POLITECNICO
MILANO 1863

Computer Science and Engineering
Project of Software Engineering 2

PowerEnJoy

# Requirements Analysis

# and

# Specification Document


**Authors:**

Alberto Zeni 813977

Manuel Parenti 876085


**Reference Professor:** Mottola Luca

Accademic Year 2016–2017

# Table of contents

# 1 Introduction

## 1.1 Description of the given problem

PowerEnJoy is a car sharing service based on a mobile and web application. It provides its users with the tools and functionalities needed to pick up a car in a parking zone near them and ride to their destination. The system should be able to register new users with their personal information, such as name, surname, address, email and payment information. Once a user is registered to the system he should be able to reserve a car using the web or the mobile app, the position of the nearest car available is calculated by the system using the user position (provided by the gps or inserted manually by the user).

In the context of smart and green cities, car sharing is a really important service that gives the opportunity of driving cars (possibly with some passengers) in a more sustainable way to people that donʾt own a car or that just need to move to areas not covered by public transport.

PowerEnJoy enables its customers to drive ecologic cars at their need, either they want to be eco-friendly or just want to reach a place and they donʾt have an available transport.

## 1.2 Actors

User: a person who, after a successful login through the mobile or web app, can select a specific area (based on gps or manual insertion) and see the cars available in that area and is able to reserve a car and benefit of the car sharing service. He first needs to register an account with his information and a valid driving license.

Dispatcher: a person or a system that can access the information of all the cars through an interface of the application on their terminal/device. He can decide which cars need an on-field operation of an operator.

Operator: A person that follows the dispatcherʾs instructions, he needs to drive the cars to safe areas or to special parking areas, where he should also plug all the cars that are not already plugged. He has access to the keys of the cars. His account is registered by the company.

Users and operators may be overlapping sets. For example, some operator, after work, could use one of the rental cars to go home, but since he can use his operator account only to move cars selected from the dispatcher, he might have a normal user account to benefit from the service.

## 1.3 Goals

Users should be able to:

• <G1> Sign up into the system;

• <G2> Log into the system;

• <G3> Reserve a car for up to one hour before they pick it up;

• <G4> Find a car near their current location or near a specified location;

• <G5> Pick up a car from a safe area and reach another safe area, as long as they pay the service;

• <G6> Drive a car if the amount of money they have on their account is enough;

• <G7> Pay for their rides with the money they put on their PowerEnJoy account;

• <G8> Receive discounts based on their good behavior;

• <G9> Pay for the additional fees based on bad behavior;

The dispatcher should be able to:

• <G9> Know the position for all the cars;

• <G10> Know if a car is on charge or not;

• <G11> Send operators to move cars that are not parked in a safe area or that have low percentage of battery, and also to place cars in charge if they are parked in a special area but the plug is not in;

## 1.4 Glossary

USER: A person who is registered on PowerEnJoy, he can register, find information about his account, search for a car and use the car sharing service using a web application via a browser or using a mobile application. Users are provided with an account ID and a password which they need to use to access the service.

SAFE AREA: Public parking areas defined by the PowerEnJoy company, these

won't cause any police officer to write a ticket for the car or even remove the car.

SPECIAL PARKING AREA: These areas are defined by the PowerEnJoy company and they are provided with electric plugs for charging the battery of the cars.

RESERVATION: This term refers to the possibility of reserving a car for a specific user. He can do this with the web application or the mobile app, after he finds a car he wants to pick up. The cars will be reserved for that user for an hour, and after the time expires if the user doesn't pick up the car he is charged of a small fee. Users can reserve a car for up to an hour before they need to drive.

DISPATCHER: A person or a system that will manage the parking and recharging of the cars. The dispatcher will send operators on field to do the operations required in order to overcome the eventual bad behavior of some users.

## 1.5 Identifying stakeholders

Our stakeholder is the professor, who gave us this project to work on during the semester. The development of the project is focused on requirement analysis, design, testing, and planning.

We will define and document the main functionalities of the system, which are requested in the assignment document, with some assumptions, generalizations and some extensions that in our opinion are needed for the system to work in the vast majority of cases that can happen.

Starting from this consideration we can imagine that a hypothetical stakeholder

for our system could be a company that would have asked us to realize a platform

to manage an electric car sharing service in an efficient way.

# 2 Overall Description

## 2.1 Product perspective

The proposed solution for the system that we analyzed is a web and mobile application that has some interaction with an online payment platform such as a bank provided service or a virtual service (I.E. PayPal). The application will have a user and an administrative interface. The application will provide API to implement new features in the future (i.e. bike or different types of vehicles sharing).

## 2.2 User characteristics

The type of user that we expect to use our application is someone who wants to be able to drive a car without owning one, or just pick up a rental car in an easy and immediate way and drive around with an electric car.

The user needs a driving license to register to the system.

## 2.3 Constraints

### 2.3.1 Regulatory policies

The system must ask the user the permission to get his position and to use his sensible data such as the information saved on his account and driving license info, respecting the privacy law.

### 2.3.2 Interfaces to other applications

PowerEnJoy should interface with multiple applications such as the payment service to manage the account balance, the insurance company's system and the national vehicle registration system.

### 2.3.3 Parallel operation

Our application must provide support for multiple parallel operations from different users.

Every user that wants to access our system can do it independently from the others.

## 2.4 Domain Properties

We suppose that the following properties hold in the analyzed domain.

• The GPS of all the cars always give the right position.

• When a client registers himself in the system the system always gives back a password.

• Special parking areas are already defined by the company.

• The system knows when a car is charging or not.

• The cars that are available on the search function of the application are always parked in safe areas or special parking areas.

• The cars have sensors that recognize the number of passengers.

• The cars have sensors for the remaining battery life.

• All the cars are the same, so every car can host the same number of passengers.

• When a client leaves the car, he will not come back.

• Each ride is independent to the others.

• Every time a user reserves a car he receives a confirmation message.

• If a user has an active reservation when he's near the car he can unlock it by inserting the license plate of the car on the unlock page of the application on his smartphone, if he's in a certain maximum range from the car.

## 2.5 Assumptions

There are a few points that aren't clear in the document, so we assumed some facts for our system:

• Users will behave correctly and will be respectful of the cars, we can trust them to not cause intentional damage and try to cheat to avoid payment.

• Cars not parked in safe areas are moved from some operator of the company.

• If a user parks in a not safe area his last ride will be charged 20% more.

• Every violation of the traffic laws will be charged on the user account.

• Whoever rides the car all the responsibilities are upon the user that reserved the car.

• If a user enters in the car the system will start charging him after 5 minutes if he doesn't start the engine.

• Users cannot have more than one active reservation.

• If an accident occurs the car will contact automatically call an affiliated insurance company, via a device provided by them, that will manage everything.

• If a user gets caught breaking laws the tickets are all going to be forwarded to him.

• The car will notify the user if a special parking area is nearby.

• There is a small window of time in which the user can exit the car and re-enter it, before the system terminates the ride and charges the user.

• The same amount of time is given to the user to place the electric plug to charge the car to obtain the relative discount.

• Users can gain the discount for taking passengers only if the passengers are in the car at the rides' start. A user can always pick up passengers during the ride but no discount will be applied for that.

• Users cannot find a car that has low battery, because when a car is left with low battery some operator of the company will place it in charge.

• The discounts are cumulative.

• All the discounts are applied if and only if the user parks the car in a safe area.

• Operators have access to internet through a smartphone.

# 3 Requirements

In order to satisfy the goals, we have written some requirements, assuming that all the domain properties and assumptions, written above, hold for our purposes.

## 3.1 Functional Requirements

### 3.1.1 <G1> Allow a person to register into the service

- <R1> The person must not be already registered in the system, with the same SSN, email or driving license information
- <R2> The person who wants to register into the system must have a valid driving license
- <R3> The system must give a unique password to the user
- <R4> Every user should have a unique login name
- <R5> Users need to accept Terms of Use

### 3.1.2 <G2> Allow user to log into the application

- <R1> The user must be already registered in the system
- <R2> The user must insert his login name and the right password in the login screen
- <R3> The system verifies the validity of the user's driving license

### 3.1.3 <G3> Reservation of a car for up to one hour before a user picks it up

- <R1> A car cannot be reserved if it is in use or already reserved
- <R2> To reserve a car a user must be registered and logged in
- <R3> When the user unlocks the car, the ride can start
- <R4> The time of the reservation is saved
- <R5> After one hour from the user's reservation request the reservation expires if the car isn't picked up
- <R6> If the car isn't picked up a fixed fee is applied by the system

• <R7> Users that have debts cannot reserve cars

• <R8> Users need to leave a deposit to handle unexpected outgoings

• <R9> To rent a car the user needs to have a specified minimum amount of money on his PowerEnJoy account

• <R10> After the reservation is accepted by the system the user is notified with a confirmation message reporting the position of the car and the time he is expected to pick up the car.

#### 3.1.4 <G4> Find a car near their current location or near a specified location

• <R1> Data from the carʾs GPS sensor is retrievable in real time

• <R2> A location picked from the user is required for a search of a vehicle

• <R3> All available (non-reserved) cars in a certain range from the userʾs selected location are displayed after the user starts the search

• <R4> Displayed cars show the exact location (address, coordinates) so that a user can easily reach them.

• <R5> Displayed cars also show some information about the state of their remaining battery charge.

• <R6> If no cars are available the user can ask the system to check periodically the availability of cars in the specified area and if a car becomes free it notifies the user

#### 3.1.5 <G5> Pick up a car from a safe area and reach another safe area

• <R1> A car can be unlocked only by the user that reserved it or by an operator

• <R2> A car can be unlocked by the user if he writes the carʾs license plate on the application

• <R3> To unlock a car a user must be inside a certain range from the car

• <R4> 5 minutes after the user has entered the car, the system starts charging him even if he didnʾt start the car

• <R5> To rent a car the user needs to have a specified minimum amount of money on his PowerEnJoy account

• <R6> When the ride cost exceeds the userʾs account balance a debt is created

• <R7> After the ride is complete the car closes itself

• <R8> A ride is complete if the user has been out of the car for more than 2 minutes

• <R9> A map of the special parking areas is accessible from the application

#### 3.1.6 <G6> Drive their car if the amount of money they have on their account is enough

• <R1> When the user is logged in the system checks its account balance before granting him to reserve a car

• <R2> If the ride cost exceeds the account balance the deposit will be used to pay the excess and a debt is assigned to the user

• <R3> Users that have debts need to recharge the account in order to make a reservation

#### 3.1.7 <G7> Pay for their rides with the money they put on their PowerEnJoy account

• <R1> When the ride is complete its cost is scaled from the account of the user who reserved the car.

• <R2> Users can charge their account with online payment methods

• <R3> If the ride cost exceeds the account balance the deposit will be used to pay the excess and a debt is assigned to the user

#### 3.1.8 <G8> Receive discounts based on their good behavior

• <R1> A discount of 10% on the ride is applied if the user picks up at least 2 passengers

• <R2> The number of passengers is retrieved via a specific sensor

• <R3>If a car is left with no more than 50% of the battery empty, the system applies a discount of 20% on the last ride

• <R4> If a car is parked in a special parking area and it is plugged by the user, the system applies a discount of 30% on the last ride

• <R5> If a car is left at more than 3 KM from the nearest special parking area or with more than 80% of the battery empty, the system charges 30% more on the last ride

#### 3.1.9 <G9> The dispatcher knows the position of all the cars

• <R1> Data from the car's GPS sensor is retrievable in real time

### 3.1.10 <G10> The dispatcher knows the battery percentage of every car

- <R1> Information about the state of their remaining battery charge is retrievable in real time

### 3.1.11 <G11> The dispatcher can send operators to move or charge cars when needed

- <R1> The dispatcher can notify an operator with information about the car that needs to be moved or placed in charge
- <R2> The operators can only pick up cars that are not in use or reserved
- <R3> The dispatcher sends operators on field to move just cars with low battery or parked in non-safe areas

## 3.2 Non-functional Requirements

The system we describe needs to have the following nonfunctional requirements that will be important for the end users:

• **Performance**: the system has to be reactive and able to answer to a high number of simultaneous requests.

