PowerEnJoy
Requirements Analysis and Specification Document
Software Engineering 2 project

POLITECNICO
MILANO 1863

Authors:
Arcari Leonardo
Bertoglio Riccardo
Galimberti Andrea

Document version: 1.1

Contents

Part I
Introduction

1 System purpose

The required product is a digital management system for PowerEnJoy, a car sharing service employing exclusively electric vehicles.

The system has to provide functionalities to support both the users of the service and the employees of the company who interact with customers and cars.

Typical functions of a car-sharing service have to be supported, such as reserving a car and finding a parking area where to plug it in at the end of the rent.

A critical issue that has to be considered is the management of the electric vehicles, which have to be continuously recharged as they get used by customers and have to be distributed in a quite uniform way around the town; hence, the user has to be incentivized through discounts to recharge the car in power plugs-equipped parking areas, don't leave the car with a low battery level and park the car where there are few other available vehicles.

The product also has to provide a way to attract more potential customers to the service, therefore sharing the car with other passengers will be incentivized through appropriate discounts.

1.1 Goals

- [G.1] The *customer* shall be registered to rent a car
- [G.2] The *customer* shall be able to find cars nearby
- [G.3] The *customer* shall be able to reserve a car when he needs it
- [G.4] The *customer* shall be encouraged to pick up the car he reserved in an hour
- [G.5] The *customer* shall be encouraged to recharge the car after the rental end
- [G.6] The *customer* shall be encouraged to share a ride with other people
- [G.7] The *customer* shall be able to do a temporary stop
- [G.8] A car shall be parked in a safe area
- [G.9] A car shall not be picked without a reservation
- [G.10] A car with low or empty battery shall be recharged by a *road operator*
- [G.11] A car involved in an accident shall be reached by a *road operator*

- [G.12] A faulty car shall be repaired

2 System scope

PowerEnJoy already possesses a fleet of electric vehicles, all equipped with sensors, constantly checking the battery level and other important status information about the car components (e.g. engine, brakes, etc.), and a touch monitor to interact with the driver.

The car-sharing service provided by the company will be restricted to the city boundaries.

The company also already owns a set of parking areas around the town, and only some of them are equipped with power plugs where to charge the electric vehicles.

The company employs assistance service operators and road operators.

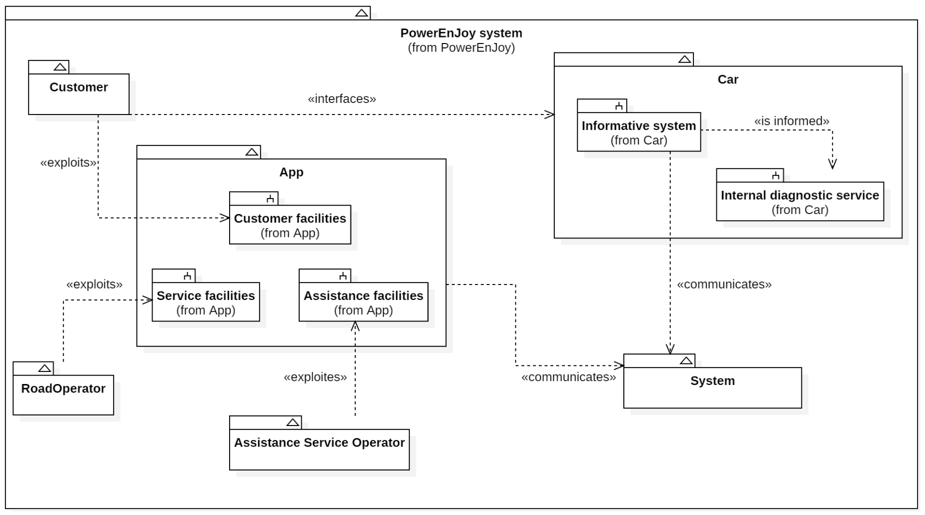


Figure 1: System context diagram

3 System overview

3.1 System context

The PowerEnJoy system has 6 major elements interacting with each other: the Customer, the Assistance Service Operator, the Road Operator, the App, the Car and the System.

The Customer is a person registered to the service that is willing to rent a car. They exploit the App customer facilities in order to reserve a car, receive guidance to reach it and unlock it.

