POLITECNICO
MILANO 1863

Politecnico di Milano

COMPUTER SCIENCE AND ENGINEERING

Software Engineering 2

Requirement Analysis and Specification Document

PowerEnJoy

Authors:
Francesco Fabiani
Jagadesh Manivannan
Niccolò Pozzolini

Professors:
Elisabetta Di Nitto
Luca Mottola

Academic Year 2016/2017

Contents

Chapter 1

Introduction

1.1 Purpose

The Requirement Analysis and Specification Document aims to provide in detail every aspect of the service PowerEnJoy, including its components, goals, constraints, functional and non-functional requirements. Use cases and scenarios for all the users involved will be provided as well to conform the product's objectives to the real world.

The last part of this document is reserved to the formalization of some features of the system involving the utilization of Alloy, a declarative specification language which provides a structural modeling tool based on first-order logic.

The high-level functionalities described in the RASD are intended for both developers and project managers. The former have to implement and test the functionalities while the latter must examine whether every requirement has been respected. It may also be useful to users in order to best take advantage of the service.

1.2 Description of the problem

The product described in this RASD is PowerEnJoy, a car-sharing service which offers to its users exclusively electric cars. It includes the common functionalities of its category: permitting to registered users to obtain the position of all the available cars, reserving one within a certain amount of time and continuously displaying the up-to-the-minute cost of the ride are just few of them. Moreover, PowerEnJoy stimulates users to behave virtuously towards the ecosystem by applying various types of discounts under specific conditions.

There are four software components that constitute the PowerEnJoy project. First, a back-end server provides APIs in order to simplify the communication related to the interactions of a user with the cars. Then two applications are available for a user to allow him/her the employment of every functionality: a web-based application (intended for visualization from desktop) and a mobile one. Lastly, every vehicle will be equipped with an on-board computer, used by the driver to manage the ride with the available options and see real-time information related to it, such as the time spent, the distance traveled and the total amount.

1.3 Goals

[G1] Registration of a user to the system

[G2] Finding the locations of the available cars

[G3] Reservation of a car

[G4] Expiration of reservation and penalization

[G5] Entry of registered user into the car

[G6] Start charging and notifying the registered user

[G7] Stop charging the registered user and lock the car

[G8] Safe areas for parking the reserved cars

[G9] Detection of extra passengers and applying discount

[G10] Detection of the battery status and applying discount

[G11] Detection of special parking areas and applying discount

[G12] Checking parking and battery constraints and penalization

1.4 Domain properties

- User's data are always valid.
- Location reported by the GPS is always accurate.
- Every user can reserve just a car per time.

- The service is only available to validated users.
- The service has employees to move the cars if their distribution in the map is unbalanced.
- The licence office provides the online service to let the system instantly check the user's documents during the registration.
- If the user's driver license expires, he gets unvalidated until he renew it.
- A call center is provided by the service to handle difficult situations.

1.5 Glossary

1.5.1 Definitions

- *Car*: electric vehicle provided by the service
- *Guest* or *Guest user*: person not registered to the service
- *Registered user*: see *User*
- *Safe area*: set of parking spots where a user can leave a car without penalization
- *User*: person with a valid driving license registered to the service

1.5.2 Acronyms

- *API*: Application Programming Interface
- *GPS*: Global Positioning System
- *OS*: Operating System, related both to desktop and mobile platforms
- *PIN*: Personal Identification Number
- *RASD*: Requirements Analysis and Specification Document
- *W3C*: World Wide Web Consortium

1.6 Constrains

1.6.1 Regulatory policies

While waiting for future conventions, at the moment toll and handicap parkings are forbidden. Timed parkings are also forbidden, since the user cannot ensure compliance with the deadline once left the car.

During the registration the system receive the user's permission to get his position and it has to handle sensible data according to the privacy law. To avoid SPAM the system can only use messages and notifications if strictly required to the proper operation of the system.

1.6.2 Hardware limitations

- User's mobile device:
  - Connection speed $\geqslant$ 3G
  - GPS
  - Enough memory available to install the app
- Car:
  - GPS
  - Weight sensor for each seat
  - Fast Internet connection
  - On-board computer with integrated system

1.6.3 Interfaces to other applications

Interface with an SMS gateway provider via standard SMS REST APIs, to verify the user's account and send important notifications.

1.6.4 Parallel operation

The server supports parallel reservations of cars from different users at the same time.

1.7 Reference documents

The Requirements Analysis and Specification Document has been composed following the indications and examples reported in the document ISO/IEC/ IEEE 29148, released by W3C, containing provisions for the processes and products related to the engineering of requirements for systems and software products and services throughout the life cycle.

With regards to the course named Software Engineering 2 and held by professors Luca Mottola and Elisabetta Di Nitto (Politecnico di Milano, a. y. 2016/17), the document conforms to the guidelines provided during the lectures and within the material of the course.

Chapter 2

Requirements

2.1 Functional requirements

The following requirements have been elicited with respect to the domain properties and assumptions mentioned above in order to satisfy the goals.

2.1.1 Basic requirements

- [G1] Registration of a user to the system:
  - The system needs to provide mandatory sign up and payment options for the guest users who wants to register to use the car sharing service.
  - Once the payment is successful and the guest user is registered, the registered user receives a password that can be used to access (login into) the system.
- [G2] Finding the locations of the available cars:
  - The system needs to provide the exact location of the cars that are available within a certain distance either from the current location of the registered users or from a specified address given(entered) by the registered users.
- [G3] Reservation of a car:
  - The system provides provision such that the registered users must be able to reserve only a single car among the available cars in a certain geographical region for up to one hour before they pick it up.

- [G4] Expiration of reservation and penalization:
  - The system checks if a reserved car is picked-up within one hour.
  - If not, the system tags the car as available again and the reservation expires.
  - The system penalizes the registered user who made the reservation and did not pick the reserved car within an hour, by making him to pay a fee of 1 €.
- [G5] Entry of registered user into the car:
  - The system must be able to identify (communicate with) the registered user when he/she is nearby the reserved car.
  - The system unlocks the reserved car and allows the registered user to enter it after identification of registered user as mentioned in the previous point.
- [G6] Start charging and notifying the registered user:
  - The system starts charging the registered user for a given amount of money per minute as soon as the engine is ignited.
  - The system must be able to notify the current charges to the registered user through a screen on the reserved car.
- [G7] Stop charging the registered user and lock the car:
  - The system must stop charging the registered user as soon as the reserved car is parked in a safe area and the registered user exits the reserved car.
  - The system must be able to lock the reserved car automatically at this point after the above operation is successfully done.
- [G8] Safe areas for parking the reserved cars:
  The safe areas are defined by the systems as follows:
  - Those areas predefined by the system in and around a specific geographical region (green areas displayed in the user interface screen)
  - The parking areas which are not private and located in the basement (underground).
  - The parking area in which the GPS signals are not very low (the system must suggest the user in this case).