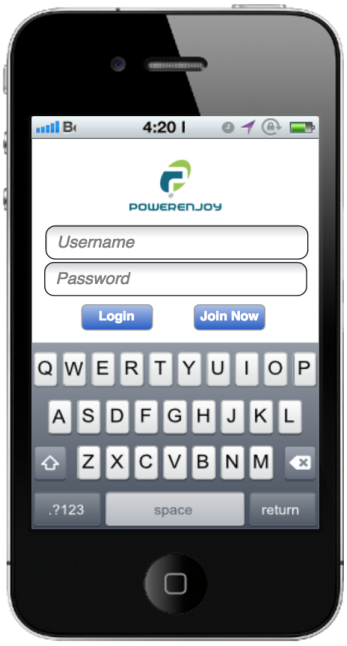
• **Reliability**: the system should be running 24/7. The more the system is available the more customers we are able to satisfy. Anyway, this is not the most critical of the systems so if there are some issues we will need to recover just for not losing money.

• **Data integrity, consistency and availability**: since the users load sensible data and even money on the system we need to guarantee that this kind of data is never lost and always accessible by the user.

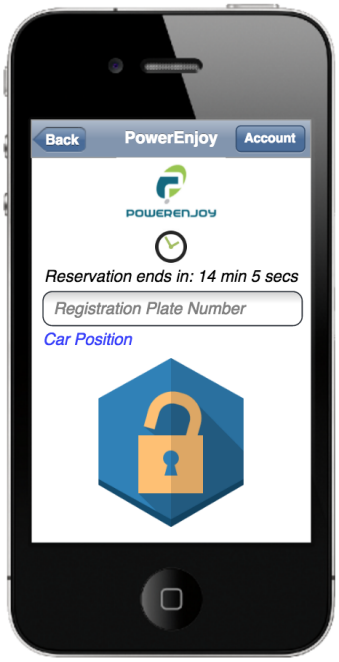
• **Security**: as previously said we need to manage sensible information, so we need to also guarantee that our system prevents unauthorized access to them.