The Assistance Service Operator is a company employee responsible for responding to Customers assistance requests and to Cars failures. They have full access to the System facilities in order to remotely control Cars and dispatch Road Operators when on-field presence is required.

The Road Operator is a company employee responsible for granting the cars availability. They exploit the App back office facilities in order to be notified of a car fault or low battery level, receive guidance to reach it and restore its correct behavior.

The App is the front-end of the PowerEnJoy system. Depending on its user, the features it provides change. For a Customer, the App represents the only way to handle car reservation, unlock and managing personal information. For a Road Operator, the App represents a way of being helped in accomplishing their task, from locating a faulty car to receive a deeper acknowledgment of the fault diagnosis.

The Car is the set of internal car sensors and on-board informative system. The car status is constantly checked and in case of an anomaly the System is immediately notified. In addition a touch screen Monitor is available to let the interaction with the Customer.

The System is the set of all the data and business logic facilities. It has to be interpreted, at this level, as an abstraction of all the background tasks to keep the system reliable and consistent.

3.2 System functions

The PowerEnJoy system provides the following functionalities:

- Users are able to register to the system by providing their credentials and payment information. They receive back a password that can be used to access the system.
- Registered users, now known as Customers, are able to find the locations of available cars within a certain distance from their current location.
- Among the available cars in a certain geographical region, Customers are able to reserve a single car for up to one hour before they pick it up.
- If a car is not picked-up within one hour from the reservation, the system tags the car as available again, and the reservation expires; the Customer pays a fee.
- A Customer that reaches a reserved car is able to tell the system they are nearby, so the System unlocks the car and the Customer may enter.
- As soon as the engine ignites, the System starts charging the Customer for a given amount of money per minute; the Customer is notified of the current charges though a Monitor on the Car.
- The System stops charging the Customer as soon as the car is parked in a safe area and the Customer exits the car; at this point, the System locks the car automatically.
- The set of safe areas for parking cars is pre-defined by the management system.
- Once an Assistance request has been opened, the System notifies the nearest road operator to the involved car.

In addition to the functionalities above, the System should incentivize the virtuous behaviors of the Customers:

- If the System detects the Customer took at least two passengers onto the Car, the System applies a discount of 10% on the last ride.
- 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.

- If a Car is left at special parking areas where they can be recharged and the Customer takes care of plugging the car into the power grid, the System applies a discount of 30% on the last ride.
- If a Car is left at more than 3 KM from the nearest power grid station or with more than 80% of battery empty, the System charges 30% more on the last ride to compensate for the cost required to re-charge the Car on-site.
- If the Customer enables the money saving option, they can input their final destination and the System provides information about the station where to leave the car to get a discount. This station is determined to ensure a uniform distribution of Cars in the city and depends both on the destination of the user and on the availability of power plugs at the selected station.

3.3 User characteristics

- Customer
- Road operator
- Assistance service operator

4 Definitions

Account password

The password chosen by the customer at the time of registration to PowerEnJoy. The account password is required to log into the mobile app, web site and manage their account information and payment options.

Assistance request

An Assistance request is opened by an Assistance service operator when a car is faulty. Once an Assistance request has been opened the System notify the nearest operator to the involved car. The Assistance request contains detailed information about the failure and also the current position of the car.

Assistance service operator

A PowerEnJoy employee that has full control on the system, interacts with the customer to provide remote assistance and dispatches road operators when needed.

Car

A 5-door electric-powered vehicle, equipped with appropriate sensors acknowledging the company of information about the Car. They include but not limits to:

- *Human presence sensors* measure the number of passengers aboard
- *GPS* retrieves latitude and longitude of the Car
- *Rotational sensor* retrieves rotation degree of the Car front
- *3G module* provides Internet connection to the Car

Customer

A person who wants to rent a PowerEnJoy car and is therefore registered to the service.

Fault

Failure of some nature (e.g. mechanical ,electrical) in a car.

PIN

A numerical key provided by the system needed to unlock the car. The customer receives the PIN by email in the registration phase.

Road operator

A PowerEnJoy employee, working in a specific repair center, responsible to react to issues related to cars availability. Their job includes getting to the faulty or discharged cars' location and taking them to the closest special parking area or to the repair center.

