Politecnico di Milano

Industrial and Information Engineering
Computer Science and Engineering

Software Engineering II Assignment
PowerEnJoy - car sharing

Academic Year: 2016/2017

Prof. Elisabetta Di Nitto

Alice Segato matr. 875045
Mark Edward Ferrer matr. 876650
Davide Bonacina matr. 876199

PowerEnJoy

Car Sharing App

TABLE OF CONTENT

LIST OF FIGURES

1 INTRODUCTION

1.1 Description of the given problem

According to the assignment document the problem consists in the definition and implementation of a system that manages the activity of an electric car sharing. The activity itself doesn't need to be explained because car sharing is a largely spread out costume.
More in depth, the system have to allow people to join a community of car sharers by registering into the system and after this, they can share cars with neighbors and friends in order to save money together and reduce pollution made by public means of transport and private cars.

1.2 Actual system

Actually the car sharing works without a computer system and there are lots of employees dislocated in the safe areas that provide car keys when users want to book cars.
The only system available in the company is a system that manages all the bureaucracy such as car revisions, car maintenance, fine delivery and car recovery.
This system stores all the data about the cars in a MySQL database and tracks continuously all the cars to provide assistance by telephone in case of failure or user request.

1.3 Goals

In this section of the document, we are presenting the main goals that the software-to-develop must fulfill in order to satisfy the requirements:

- Users must have the possibility to register to the system to become part of the community. They must provide their credentials (such as first name, last name email address and so on) and a valid method of payment that can be an IBAN code or a credit card code. After the online registration, users will receive a confirmation email that includes a password for the access to the system and a link that will conclude the registration task and will show the user his personal area from where he/she can decide to do stuff.

- The user can select the parking lot from a subset of them where to pick the car. The user's position or the input of an address will be the base on which the system will show a subset of parking lots: the system will choose the parking lots with at least one available car in a range of 1 km from the given position.

- The user will be able to reserve a car for up to one hour from the pickup, from the list of the available ones in the selected parking lot. Then the system will lock automatically the selected car to everyone

until the reservation time, with except to the user that will use his mobile device to identify himself and let the system know that he's picking up the car.

- The system must control that every reserved car will be picked in the time range: if this condition is not satisfied, it will assign an extra fee of 1€ to the owner of the unpicked car and then it will set the car as available in order to allow other people to pick that car.

- The system must charge the user by a certain amount of Euros per minute. The time outside the predefined areas will contribute to calculate the final amount for the user to pay and a built-in-car screen will show this information to the user/driver. This amount can be decreased or increased by:
  - The number of passengers of the car (a sensor will count the passengers in order to apply a 10% discount on the final amount);
  - The battery charge remaining (if the battery has more than 50% of the charge, the system must apply a 20% discount);
  - If the car is parked in a charging station and the user plugs the car BEFORE he/she registers the exit, a 30% discount will be applied;
  - If the car has less than 20% of the battery charge or the user leaves it in an unsafe area further that 3 Km from a charging station, a 30% extra free will be applied (the car goes in recovery mode);

- The system must provide the user the possibility to select the money saving mode: with this option activated, the user has to input the final destination and the system will tell the user the best charging station where to leave the car in order to receive a discount. The system chooses the right charging station based on the position of the other cars and on the availability of power plugs to provide a balanced number of vehicles on each zone of the city.

- To close the bill, the user must park the car in one of the predefined parking areas. Then the system will detract the correct amount of money from user's payment method. The car will then set as available again when the battery charge reaches 100%.

1.4 Domain properties

Since the world is the base of the software, we assumed it in this way:

- Users have all a smartphone from where they can use the mobile application and their smartphones have GPS;
- Users have always internet connection from their devices in order to tell to the system that he/she is nearby;
- GPS give always precise information about car position;
- GPS cannot be faked or modified by anyone;
- All cars have GPS and the system can track them;
- All cars are the same and they consume the same amount of battery charge;
- If a car has a failure, it can't be used until it is repaired.
- When in maintenance, a car is out of service;
- Cars have a screen that shows,
- Cars have different sensors such as charging sensor and battery level sensor;
- To check if there is more than 1 passenger in the car, car seats have a sensor that observes a passenger's weight: if it gets a value greater that 40 Kg, the system will evaluate it as a passenger.
- There are 2 different kinds of parking: "safe areas" and "special parking areas";
- Special parking areas have sensors that measure the number of in-use plugs;
- The only compatible chargers for the cars are in the special parking areas.

1.5 Glossary

- Agreed zone: general name that identifies both the safe area and the special parking area (explained later).
- Basic tariff: is the time-based rate that charges a user when is using a car (moving mode and also pit stop mode).

- Car state: is the mode in which the car is set by system in order to distinguish all the behaviors that the system must adopt on them.
We defined 13 states that can overlap each other following the table represented below:

- Available = state in which a car is when is in a safe parking and it can be booked by a user through the app. When the car is available is also locked and fully charged.

- Unavailable = state in which a car cannot be reserved by users because it is actually busy. It could be busy for many reasons: a user is using it, a user has just reserved it, is out of service, is in recovery mode or is plugged.

- Locked = state in which a car has no one on board and the system puts it in this state. In general a car is locked when someone leaves the car and closes the door. The user can unlock the car if has reserved it (and one hour is not passed yet) or if was in pit stop mode (and the timer is not expired).

- Unlocked = state in which the car is unlocked when a user is on board or he has just unlocked it (if anyone unlocks the car within 5 minutes, the systems sets it in lock mode), or an operator is recovering the car (with the help of the third party system).

- Safe Parked = a car is in this state when is in a safe parking (both types). A car could be here also because is out of service.

- Unsafe Parked = when a user is not in the car, the engine is stopped and the parking is not safe. The car could be in this state when the user did not want to bring it to a safe parking so he left the car somewhere outside of it.

- Pit Stop = the user has the possibility to keep the car for several days. he can park wherever he wants by selecting on the car screen the option “pit stop”. In this way the car will be reserved for him for other 24 h (at the end of this timer, the car is set in Recovery mode). If a user did not select anything from the two option on the screen (stop and pit stop) system sets the car in Stop mode by default and if it is in an unsafe parking, set it in Recovery mode. Pit stop mode is available only out of safe parking. When a car is in pit stop mode, the basic tariff is still of force.
- Pit Stop Timer = is equal at the pit stop mode, but in this case the user could when is out of the car and when the car is in pit stop mode, extend the timer of the pit stop over the 24 h. He can do this through the app.

- Moving = This mode lasts from when the user start the engine until he stops the engine.

- Out of Service = when the car doesn’t work for any reason (a damage, a maintenance or something else) the system puts the car in this state and a third party system will take care of it.

- Recovery = state in which a car goes automatically when a user parks the car in an unsafe parking without selecting the pit stop mode or when the pit stop timer has expired. The System sets the car in this mode and the third party system will take care of it.

- Plugged = when a car is a special parking and is in charge.

- Charging = when the system is applying the basic tariff to the user.

TABLE OF CAR STATES

Caption:
Grey = the two states cannot overlap;
Green = the two states overlap;
Yellow = the two states can possibly overlap in some cases.

	AVAILABLE	UNAVAILABLE	LOCKED	UNLOCKED	SAFE PARKED	PARKED	PIT STOP	PIT STOP TIMER	MOVING	OUT OF SERVICE	RECOVERY	PLUGGED	CHARGING
AVAILABLE													
UNAVAILABLE													
LOCKED													
UNLOCKED													
SAFE PARKED													
PARKED													
PIT STOP													
TIMER PIT STOP													
MOVING													
OUT OF SERVICE													
RECOVERY													
PLUGGED													
CHARGING													

- Charging spot: is a parking slot provided with an electrical plug used to charge the car battery.
- Charging station: synonym for special parking area.
- Credentials: these are the personal information of a user. They include first name, last name, email address, password, IBAN (explained later) or credit card number and geographical position.
- GPS (Global Positioning System): is a positioning system based on triangulation with satellites in order to give the exact position of a device in the world.
- IBAN (International Bank Account Number): is a unique string that locates a bank account.
- Parking lot: synonym for safe area.
- Safe area: is a parking agreed by the car sharing company where users should park rented cars. There are a certain number in the city and they are equally distributed in the territory.
- Special parking area: is a parking with charging spots agreed by the car sharing company where users should park rented cars. Users can park cars here and they can charge them. Special parking areas ARE safe areas, on the contrary, safe areas are not all special parking areas.
- UU: abbreviation for “unregistered user”.

1.6 Assumptions

- The only actor is the User: we decided to assume that all the system management and bureaucracy part (intended as fine payment, car maintenance, car retrieve and stuff like this) will be handled by a third party, so it concerns another system that will not be developed.
- There is a limited number of cars, stations and plugs so, if a user selects the money saving options and there are no available plugs, he/she will not receive a discount.
- The discounts can be accumulated and the system applies all of them on the amount calculated by time. The maximum discount that a user can get is 90%.
- When the user uses the service from mobile application, he/she has to share his/her location.
- When users book cars, they cannot revert the operation.
- Users can park the car in unsafe areas.
- If the user has no money on his payment method, he cannot book a car.
- If the user has some not payed trips, he has to fulfill them first, then he can book another car.
- If a user wants only to park the car, he has to select the “Pit Stop” option on the screen and then he can take the car back within a day if it is parked outside of the predefined zones.
- When a user wants to rent a car, the system will show only the full charged ones.
- All cars, after users end their trip a a safe area, must be at full battery charge before being set as available again.
- While the car is in pit stop mode, the basic tariff is still in force.
- In case of failures on the road, the user will not pay because he doesn’t receive the correct service. Then the car goes automatically in recovery mode.

1.7 Constraints

The system can run on a mobile device (such as smartphones and tablets via the mobile app): the mobile device should have GPS, internet connection and some storage space where to install the application. When users access the service, they have to share their position in order to allow the system to show up the nearest parking.
Other important constraints to the system are the DBMS, that will store all user data, such as their credentials, payment method and email address, and the access to a SMTP server that allows the system to send email to the users.
The system interfaces with a third party system that relies on a different database that stores all information about cars such as fines written on them, car documentation (like revision papers), position and other important data.

1.7.1 Interfaces with other applications

The cooperation between this third party system and the software-to-develop will be explained better in the “functional requirements” section of this document.

1.8 Proposed System

We think that the best solution for the implementation is to build the software for a mobile application in order to allow everyone to use the service anywhere and anytime.
This implementation will allow users to benefit of the service everywhere in order to allow maximum accessibility in any condition.
The application will be available for all platforms (Android and iOS) in order to cover the most part of the smartphone users, and maybe, in future development, will be implemented a Windows Phone version too.

1.9 Identifying Stakeholders

In case this software has to be released, the stakeholders could be a car sharing company that needs a software system to manage the activity of the enterprise, but in this particular case, where the software has to be developed to demonstrate the ability to find requirements, test and other things, the stakeholders are the professors.
Our main goal is to demonstrate that we are capable of a work like this and that we can organize ourselves to perceive the same objective, even if we cannot encounter in real life due to different personal obligations.
Talking about the final user of this kind of system, we think that the most probable is the common citizen that lives in a city with a lot of traffic that doesn’t want to use public means of transport for delays and other discomforts. Our solution could resolve a lot of people’s problems.

1.10 Reference documents

- Assignments AA 2016-2017.pdf;
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications;
- RASD sample from Oct. 20 lecture.pdf;
- RASD ExampleSWIMv2.pdf;
- RASD_meteocal-example1.pdf;
- RASD_meteocal-example2.pdf;
- RASD_meteocal-example3.pdf;

2 ACTORS IDENTIFYING

The only actor that will use our system is the User. It is the representation of the common citizen that uses the service for moving/traveling purpose. It will use the system by registering an account, renting cars, notifying the system that it is nearby the car, locking and unlocking the car, checking out and eventually call assistance;

3 REQUIREMENTS

3.1 Functional requirements

3.1.1 Requirement 1

Allow the users to register in the system and to become part of the community:

- The system must be able to check driving license and payment method authenticity and validity.
- The system must send a password for log in at the e-mail address provided.
- The system must provide to the user a redirection link for the activation of the account within the welcome e-mail.

3.1.2 Requirement 2

Allow the user to select a parking lot from a subset of them where to pick up a car:

- The system must be able to provide the list of all parking lots with at least one free vehicle, from the nearest to the furthest, within 1 km from the user's position or from the one inserted by user.
- The system must be able to detect user's location according to user's GPS.
- The system must be able to provide basic information about the car, such as tariff.

3.1.3 Requirement 3

Allow the user to reserve a car:

- The system must allow the user to book a car from the selected parking lot.
- The system must show the remaining pickup time to user.
- The system must remove the car from the available ones according to the user's choice.
- The system must be able to lock a reserved car until the user that booked it, unlocks it.

3.1.4 Requirement 4

Allow the user to unlock the car:

- The system must remove the car from the available ones until the user ends to use it.
- The system must be able to detract 1€ from the user's payment method if he does not pick up the car within the hour.
- The system must provide a feature in order to unlock the car.
- The system must be able to detect the car's location according to the GPS of the car.
- The system must be able to detect user's location according to user's GPS.
- The system must notify the user when the time is up if the car has not been unlocked and about the detraction.
- The system must add the car to the available ones if the car has not been unlocked in time.
- The system must be able to nullify the reservation in order to not permit the user to pick up the car if the time is up.