2.1.2 Value added requirements

In addition to the above requirements, the system should motivate the virtuous behaviors of the registered user by satisfying the following requirements:

- [G9] Detection of extra passengers and applying discount:
  - The system must detect if the registered user has taken at least two other passengers onto the reserved car.
  - The system must calculate and apply a discount of 10% on the last ride if the above-mentioned point is true or satisfied.
- [G10] Detection of the battery status and applying discount:
  - The system must detect the percentage of the battery charge that has been consumed in the reserved car by the registered user during the last ride.
  - The system must calculate and apply a discount of 20% if the reserved car is left with no more than 50% of the battery empty.
- [G11] Detection of special parking areas and applying discount:
  - The system should detect if the reserved car is left in the special parking areas-where they can be recharged and the registered user takes care of plugging the car into the power grid.
  - The system must calculate and apply a discount of 30% on the last ride if the above check is true or satisfied.
- [G12] Checking parking and battery constraints and penalization:
  The system must check if either of the following conditions are true:
  - The distance between the reserved car (parked after the ride) and the nearest power grid station is more than 3KM (Kilometers).
  - The battery of the reserved car (parked after the ride) is consumed more than 80%.
  - The system must penalize the registered user by charging 30% more on the last ride if either of the two conditions mentioned above are true to compensate for the cost required to recharge the reserved car (parked after the ride).

2.1.3 Operations allowed to users

We have clearly elicited the requirements of PowerEnJoy for which the functional requirements are stated as follows:

- *Guest user*:
  - Sign up.
- *Registered user*:
  - Login.
  - Find the location of available cars.
  - Select his/her final destination.
  - Reserve an available car.
  - Receive notification of the reservation expiry and penalty.
  - Enter the reserved car (by communicating with the system).
  - Receive notification of the current charges.
  - Park the reserved car in safe areas.
  - Take at least two other passengers onto the reserved car and avail discount.
  - Minimize the consumption of the battery's charge in the reserved car to avail discount.
  - Park the reserved car in special parking areas and avail discount.
  - Select (Enable) the money saving option to get discount.

2.2 Non-functional requirements

2.2.1 User interface

The user interface of our application is thought to be used via web as well as a mobile application. There are two sketches of the UI screen which are displayed below.

The figure 2.1 shows the features and option in a mobile application. The first picture in the below area shows the homepage of the application before login. It shows the sign up and login option for the guest user. It also displays the availability of cars in nearby area through GPS.

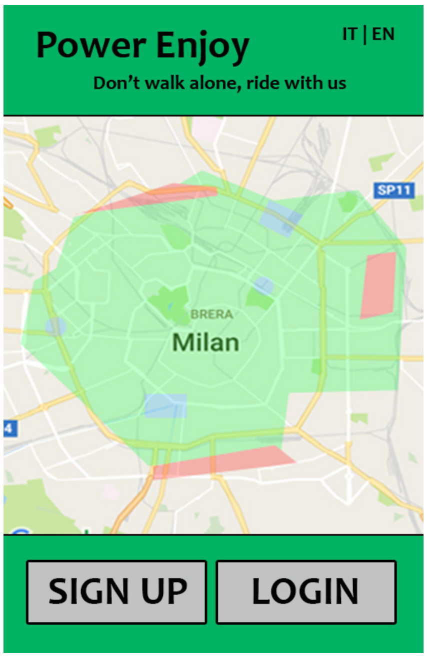
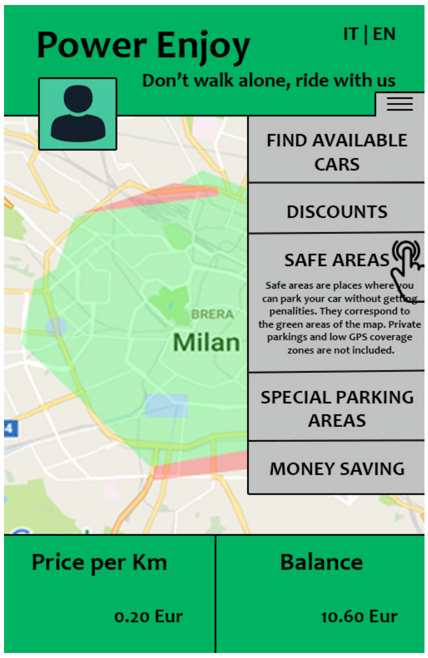


Figure 2.1: Mobile UI of PowerEnJoy login page



Safe areas are places where you can park your car without getting penalities. They correspond to the green areas of the map. Private parkings and low GPS coverage zones are not included.

Figure 2.2: Mobile UI of PowerEnJoy homepage

The figure 2.2 displays the screen after a registered user logins. It has options like: find the available cars, reserve a car, discounts, safe area, special parking areas, money saving option, price per km with current charge.

The figure 2.3 shows the homepage that any guest user can see. It displays the sign up and login option; options like finding available cars, reserving an available car, money saving option, charge per km with current charge, homepage of the desktop version.

The figure 2.4 represents the logged screen of a registered user. It displays the map of the city with safe areas; the current balance of the registered user; the battery status after the ride; and other options to find available cars; to know information about discounts, safe areas, special parking areas, etc. When the cursor is hovered on any of those options, a small area expands to provide information about that option as similar to the figure 2.2.

2.2.2 Documentation

We will release the following documents in order to organize our work in each phase of the development process and keep in touch with the stakeholders.

- RASD, Requirement Analysis and Specification Document, which de-

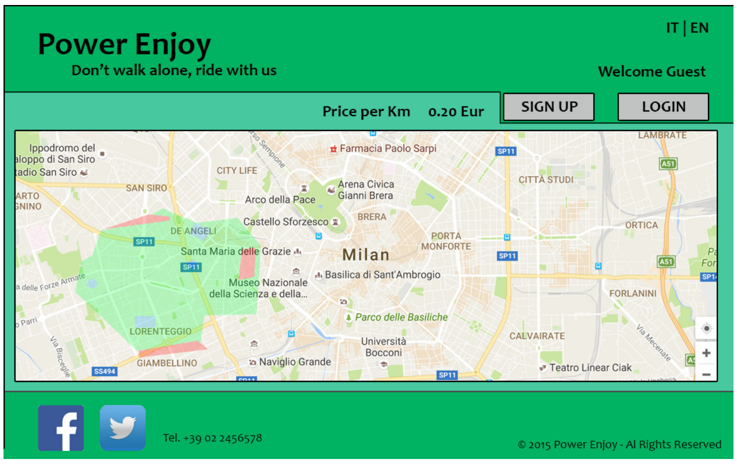


Figure 2.3: Desktop UI of PowerEnJoy login page

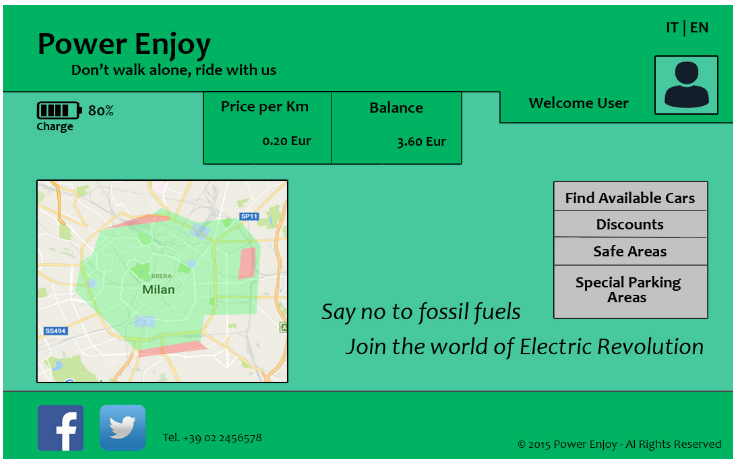


Figure 2.4: Desktop UI of PowerEnJoy homepage

fines our goals and assumptions and contains an overall description of the system (using scenarios and use-cases) and the models describing requirements and specifications.