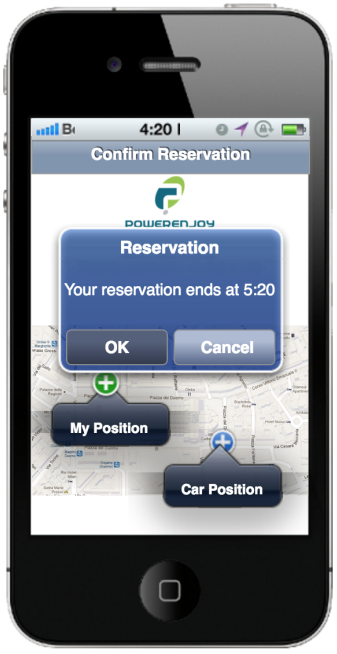
**Client Interface:**

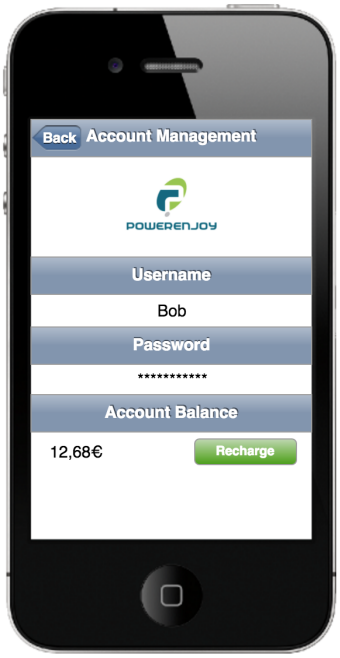
**mobile**





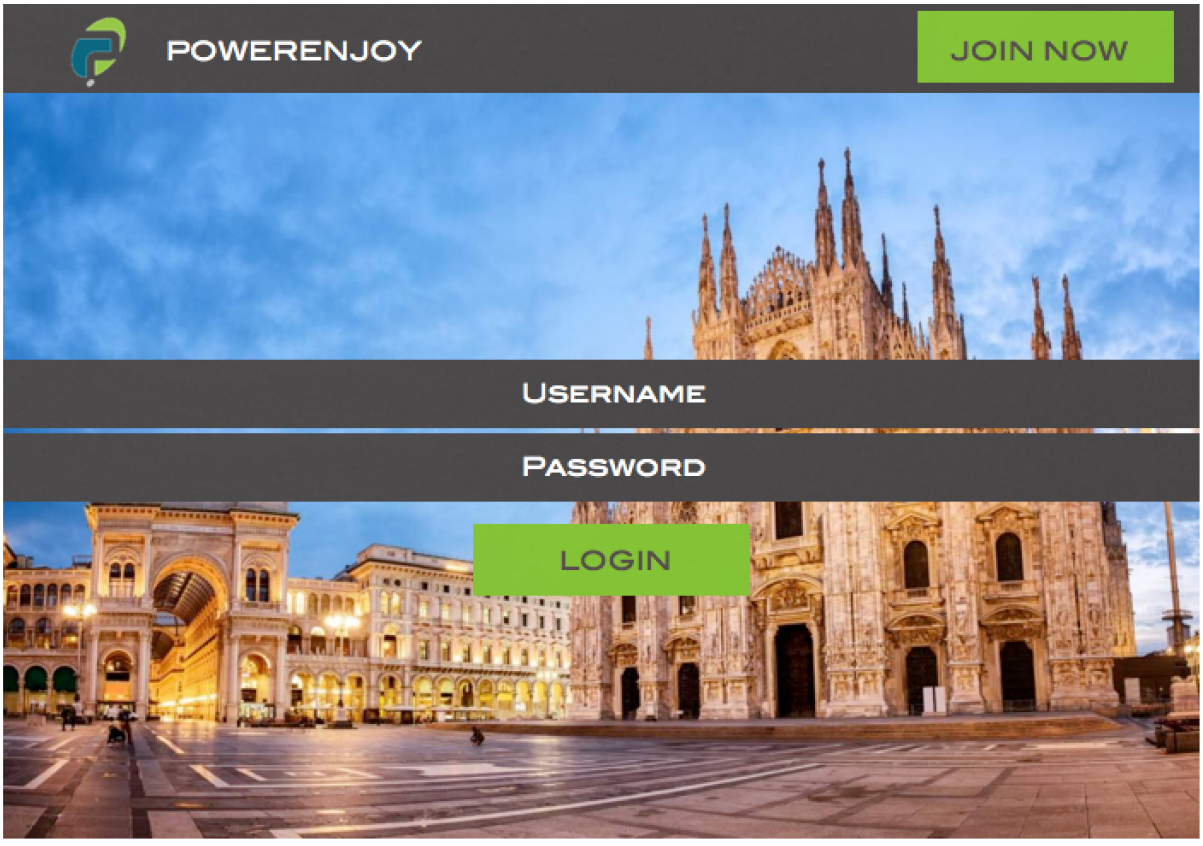


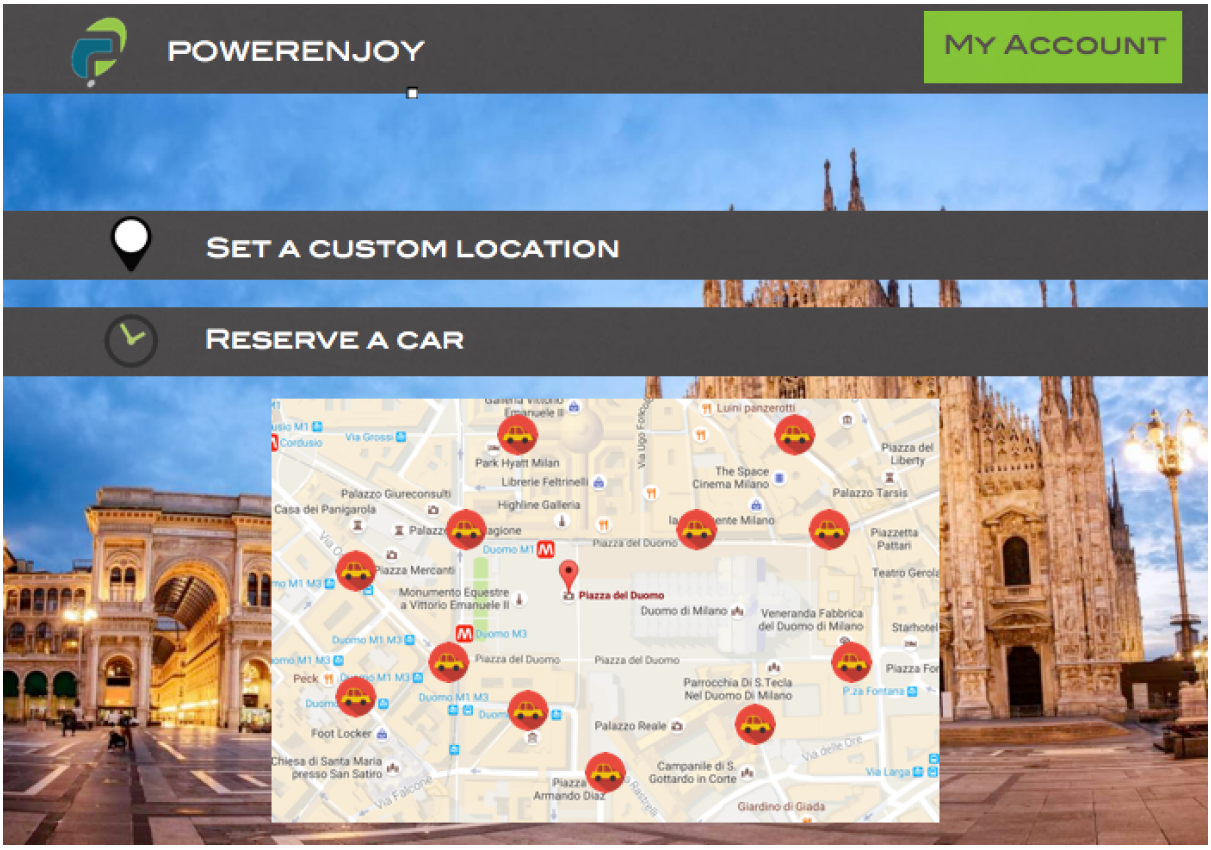



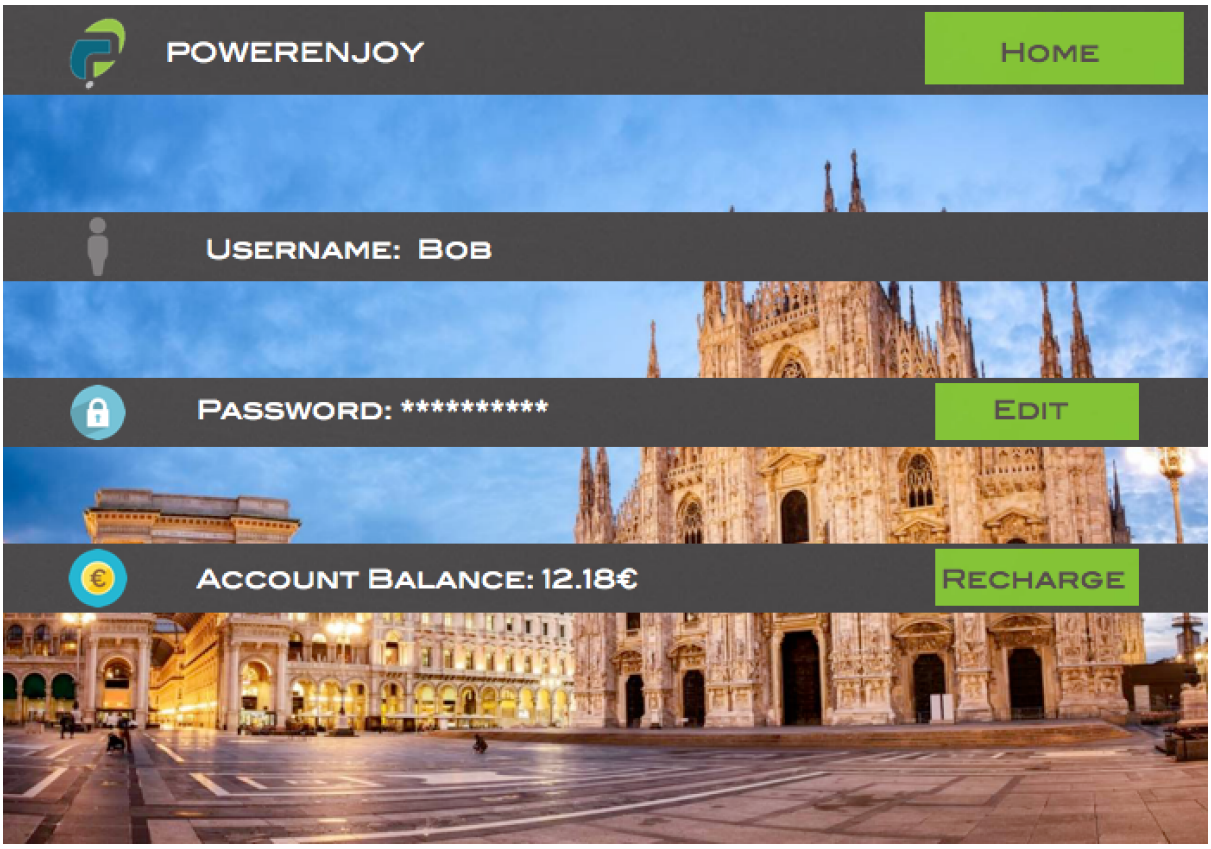



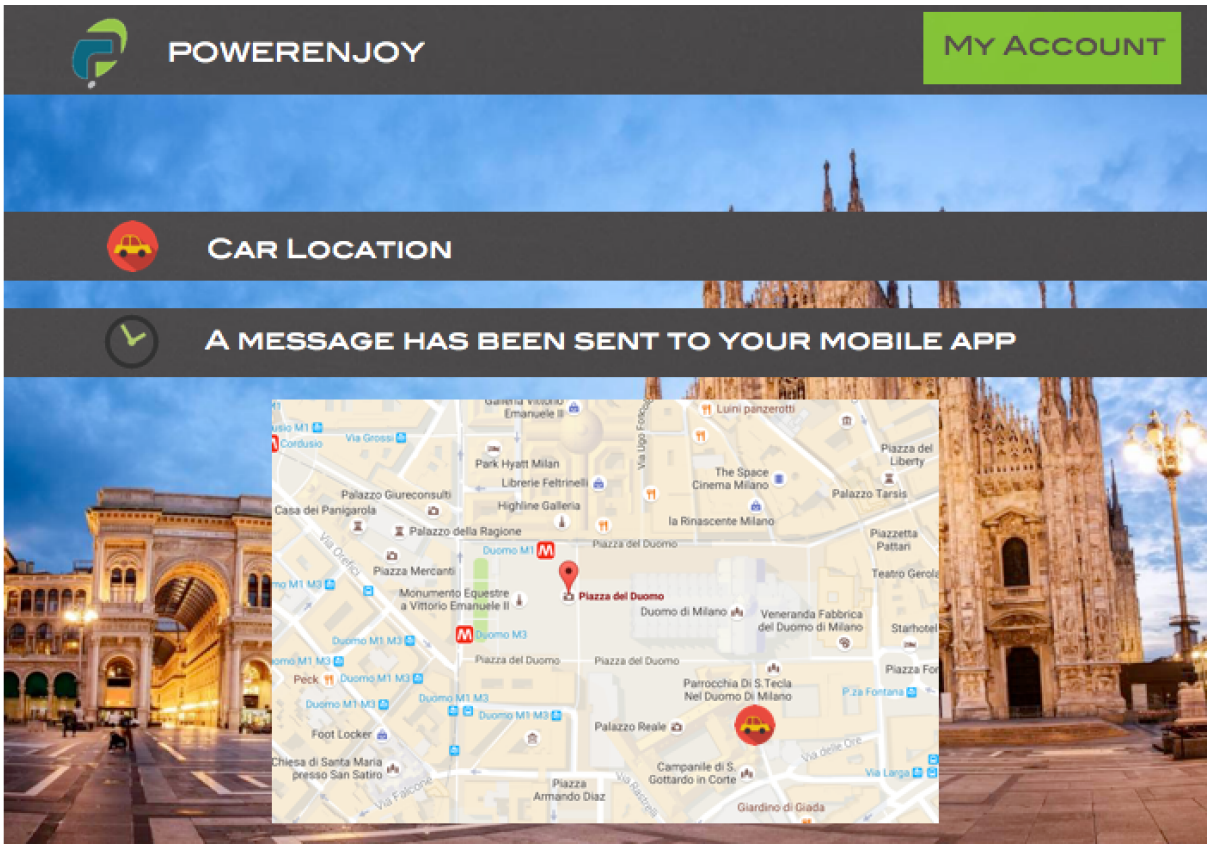
**desktop**


POWERENJOY
LOGIN
THE FUTURE OF CAR SHARING
JOIN NOW


POWERENJOY
JOIN NOW
USERNAME
PASSWORD
LOGIN


POWERENJOY
MY ACCOUNT
SET A CUSTOM LOCATION
RESERVE A CAR
Park Hyatt Milan
Librerie Feltrinelli
Highline Galleria
Palazzo Giureconsulti
Casa degli Panigarola
Piazza Mercanti
Monumento Equestre a Vittorio Emanuele II
Piazza del Duomo
Duomo di Milano
Veneranda Fabbrica del Duomo di Milano
The Space Cinema Milano
Palazzo Tarsis
Piazzetta Pattari
Parrocchia Di S.Tecla Nel Duomo Di Milano
Palazzo Reale
Foot Locker
Chiesa di Santa Maria presso San Satiro
Campanile di S. Gottardo in Corte
Giardino di Giada


POWERENJOY
HOME
USERNAME: BOB
PASSWORD: **********
EDIT
ACCOUNT BALANCE: 12.18€
RECHARGE




**Operator interface:**

Operators have the same interface as the customers but they can unlock cars without passing through a reservation.

## 3.3 Scenario Identifying:

Here we describe some situations of use of the application.

### Scenario 0: account registration

Alberto finds out that in his city there is a new car sharing service, that has nice electric cars available for everyone that has a driving license and wants to use one of these for going to work, appointments, even in places located inside limited traffic areas.

So, he pulls out his phone, downloads the PowerEnJoy app and clicks on the registration button.

He is asked for general information about himself, his SSN and his driving license information, that will be checked by the system. He is then asked to create a username and a password, that can be used to access the application. He doesn't have already an account so this process ends with a confirmation notification.

Alberto will now be able to access the application from his smartphone or his computer and start using the PowerEnJoy service.

### Scenario 1: account management

Alberto wants to change the information saved on the application about himself because they are no longer correct. He clicks on the account management button and starts correcting them. After he's done he clicks the save button and a confirmation message appears.

He also wants to start using the service, so he needs to charge his account's balance with some money. Alberto reaches the account management section of the application, where an interface recalling online payment methods is displayed. He then selects a payment method and inserts the information needed to complete the payment. He then sees a confirmation screen that shows him the new balance.

He is now ready to fully take advantage of the car sharing service.

### Scenario 2: search and reservation of a car

Alberto needs to go to the university and he wants to use the car sharing service because he thinks it's convenient and really nice. So, he logs into the application and starts the search of a car in the reservation page of the application.

He inserts its current location through a specific button and a map shows the nearest cars available, he chooses the one parked in the nearest special parking area and makes the reservation. The system checks Alberto's account balance and his information, everything is alright, so a confirmation screen appears, as well as a notification saying that he has one hour to pick up the car and that he will need to use the app to unlock the car. The car disappears from the available cars in that zone so nobody else can reserve it.

Alberto can now reach the car and drive to his university.

### Scenario 3: no cars available

Alberto is at home and needs to go to a place but after logging in, putting his location in the search screen, and starting the search of the nearby cars cannot find any car available. The application asks if he wants to be notified when a car becomes available, he has some more time to wait so he accepts. After 10 minutes a car has been parked near his home, so the system notifies Alberto and he immediately reserves the car for himself. He then runs to the car because he doesn't want to be late.

### Scenario 4: missed reservation

Manuel reserved a car at 12:00 because he has to attend a lecture at 13:30, but after lunch he fell asleep on the couch and woke up at 13:10, the system charged him for 1€ at 13:00 because he didn't pick up the car. He realizes he can make another reservation and still reach the university without making any accidents.

### Scenario 5: complete ride with no errors

Alberto reserved a car for himself at 11:00 and he is just by the car he reserved at 11:10.

He remembers that the application told him to open the app to unlock the car, so he pulls out his phone and opens the app. He logs in, goes to the unlock page and he sees that the carʾs license plate is needed, so he looks at the car, writes the letters and numbers in the specific text field and pushes the unlock button. Since the system detects that Alberto is near the car, through his phoneʾs gps sensor, the car is unlocked.

Alberto sees the car lights turning on and enters the car. He then starts the ride and reaches his destination. He doesnʾt have much time so he parks the car in the first free parking zone he finds. He gets out of the car, closes the door and runs to its lecture.

### Scenario 6: ride with good behavior

Alberto reserved a car, he wants to pick up two friends to go to the theatre. They meet where the car is parked and they ride all together to the theatre. Alberto parks the car and looks at the last ride checkout on the app. He notices that a discount has been applied to his ride for the number of passengers and the percentage of the battery charge left on the car. He then thinks that he can benefit from the service with the lowest cost possible so that he can drive around more often with the PowerEnJoy cars without hitting too hard on his wallet.

### Scenario 7: ride with bad behavior with dispatcher action

Tommaso, who is a bad guy, at the end of a ride parks the car he previously reserved in an unsafe area. More precisely in a payment parking zone, because he thinks that the car is not his personal car and that the PowerEnJoy company can pay for the parking time. After he finds out that the system charged him for the bad behavior he yells.

Fortunately, the dispatcher has been notified of the unsafe park and sends an operator to move the car to a safe area before a police officer notices that the car shouldnʾt be parked there without paying. In that case Tommaso would have received the letter containing the ticket wrote by the officer, and would have yelled more.

The operator moves the car to a special parking area and while he is there he plugs all the cars in the electric plugs.

### Scenario 8: ride completion with debts

Manuel is a forgetful guy and he has been riding the car rented with the PowerEnJoy app for too long. The ride cost exceeds Manuelʾs balance on his account. So, when he decides to park the car, the system creates a debt for him. After Manuelʾs appointment is over he wants to go back home with another rental car because he really enjoys the service. He notices he cannot reserve any car because the app is telling him that he needs to repay the debt. So, he charges his account with the right amount of money, plus the minimum money needed in order to reserve any car.

He then can enjoy the ride back home, but he will remember not to dwell in travel anymore.