3.1.5 Requirement 5

Allow the user to use the car and to get a discount:

- The system must be able to check the time that passed from the first ignition of the motor to the checkout, in order to evaluate the final ride cost.
- The system must provide the update information about the actual ride cost to the user in any moment from the app and the built-in screen
- The system must apply the discount at the end of the ride and let know the actual cost to the user from the application
- The system must check the value of the sensors under the seats of the car.
- The system must check the car battery charge at the end of the ride.
- The system must check if a car is plugged or unplugged before the checkout
- The system must check the final position of the car after the checkout, from the car's GPS, and if it's in a safe park or not.

3.1.6 Requirement 6

Allow the user to select the money saving mode:

- The system must provide to the user the possibility of activating the money saving option.
- The system must be able to locate the final destination provided by the user
- The system must be able to locate the nearest safe parking areas to the final destination
- The system must be able to tell the user of the destination provided exists
- The system must be able to provide to the user a list of possible safe parking areas ordered according to: number of free parking slots, number of plugged, charged and uncharged cars and the distance of the single parking area from the final destination.
- The system must be able to check the number of charged, uncharged and plugged cars in any parking area of the Company
- The system must be able to know the number of parking slots in any parking area of the company

3.1.7 Requirement 7

Allow the user to finish to use the car:

- The system must be able to detect if a car is in a safe parking area

- The system must allow the user to lock the car
- The system must be able to set the car as "to be charged" mode after the user locks it
- The system must be able to detract the right amount of money with the method of payment provider by the user after the locking of the car in "stop" mode.

3.1.8 Requirement 8

Allow users to pay via IBAN or credit card:

- System should automatically detract the correct amount of money from the payment method.
- System adds a not-payed receipts to user's account if the transaction fails.
- System must allow users to pay not-payed bills by clicking the relative button in the application.

3.1.9 Third-Party System

In order to satisfy some conditions requested by the goals of the system-to-develop, this one will be flanked by a third party system that manages the administrative side of the service and the maintenance of the cars.
This additional system is supposed to be already implemented and already fully working in the company where the system-to-develop will be finally settled and in this document, we explain only which tasks it has to fulfill and how it interfaces with our system.
This system will be used mainly by operators that has the task to retrieve unsafe parked cars, they will put in charge cars in order to set them as available again when fully charged and they will answer as assistance.
In particular this system will show possibly 3 lists of cars: one will contain the cars that have to be retrieved for expiring of the pit stop timer, one will show the cars that have to be maintained (the ones that will go in "Out of Service" mode) and the final one will show the cars that received a fine.
Car will be listed accordingly to the "car state" in which they are: if the car goes in recovery mode after pit stop timer mode is over, the on board computer will send a "change state" event. This event will trigger certain conditions that will show that particular car in one of the previously described lists, in order to allow operator to take care of it in the best way.
The system-to-develop and the third party system will interface with a common database that holds all cars information: the communication between them will work on data exchange through the database so the users will not notice this interaction.
In this way the system-to-develop will handle only the interaction with users, while the third party system will work behind the scenes to provide some features of the service.

3.2 Non-functional requirements

3.2.1 App Icon

Figure 1

3.2.2 Main page - App interface

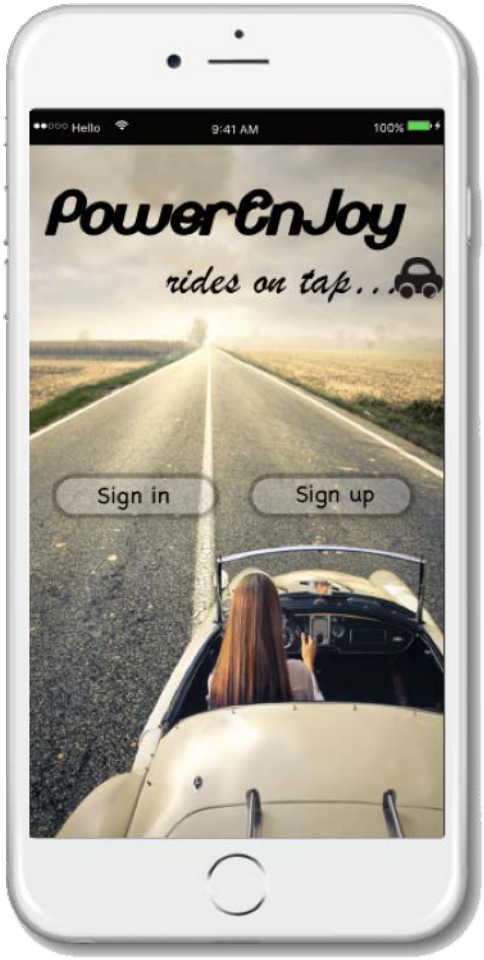
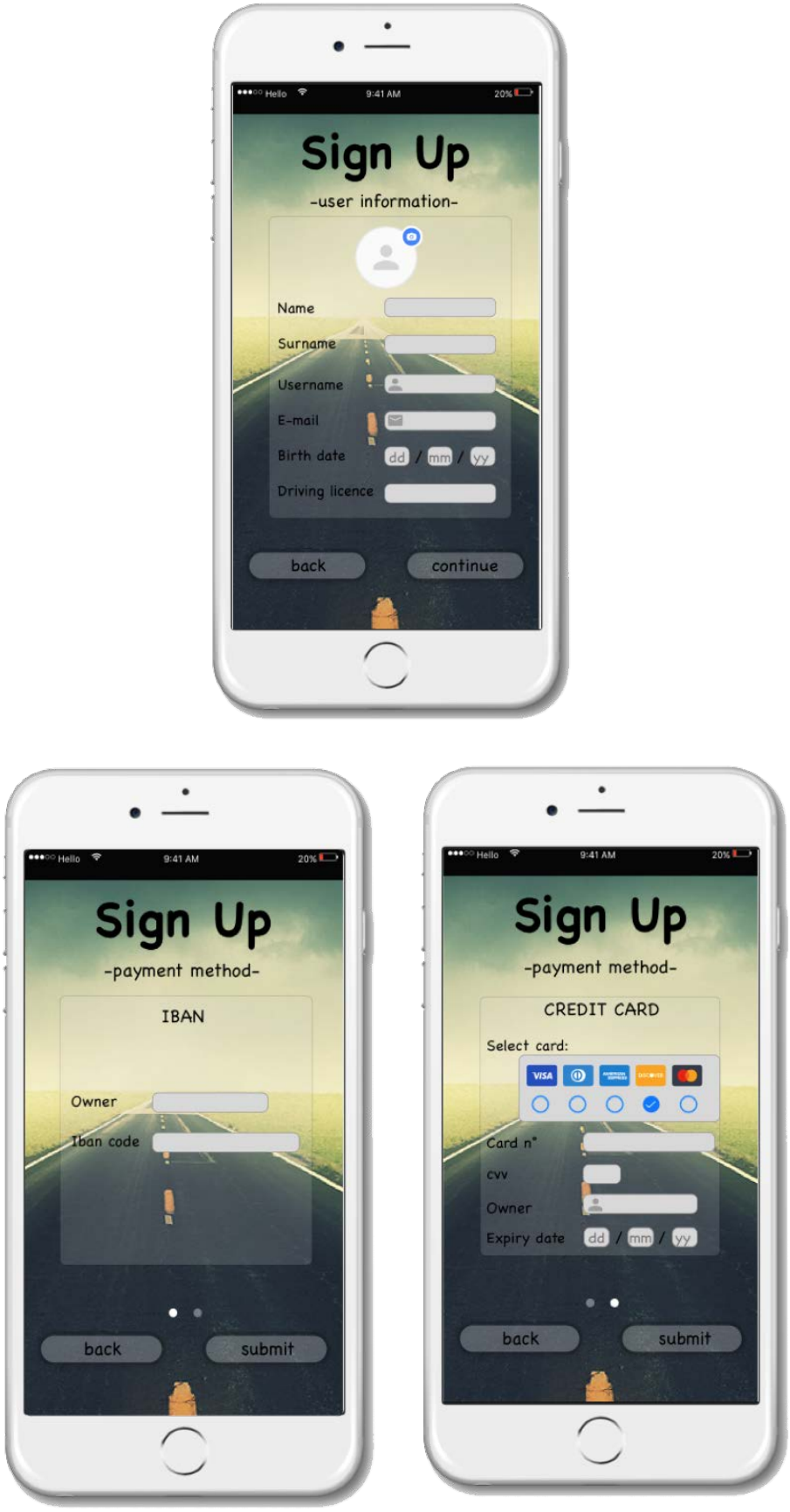


Figure 2

3.2.3 Registration - App interface



Figures 3, 4 and 5

3.2.4 Log in - App interface

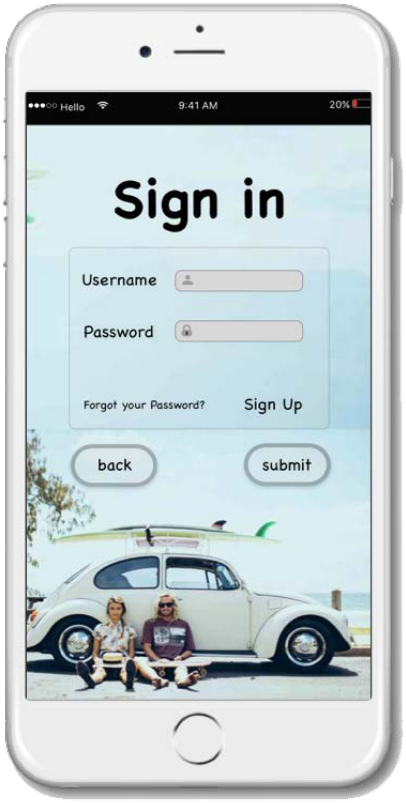


Figure 6

3.2.5 Main menu - App interface

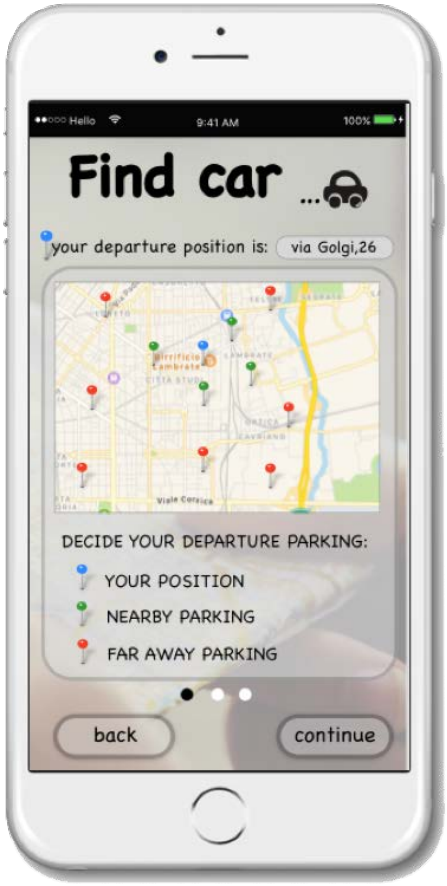
Figure 7

3.2.6 User Info - App interface

Figure 8

3.2.7 Find Car - App interface

Figure 9



Figures 10, 11 and 12

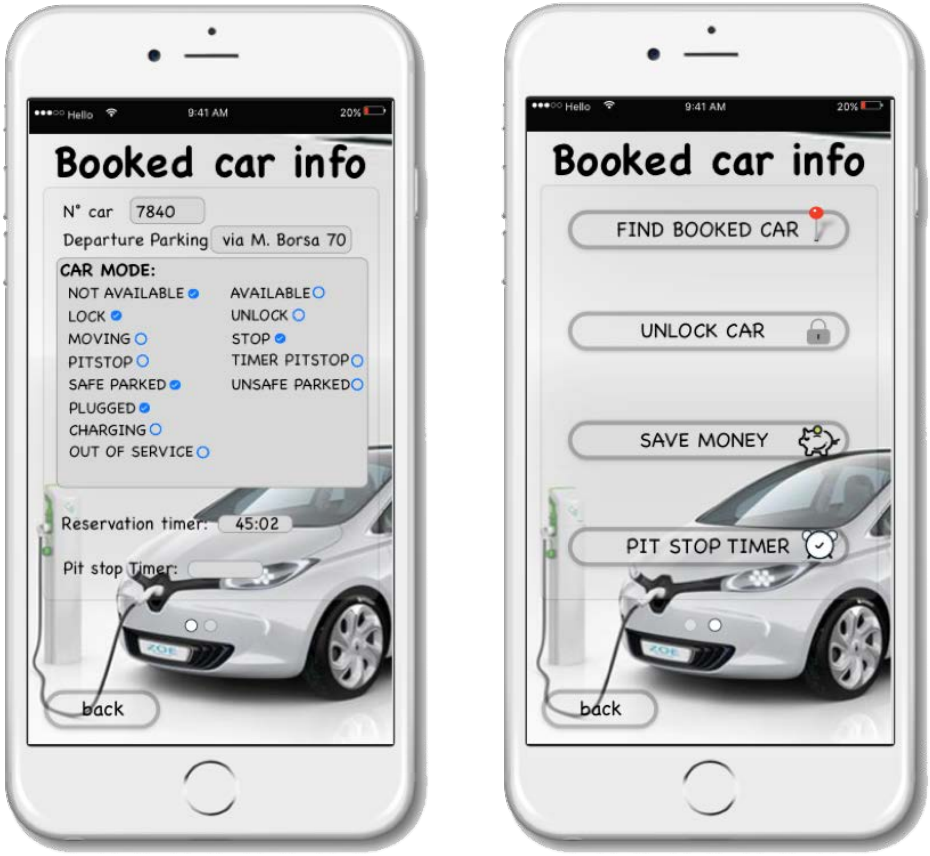
3.2.8 Book Car - App interface

Figure 13

3.2.9 Save money - App interface

Figure 14

3.2.10 Check Car Reservation - App interface



Figures 15 and 16

3.2.11 Welcome mode - Car interface

Figure 17

3.2.12 Moving Mode - Car interface

Figure 18

3.2.13 Stop Mode - Car interface

Figure 19

4 SCENARIO IDENTIFYING

4.1 Scenario 1 - Base Case

John is an employee who works in an important company in Milan but he lives in the suburbs, far from his workplace. John has already obtained the driving license but he haven't bought yet a car of his own . Today John has to go to work early so he decides to rent a car to go to work.
He downloads the app, registers an account by inserting his credentials and his payment method and books a car from the nearest parking, within 10 minutes he gets out and pick up the car; later, he arrives at the workplace and parks the car in the safe area in front of the building and 5 minutes later he receives the payment receipt via email.