- DD, Design Document, which contains a functional description of the system using models such as UML diagrams.
- Installation manual, which explains how to deploy the web site.
- User manual, which explains users how to use the main functionalities of the web site.

- Testing report, which contains the results of the testing activity performed on a system developed by another group.
- Project reporting, which is the result of some analysis done on the project activity.

Chapter 3

Scenarios identifying

In this section we are going to analyze some of the possible scenarios derived from the use of PowerEnJoy.

3.1 Scenario 1

Nick wants to go to his friend's wedding anniversary. He has come to know about the PowerEnJoy car-sharing service through some advertisements. He visits the website for the first time. He realizes that he needs to become a registered user in order to reserve an available car. He decides to register in the website and becomes a registered user after filling the mandatory details (driving license, codice fiscale, etc.) and completing the payment successfully. He receives his credentials (user ID, password) for further sign in.

Without knowing about the benefits and constraints of the PowerEnJoy, he just reserves an available car and travels to his destination and ends his ride after parking the reserved car in a nearby area to the place of his destination. Fortunately, it turns out to be a safe parking area and he has left the car with 60% of battery. He is notified of the final price mentioning that he has received 20% discount. Nick leaves the reserved car happily.

3.2 Scenario 2

After getting the knowledge of the benefits and constraints from the website of PowerEnJoy, Nick wants to reserve a car to attend a meeting in his office. This time, two of his colleagues are there with him and he reserves a nearby available car.

He takes a ride with the two colleagues, reaches the destination with more than 50% of battery charge remaining, parks in a safe area and gets an overall discount for 30% by the system (10% for riding with at least two extra passengers and 20% for not draining the battery more than 50%). Nick leaves the reserved car happily.

3.3 Scenario 3

Right after the scenario 2, Nick receives a telephone call saying that his wife is in maternity pain and he needs to take her to the hospital immediately. Meanwhile, the parked car (after Nick's ride) is the only nearest car for him. He decides to open the car and then reserve it.

Meanwhile, this parked car has been reserved by someone else and Nick is not even able to open the door of the parked car (since he was not able to reserve that car and hence the communication with that car is not possible). Finally, he searches for the other available cars and then takes a ride.

3.4 Scenario 4

Frank is a registered user to the PowerEnJoy service. He wants to make a ride to his university which is very far from his home. Frank reserves an available car with PowerEnJoy and rides to his college where he parks the reserved car in a safe area but has consumed more than 80% of the battery.

So, he is charged 30% more on his ride and Frank is notified about the final charge.

3.5 Scenario 5

Frank needs to take his girlfriend to a pub on a Saturday night. He decides to use the PowerEnJoy service. As he is a registered user, he reserves an available car. But his girlfriend arrives late. Also, so it has been more than an hour since he made the reservation and did not pick the reserved car.

Hence he is charged or penalized with 1€ and the reservation he made has been expired. So, Frank needs to start the process again.

3.6 Scenario 6

James is a registered user to the PowerEnJoy service. He has gone to Lugano (Switzerland) for a business visit. He has found PowerEnJoy to be cheaper than the train and decides to reserve an available car from Lugano (Swiss) to Milan (Italy), which is a cross country travel.

When he searches for the available cars to make a reservation, he is not able to spot any cars, in his geographical region as PowerEnJoy restricts cross country travel which are against its terms and conditions.

3.7 Scenario 7

James wants to save money, hence he activates the money saving option. As a registered user, he selects his destination and picks an available car. He is notified from the system about the station where he needs to park the reserved car in order to avail of the discount.

But, in a hurry, James leaves the reserved car in a safe area and ends the ride. Thus, he did not receive any discount in his final charge after the end of his ride.

3.8 Scenario 8

Mr. Potter is a guest to the PowerEnJoy service and he wants to register himself to make a ride when he needs. He uploads all the mandatory documents in the web page and registers himself after completing the payment successfully. He reserves an available car and travels more than the amount which he has paid while registering. He parks the reserved car in a safe area and ends his ride.

Mr. Potter is notified by the system that he must pay the balance by three days from the end of the ride otherwise his documents (driving license, codice fiscale, etc.) will be sent or notified to the local police, stating the issue and serious action will be taken against him as per the law. Also, he loses the privilege of being a prestigious member of PowerEnJoy and its services.

Chapter 4

UML models

4.1 Use cases

In this section we are going to analyze some of the use cases mentioned in the diagram of the figure 4.1 and derived from the scenarios described in chapter 3 of this document.

4.1.1 Guest sign-up

- *Actors*: Unregistered user
- *Entry conditions*: User must not be registered.
- *Flow of events*:
  - The user arrives at the home page of the mobile app or desktop version.
  - The user clicks on the Sign Up button.
  - The user inputs its personal data, including one valid ID and a valid drive license.
  - The user must confirm the registration browsing a link sent by the system to the email address specified by the user.
  - The system checks the profile and the input data are validated.
  - The user must input in the app's homepage the 6 digit code sent by the system to the specified mobile number.
  - The system validates the account, the user gets notified by sms and notification.

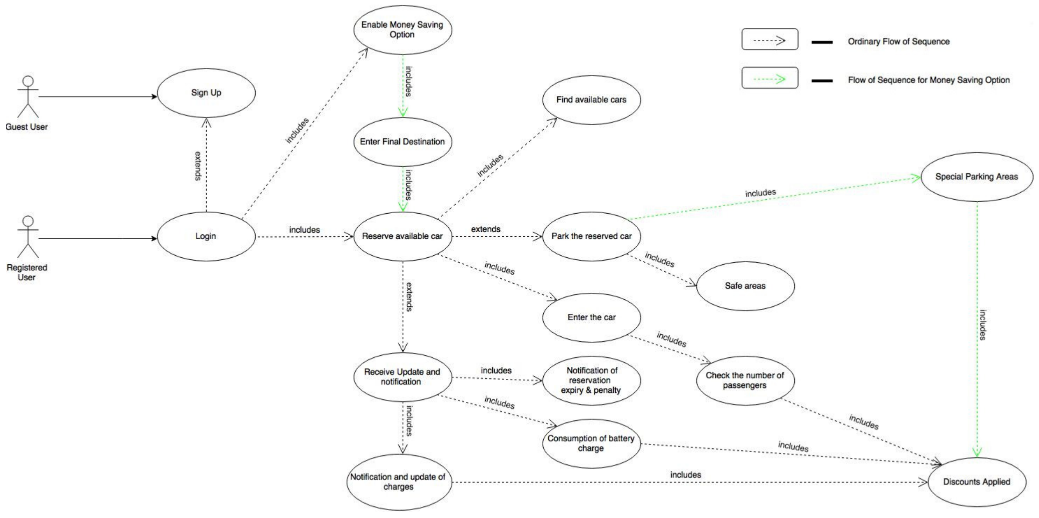

Guest User
Registered User
Sign Up
Login
Enable Money Saving Option
Enter Final Destination
Reserve available car
Find available cars
Park the reserved car
Special Parking Areas
Safe areas
Enter the car
Check the number of passengers
Receive Update and notification
Notification of reservation expiry & penalty
Consumption of battery charge
Notification and update of charges
Discounts Applied
includes
extends
Ordinary Flow of Sequence
Flow of Sequence for Money Saving Option