Safe area

A parking area owned by the company providing a set of parking spots, located in a certain zone.

Special parking area

A safe area equipped with power grid to charge cars.

System

The set of all the back-end facilities that the service is composed of. That includes, but not limited to, data bases, business logic and signal handling.

Touchscreen Monitor

A touchscreen display located in every car to provide the user a convenient way to interact with the PowerEnJoy service during the rental.

Zone

A bounded geographical area of the town. Each zone is provided with a repair center.

Part II
System requirements

5 Functional requirements

5.1 Customer

[U.1] Register to the service

- [R.1] When the *customer* starts to interact with the service the system shall provide a login/register area.
- [R.2] When the *customer* enters into the registration area the system shall provide a form to compile.
- [R.3] The system shall confirm the registration if and only if the data provided are valid and the *user* isn't registered yet.
- [R.4] When the *customer* enters into the login area the system shall allow the client to insert their email and password and redirect them to the personal area if and only if the password is correct.
- [R.5] When the *customer* registers to the service the system shall send to the customer an email with a confirmation link that expires in 7 days by the moment it is created.
- [R.6] When the *customer* confirms the registration clicking on the confirmation link the system shall send to the *customer* an email containing a valid PIN to unlock the car.

[U.2] Reserve a car

- [R.7] When the *customer* with no reservation in progress opens the app, it shall show them a city map with available cars geographical position on.
- [R.8] When the *customer* taps on *list* button the app shall show a list of all available cars sorted by the distance from *customer*'s location.
- [R.9] When the *customer* types their desired destination the app shall filter the available cars showing only whose battery charge level is enough to reach the input destination.
- [R.10] When the *customer* taps on an available car icon the app shall show the car status, which involves license plate, battery charge level, number of km available and distance from the *customer*'s location.
- [R.11]When the *customer* taps on an available car icon the app shall show a button to reserve a car.

- [R.12] When the *customer* taps on the *reserve car* button the system shall make the car unavailable to other *customers* for at least 1 hour.
- [R.13] When the *customer* taps on the *reserve car* button the system shall make the car unlock-able from the *customer*'s *PIN* only for maximum 1 hour.

[U.3] Car reservation expires

- [R.14] When the *customer* doesn't pick up the car in an hour the system shall delete the reservation and charge the *customer* a fee.

[U.4] Drive a rented car

- [R.15] When the *customer* with a reservation in progress opens the app, it shall show them the reserved car status which involves license plate, battery charge level, number of km available and distance from customer location.
- [R.16] When the *customer* with a reservation in progress opens the app and they are within 10 meters from the car, the app shall show them a button to unlock the car.
- [R.17] When the *customer* taps the *unlock car* button the app shall prompt them to type their *PIN.*
- [R.18] When the *customer* types their *PIN* and it matches with the one sent during the registration process the car shall unlock.
- [R.19] When the *customer* ignites the engine the Car shall send a message to the System saying that the ride has started.
- [R.20] When the System receives a message from the Car saying that the ride has started, the System shall start charging the *customer* according to the PowerEnJoy company payment policies.
- [R.21] When the *customer* is driving a car the Monitor shall show their location.
- [R.22] When the *customer* is driving a car the Monitor shall show the safe areas nearby.
- [R.23] When the Car is parked in a safe area and the car windows are closed and the engine is switched off, the Monitor shall show a *End rental* button.
- [R.24] When the *customer* taps the *End rental* button the Car shall send a message to the System saying that the rent is over.
- [R.25] When the System receives a message from the Car saying that the ride is over, the System shall contact the payment service provider charging the user according to the PowerEnJoy company payment policies.

[U.5] Park and plug the car in

- [R.26] When a *customer* presses the *End rental* button on the touch screen monitor the system shall make the car available to rent.
- [R.27] When a *customer* plugs a car the system shall apply a discount on the last ride fee.

[U.6] Temporarily park the car

- [R.28] When a *customer* press the *Temporary stop* button the system shall continue to show the car unavailable to rent.
- [R.29] When the car is parked in a safe area and the car windows are closed and the engine is switched off, the monitor shall show a *Temporary stop* button.