4.2 Scenario 2 - The Forgetful

Today John goes to work with a colleague of his own that has a car so he doesn't need to book a car.
Two hours before the end of the work shift, John's colleague remembers him that he has to remain in the office a little longer so he cannot bring him home. Now he has to book a car from the car sharing for the end of the work shift (18.00) but the app doesn't permit it: he can book a car at most one hour before he picks up the car so he has to wait until 17.00 to book a car. At 17.00 John tries to reserve a car, but the system doesn't allow it because system failed to detract money from his payment method for lack of funds. So John enters his account and changes the payment method to pay the pending trip. After the transaction went good, the system allows Johns to book the car for around 18.20.
At the end of the work shift (18.00) John exits form the working place and encounters an old friend of his and chats with her for 30 minutes. After saying goodbye to each other, John goes towards the car but he realizes that the reservation has expired nad an email address confirms the detraction of 1 € for the missed pick up. Fortunately, in that parking is plenty of cars, so he books another one through the application in order to finally go back home. The only thing is when he approaches to the car, his smartphone runs out of battery charge and switches off. Now he has to find an alternative way to arrive at home.

4.3 Scenario 3 - The Money Saver

Today is an important day for John because he has an interview in Rome. For the special occasion, he decides to book a car to go to Rome spending less money: in order to save more, he invites 3 of his friends for a daytime visit of the capital city and he also activates the money saving option in order to get the address of the best special parking area where to leave the car.
When they arrive in Rome, the find immediately a parking spot in the previously mentioned special parking area: furthermore the battery charge is greater than 50% but John plugs the car to the charging plug anyway. so he will get a lot of discounts when he checks out from the car.

After one hour he and his friends book another car and they go around the city to visit the most famous places of Rome.

4.4 Scenario 4 - The Unlucky

Today is a bad day for John: it's rainy and when he tries to book a car from the nearest parking, he realizes that it is empty and he has to book a car from a 2 km further parking. Furthermore the traffic is particularly heavy due to the bad weather to he gets caught in it. The battery of the car runs out of charge in the middle of the road, 3,1 Km far away from the nearest charging station so he has to leave the car on the road in pit stop mode (he has 24h to retrieve the car but he decides lo leave it there until an operator recovers it) and he has to go to his workplace by feet or by public means of transport. Unfortunately he has to pay a little bit more at the end of the pit stop time because of the low battery and the parking in a not agreed zone. At the end of the work shift it's still raining so he decides to rent another car. After 20 minutes of driving, the car has a mechanical failure, he calls the assistance but it looks that the problem cannot be resolved immediately so he has to take public means of transport to go home.

4.5 Scenario 5 - The High Tech Housewife

Mary and her husband have only one car. Today Mary has a lot of obligations and without a car she could have lost the entire day and she couldn't come back home in time before their sons return from school.
So Mary decides to rent a car from PowerEnJoy, a car sharing company that the have already listened about from her colleagues. She downloads the app and registers a new account and, since she never used the service before, she has to input all the credentials and a valid payment method.
A few seconds later, she receives an email with the password and books a car at 9:45 from a 5-minute-walk parking near her house.
At 10.15 Mary retrieves the car that she rented, she unlocks it and gets out of the safe parking: now she can go to the supermarket for shopping with the help of the "Pit Stop" mode that allows her to keep the car for herself whole day.
After all the commissions, the brings back the car in the safe area where she picked it up.

4.6 Scenario 6 - The weekend in the mountains

Luke, Lewis and Martha live in Milan and they decide to spend a weekend in the mountains but anyone of them has a car and it would be impossible to reach the location with public means of transport.
So they decide to book a car from PowerEnJoy: Luke was already registered so he logs in, books a car, suddenly he pick it up and goes to get his friends.
They arrive to the location in one hour and Luke sets the Pit Stop mode. After switching of the car, Luke enhances the pit stop time for 2 more days instead of 1 from the app in order to get the car at the and on the vacation and go back in town.
When Luke arrives at home, he's so much tired that he decides to leave the car in a parking near his house in stop mode in order to let an operator to retrieve the car. He knows that he will have to pay 30% more due to the low battery and the distance from a safe area but he ca afford this.

4.7 Scenario 7 - Remember to put in Pit Stop the car

Sarah is on a car of PowerEnJoy Company and she's headed to the supermarket. Once she arrived at her final destination, she forget to set the Pit Stop mode. Poor Sarah, the car locks and goes in recovery mode, so an operator will retrieve it, and she will have to go back home with the underground with all the shopping bags!

5 UML MODELS

5.1 Use case diagrams

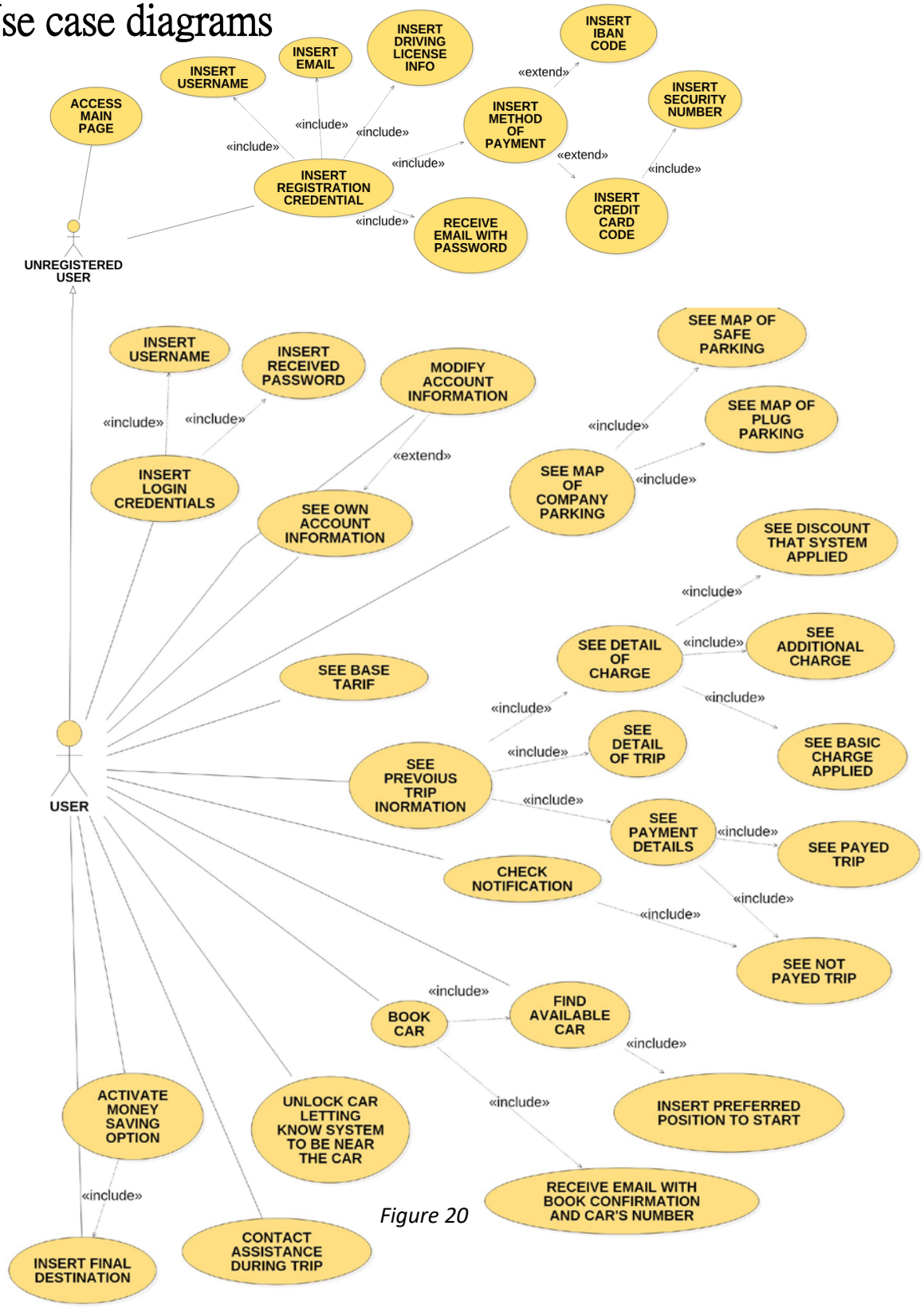


Figure 20

5.1.1 Insert registration credential

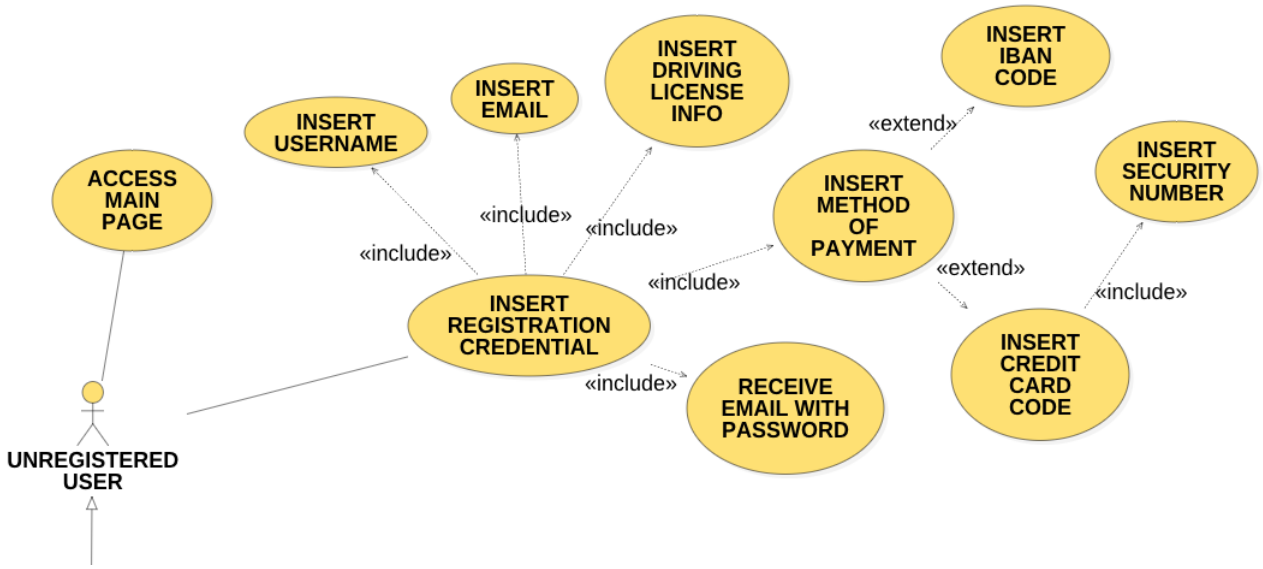


Figure 21

Actors: Unregistered User.
Entry conditions:

- The user selects to register himself into the system from the main page.

Flow of events:

- The user inserts his credentials in the featured form.
  His credentials include:
  - First Name;
  - Last Name;
  - Date of birth;
  - User name;
  - E-Mail;
  - Driving license info;
  - Payment method.
- The user clicks the registration button;
- The user is successfully registered.
- The system sends a welcome mail to the user that contains his password;

Exit conditions:

- The user is redirected to the log in page.

Exceptions:

- Some of the information the user provided are incorrect. The user is not redirected but is notified of the error fields.

5.1.2 Insert payment method

Actors: Unregistered user.
Entry conditions: No entry conditions.
Flow of events:

- The user chooses the payment between the available options and completes the related fields:
  - If he chooses IBAN, he inserts the IBAN code
  - If he chooses credit card, he fills up the following fields:
    - Credit card number;
    - CVV;
    - Holder name;

- Expiration date.