Figure 4.1: Use case diagram for PowerEnJoy

- *Exit conditions*: User's account is validated.
- *Exceptions*: If the system rejects the input data, the user can find explanations in the app's home page, where he can insert the missing data. If the user doesn't receive the SMS he can ask the app to send it again.

4.1.2 User login

- *Actors*: Registered user
- *Entry conditions*: User must have a confirmed account.
- *Flow of events*:
  - The user arrives at the homepage of the mobile app or desktop version.
  - The user inputs its credentials in the drop-down menu.
  - The user clicks on login.
  - The user is redirected to his personal page.
- *Exit conditions*: The user is successfully redirected to his personal page.
- *Exceptions*: The credentials specified by the user were wrong. The user is notified and he can input them again.

4.1.3 User reserves an available car

- *Actors*: Registered user
- *Entry conditions*: User must have a validated account and be logged in.
- *Flow of events*:
  - The user arrives at his personal page of the mobile app or desktop version.
  - The user clicks on Find Available Car to full size the map.
  - The user selects a car and clicks Reserve.
  - The user is notified with both SMS and app notification about the reservation and the instructions to abort it.
  - The user approaches the car.
  - The user asks the system to unlock the car.
  - The system checks the user's position through the GPS.
  - If the user's GPS position matches with the car's one, the car is unlocked.
  - The user can specify whether to enable the Money Saving Option and the destination (mandatory if this option is enabled).
  - The user ignites the engine and the system starts charging.
  - The system takes notes about the number of passengers.
  - The user drive to his destination.
  - If the Money Saving Option is enabled, the system suggests the user the special parking area where to park.
  - The user looks for an available parking spot, while verifying on the app's map that it's not forbidden by the service's policies.
  - The user parks the car.
  - The user stops the engine and the system stops charging.
  - The user leaves the car.
  - If the parking area isn't forbidden the car gets locked by the system, otherwise the user will be asked to get back in the car and change parking spot.

  - The system checks the battery status, the distance between the parking slot and the closest power station grid, and along with the previously acquired number of passengers, it applies penalties and discount according to the terms and conditions.
  - The user gets notified about the final charge along with the discounts.

- *Exit conditions*: The user gets notified by the system and the car is locked.

- *Exceptions*: If the GPS positions of the car and the user don't match, the user is notified to get closer to the car. If the user parks in a low GPS coverage area, the system will notify the user in time. The same if the user parks in a non-safe area. If the user doesn't get back in the car within X minutes, the car gets locked and the user gets penalties.

4.2 Sequence diagrams

4.2.1 Registration of a user

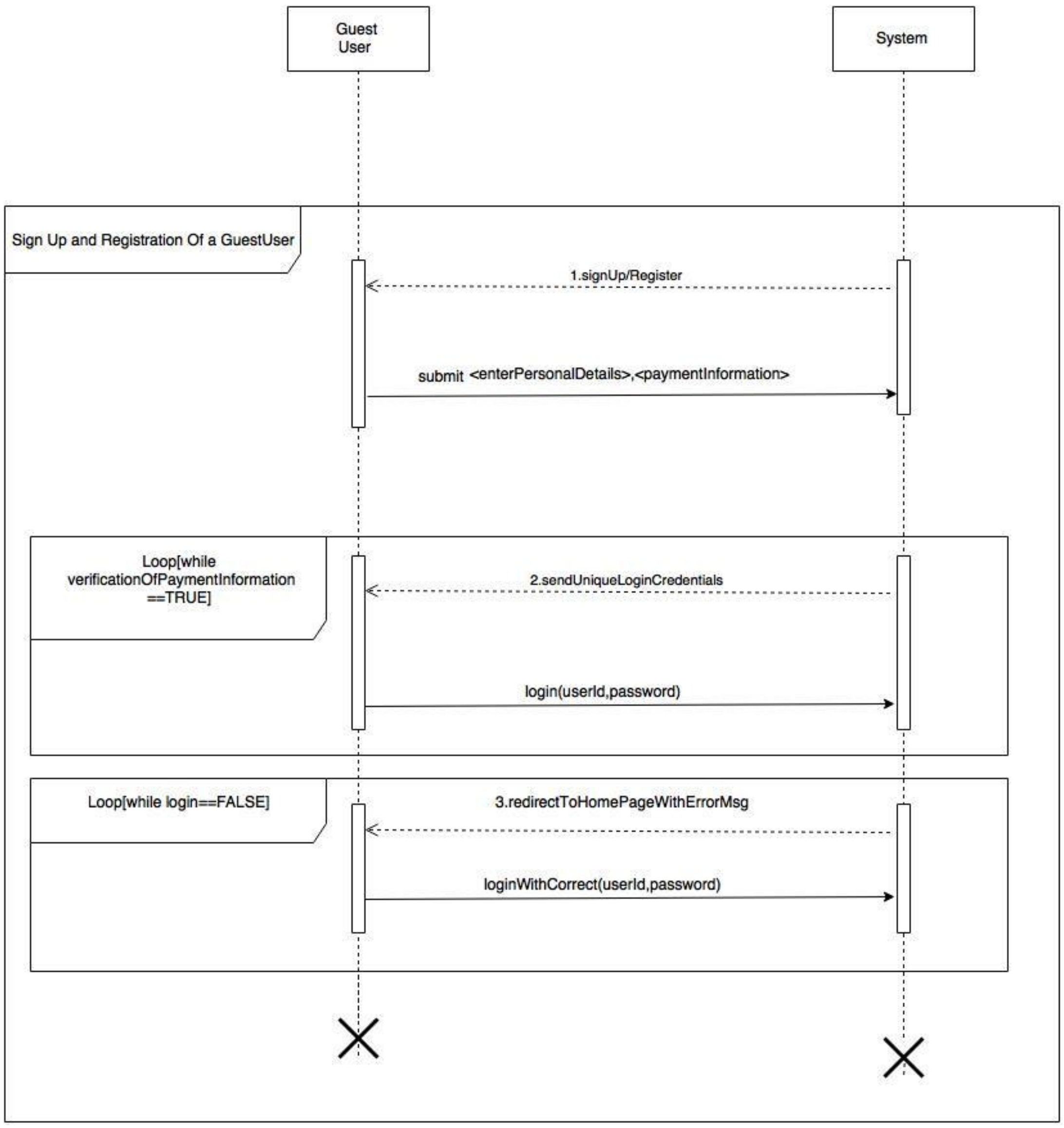

Guest User
System
Sign Up and Registration Of a GuestUser
1.signUp/Register
submit <enterPersonalDetails>,<paymentInformation>
Loop[while verificationOfPaymentInformation ==TRUE]
2.sendUniqueLoginCredentials
login(userId,password)
Loop[while login==FALSE]
3.redirectToHomePageWithErrorMsg
loginWithCorrect(userId,password)

