UML Diagrams
- **Alloy Analyzer 4.2:** for Alloy Modelling and proving consistency
- **TeXworks:** for LaTeX editing of this Document
- **Microsoft PowerPoint:** for Proposed System and Presentation
- **Draw.io:** for Activity and Statechart Diagrams
- **Balsamiq Mockups 3:** for UI Screen Mockups

## 9 Hours of Work

Date	Domenico	Caio	Matheus
25/10/16	30m	30m	30m
26/10/16	1h	-	-
27/10/16	-	-	-
28/10/16	2h.30m	4h.30m	4h.30m
29/10/16	1h	-	-
30/10/16	-	-	2h
31/10/16	-	2h	-
01/11/16	-	-	2h
02/11/16	2h	-	-
03/11/16	2h	3h	3h
04/11/16	-	-	2h
05/11/16	1h	4h	6h
06/11/16	-	5h	6h
07/11/16	4h	2h	6h
08/11/16	3h	-	-
09/11/16	2h	2h	3h
10/11/16	2h	3h	2h
11/11/16	2h	-	-
12/11/16	-	-	-
13/11/16	-	-	-

## 10 Changelog

As the project and design decisions may change during the development this document is also prone to change. We'll document every version in this part.

- **Version 1.1:** 13/11/2016