Exit conditions:

- The system displays a notification of the correctness of the information provided and enables the registration button.

Exceptions:

- The system displays an error notification associated to the incorrect fields and disables the registration button.

5.1.3 Insert log in credentials

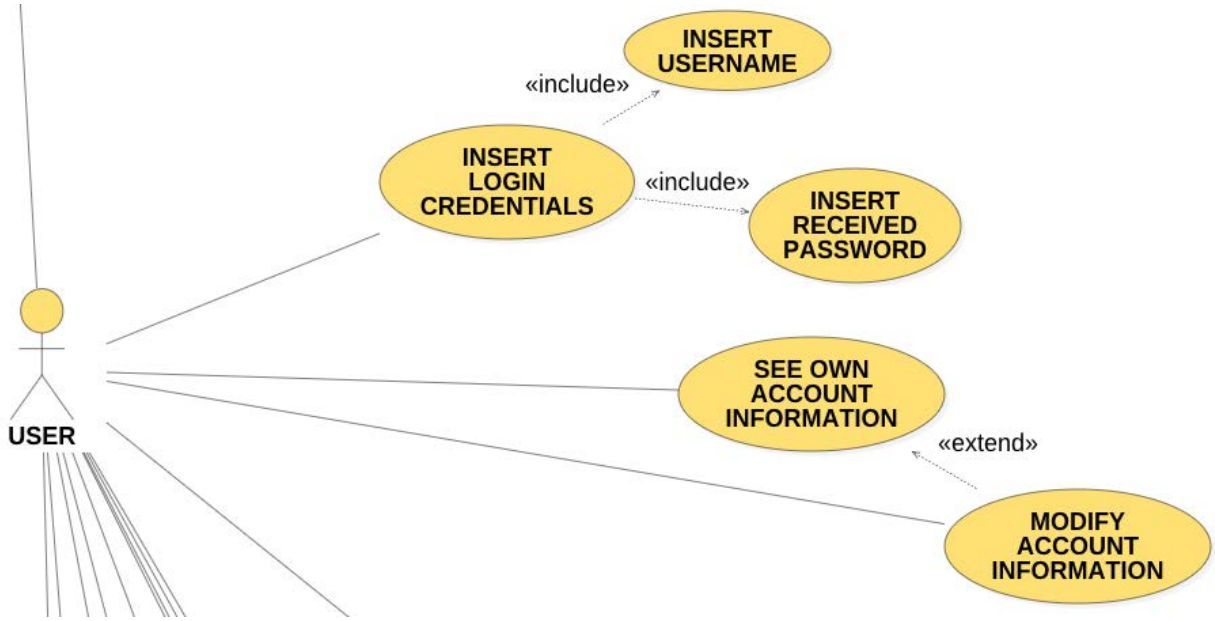


Figure 22

Actors: User.

Entry conditions:

- The user is registered and has received the password.

Flow of events:

- The user opens the app for the first time;
- The user inserts his credentials that include:
    - User name or e-mail;
    - Password.

Exit conditions:

- the user is redirected to the research page of the app where he can now book a car.

Exceptions:

- Some of the information the user provided are incorrect. The user is not redirected but is notified of the error fields.

5.1.4 See own account information

Actors: User.

Entry conditions:

- The user is successfully logged.

Flow of events:

- The user clicks on the account button of the app;
- The user sees his information and eventually edits them.

Exit conditions:

- The user successfully goes on another page.

Exceptions: there are no exceptions.

5.1.5 Find available car

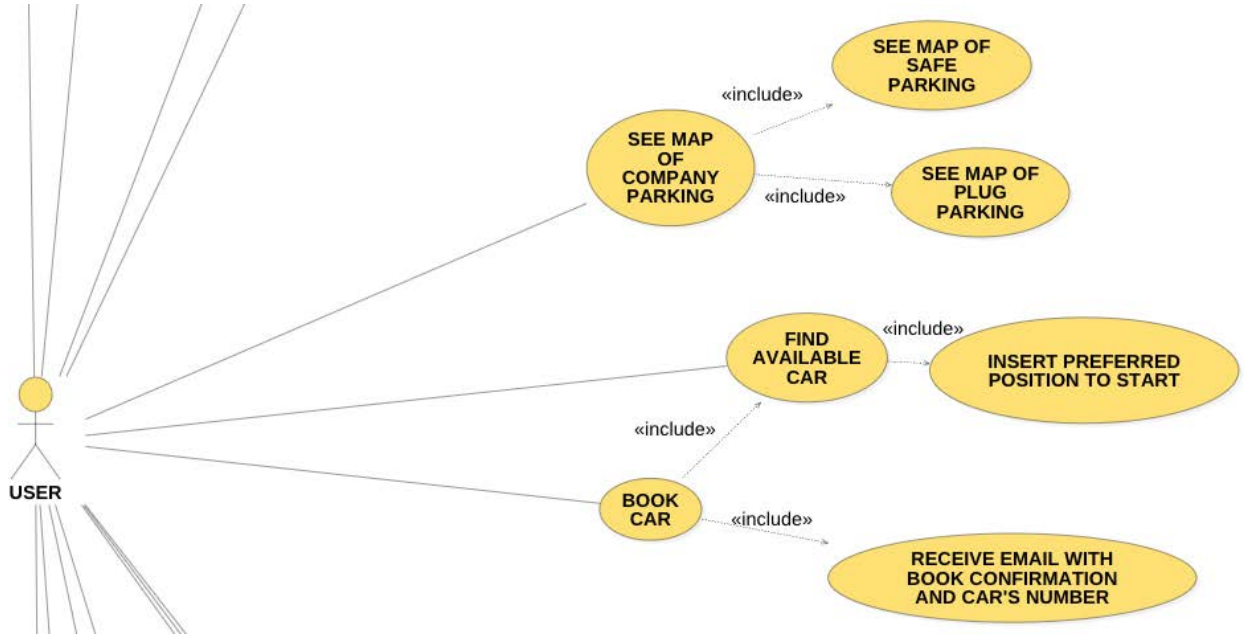


Figure 23

Actors: User.

Entry conditions:

- the user is successfully logged.

Flow of events:

- The user goes into the research section;
- The system shows the available cars near his current position;
- If the user inserts a different position or place, then the system displays the available cars near that position;

Exit conditions:

- The user changes page;
- The user selects an available car from the system displayed options;
- The user presses "book" button;
- The user presses "map" button.

Exceptions:

- The GPS is not activated: The system doesn't display any car but an option link that redirect the user to the device settings in order to activate the GPS;
- The user inserts an unrecognized position or place. The system displays a message that no available cars are found.

5.1.6 Book car

Actors: User.

Entry conditions:

- The user selects an option from the research page.

Flow of events:

- The user sees the information related to the car;

- The user can eventually activate the money saving option;
- The user clicks on the book button;
- The system checks the availability of the payment method;
- The system checks for previous not yet payed trips.

Exit conditions:

- The system notifies the user of the successful booking, starts 1-hour countdown visible on the app as picking up threshold, redirects the user to the "car" page where he can see the vehicle information.

Exceptions:

- The user inserts incorrect information. The system notifies the user about the incorrect fields;
- There's a pending payment related to a previous trip not payed yet. The system redirects the user into the payment details section.

5.1.7 Unlock car

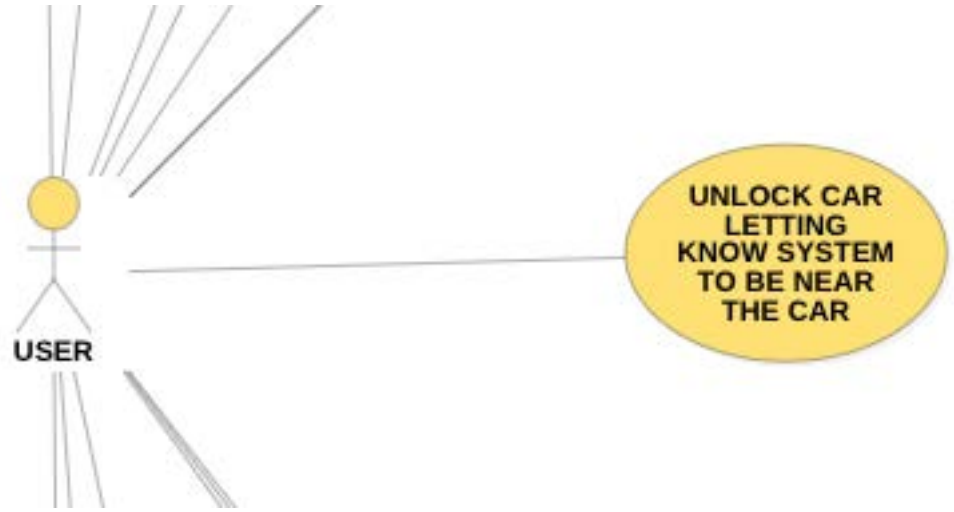


Figure 24

Actors: User.

Entry conditions:

- The user has booked the car less than an hour ago;
- The user has parked the car and less than the pit stop time limit of the car has passed.

Flow of events:

- The user accesses to the system;
- The user goes on "car" page;
- The system checks the user position by GPS and enables the unlock button if the user is near the car;
- The user clicks on the unlock button.

Exit conditions:

- The user can now use the car.

Exceptions:

- the GPS is not working. The system notifies the user and redirects him to the device settings in order to activate it;
- The internet connection is not available. The system notifies the user to activate the Bluetooth and to put the device near the car in order to unlock it;
- The user cannot unlock the car using his device. The user could ask for support to the authorized personnel.

5.1.8 Pay for a car

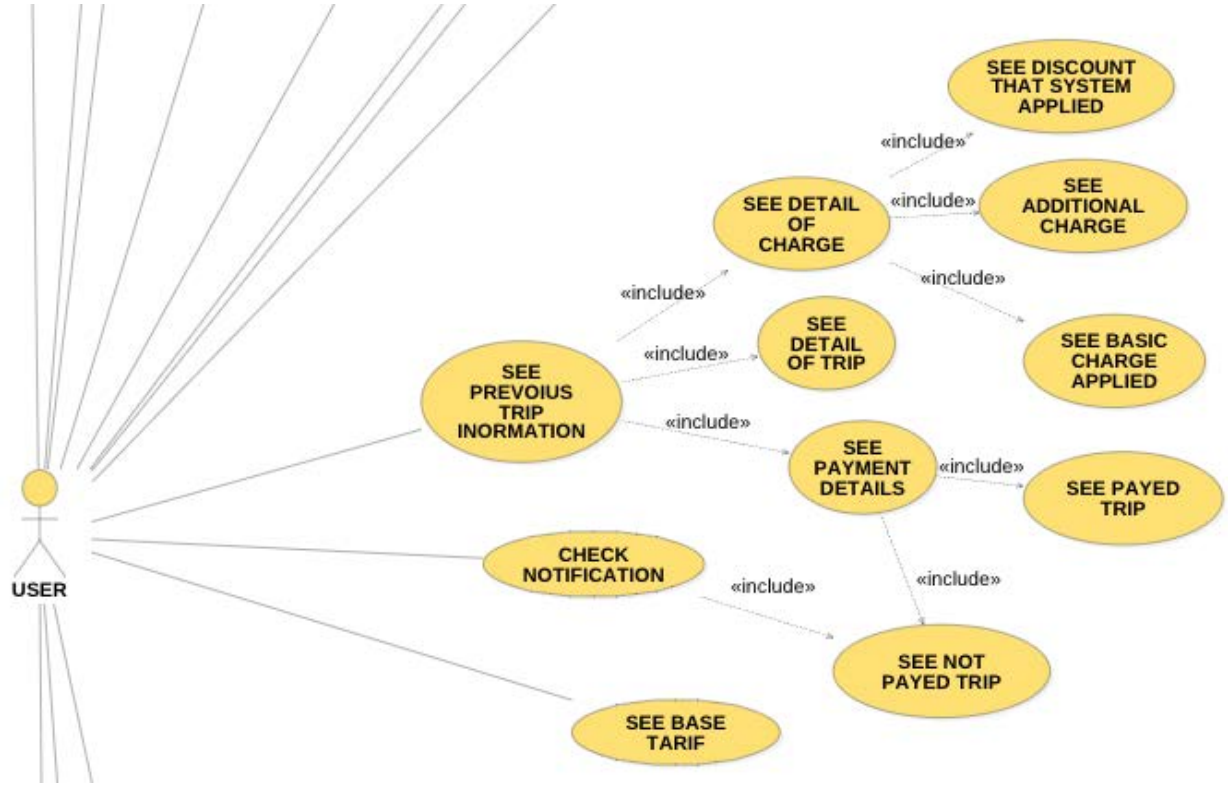


Figure 26

Actors: User.
Entry conditions: The user has to pay the last ride but an error occurs during the automatic checkout.
Flow of events:

- The user goes into the payment section;
- The user clicks on "Pay" button;
- The system checks that the method of payment is correct;
- The system detracts the amount of money of the last ride;

Exit conditions:

- The system notifies the user of correctness of the transaction.

Exceptions:

- The check of the method of payment fails: the system notifies the user about it and redirect him to the account settings;
- The system cannot detract the amount of requested: it notifies this to the user suggesting to control the residual credit.