[U.7] Share a ride

- [R.30] When the system is notified by a car that there are two or more passengers aboard it shall apply a discount on the ride fee.

[U.8] Managing a failure

- [R.31] When the system is notified by a car of a failure it shall stop to charge the *customer*

5.2 Road operator

[U.8] Manage a failure

- [R.36] When an *assistance request* has been opened the system shall notify the *road operator* nearest to the car position.
- [R.37] When the *road operator* accepts the *assistance request* clicking on the *"Accept"* button the system shall mark the *assistance request* as accepted.

[U.9] Recover a car involved in an accident

- [R.32] When a car detects an accident, it shall notify the system of this event.
- [R.33] When the system is notified of an accident, in turn it shall notify an *assistance service operator.*
- [R.34] When a *road operator* doesn't accept an *assistance request* in 1 minute the system shall send the same request to another *road operator* nearest to the accident position.

- [R.35] When multiple *road operators* are notified of an accident and one of them accept the *assistance request* the system shall delete all other notifications in order to ensure that only one *road operator* is in charge of the operation.

[U.10] Recharge a car with low battery

- [R.38] When a car has been left with less than 10% of charge in a safe area without power plugs and hasn't been used for more than 4 hours the system shall notify the nearest *road operator* .
- [R.39] When a car unavailable for rent due to the low charge of battery is plugged, the system shall make the car available to rent.

5.3 Assistance service operator

[U.8]Manage a failure

- [R.41] When a car notifies the system that it has a failure, the system shall notify an *assistance service operator* about this failure.
- [R.42] When a car notifies that it's faulty, the system shall make it unavailable to rent.
- [R.43] When the *road operator* click on the *Solved* button in the context of an *assistance request* the system shall mark the *assistance request* as closed.

[U.9]Recover a car involved in an accident

- [R.40] When an *assistance service operator* dispatches a *road operator* to an accident location the system should notify the *road operator* nearest to the accident position.

6 Performance requirements

- The reservation service shall be available 99.999% of the time.
- The Car shall unlock within 10 seconds from the *customer*'s *PIN* input.
- The rental shall end within 5 seconds from monitor *End Rental* button tap.
- The car shall lock after 15 seconds from the door closing.

7 System security

No security concern have been taken into account in the requirement analysis. This is a lack of the system specification, although the knowledge of the requirement analysis team was not enough to produce a valid option. A consultancy with field experts is due.

8 System interface

8.1 Internal interfaces

System communication The interaction showed in the system context between the App and the System and between the Car and the System happens over the internet. The System exposes a set of APIs that App and Car informative system exploit in order to communicate user intentions, status updates and perform requests. A cross-platform cross-language communication protocol is due in order to face the PowerEnJoy heterogeneous system.

Human interaction A human being interacts with the PowerEnJoy system in the following ways:

- through the App
- through the sensors inside the Car that perceive human presence
- through the touch screen of the informative system inside the Car

8.2 External interfaces

The system to be developed has to interface with external systems to adequately provide all the required functionalities.

Payment service Billing operations are dealt by a payment service provider through a convenient communication protocol.

The payment service provider verifies if the customer's credit card is valid and manages the payment of rental fees and additional charges.

Geo-localization service An external geo-localization service provides a map of the city, complete with localization of the safe areas and real time localization of the cars, and distances calculation.

Car internal diagnostic system The cars manufacturer provides a car internal diagnostics system that signals eventual failures, the battery level, the locking status of doors and windows, and the presence of passengers through sensors.