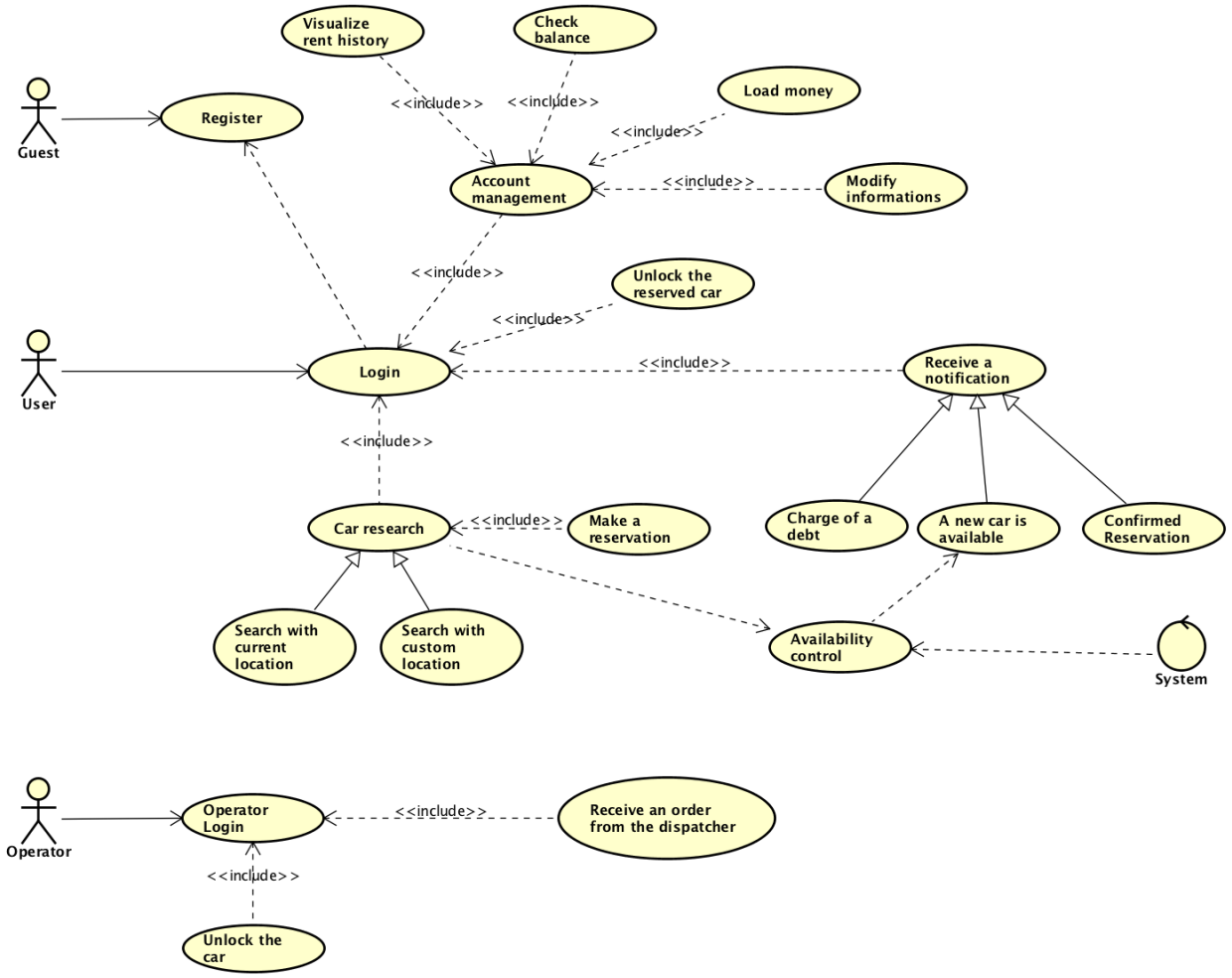
### Scenario 9: try to make a reservation without any money on the account

Tommaso wants to drive to a meeting because heʾs really late. He opens the app, reaches the reservation screen and notices that he cannot reserve any car because he doesnʾt have the minimum amount of money requested in order to use the service.

The app suggests him to load some money on his account and gives him a shortcut to the account management page. Tommaso loads the requested money on his account, tries again the reservation procedure and when the confirmation screams appear he is relieved, he can still reach the place of the meeting in time.

## 3.4 UML Models

### 3.4.1 Use Case Diagram



### 3.4.2 Use Case Description

We provide some detailed description of the use cases derived scenarios indicated above. These descriptions want to be as clear as possible to help the stakeholder to understand what we plan to do, although the real structure will be indicated in the Design Document.

Use case: ‘‘Register’’

| | |
|---|---|
| Actors | Unregistered user or ‘‘guest’’ |
| Goal | G1 |
| Entry conditions | The guest is not already registered to the application |
| Flow of events | ● The guest opens the PowerEnJoy website or the mobile application<br>● The guest clicks on the ‘‘Join Now’’ button<br>● The guest enters all the mandatory information about himself, such as: name, surname, SSN, email address, driving license information and so on<br>● The guest confirms all the information and accepts the ‘‘terms and conditions of use’’ of the application<br>● The application saves all information and confirms the registration |
| Exit conditions | The guest becomes a PowerEnJoy user. |
| Exceptions | A user could skip mandatory fields.<br><br>Another user may be registered with the same email or SSN or driving license.<br><br>The user closes the application prematurely. |

Use case: ‘‘Login’’

| | |
|---|---|
| Actors | User |
| Goal | G2 |
| Entry conditions | The user is already registered to the application |
| Flow of events | ● The user opens the PowerEnJoy website or the mobile application<br>● The user enters in the login page of the application or website<br>● The user confirms his information by clicking the login button |
| Exit conditions | The user logs into the system successfully or he leaves the web page or the app. |
| Exceptions | A user inputs incorrect information or a wrong password. The System shows to the user an error message and some options to recover his account.<br><br>The user closes the application prematurely. |

Use case: ‘‘Car research with custom location’’

| | |
|---|---|
| Actors | User |
| Goal | G4 |
| Entry conditions | The user is logged in and clicks on the ‘‘reserve a car’’ button in the menu |
| Flow of events | ● The user clicks on the text field regarding the location for the car research<br>● The user inserts a custom location writing an address and clicking on the suggestion that appears<br>● The user starts the search, confirming the address<br>● The application displays all the available cars near the custom location<br>● The user clicks on the nearest car to check its battery state and to learn its exact location |
| Exit conditions | The user decides to make a reservation or to exit the research page |
| Exceptions | No cars are available. The system asks the user if he wants to wait for a car, if the user accepts the system checks periodically the available cars and notifies the user when some car is available.<br>Another use case describes this scenario.<br>The user closes the application prematurely. |

Use case: ‘‘Car research with current location’’

| | |
|---|---|
| Actors | User |
| Goal | G4 |
| Entry conditions | The user is logged in and clicks on the ‘‘reserve a car’’ button in the menu |
| Flow of events | ● The user clicks on the text field regarding the location for the car research<br>● The user clicks on the ‘‘use current location’’ button<br>● The system detects his location through the gps information retrieved from the user’s device<br>● The user starts the search, confirming the address<br>● The application displays all the available cars near the user’s location<br>● The user clicks on the nearest car to check its battery state and to learn its exact location |
| Exit conditions | The user decides to make a reservation or to exit the research page |
| Exceptions | The location detected by the GPS sensor may be incorrect, the user can correct it by inserting a custom location.<br><br>No cars are available. The system asks the user if he wants to wait for a car, if the user accepts the system checks periodically the available cars and notifies the user when some car is available. Another use case describes this scenario.<br><br>The user closes the application prematurely. |

Use case: ‘‘A new car is available notification’’

| | |
|---|---|
| Actors | User |
| Goal | G3, G4 |
| Entry conditions | The user searched for a car but none was available, so he asked the system to be notified when a car becomes available |
| Flow of events | ● A user parks a car near the location selected for the search from the user who’s waiting for a car.<br>● The system notifies the user that a car is available near him, with the information regarding its location. |
| Exit conditions | The user decides to reserve the car or to ignore the notification. |
| Exceptions | The user might have searched for a car in another location and reserved it by the time that the new car became available. |

Use case: ‘‘Load money’’

| | |
|---|---|
| Actors | User |
| Goal | G3, G6, G7, G9 |
| Entry conditions | The user is already logged into the application and clicks on ‘‘recharge’’ in the account management page. |
| Flow of events | ● The user reaches the online payment page of the website<br>● The system shows all the payment methods available to the user<br>● The user chooses its favorite payment method<br>● The user inserts the information about the chosen method<br>● The user confirms his information by clicking the ‘‘confirm’’ button<br>● The system confirms to the user that the payment has been completed successfully. |
| Exit conditions | The user recharges successfully his account or he leaves the page. |
| Exceptions | The information written by the user in the payment page are wrong.<br><br>The payment transaction is not completed successfully (the user has not enough money or the connection is interrupted).<br><br>The user closes the application prematurely. |

Use case: ‘‘Modify informations’’

| | |
|---|---|
| Actors | User |
| Entry conditions | The user is already logged into the application and clicks on ‘‘modify information’’ in the account management page. |
| Flow of events | ● The user chooses some information that is not correct and wants to update it<br>● The user clicks on the text field of the information he wants to change<br>● The user inserts new data<br>● The user confirms all the updates by clicking on the ‘‘save’’ button<br>● The system saves the update and shows a confirmation message to the user |
| Exit conditions | The user saves the update or cancels the operation |
| Exceptions | The user inputs incorrect information.<br><br>The user deletes information on mandatory fields and does not insert new ones.<br><br>The user closes the application before saving, in this case the operation is canceled and no change is applied. |

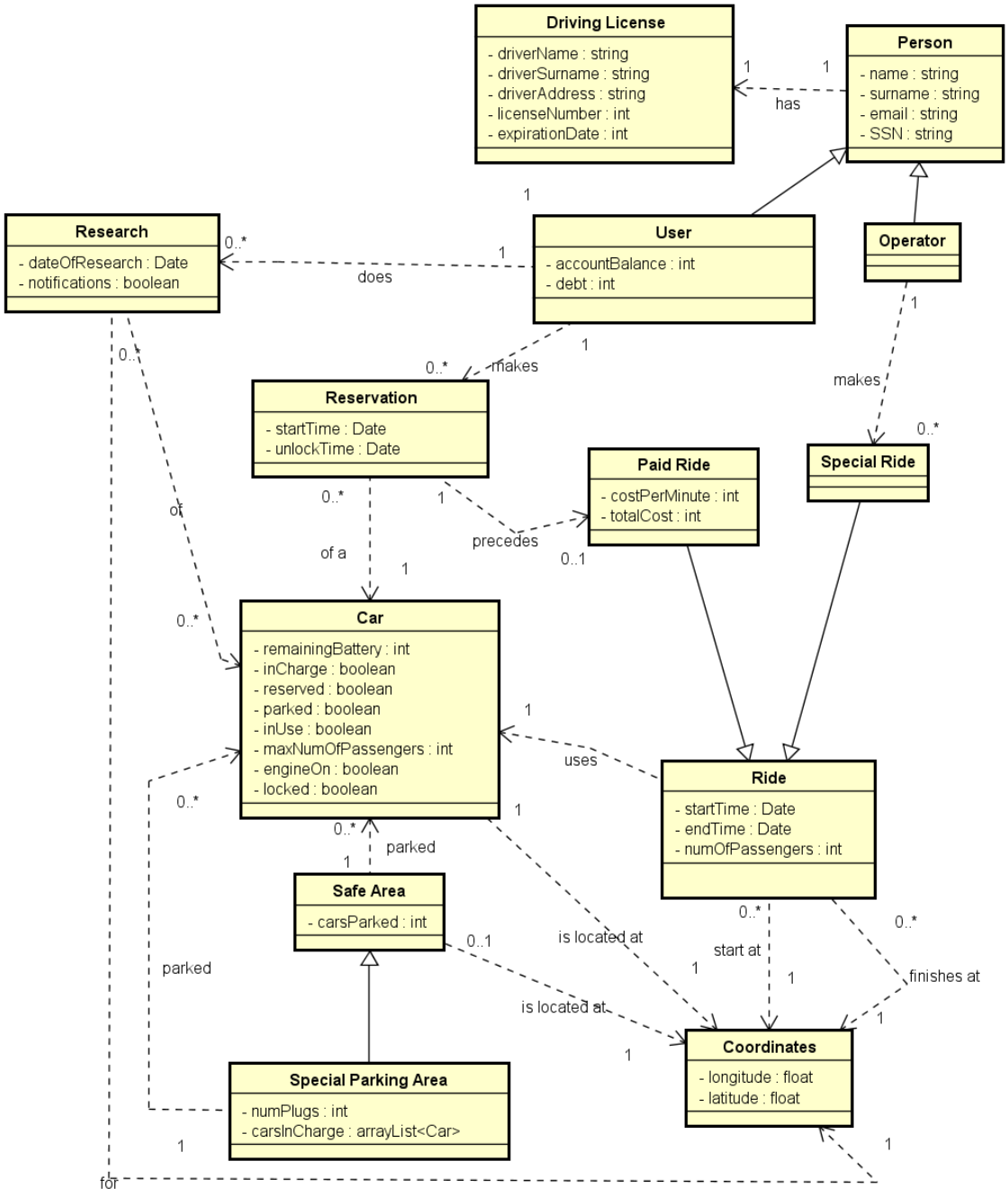
Use case: ‘‘Make a reservation’’

| | |
|---|---|
| Actors | User |
| Goal | G3 |
| Entry conditions | The user is already logged into the application and clicks on ‘‘reserve a car’’ in the menu and he has already inserted his position. |
| Flow of events | ● The user reaches the reservation page of the app/site<br>● The system shows all available cars to the user<br>● The user choose a car<br>● The system notifies the user that the reservation has been done successfully and he has an hour to pick up the car |
| Exit conditions | The user reserves successfully a car or he leaves the page. |
| Exceptions | The user closes the application prematurely.<br><br>By the time the user has completed the operation, the car might be already taken, so the list of cars displayed is updated and the user can choose another one.<br><br>The user has not enough money on his account.<br><br>No cars are available for the selected position. The system asks the user if he wants to wait for a car, if the user accepts the system checks periodically the available cars and notifies the user when some car is available. |