5.1.9 Contact assistance during trip

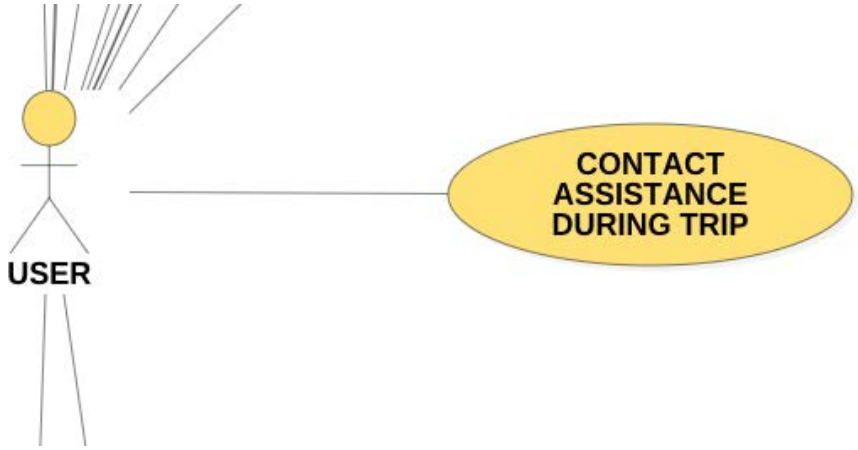


Figure 27

Actors: User.
Entry conditions:

- The user is using the car.

Flow of events:

- The user selects the assistance from the display in the car;
- The system notifies the personnel of the call.

Exit conditions:

- The user is contacted by the personnel immediately.

Exceptions:

- The internet connection or the telephone network does not reach the car;
- The personnel will contact the user as soon as an internet or telephone connection will be available.

5.1.10 Activate money saving option

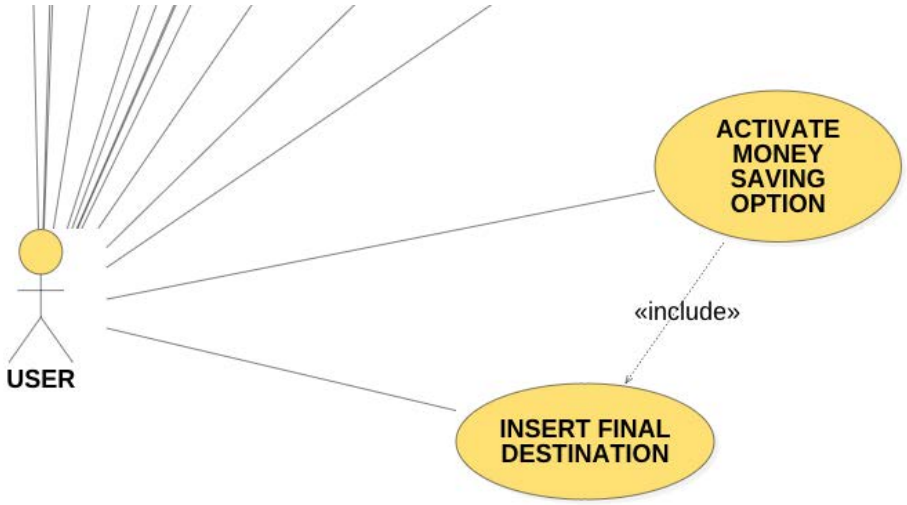


Figure 28

Actors: User.
Entry conditions:

- The user has booked a car.

Flow of events:

- The user access to the system;
- The user selects the money saving option from the "car" page;
- The system shows a form where the user chooses the final destination;
- The system provides information about the nearest station to the final destination where to leave the car to get a discount.

Exit conditions:

- Deactivate the money saving option;
- Drive to the station indicated by the system;
- The user makes a pit stop in an unsafe area and passes the time limit.

Exceptions:

- The internet connection is not available;
- The system shows a connection error and allows the user to retry;
- The user inserts an unrecognized position or place. The system shows a "not found" error asking the user to check if the destination in correct;
- The user parked in the right place but doesn't receive a discount. The user could contact the personnel for more information.