8.3 App mockups

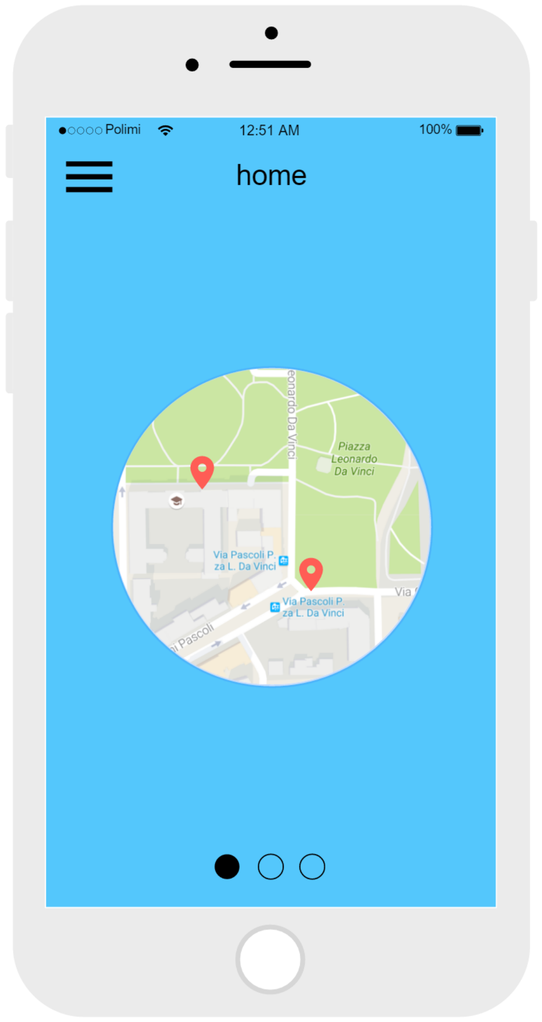
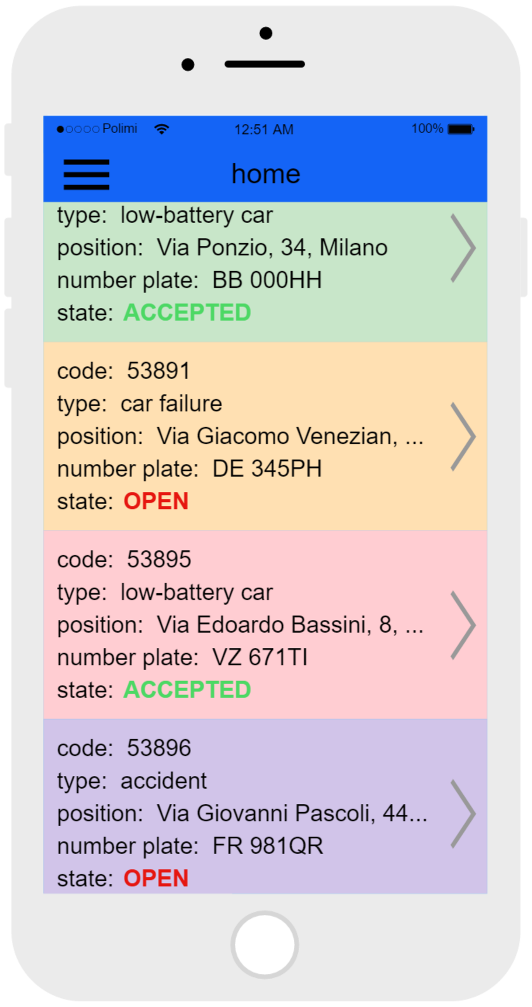


Figure 2: *Home* view



type: low-battery car
position: Via Ponzio, 34, Milano
number plate: BB 000HH
state: ACCEPTED

code: 53891
type: car failure
position: Via Giacomo Venezian, ...
number plate: DE 345PH
state: OPEN

code: 53895
type: low-battery car
position: Via Edoardo Bassini, 8, ...
number plate: VZ 671TI
state: ACCEPTED

code: 53896
type: accident
position: Via Giovanni Pascoli, 44...
number plate: FR 981QR
state: OPEN