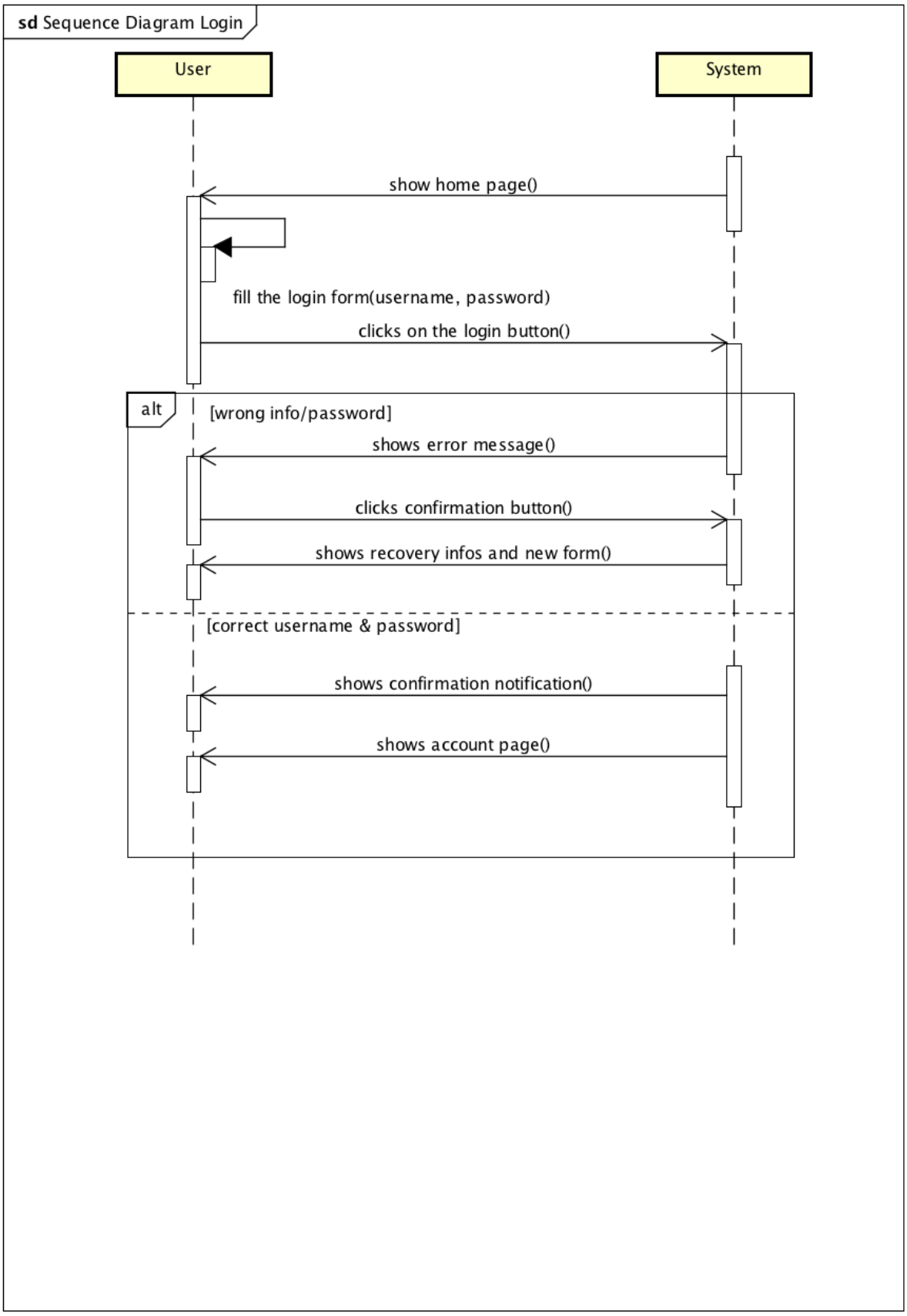
Use case: ‘‘Unlock the car’’

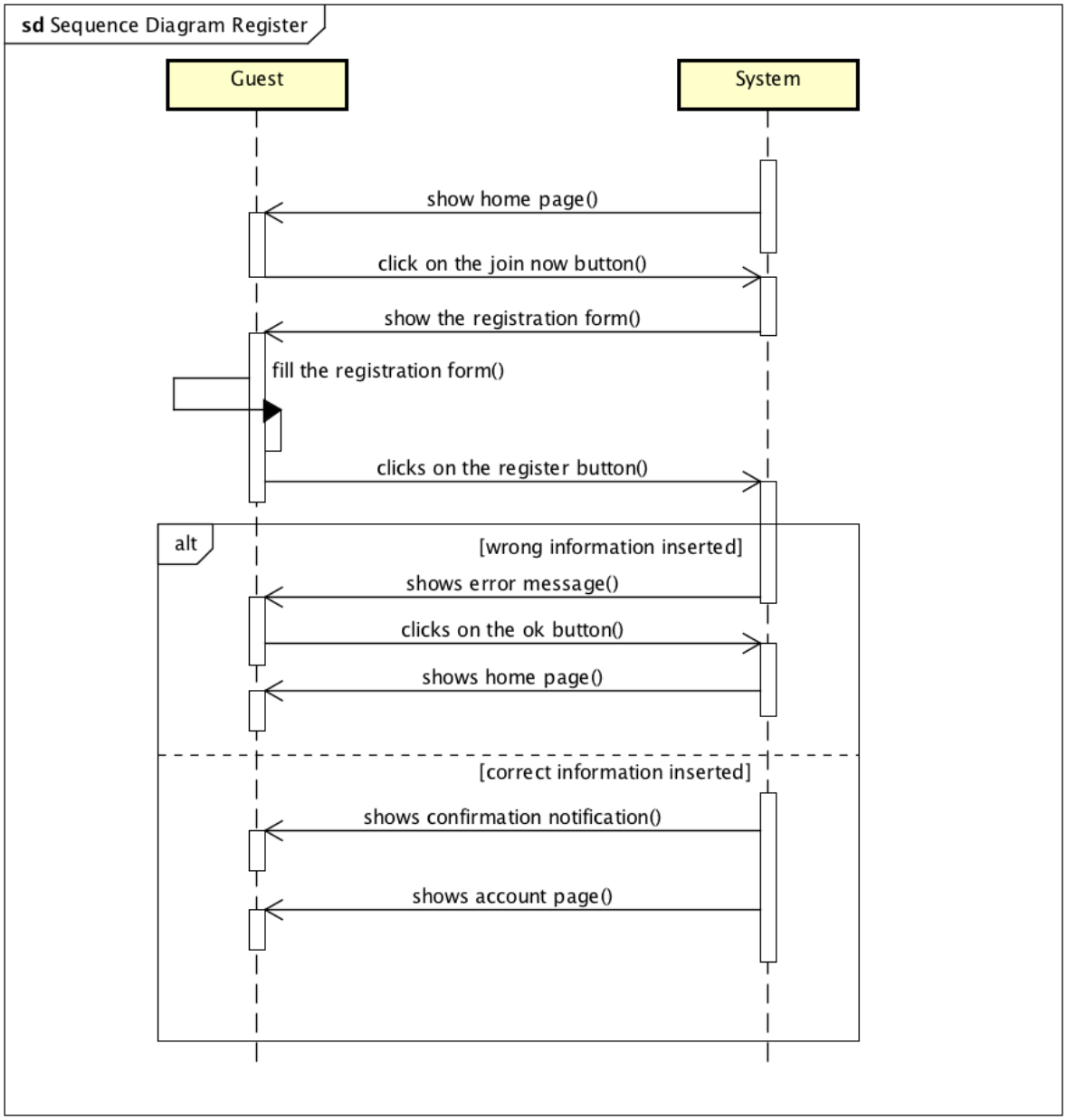
| | |
|---|---|
| Actors | User |
| Goal | G5 |
| Entry conditions | The user has reserved a car, he is already logged into the application and reaches the ‘‘unlock reserved car’’ section. |
| Flow of events | ● The system asks the user to have access to his phone’s gps information<br>● The user accepts<br>● The user notices that he needs to insert the car’s license plate in a text field to open the car<br>● The user looks at the car and inserts the data required<br>● The user clicks on the ‘‘unlock car’’ button of the page<br>● The system determines the distance between the user and the car<br>● The system notices that the user is near the car<br>● The system unlocks the car for the user |
| Exit conditions | The user reaches the car and starts the ride |
| Exceptions | The reservation expired so the user can’t see any car to unlock in the application page.<br><br>The user might not be close enough to the car.<br><br>The user closes the application prematurely. |