5.2 Class diagram

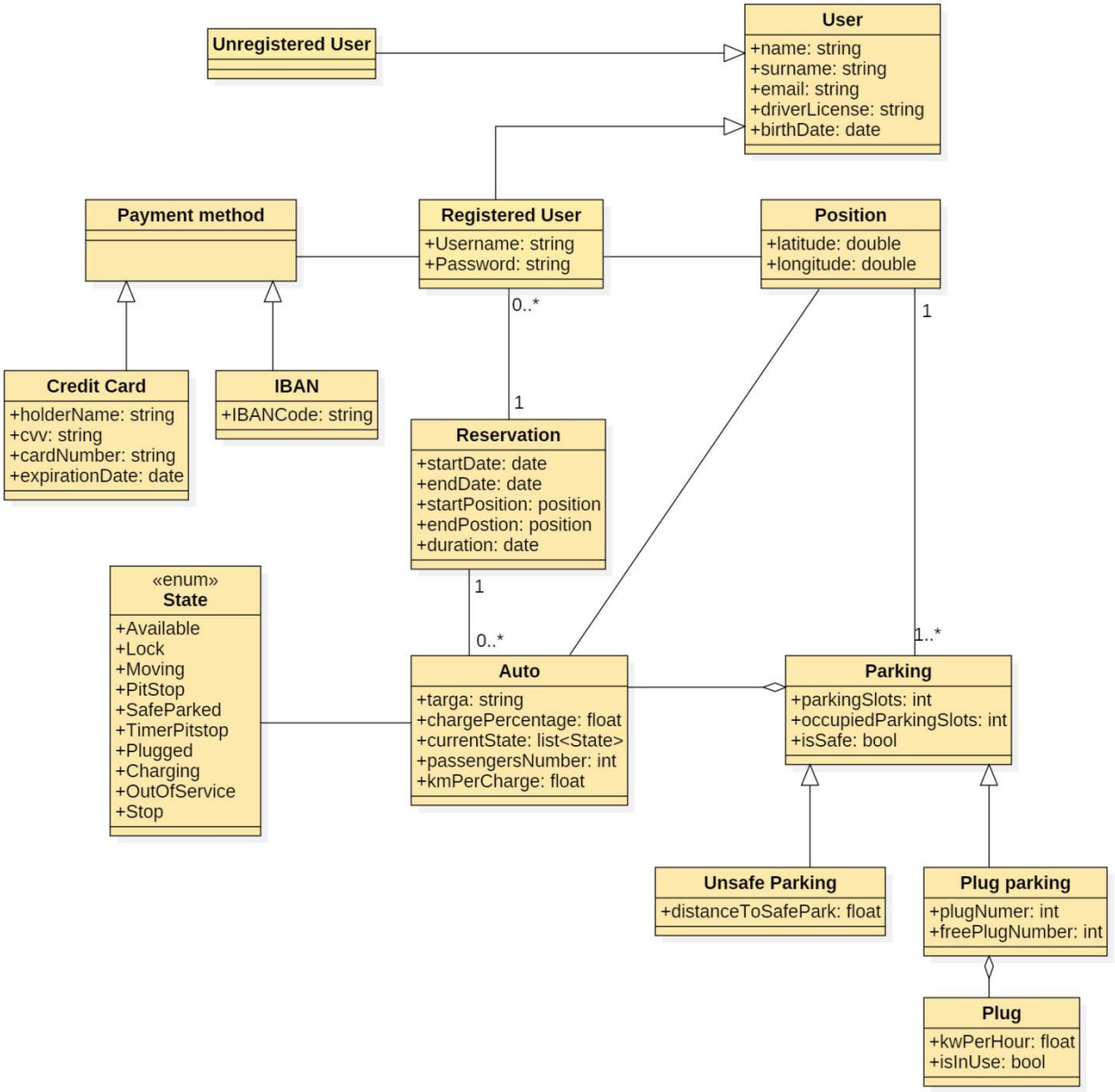


Figure 29

5.3 Sequence diagrams

5.3.1 User's Checkout - Sequence diagram

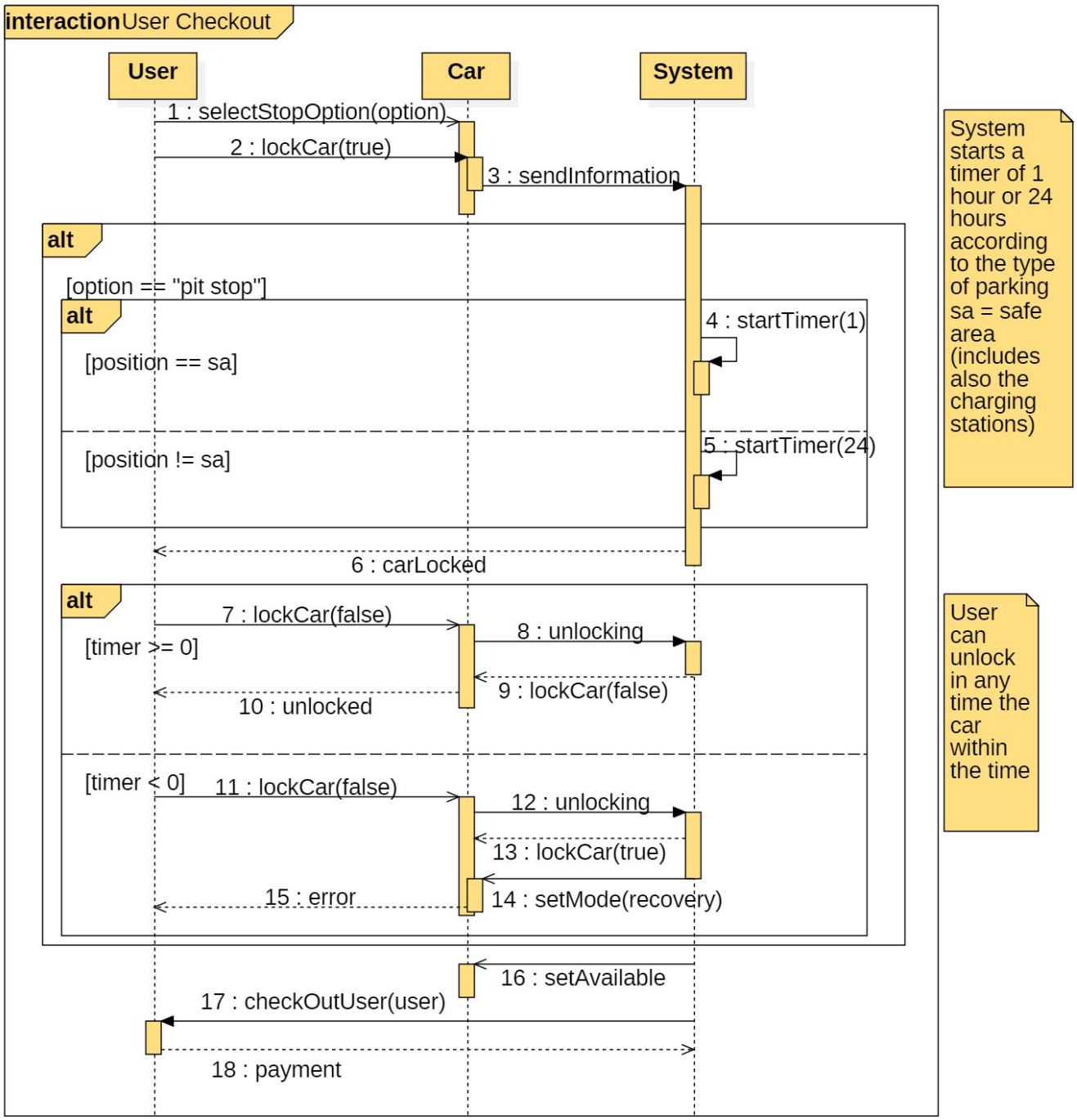


Figure 30

5.3.2 System applies discount - Sequence diagram

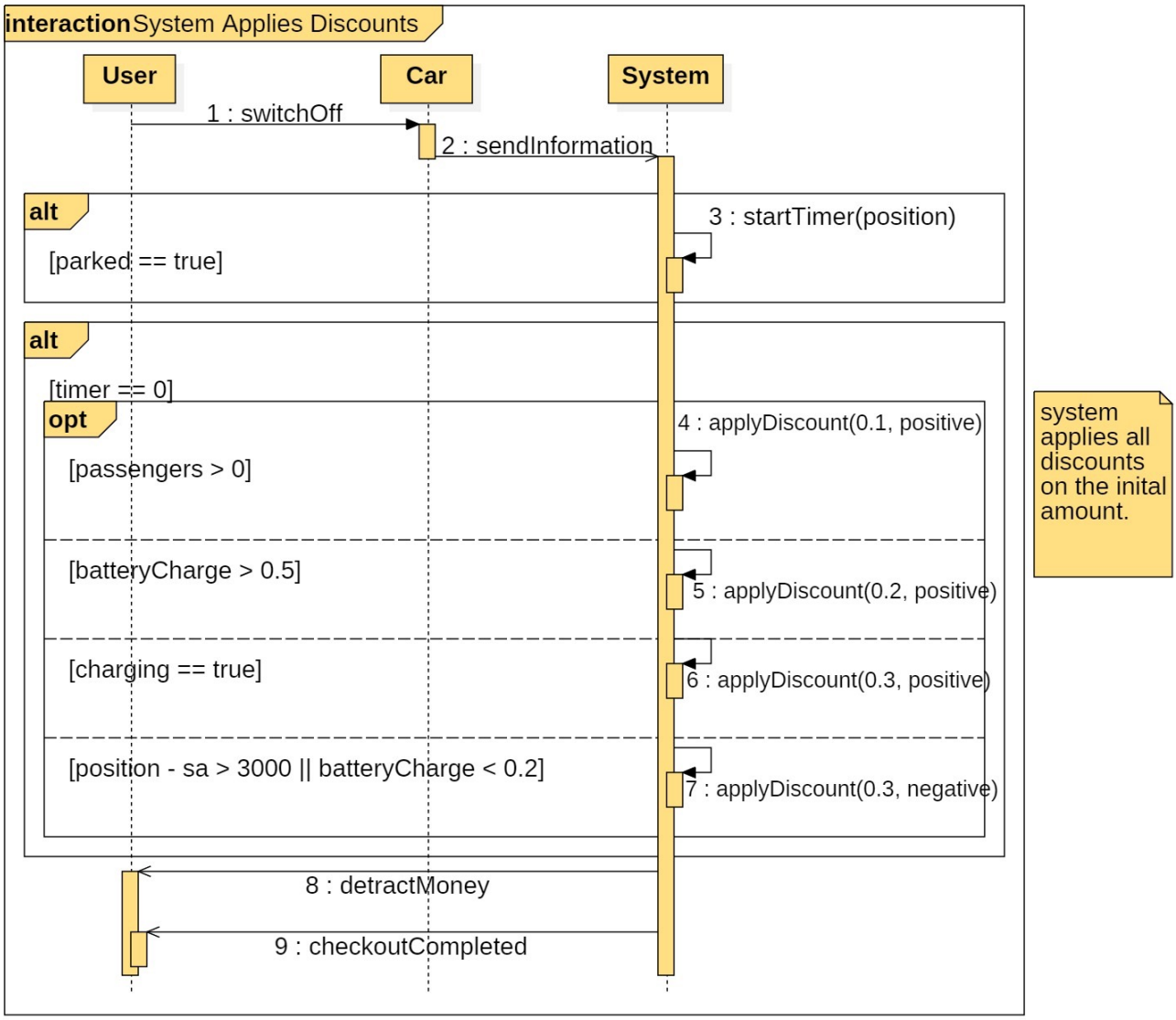
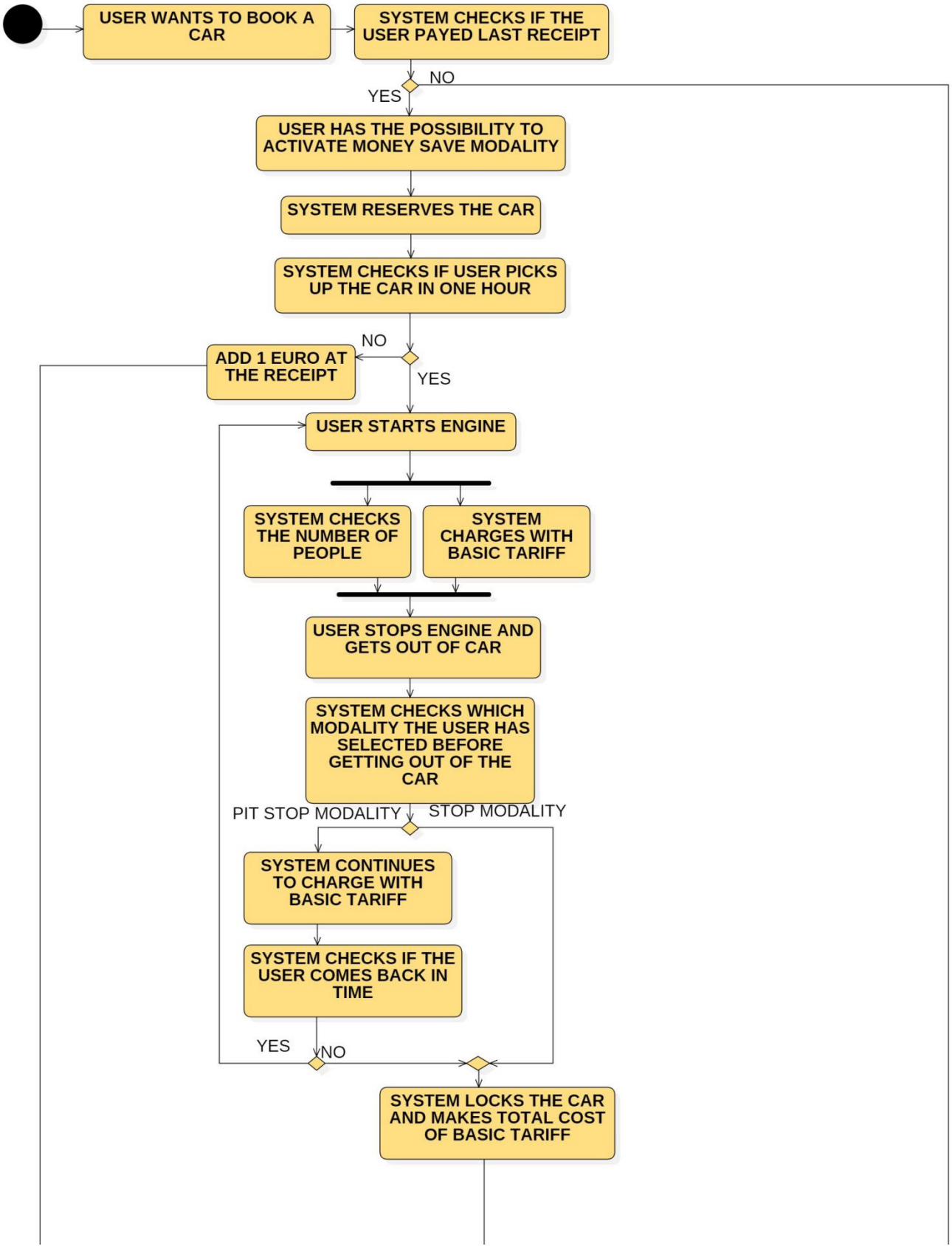


Figure 31

5.4 Activity diagram



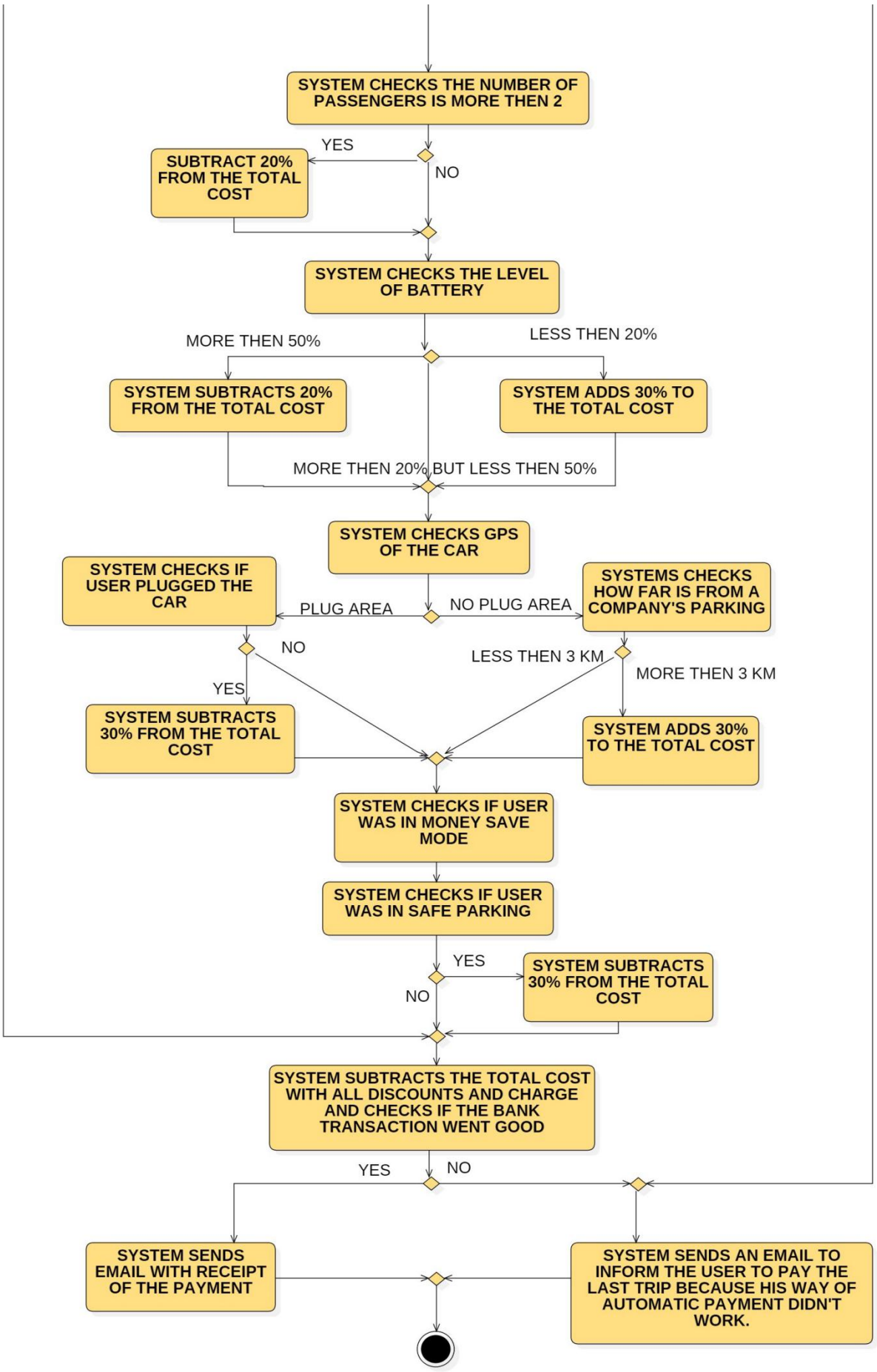

SYSTEM CHECKS THE NUMBER OF PASSENGERS IS MORE THEN 2
YES
NO
SUBTRACT 20% FROM THE TOTAL COST
SYSTEM CHECKS THE LEVEL OF BATTERY
MORE THEN 50%
LESS THEN 20%
SYSTEM SUBTRACTS 20% FROM THE TOTAL COST
SYSTEM ADDS 30% TO THE TOTAL COST
MORE THEN 20% BUT LESS THEN 50%
SYSTEM CHECKS GPS OF THE CAR
SYSTEM CHECKS IF USER PLUGGED THE CAR
PLUG AREA
NO PLUG AREA
SYSTEMS CHECKS HOW FAR IS FROM A COMPANY'S PARKING
NO
YES
LESS THEN 3 KM
MORE THEN 3 KM
SYSTEM SUBTRACTS 30% FROM THE TOTAL COST
SYSTEM ADDS 30% TO THE TOTAL COST
SYSTEM CHECKS IF USER WAS IN MONEY SAVE MODE
SYSTEM CHECKS IF USER WAS IN SAFE PARKING
YES
NO
SYSTEM SUBTRACTS 30% FROM THE TOTAL COST
SYSTEM SUBTRACTS THE TOTAL COST WITH ALL DISCOUNTS AND CHARGE AND CHECKS IF THE BANK TRANSACTION WENT GOOD
YES
NO
SYSTEM SENDS EMAIL WITH RECEIPT OF THE PAYMENT
SYSTEM SENDS AN EMAIL TO INFORM THE USER TO PAY THE LAST TRIP BECAUSE HIS WAY OF AUTOMATIC PAYMENT DIDN'T WORK.