Figure 4.2: Sequence diagram for the registration of a user

4.2.2 Reservation of a car

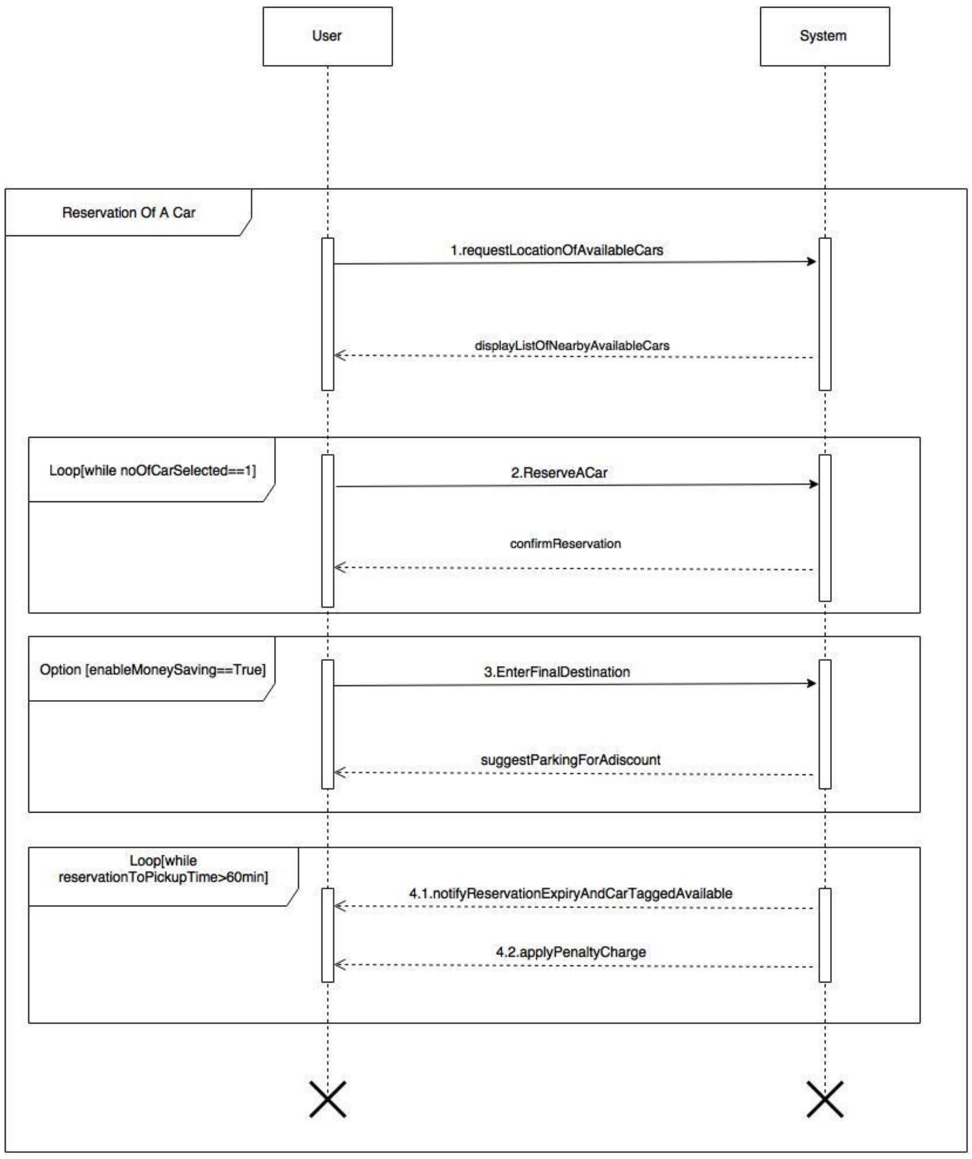

User
System
Reservation Of A Car
1.requestLocationOfAvailableCars
displayListOfNearbyAvailableCars
Loop[while noOfCarSelected==1]
2.ReserveACar
confirmReservation
Option [enableMoneySaving==True]
3.EnterFinalDestination
suggestParkingForAdiscount
Loop[while reservationToPickupTime>60min]
4.1.notifyReservationExpiryAndCarTaggedAvailable
4.2.applyPenaltyCharge