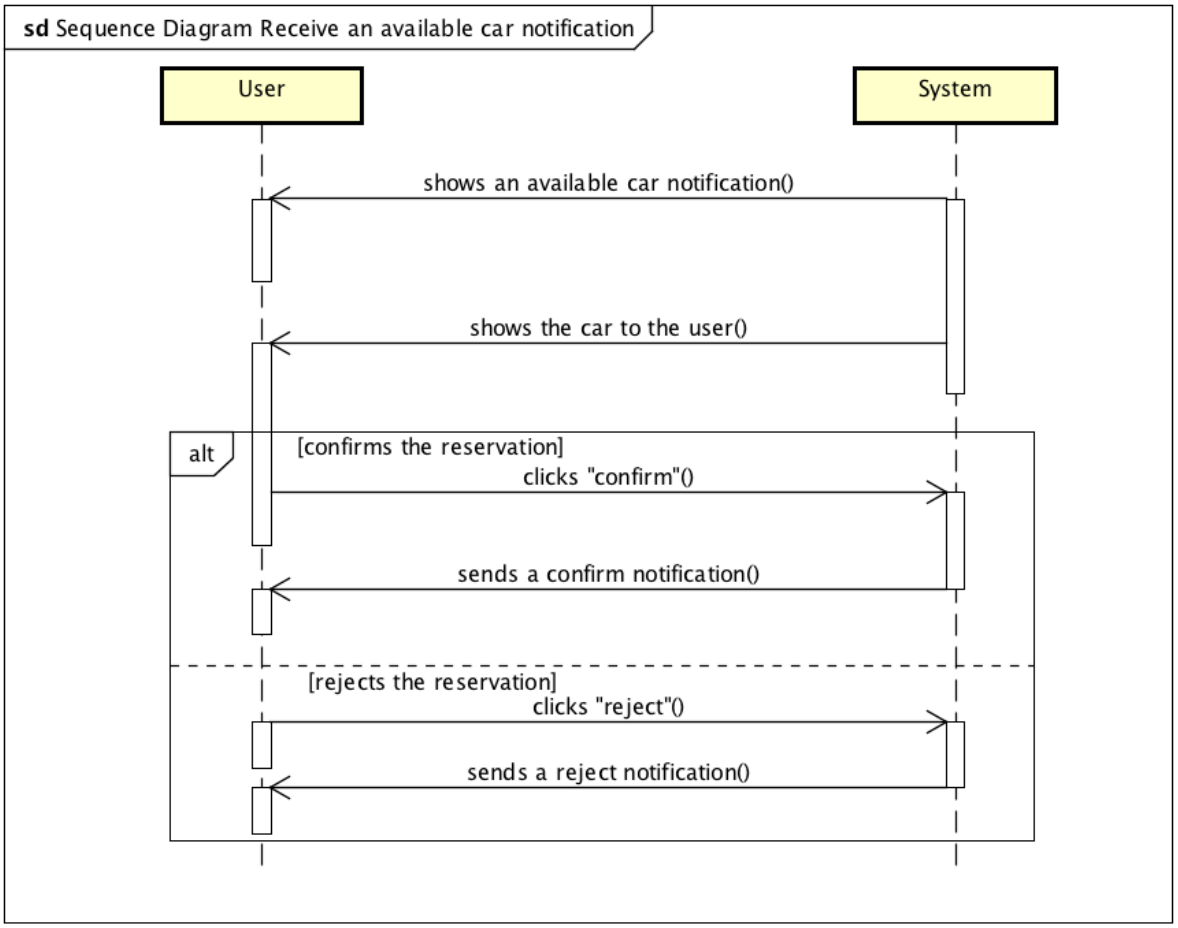
### 3.4.3 Class Diagram

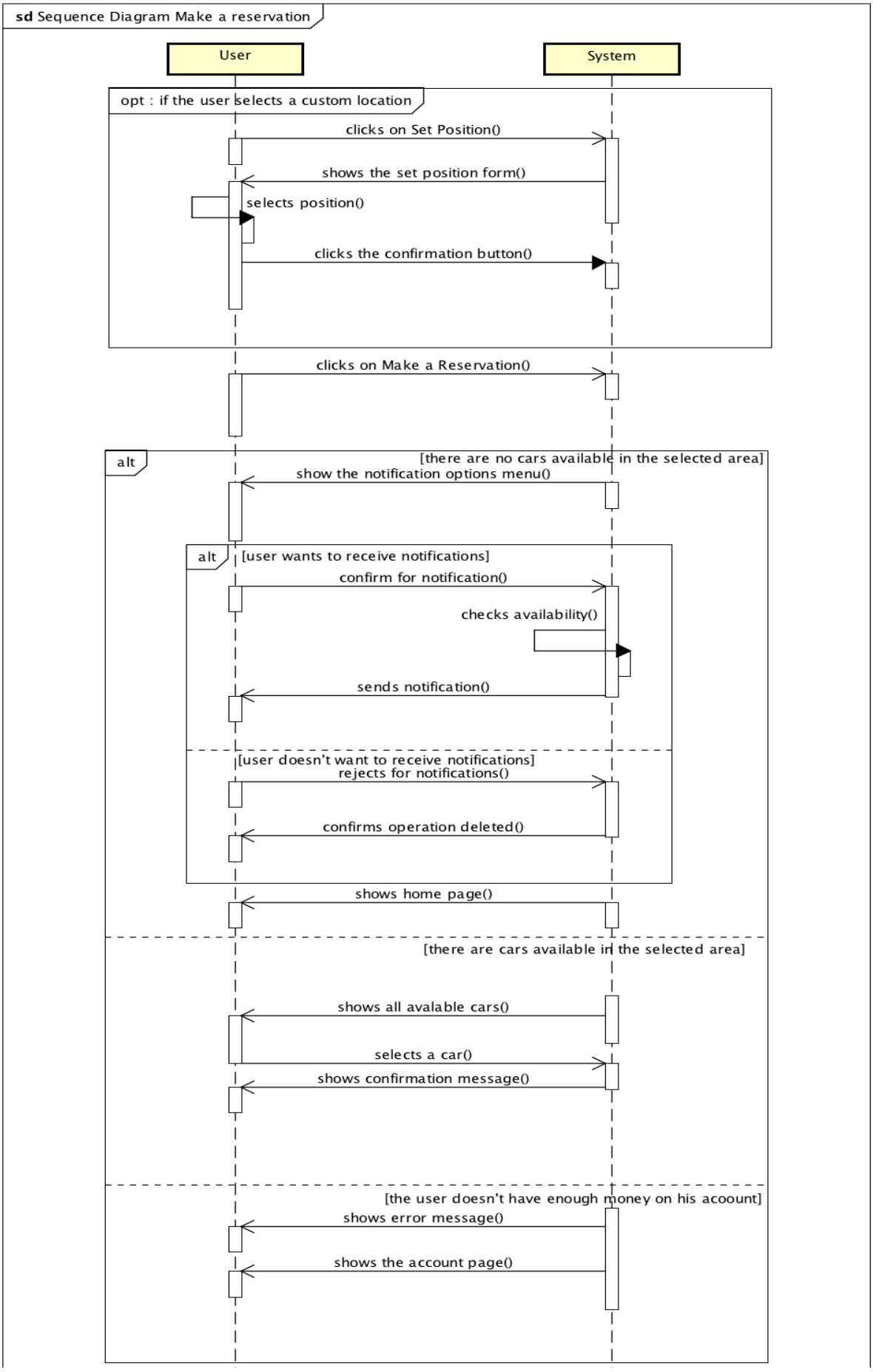


### 3.4.4 Sequence Diagrams

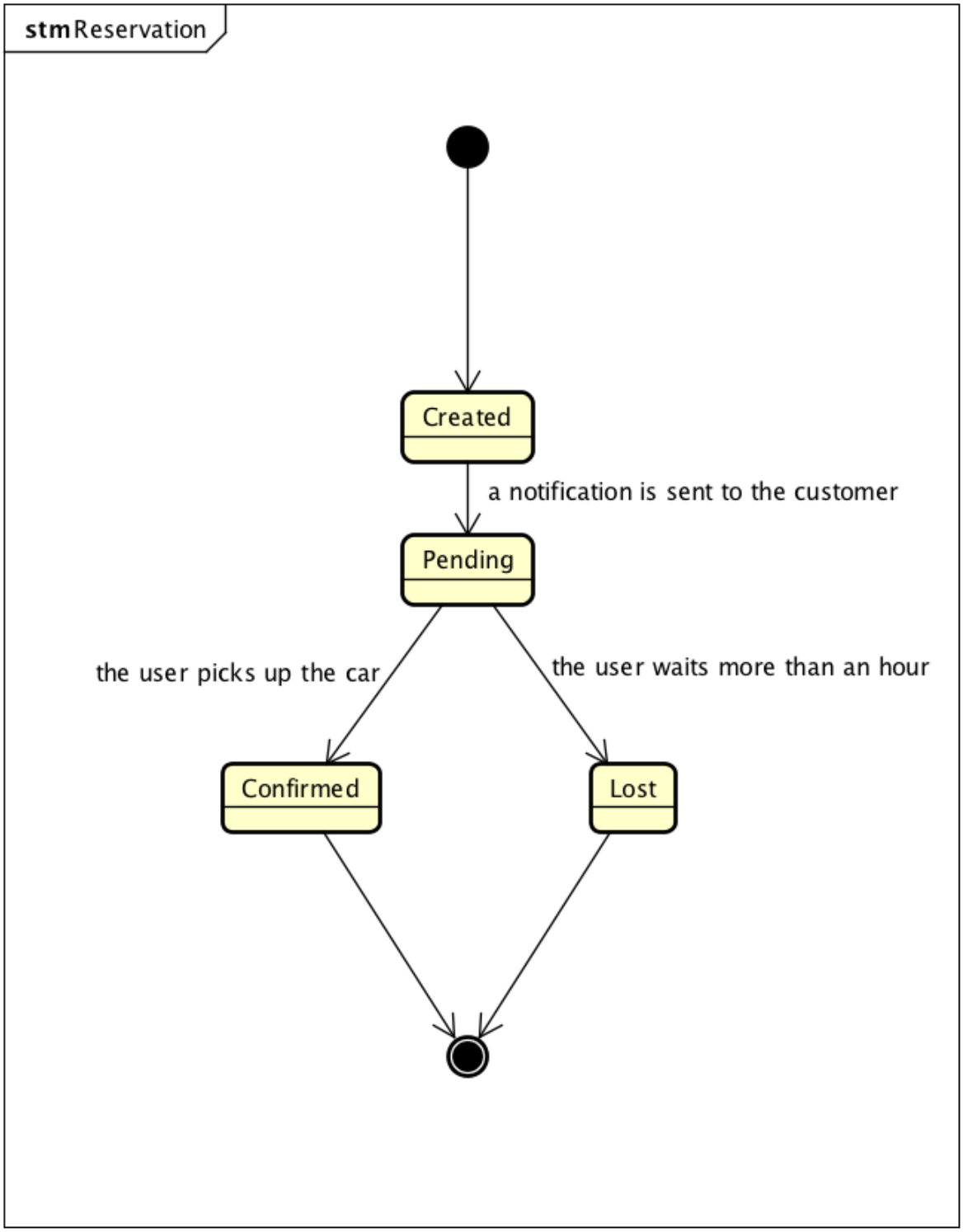


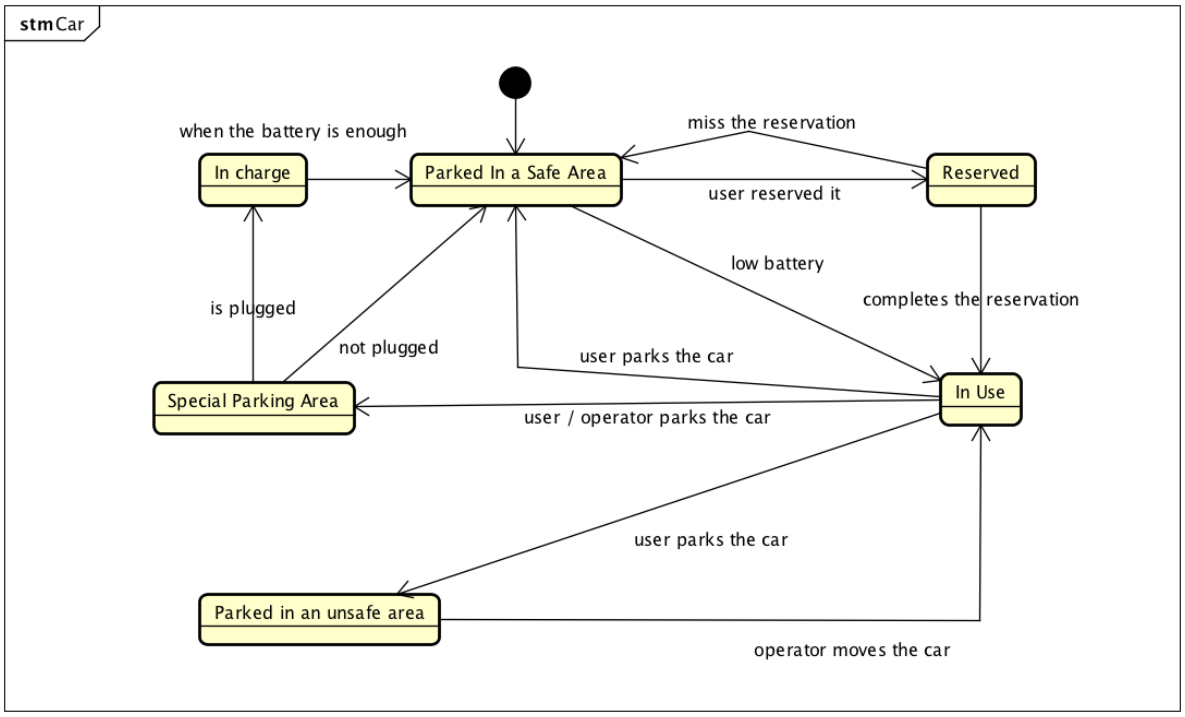

sd Sequence Diagram Register
Guest
System
show home page()
click on the join now button()
show the registration form()
fill the registration form()
clicks on the register button()
alt
[wrong information inserted]
shows error message()
clicks on the ok button()
shows home page()
[correct information inserted]
shows confirmation notification()
shows account page()


sd Sequence Diagram Receive an available car notification
User
System
shows an available car notification()
shows the car to the user()
alt
[confirms the reservation]
clicks "confirm"()
sends a confirm notification()
[rejects the reservation]
clicks "reject"()
sends a reject notification()


sd Sequence Diagram Make a reservation
User
System
opt : if the user selects a custom location
clicks on Set Position()
shows the set position form()
selects position()
clicks the confirmation button()
clicks on Make a Reservation()
alt
[there are no cars available in the selected area]
show the notification options menu()
alt
[user wants to receive notifications]
confirm for notification()
checks availability()
sends notification()
[user doesn't want to receive notifications]
rejects for notifications()
confirms operation deleted()
shows home page()
[there are cars available in the selected area]
shows all avalable cars()
selects a car()
shows confirmation message()
[the user doesn't have enough money on his acoount]
shows error message()
shows the account page()

### 3.4.5 State Chart Diagrams




stmCar
when the battery is enough
miss the reservation
In charge
Parked In a Safe Area
Reserved
user reserved it
low battery
completes the reservation
is plugged
not plugged
user parks the car
Special Parking Area
In Use
user / operator parks the car
user parks the car
Parked in an unsafe area
operator moves the car

# 4 Alloy

**Assumptions for the model analyzed in alloy:**

The world is represented in a possible instant of time, so it doesn't show all the rides that a user has taken, but only the one that's currently in progress. Time is represented by time slots of one hour each.