Figure 32

5.5 State diagram

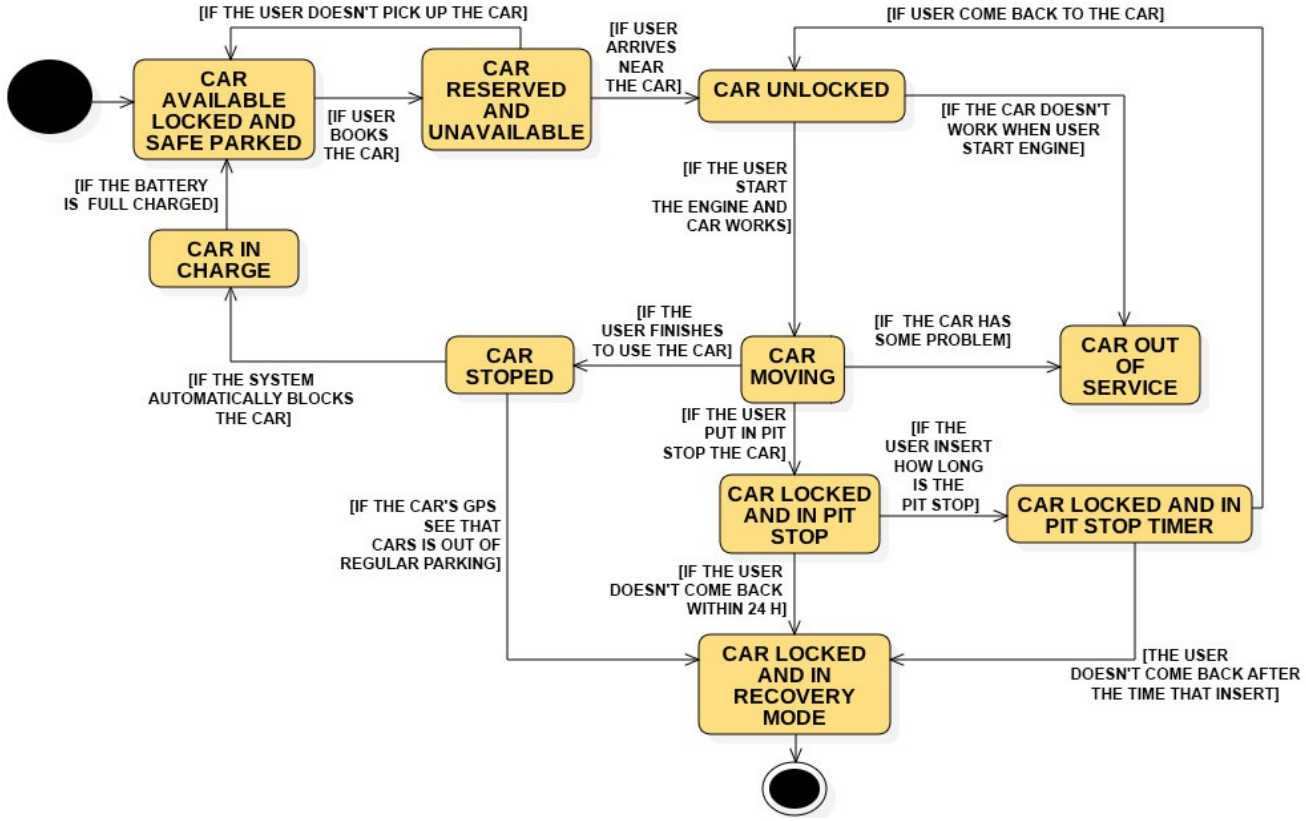


Figure 33

6 ALLOY MODELING

6.1 Model

```
open util/boolean

abstract sig State {}
sig Available extends State {}
sig Moving extends State {}
sig SafeParked extends State {}
sig PitStopped extends State {}
sig Plugged extends State {}
sig OutOfService extends State {}
```

```
sig Recovery extends State {}
abstract sig PaymentMethod{}
sig CreditCard extends PaymentMethod{}
sig IBAN extends PaymentMethod{}
sig Plug{}

sig Car{
	driver: lone User,
	passengers: Int,
	position: lone Position,
	fullCharged: Bool,
	plugged: lone Plug,
	carState: one State
}{
	passengers >= 0
	passengers < 5
	carState != none
}

sig User{
	payment: one PaymentMethod,
	position: one Position,
	reserve: lone Reservation
}

sig Position{
	latitude: Int,				//should be float
	longitude: Int		//should be float
}

sig Reservation{
	startPosition: one Position,		//safe parking
	car: one Car
}

sig Parking{
	bounds: one Position,
	slotNumber: Int,
	parkedCarNumber: Int,
}{
	slotNumber > 0
	parkedCarNumber >= 0
```

```
	slotNumber >= parkedCarNumber
}

sig PlugParking extends Parking{
	plugs: set Plug
}{
	#plugs > 0
}

sig Company {
	safeAreas: set Parking,
	cars: set Car
}{
	#safeAreas > 0
	#cars > 0
}

//==================FACTS==================

fact paymentMethodFacts {
	all pm: PaymentMethod {one u: User | pm in u.payment}
}

fact carStates {
	all c: Car, a: Available {one p: Parking | c.carState = a => (c.position =
p.bounds)		&& (c.driver = none) && (c.fullCharged.isTrue)}
	all c: Car, m: Moving {no p: Parking | c.carState = m => (c.position = p.bounds)
		or (c.driver = none) or (c.driver.position = c.position)}
	all c: Car, sp: SafeParked {one p: Parking | c.carState = sp => ((c.position =
		p.bounds) && (c.driver = none) && (c.fullCharged.isFalse))or((c.position
		= p.bounds) && (c.driver != none) && (c.fullCharged.isTrue))}
	all c: Car, ps: PitStopped {one p: Parking | c.carState = ps => (c.position !=
		p.bounds) && (c.driver != none)}
	all c: Car, pl: Plugged {one pp: PlugParking | c.carState = pl => (c.position =
		pp.bounds)}
	all c: Car, pl: Plugged | c.carState = pl => (c.driver = none) &&
		(c.fullCharged.isFalse)
	all c: Car, o: OutOfService | c.carState = o => (c.position = none) and (c.driver =
		none)
	all c: Car, r: Recovery {no p: Parking | c.carState = r => (c.position = p.bounds)
		or (c.driver != none)}
	all c: Car, o: OutOfService | c.carState != o => c.position != none
```

```
	all c: Car {one r: Reservation | c.driver != none iff r.car = c}
	all c: Car {one comp: Company | c in comp.cars}
	all c: Car | c.driver = none => c.passengers = 0
	all c: Car | c.driver != none => c.passengers > 0
	all c1,c2: Car | c1 != c2  && (c1.driver != none or c2.driver !=none) =>
		c1.driver != c2.driver
}

fact reservationFacts {
	all r: Reservation {one p: Parking | r.startPosition = p.bounds}
	all r: Reservation {one u: User | r in u.reserve}
	all r: Reservation, u: User | r in u.reserve => r.car.driver = u
	all r1, r2: Reservation | r1 != r2 => r1.car != r2.car
}

fact parkingFacts {
	all p1, p2: Parking | p1 != p2 => p1.bounds != p2.bounds
}

fact plugParkingFacts {
	all p: Plug {one pp: PlugParking | p in pp.plugs}
	all c: Car, pl: Plugged | c.carState != pl => c.plugged = none
	all c: Car{one pl: Plugged | c.carState = pl => c.plugged != none}
	all c: Car, pp: PlugParking | c.plugged in PlugParking.plugs => c.position =
		pp.bounds
	all c: Car, pl: Plugged {one p: Plug | c.carState = pl => c.plugged = p}
	all c1,c2: Car | c1 != c2 && (c1.plugged != none or c2.plugged != none) =>
c1.plugged != c2.plugged
}

fact companyFacts {
	all p: Parking {one c: Company | p in c.safeAreas}
	all car: Car {one c: Company | car in c.cars}
}

assert allReservationHaveACarWithADriver{
	all r: Reservation | r.car.driver != none
}

assert noSafeParkedPluggedCars{
	all c: Car, s: SafeParked | c.carState = s => c.plugged = none
}
```

```
pred show() {
}

check noSafeParkedPluggedCars
check allReservationHaveACarWithADriver
run show for 20 but 1 Company, 3 PlugParking, 15 Car
```

6.2 Alloy result

3 commands were executed. The results are:
#1: No counterexample found. noSafeParkedPluggedCars may be valid.
#2: No counterexample found. allReservationHaveACarWithADriver may be valid.
#3: **Instance found.** show is consistent.