Figure 4.3: Sequence diagram for the reservation of a car

4.2.3 Discounts

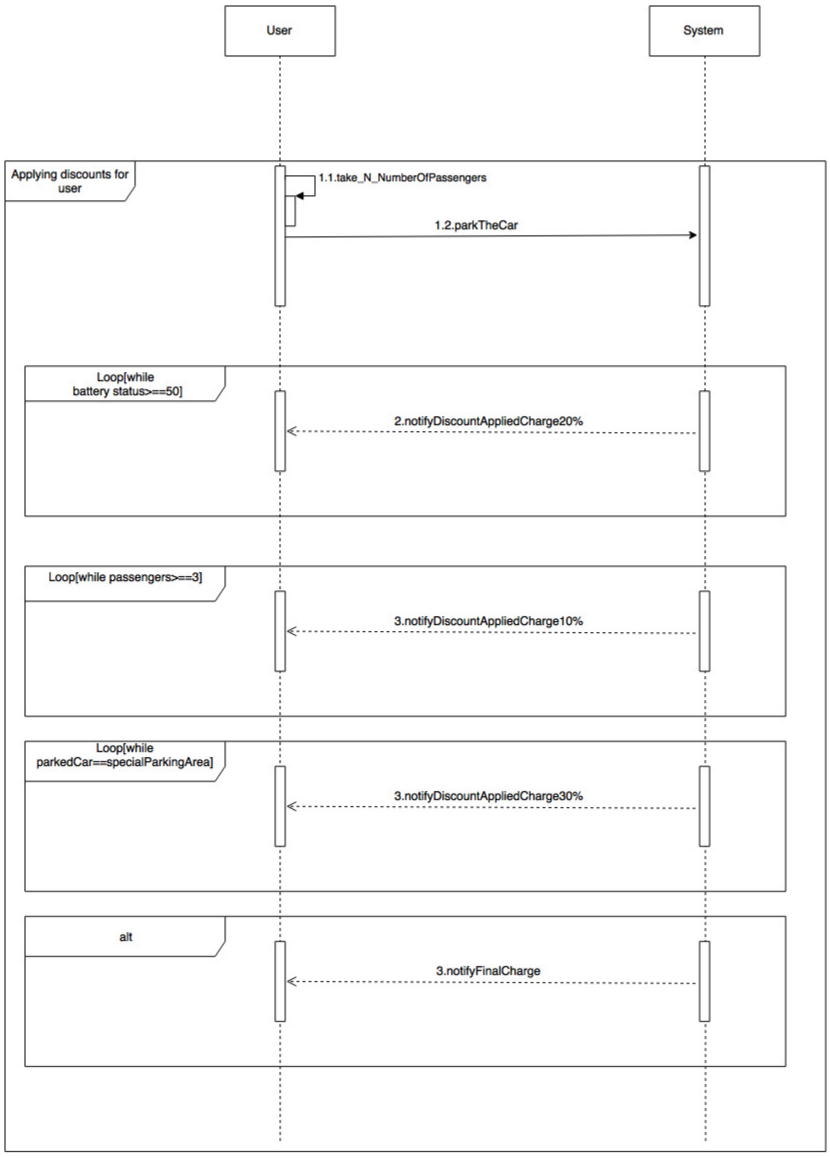

User
System
Applying discounts for user
1.1.take_N_NumberOfPassengers
1.2.parkTheCar
Loop[while battery status>==50]
2.notifyDiscountAppliedCharge20%
Loop[while passengers>==3]
3.notifyDiscountAppliedCharge10%
Loop[while parkedCar==specialParkingArea]
3.notifyDiscountAppliedCharge30%
alt
3.notifyFinalCharge

Figure 4.4: Sequence diagram for the application of discounts

4.3 Class diagram

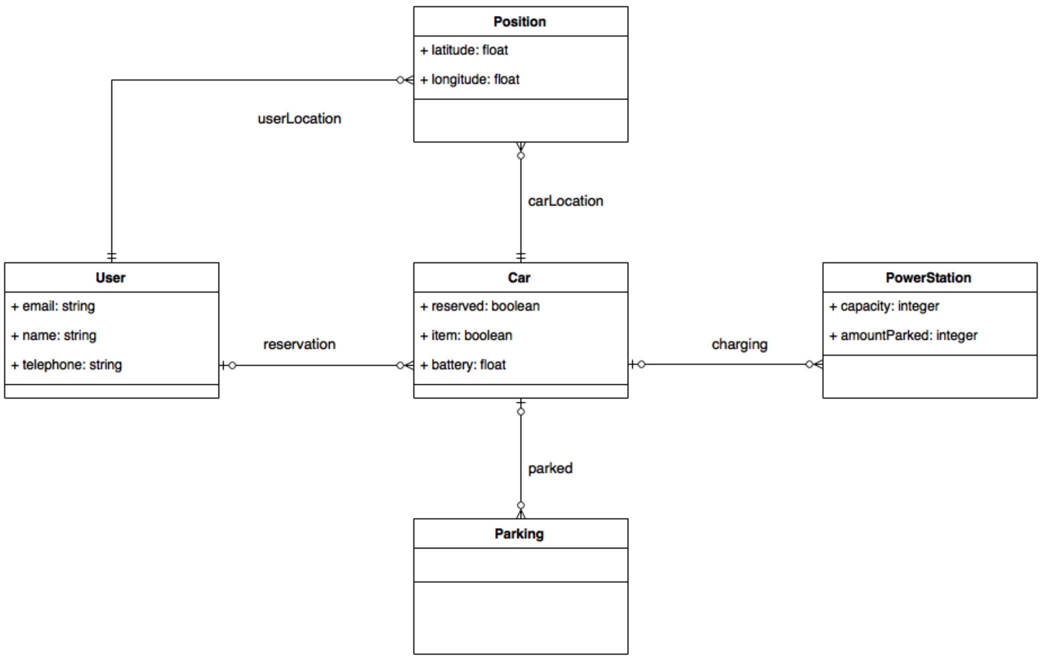


Figure 4.5: Class diagram for PowerEnJoy

4.4 Activity diagrams

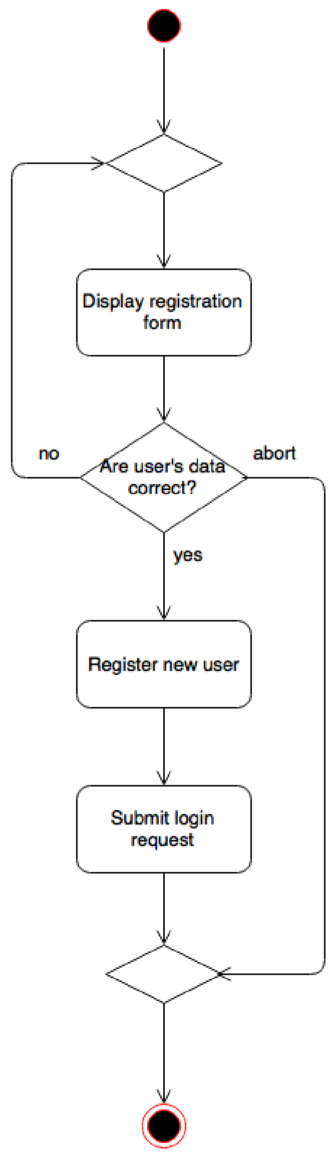


Figure 4.6: Activity diagram for the registration of a user

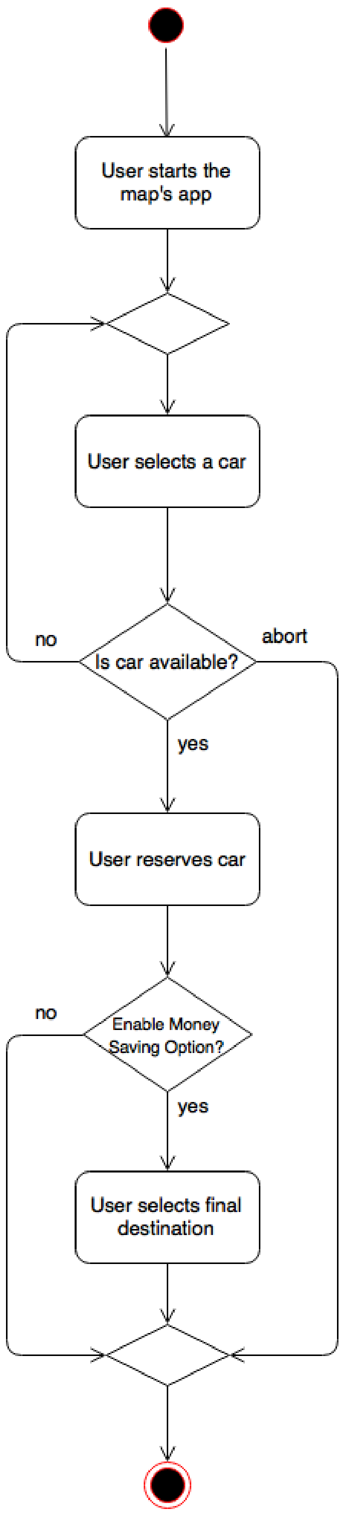

User starts the map's app
User selects a car
no
Is car available?
abort
yes
User reserves car
no
Enable Money Saving Option?
yes
User selects final destination