## 4.1 Alloy Code

```
open util/boolean
open util/integer

//SIGNATURES

sig Coordinates{
	latitude : Int,
	longitude : Int
}

sig SafeArea{
	position: Coordinates,
	parkedCar : set Car
}

sig SpecialParkingArea extends SafeArea{
	numPlugs : Int,
	carsInCharge : set Car
}
{
	numPlugs > 0
}
sig LicensePlate{}
sig Car{
	position : Coordinates,
	maxPassengers: Int,
	licensePlate : LicensePlate,
	inCharge : Bool,
	reserved : Bool,
	Parked : Bool,
	inUse : Bool,
	locked : Bool,
	lowBattery : Bool
}
{
	maxPassengers=4
}
```

```
abstract sig Person{
	drivingLicense: DrivingLicense,
	SSN: Int
}

sig DrivingLicense{
}

sig User extends Person{
	accountBalance: Int,
	debt:Int,
}
{
	accountBalance=0 implies debt>=0
	accountBalance>0 implies debt=0
	accountBalance>=0
}
sig Operator extends Person{
}
{}

abstract sig Ride{
	startTime: Int,
	startPosition: Coordinates,
	car: Car
}
{
	startTime>=0 startTime <=23
}
sig PaidRide extends Ride{
	costPerMinute: Int,
	totalCost: Int,
	reservation: Reservation,
	passengers: Int
}
{
	passengers>=0 passengers<=3
	totalCost>=costPerMinute
	costPerMinute=1
}
sig SpecialRide extends Ride{
	operator: Operator
}
sig Reservation{
	startTime: Int,
	unlockTime: lone Int,
	user: User,
	car: Car
}
{
	unlockTime>=startTime
	unlockTime<=startTime+1
	startTime>=0 startTime<=22
}
```

```
//FACTS

fact OperatorsCannotHaveTheSameSSNOrDrivingLicense{
        //each operator as an unique SSN and Driving License
        no o1,o2 : Operator| o1.SSN=o2.SSN and o1!=o2
        no o1, o2 : Operator | o1.drivingLicense = o2.drivingLicense and o1 != o2
}
fact No2SafeAreasInOneLocation{
        //two safe areas cannot have the same coordinates
        no s1, s2 : SafeArea | s1.position = s2.position and s1 != s2
}
fact PlugNumberConstraint{
        //the number of cars parked in the same special parking
        //areas are equal or less the number of plugs
        all s1: SpecialParkingArea | #(s1.carsInCharge) <= s1.numPlugs
}
fact LicensePlateBelongsToACar{
        all l1 : LicensePlate | one c1: Car | l1=c1.licensePlate
        //each license plate belongs to a car
}
fact UniqueLicensePlatesForCars {
        //every car has a unique license plate to identify it
        no disj c1,c2 : Car | c1.licensePlate = c2.licensePlate
}
fact CarsInUseIfTheyAreInARide{
        //the attribute InUse of the car is true only if they are present
        //in a ride when we analyze the system
        all c1: Car | one r1: Ride | c1.inUse.isTrue implies (r1.car = c1)
        all r1: Ride, c1: Car | (r1.car = c1) implies c1.inUse.isTrue
}

fact ParkedCarsCannotBeInUse{
        //the cars that are parked cannot be indicated as in use
        all c1 : Car | c1.Parked.isTrue implies c1.inUse.isFalse
}

fact OnlyParkedCarsAreLocked{
        //only the cars that are parked are Locked
        all c1 : Car | c1.Parked.isTrue implies c1.locked.isTrue
        all c1 : Car | c1.inUse.isTrue implies c1.locked.isFalse
}

fact CarsAlwaysHaveACurrentState{
        //every car has one boolean assigned as true, this ensure consistency
        all c1 : Car | c1.inCharge.isTrue or c1.Parked.isTrue or c1.inUse.isTrue
}

fact CarsInChargeAreParked{
        //all the cars that are in charge are signed as parked cars
        all c1 : Car | c1.inCharge.isTrue implies c1.Parked.isTrue
}
```

```
fact CarsInUseAreNotInChargeNorParked{
	//if a car is in use it cannot be neither in charge nor parked
	all c1 : Car | c1.inUse.isTrue implies (c1.inCharge.isFalse and c1.Parked.isFalse)
}

fact ACarIsInSafeAreaIfItsPositionIsCorrect{
	//a car is parked in a safe area if and only if it's coordinates are the same as the
safearea ones
	all c1 : Car, s1 : SafeArea | c1 in s1.parkedCar iff (c1.position = s1.position and
c1.Parked.isTrue)
}

fact CarsAreInOnlyOneArea{
	//each car cannot be in multiple safeareas simultaneously
	no disj s1,s2 : SafeArea | #(s1.parkedCar & s2.parkedCar) > 0
	no disj s1,s2 : SpecialParkingArea | #(s1.carsInCharge & s2.carsInCharge) > 0
}

fact CarsInChargeAreParkedInSpecialParkingAreas{
	//the cars that are in charge are exclusively parked in special parking areas
	all c1: Car | one s1: SpecialParkingArea | c1.inCharge.isTrue implies (c1 in
s1.carsInCharge and c1 in s1.parkedCar)
	all c1 : Car | one s1: SpecialParkingArea | c1 in s1.carsInCharge implies
c1.inCharge.isTrue
}

fact OperatorsCannotUseReservedCars{
	//an operator cannot use a reserved car
	all c1: Car | no sr: SpecialRide | c1.reserved.isTrue and sr.car = c1
}

fact DrivingLicensesBelongToSomeone{
	//every driving license as an owner
	all d1 : DrivingLicense | one p1: Person | p1.drivingLicense = d1
}

fact UsersCannotHaveTheSameSSNOrDrivingLicense{
	//each user as an unique SSN and Driving License
	no u1,u2 : User| u1.SSN=u2.SSN and u1!=u2
	no u1, u2 : User | u1.drivingLicense = u2.drivingLicense and u1 != u2
}

fact NoRidesCanBeDoneWithTheSameCarInTheSameMoment{
	//each ride has an unique car
	no disj r1, r2 : Ride | r1.car = r2.car
}

fact AllPaidRidesStartInASafeArea{
	//all paid rides starts in a safe area, because a reservation can be done
	//only if a car is in a safe area
	all r1 : PaidRide | one s1 : SafeArea | r1.startPosition=s1.position
}
```

```
fact NoRidesCanBeDoneInTheSameMomentByTheSameUser{
	//a user can do only one ride at a time
	no disj r1, r2 : PaidRide | r1.reservation.user = r2.reservation.user
}

fact NoRidesCanBeDoneInTheSameMomentByTheSameOperator{
	//an operator can do only one ride at a time
	no disj r1, r2 : SpecialRide | r1.operator = r2.operator
}

fact NoReservationsCanBeDoneInTheSameMomentByTheSameUser{
	//a user can do only one reservation at a time
	no disj r1, r2 : Reservation | r1.user = r2.user
}

fact UsersCannotBeOperatorsAtTheSameTime{
	//an operator can be a user, but it needs a separate account
	//with this fact we ensure that someone cannot drive a car as both at the same time
	all u: User, o:Operator | no pr: PaidRide, sr: SpecialRide | u.SSN = o.SSN and
pr.reservation.user = u and sr.operator = o
}

fact UnlockTimeOfReservationIsTheStartTimeOfThePaidRide{
	//the moment when the car is unlocked corrisponds with
	//the moment in which the ride starts
	all p: PaidRide | p.startTime = p.reservation.unlockTime
}

fact ReservationsAndPaidRidesHaveTheSameCar{
	//reservation and car have the same car as attribute
	all r1: Reservation,  p1: PaidRide | p1.reservation = r1 iff p1.car = r1.car
}

fact ACarHasAReservationIfItIsReserved{
	//the boolean attribute in car that says the car is reserved it's true
	//if and only if the car is inside a reservation
	all c1: Car | one r1:Reservation | c1.reserved.isTrue implies r1.car = c1
	all r1: Reservation | r1.car.reserved.isTrue
}

fact NoReservationCanBeDoneByAUserWithLowBalance{
	//a user that has low balance cannot make a reservation
	//this is done to prevent that users make debits
	all r1 : Reservation | r1.user.accountBalance>=2
}

fact NoReservationCanBeDoneByAUserWithADebt{
	//the users with a debt cannot make any reservation
	no r1 : Reservation | r1.user.debt > 0
}

fact NoReservationCanHaveTheSameRide{
	//each ride corrispond to an unique reservation
	no disj p1,p2 : PaidRide | p1.reservation=p2.reservation}
```

```
fact NoReservationCanHaveTheSameCar{
	//each car corrispond to an unique reservation
	no disj r1,r2 : Reservation | r1.car = r2.car
}

fact NoReservationCanBeDoneOnACarParkedInAnUnsafeArea{
		//all reservations are done only for cars parked in safe areas
		no r: Reservation | r.car.Parked.isTrue and not r.car in SafeArea.parkedCar
}

fact NoReservationCanBeDoneOnACarWithLowBattery{
	//a car that has low battery cannot be reserved
	all c: Car | c.lowBattery.isTrue and c.inUse.isFalse implies c.reserved.isFalse
}

//PREDICATES

pred ActiveReservation{
	some r1 : Reservation | no p1 : PaidRide | p1.reservation=r1
}

run ActiveReservation

pred CarsCanBeInUnsafeAreas{
	some c1: Car | c1.Parked.isTrue and not c1 in SafeArea.parkedCar
}

run CarsCanBeInUnsafeAreas


pred CarsInUseCorrispondsToReserveredCars{
	some c1: Car | c1.inUse.isTrue
	some r1: Ride | r1.car.inUse.isTrue
	some r: Reservation | no p1 : PaidRide | p1.reservation=r
}
run CarsInUseCorrispondsToReserveredCars

//ASSERTS

assert NoReservedCarIsInAnUnsafeArea{
	//ensures that every reserved car is in a safe
	no c1: Car | all s1: SafeArea | c1.reserved.isTrue and c1.Parked.isTrue and
c1.position!=s1.position
}
assert NoUserWithLowBalanceHasAReservation{
	//ensure that only users with wnough money can do reservations
	no r1: Reservation | r1.user.accountBalance<2
}
assert NoUserCanDoMultipleReservations{
	//enusures each user can do only one reservation at a time
	no disj r1,r2 : Reservation | r1.user=r2.user
}
```

```
assert NoCarIsPickedUpInAnUnsafeAreaForAPaidRide{
        //ensures that all the cars picked up for the paid rides were in safe areas
        no r1: PaidRide | all s1: SafeArea | r1.startPosition!=s1.position
}
assert NoCarInChargeIsNotInASpecialParkingArea{
        //ensures that all the cars that are in charge are in special parking areas
        no c1: Car | all s1: SpecialParkingArea | c1.inCharge.isTrue and c1.Parked.isTrue
and c1.position!=s1.position
}

check NoReservedCarIsInAnUnsafeArea
check NoUserWithLowBalanceHasAReservation
check NoUserCanDoMultipleReservations
check NoCarIsPickedUpInAnUnsafeAreaForAPaidRide
check NoCarInChargeIsNotInASpecialParkingArea

pred show(){}

run show
```

### 4.1.1 Results

9 commands were executed. The results are:

#1: **Instance found.** ActiveReservation is consistent.