Figure 3: *Road Operator Home* view

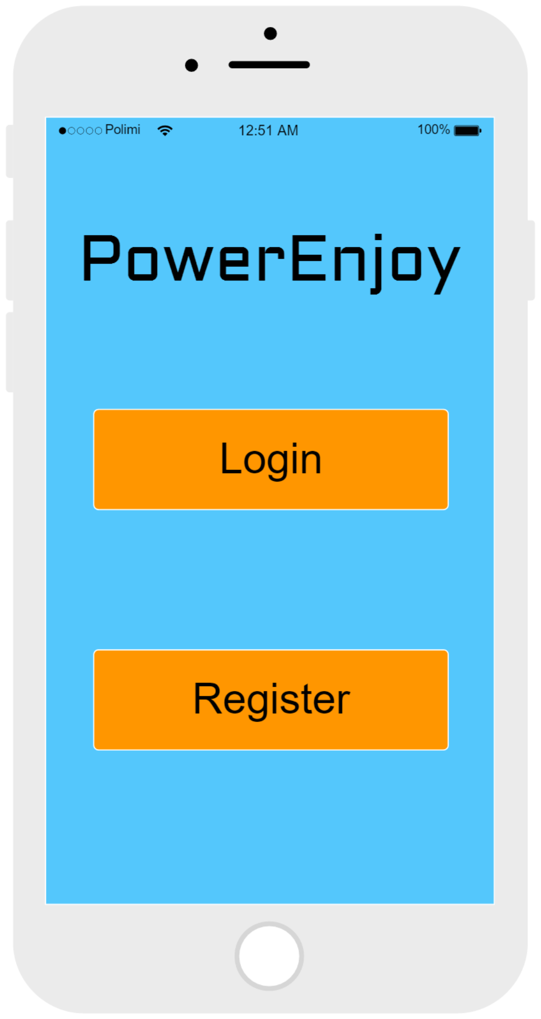


Figure 4: *Login-register* view

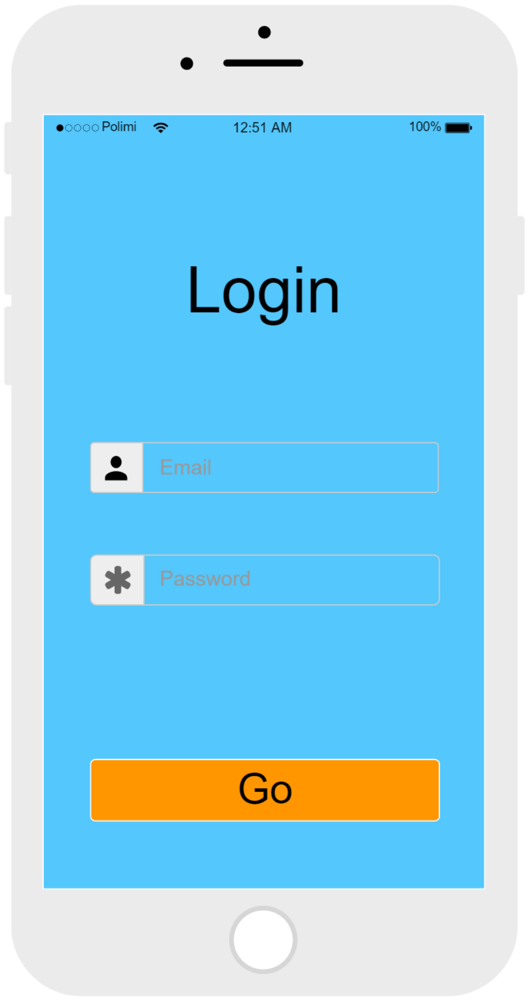


Figure 5: *Login* view

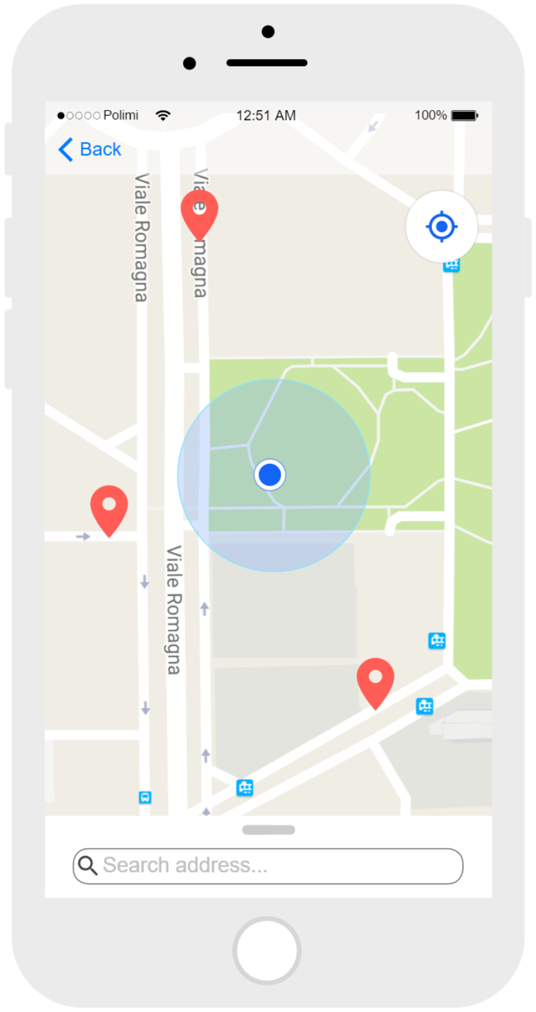