Figure 34

6.3 Worlds generated

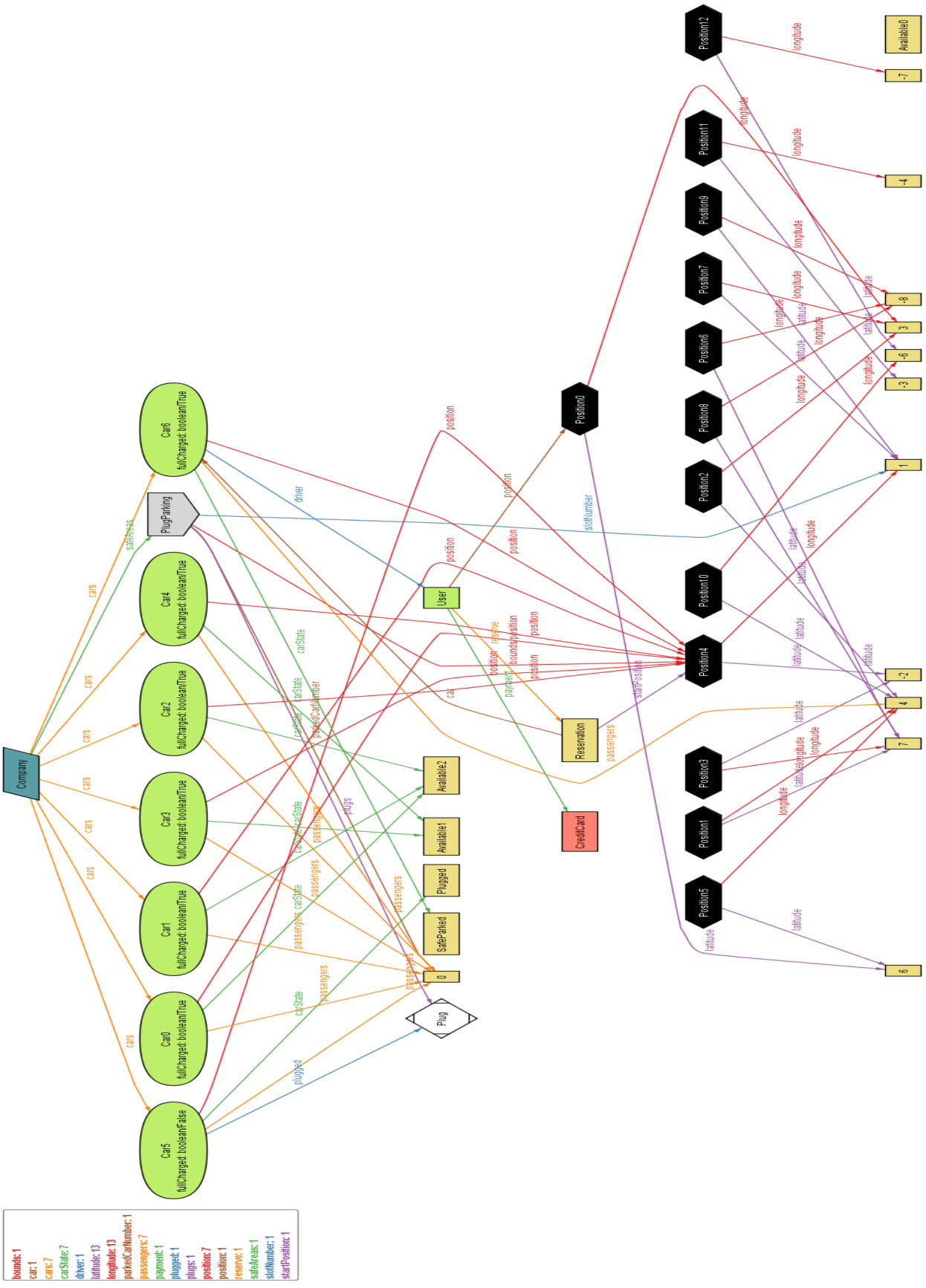


Figure 35

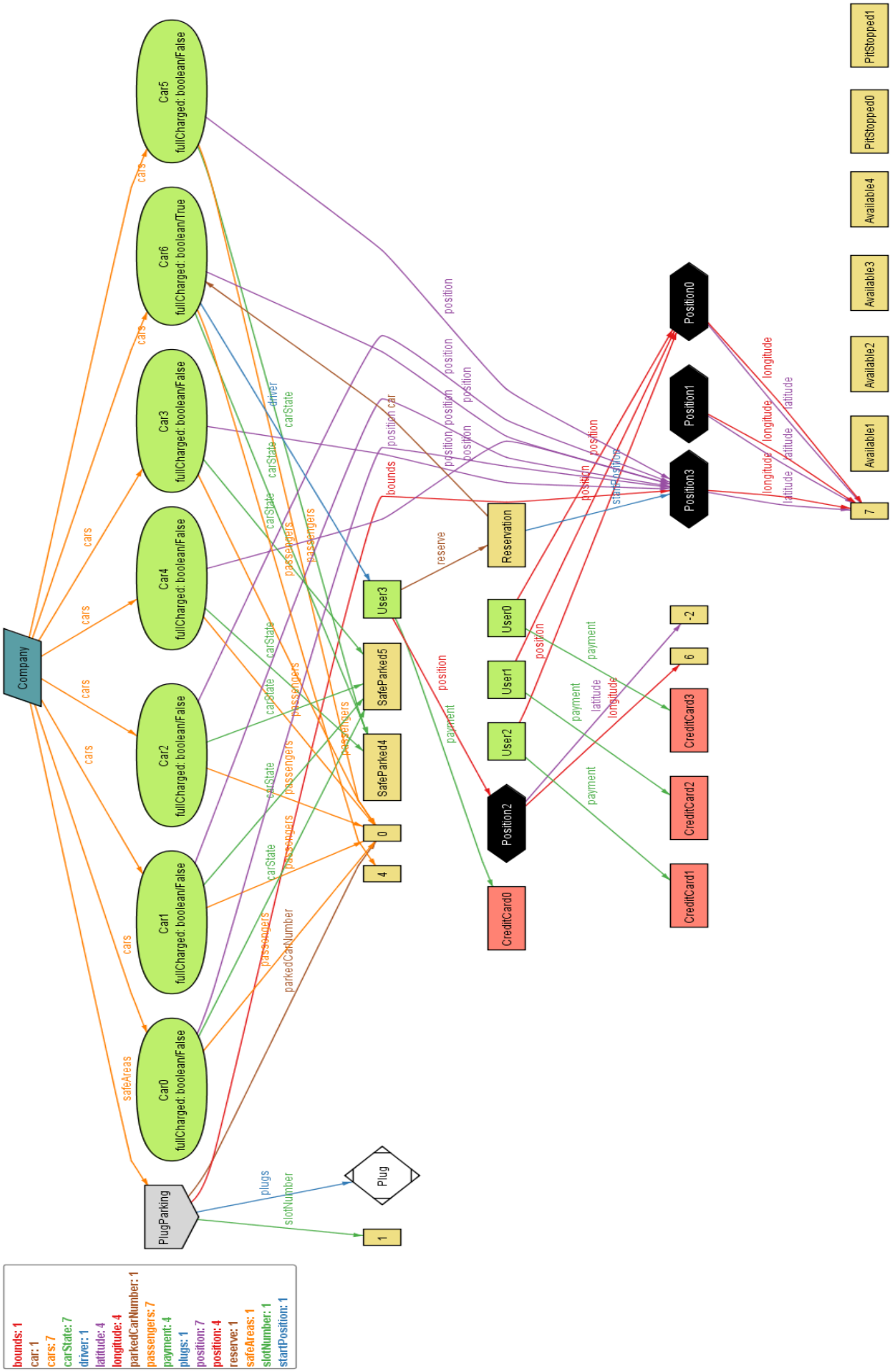

bounds: 1
car: 1
cars: 7
carState: 7
driver: 1
latitude: 4
longitude: 4
parkedCarNumber: 1
passengers: 7
payment: 4
plugs: 1
position: 7
position: 4
reserve: 1
safeAreas: 1
slotNumber: 1
startPosition: 1
Company
PlugParking
Car0
fullCharged: boolean/False
Car1
fullCharged: boolean/False
Car2
fullCharged: boolean/False
Car3
fullCharged: boolean/False
Car4
fullCharged: boolean/False
Car5
fullCharged: boolean/False
Car6
fullCharged: boolean/True
Plug
1
0
4
SafeParked4
SafeParked5
User3
Reservation
User0
User1
User2
Position2
CreditCard0
CreditCard1
CreditCard2
CreditCard3
6
-2
Position3
Position1
Position0
7
Available1
Available2
Available3
Available4
PitStopped0
PitStopped1
safeAreas
cars
plugs
slotNumber
parkedCarNumber
passengers
carState
driver
position
reserve
payment
bounds
startPosition
latitude
longitude

Figure 36

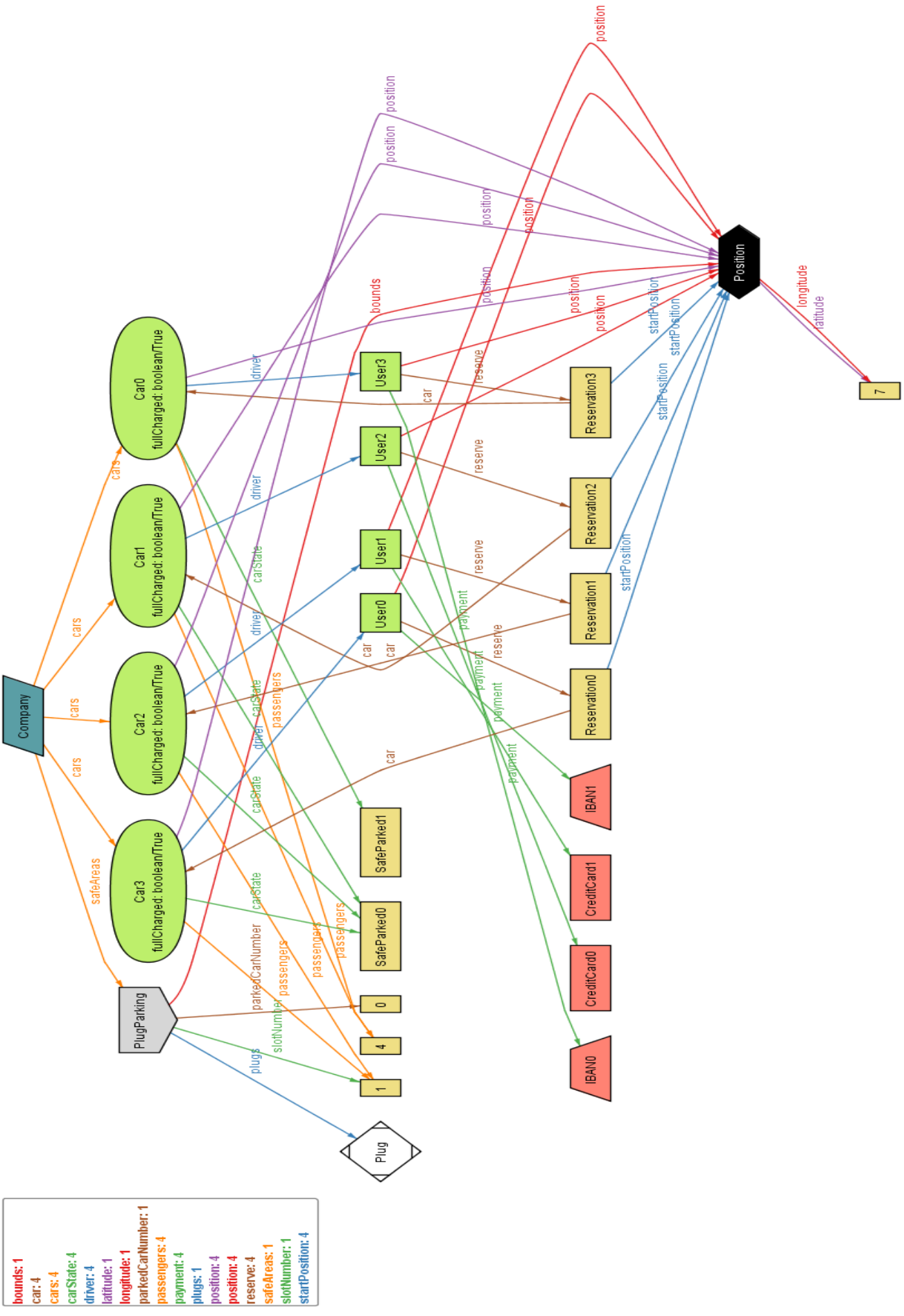

Company
PlugParking
Plug
Car0
fullCharged: boolean/True
Car1
fullCharged: boolean/True
Car2
fullCharged: boolean/True
Car3
fullCharged: boolean/True
User0
User1
User2
User3
Reservation0
Reservation1
Reservation2
Reservation3
IBAN0
IBAN1
CreditCard0
CreditCard1
SafeParked0
SafeParked1
Position
0
4
1
7
cars
safeAreas
plugs
driver
car
reserve
payment
position
bounds
carState
passengers
parkedCarNumber
slotNumber
startPosition
longitude
latitude
bounds: 1
car: 4
cars: 4
carState: 4
driver: 4
latitude: 1
longitude: 1
parkedCarNumber: 1
passengers: 4
payment: 4
plugs: 1
position: 4
position: 4
reserve: 4
safeAreas: 1
slotNumber: 1
startPosition: 4

Figure 37

8 USED TOOLS

The tools that we used to write this document are:

- WPS Writer and Microsoft Office Word: document text editors;
- Sketch: mobile apps design program, used for the mockups;
- Photoshop CC: Graphic program, used for high level graphic works;
- StarUML: UML diagram editor, used to design the UML diagrams;
- Alloy Analyzer 4.2: program for domain definition, used to verify the consistency of our model;
- GitHub: hosting website for software development, used as shared working directory.

9 HOURS OF WORK

9.1 Alice Segato

ALICE				
GIORNO	ORA INIZIO	ORA FINE	OGGETTO	ORE LAVORO
23/10	17.30.00	21.00.00	Read carefully the assignment	3.30.00
29/10	14.00.00	18.12.00	Global useCase	4.12.00
29/10	21.00.00	23.30.00	read and managed arrangements	2.30.00
30/10	7.50	9.00.00	activity diagram	1.10.00
31/10	10.00	18.00	Global useCase	8.00.00
31/10	23.00.00	0.00.00	domain assumption and userUse(	1.00.00
01/11	14.00	17.00.00	activity diagram and assumption	3.00.00
02/11	23.00.00	0.00.00	state diagram and assumption	1.00.00
03/11	15.00.00	18.00.00	activity diagram and state diagram	3.00.00
03/11	19.00.00	23.00.00	activity diagram and state diagram	4.00.00
04/11	14.00.00	17.20.00	class diagram	3.20.00
05/11	0.00.00	6.00	non functional requirments	6.00.00
06/11	14.00.00	18.30.00	non functional requirments	4.30.00
06/11	21.00.00	23.00.00	non functional requirments	2.00.00
08/11	15.00	21.00.00	non functional requirments	6.00.00
09/11	10.00.00	11.43.00	useCase and activity arrange	1.43.00
09/11	13.00	17.00	arrangements and alloy	4.00.00
10/11	16.00.00	18.00.00	car modality	2.00.00
11/11	11.00.00	13.00.00	car modality	2.00.00
11/11	21.00.00	23.00.00	app icon	2.00.00
12/11	11.00.00	21.30.00	scenarios, arrangements and rasc	10.30.00
13/11	15.00	18.00	scenarios and finish rasd	3.00.00
				TOTALE
				78.25.00

Figure 38

9.2 Mark Edward Ferrer

MARK				
GIORNO	ORA INIZIO	ORA FINE	OGGETTO	ORE LAVORO
24/10/2016	16.00	18.30.00	goals definition and started domai	2.30.00
25/10	19.10.00	20.00.00	added other things to RASD	0.50.00
29/10	11.00.00	13.30.00	Arranging chapter and started cha	2.30.00
29/10	15.20.00	17.10.00	added assumptions and use case	1.50.00
30/10	15.30.00	17.00.00	start of the class diagram	1.30.00
31/10	16.00.00	21.00.00	modifications of RASD and admin	5.00.00
01/11	13.00	17.00.00	RASD and sequence diagrams	4.00.00
03/11	10.20.00	11.20.00	done a sequence diagram	1.00.00
04/11	14.00.00	17.20.00	other sequence diagram and othe	3.20.00
06/11	15.00.00	18.00.00	writing of scenarios	3.00.00
07/11	12.00.00	13.00	Rewriting of scenarios (thanks da	1.00.00
07/11	15.00.00	16.40.00	functional requirements	1.40.00
08/11	15.30.00	19.30.00	rasd	4.00.00
09/11	15.00.00	17.30.00	alloy	2.30.00
10/11	14.30.00	16.00.00	correction of mistakes in rasd and	1.30.00
10/11	18.30.00	23.30.00	alloy	5.00.00
11/11	11.00.00	15.30.00	arrangement alloy	4.30.00
11/11	20.30.00	22.30.00	modifications to layout, alloy and	2.00.00
12/11	10.40.00	13.15.00	alloy	2.35.00
12/11	21.40.00	0.00.00	alloy revision	2.20.00
13/11	0.00.00	1.15.00	added RASD	1.15.00
13/11	15.00.00	19.00.00	finished rasd	4.00.00
				TOTALE
				57.50.00

Figure 39

9.3 Davide Bonacina

DAVIDE				
GIORNO	ORA INIZIO	ORA FINE	OGGETTO	ORE LAVORO
28/10	11.30.00	13.30.00	Arranged domain properties and	2.00.00
28/10	15.20.00	17.20.00	Arranged things and started usec	2.00.00
28/10	18.30.00	20.45.00	Created different usecases	2.15.00
29/10	15.20.00	17.10.00	Added assumptions and use case	1.50.00
01/11	18.30.00	21.00.00	Started functions of usecase	2.30.00
03/11	10.20.00	11.20.00	Sequence diagram	1.00.00
03/11	21.00.00	22.00.00	Modified description of the usecas	1.00.00
04/11	14.00.00	17.20.00	Class diagram and description us	3.20.00
06/11	14.00.00	17.00.00	Started functional requirements a	3.00.00
07/11	8.30.00	10.00.00	Finished functional requirements	1.30.00
08/11	15.30.00	19.30.00	Rasd, screen, Started alloy	4.00.00
08/11	21.00.00	22.30.00	Alloy	1.30.00
10/11	10.10.00	11.50.00	Alloy	1.40.00
11/11	14.38.00	19.23.00	Alloy	4.45.00
12/11	10.40.00	13.15.00	Alloy	2.35.00
12/11	21.40.00	0.00.00	Alloy inspection	2.20.00
13/11	0.00.00	2.40.00	Alloy optimization	2.40.00
13/11	15.00.00	19.00.00	Finished Alloy	4.00.00
				0.00.00
				0.00.00
				0.00.00
				0.00.00
				TOTALE
				43.55.00

Figure 40