#2: **Instance found.** CarsCanBeInUnsafeAreas is consistent.

#3: **Instance found.** CarsInUseCorrispondsToReserveredCars is consistent.

#4: No counterexample found. NoReservedCarIsInAnUnsafeArea may be valid.

#5: No counterexample found. NoUserWithLowBalanceHasAReservation may be valid.

#6: No counterexample found. NoUserCanDoMultipleReservations may be valid.

#7: No counterexample found. NoCarIsPickedUpInAnUnsafeAreaForAPaidRide may be valid.

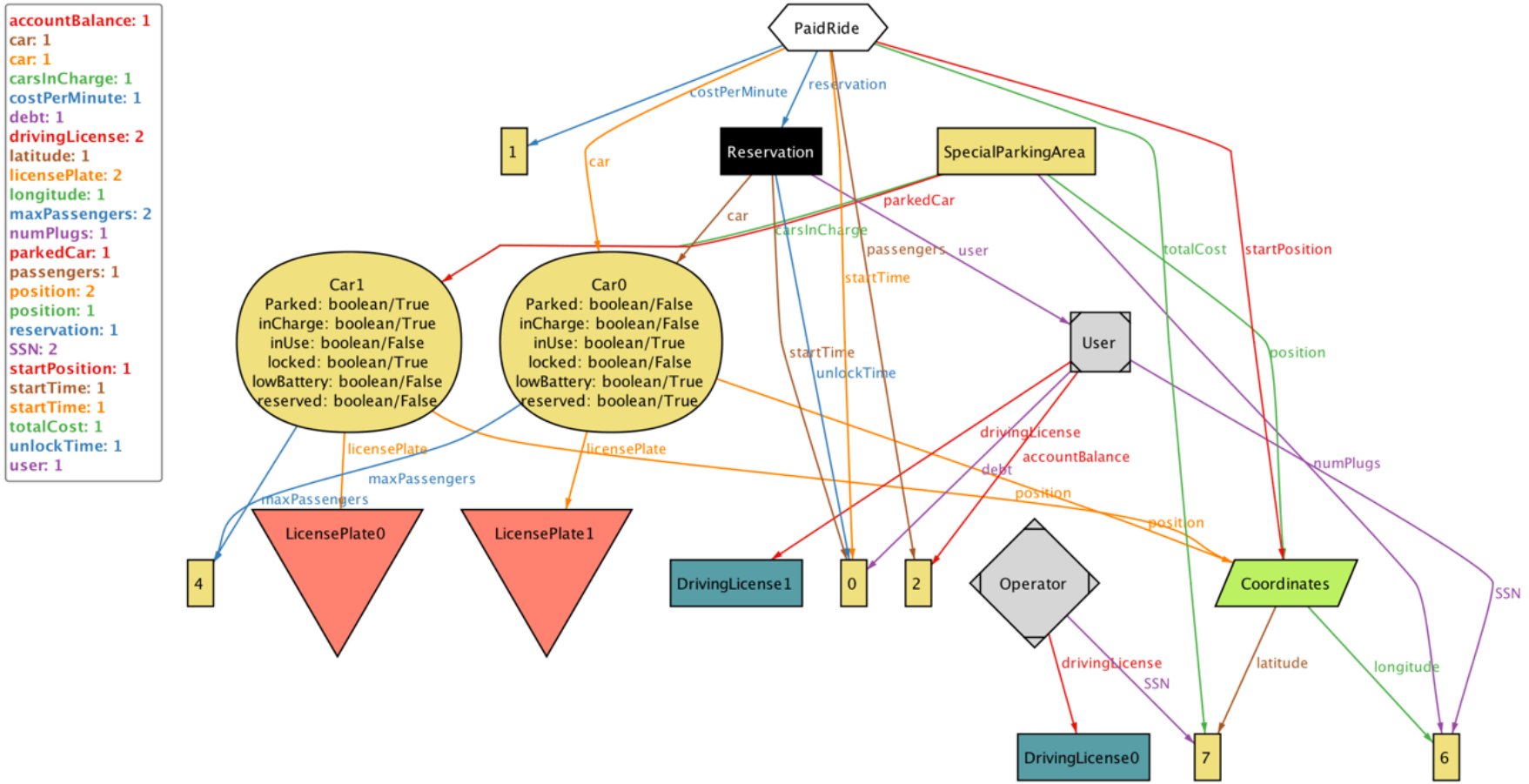
#8: No counterexample found. NoCarInChargeIsNotInASpecialParkingArea may be valid.

#9: **Instance found.** show is consistent.

## 4.2 Alloy Worlds

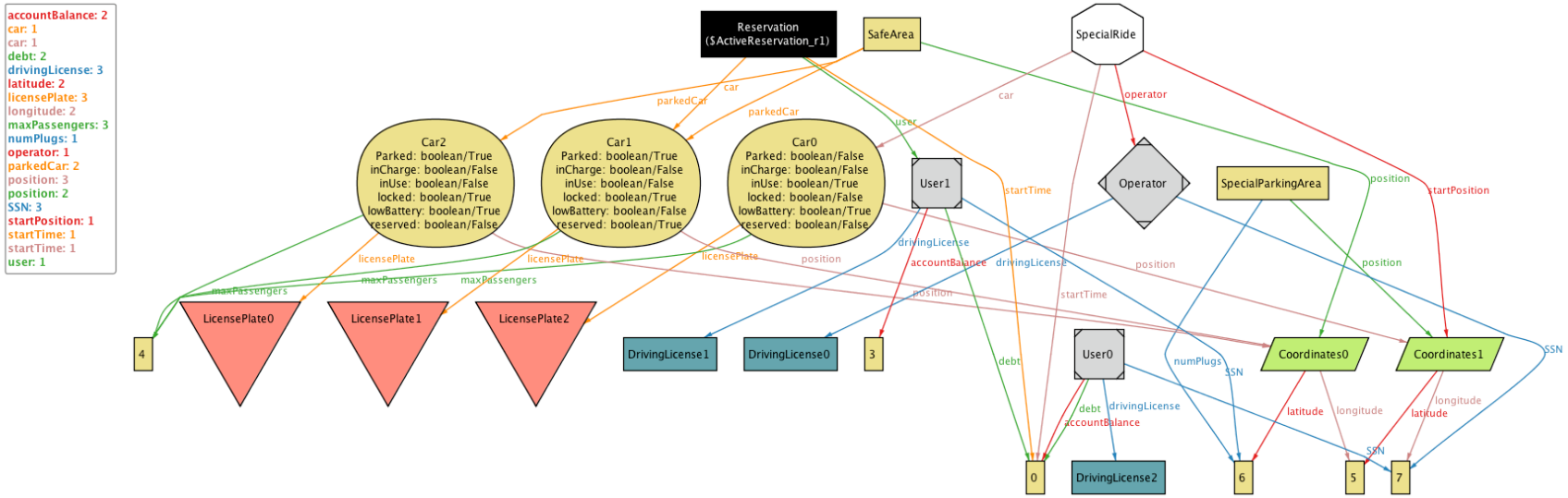
### 4.2.1 General World

The following world is a general world generated with the analyzer. It has been generated using the predicate show.



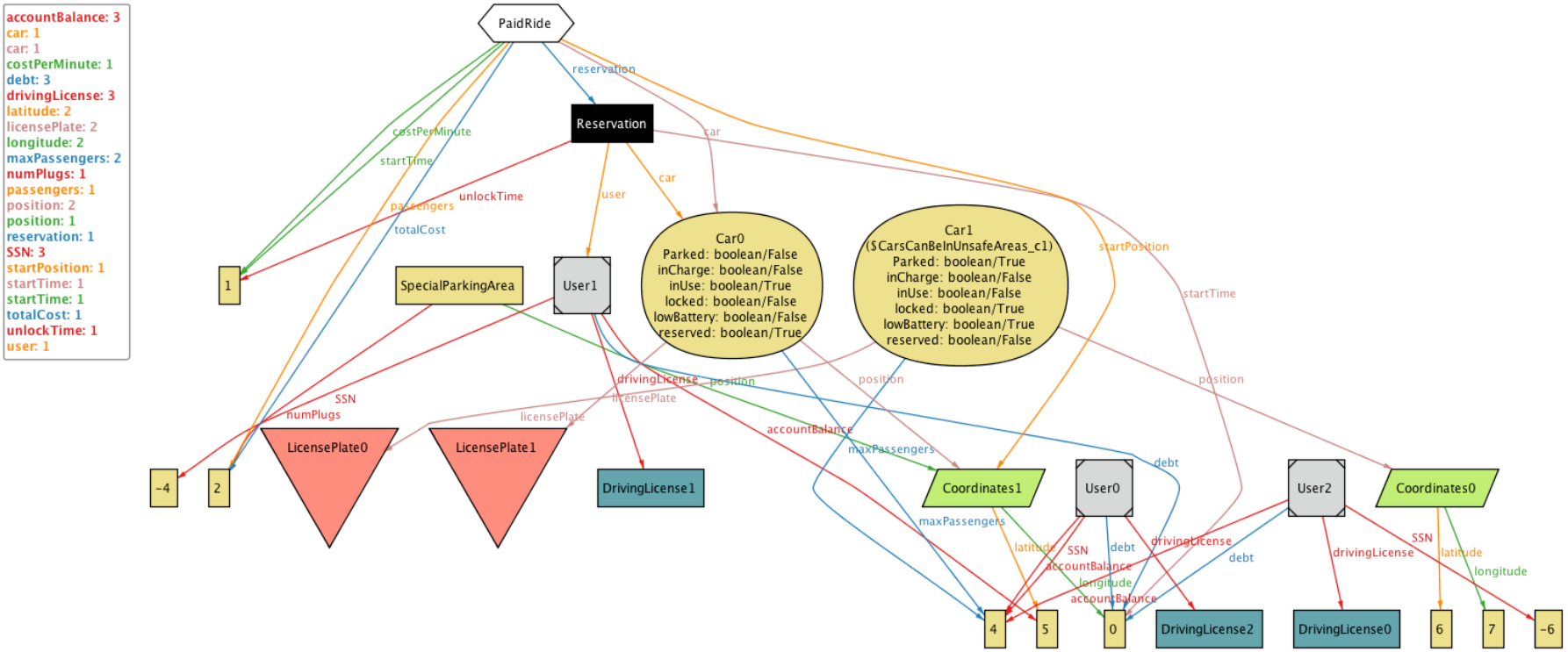
### 4.2.2 Active Reservation

The following world is a world in which there is a reservation without a ride. It has been generated using the predicate activeReservation



### 4.2.3 Cars parked in unsafe areas

The following world is a world in which some cars have been parked in unsafe areas by users with bad behavior. It has been generated by the predicate CarsCanBeInUnsafeAreas.



# 5 Additional Information

## 5.1 Hours of work

Time spent to write this document:

Manuel Parenti : 30 hours

Alberto Zeni : 30 hours

## 5.2 Used Tools

The tools that supported us are:

MS Office Word 2016: to write and redact this document;

Astah: to create UML diagrams;

Alloy Analyzer 4.2: to prove the consistency of our model;

Pencil: to create mockups of the interface.