Figure 6: *Map* view

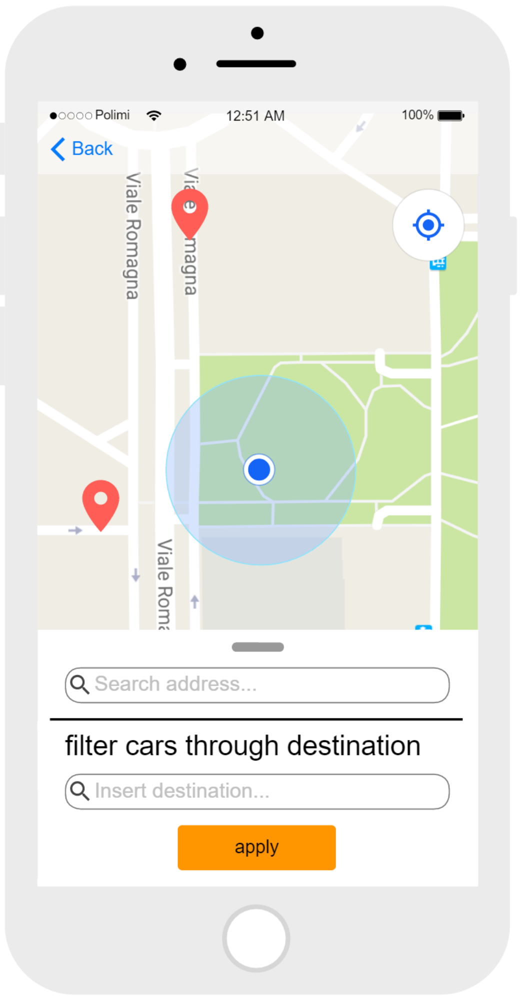


Figure 7: *Map with filter* view

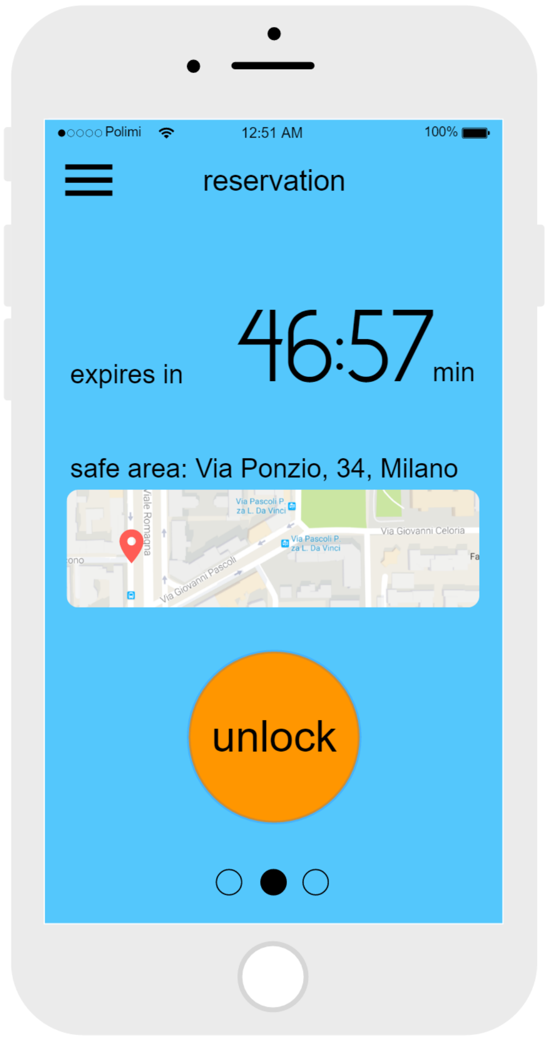


Figure 8: *Reservation* view

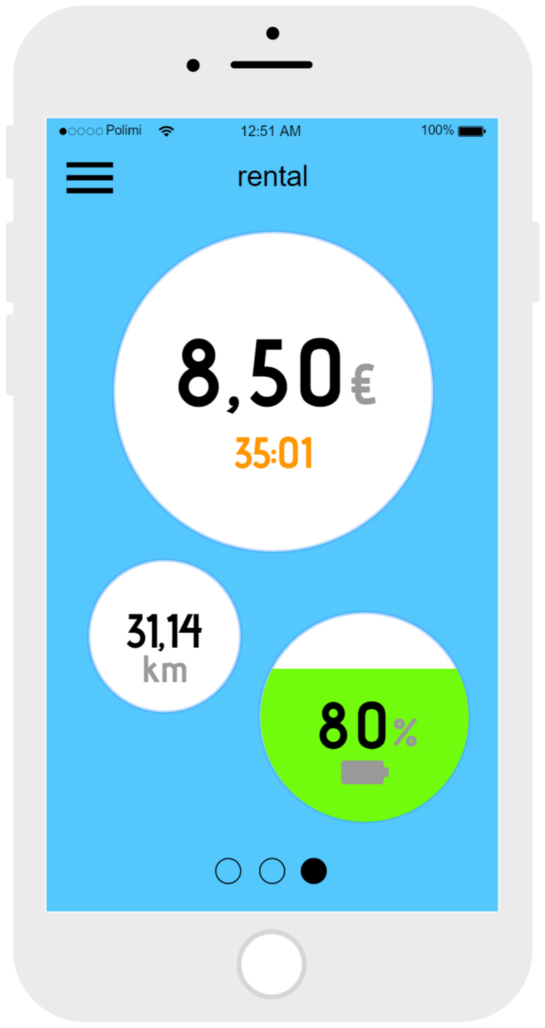