Figure 4.7: Activity diagram for the reservation of a car

Chapter 5

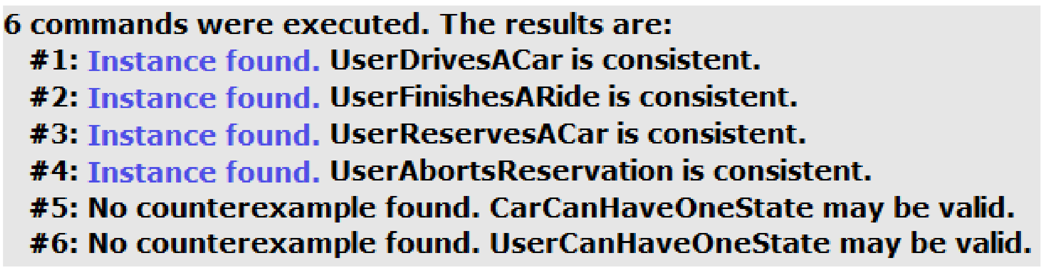
Alloy modelling

5.1 Model

```
open util/boolean

sig Car {
    position: one Position,
    reserved: one Bool,
    running: one Bool,
    user: one User,
    battery: one Float
} {
    battery > 0
    battery < 100
}

sig User {
    email: one String,
    name: one String,
    telephone: one String,
    resvCar: lone Car,
    drivingCar: lone Car,
    position: one Position
}

sig Position {
    latitude: one Float,
    longitude: one Float
```

```
}

sig String {}

sig PowerStation {
    carParked: set Car,
    capacity: one Int,
    position: one Position,
    amountParked: one Int
} {
    capacity > 0
    amountParked > 0
}

sig Parking {
    car: lone Car
}

fact capacityConstraint {
    all pw: PowerStation | #pw.carParked < pw.capacity
}

fact carLocationConstraint {
    all c: Car, ps:PowerStation, p: Parking | (c in
    ↪  ps.carParked => c != p.car) and (c = p.car => not c in
    ↪  ps.carParked)
}

fact coordinatesConstraint {
    all u: User | #u.drivingCar = 1 implies u.position =
    ↪  u.drivingCar.position
    all c:Car, pw: PowerStation | (c in pw.carParked) implies
    ↪  c.position = pw.position
    all pw1, pw2 : PowerStation | pw1 != pw2 implies
    ↪  pw1.position != pw2.position
}

fact carConstraints {
    all c: Car | (c.running = True and #c.user = 1) implies
    ↪  c.user.drivingCar = c
    all c: Car | (c.reserved = True and #c.user = 1) implies
     ↪  c.user.resvCar = c
    all c: Car | (c.running = True implies c.reserved = False)
     ↪  and (c.reserved = True implies c.running = False)
    all c: Car, ps:PowerStation, p: Parking | ((c in
     ↪  ps.carParked) or (c = p.car)) implies c.running = False
    all c: Car | (c.running = True <=> #c.user.drivingCar = 1)
     ↪  and (c.reserved = True <=> #c.user.resvCar = 1)
}

fact userConstraints {
    all u: User | #u.drivingCar = 1 implies u.drivingCar.user =
     ↪  u
    all u: User | #u.resvCar = 1 implies u.resvCar.user = u
    all u: User | (#u.resvCar = 1 implies #u.drivingCar = 0)
     ↪  and (#u.drivingCar = 1 implies #u.resvCar = 0)
}

pred PowerStationUpdate(c,c':Car) {
    all p:PowerStation | (c in p.carParked => (one
     ↪  p':PowerStation | (
        p'.carParked = p.carParked - c + c' and
        p.capacity = p'.capacity and
        p.position = p'.position and
        #p.carParked = #p'.carParked)
        )
    )
}

pred UserReservesACar(c,c':Car,u,u':User) {
    #u.resvCar = 0
    #u.drivingCar = 0
    c.running = False
    c.reserved = False
    c'.running = False
    c'.reserved = True
    c'.user = u'
    c.position = c'.position
    c.battery = c'.battery
    u'.resvCar = c'
    #u.drivingCar = 0
    powerStationUpdate[c,c']
    powerStationUpdate[c',c]
}

pred UserAbortsReservation(c,c':Car,u,u':User) {
    UserReservesACar[c',c,u',u]
}

pred UserDrivesACar (c,c':Car, u,u':User) {
    c.user = u
    c'.user = u'
    c.position = c'.position
    c.battery = c'.battery
    c.reserved = True
    c'.running = True
    u.resvCar = c
    u'.drivingCar = c'
    userUnplugsACar [c]
}

pred UserUnplugsACar(c:Car) {
    all p:PowerStation | c in p.carParked => one
     ↪  p':PowerStation | (
        p.capacity = p'.capacity and p.position = p'.position
        and p'.carParked = p.carParked - c and p.amountParked =
         ↪  plus[p'.amountParked,1])
}

pred UserFinishesARide(c,c':Car) {
    c.battery = c'.battery
    c.position = c'.position
    c.running = True
    c'.reserved = False
    c'.running = False
    UserPlugsACarIn[c']
}

pred UserPlugsACarIn(c':Car) {
    all p,p':PowerStation | p.capacity = p'.capacity and
     ↪  p.position = p'.position
    all p:PowerStation | p.position = c'.position and c' not in
    ↪ p.carParked =>
    one p':PowerStation | (p'.carParked = p.carParked + c' and
    ↪ p'.amountParked = plus[p.amountParked,1])
    all p:PowerStation | c' in p.carParked => one
    ↪ p':PowerStation | (p'.carParked = p.carParked - c'
    and p'.amountParked = minus[p.amountParked,1])
}

assert CarCanHaveOneState {
    all c:Car | not (c.reserved = True and c.running = True)
}

assert UserCanHaveOneState {
    all u:User | not (#u.drivingCar = 1 and #u.resvCar = 1)
}

run UserDrivesACar for 5
run UserFinishesARide for 5
run UserReservesACar for 5
run UserAbortsReservation for 5
check CarCanHaveOneState for 5
check UserCanHaveOneState for 5
```

5.2 Alloy result



```
6 commands were executed. The results are:
  #1: Instance found. UserDrivesACar is consistent.
  #2: Instance found. UserFinishesARide is consistent.
  #3: Instance found. UserReservesACar is consistent.
  #4: Instance found. UserAbortsReservation is consistent.
  #5: No counterexample found. CarCanHaveOneState may be valid.
  #6: No counterexample found. UserCanHaveOneState may be valid.
```

Figure 5.1: Alloy result of the model

5.3 World generated

5.3.1 UserDrivesACar

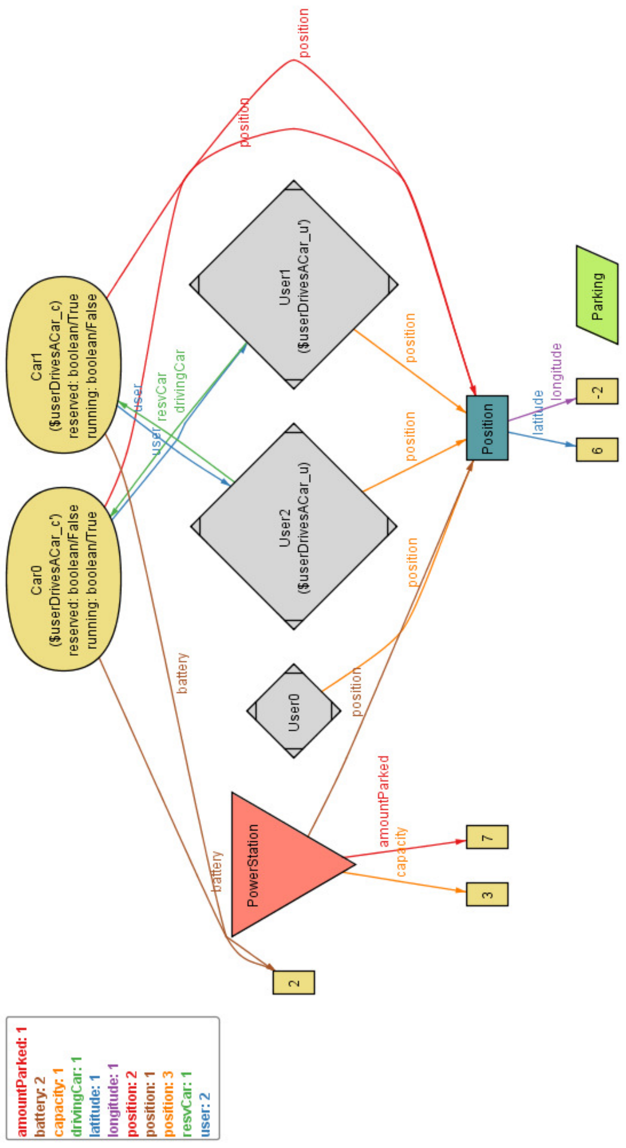


Figure 5.2: World generated for predicate #1

5.3.2 UserReservesACar

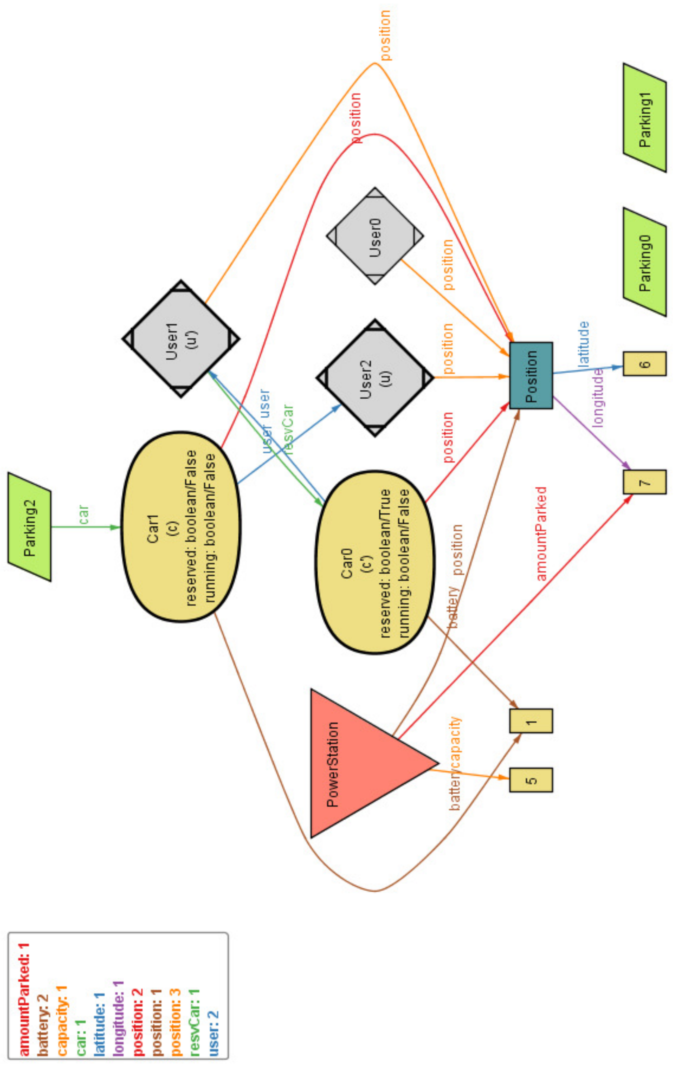


Figure 5.3: World generated for predicate #3

Chapter 6

Used tools

During the creation of this Requirement Analysis and Specification Document the following tools has been utilized:

- *Alloy Analyzer*: verification of the consistency of our model.
- *Draw.io*: design of the UML diagrams.
- *GitHub*: version control for the development of this document.
- *Photoshop*: drawing of the mock-ups of both versions of the app.
- *TeXstudio*: drafting of this document.

Chapter 7

Hours of work

7.1 Francesco Fabiani

- 23/10/2016: 1h30min
- 24/10/2016: 2h
- 25/10/2016: 1h
- 26/10/2016: 1h
- 27/10/2016: 1h
- 29/10/2016: 2h
- 02/11/2016: 3h
- 03/11/2016: 1h30min
- 05/11/2016: 2h
- 07/11/2016: 1h
- 08/11/2016: 2h30min
- 10/11/2016: 2h
- 11/11/2016: 3h
- 12/11/2016: 2h30min
- 13/11/2016: 4h

7.2 Jagadesh Manivannan

- 22/10/2016: 1h
- 23/10/2016: 1h
- 25/10/2016: 1h30min
- 26/10/2016: 1h
- 27/10/2016: 2h
- 29/10/2016: 1h
- 30/10/2016: 2h30min
- 31/10/2016: 1h
- 02/11/2016: 2h30min
- 03/11/2016: 1h
- 04/11/2016: 2h
- 06/11/2016: 1h
- 07/11/2016: 2h
- 08/11/2016: 1h
- 09/11/2016: 2h
- 10/11/2016: 2h
- 11/11/2016: 1h30min
- 12/11/2016: 3h
- 13/11/2016: 2h30min

7.3 Niccolò Pozzolini

- 24/10/2016: 2h
- 25/10/2016: 1h
- 26/10/2016: 2h
- 27/10/2016: 1h
- 28/10/2016: 1h30min
- 29/10/2016: 2h
- 01/11/2016: 2h
- 02/11/2016: 3h
- 05/11/2016: 1h
- 06/11/2016: 1h
- 08/11/2016: 3h
- 10/11/2016: 2h
- 11/11/2016: 1h30min
- 12/11/2016: 3h30min
- 13/11/2016: 4h

Chapter 8

Changelog

8.1 v1.1

- Changed the alloy results
- Added world generated diagrams for the relevant predicates