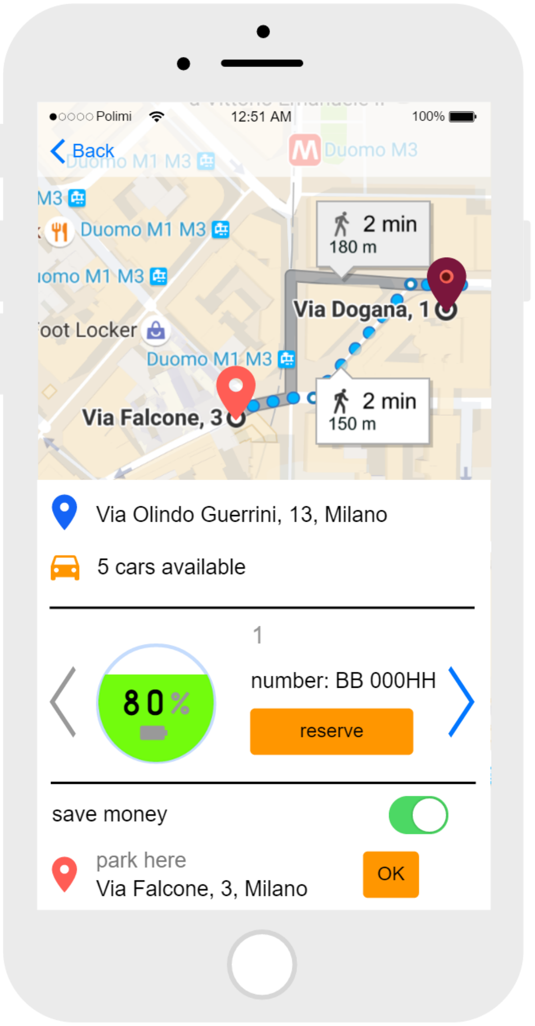


Figure 9: *Rental* view



Via Olindo Guerrini, 13, Milano

5 cars available

1

80%

number: BB 000HH

reserve

save money

park here

Via Falcone, 3, Milano

OK

Figure 10: *Money saving option* view

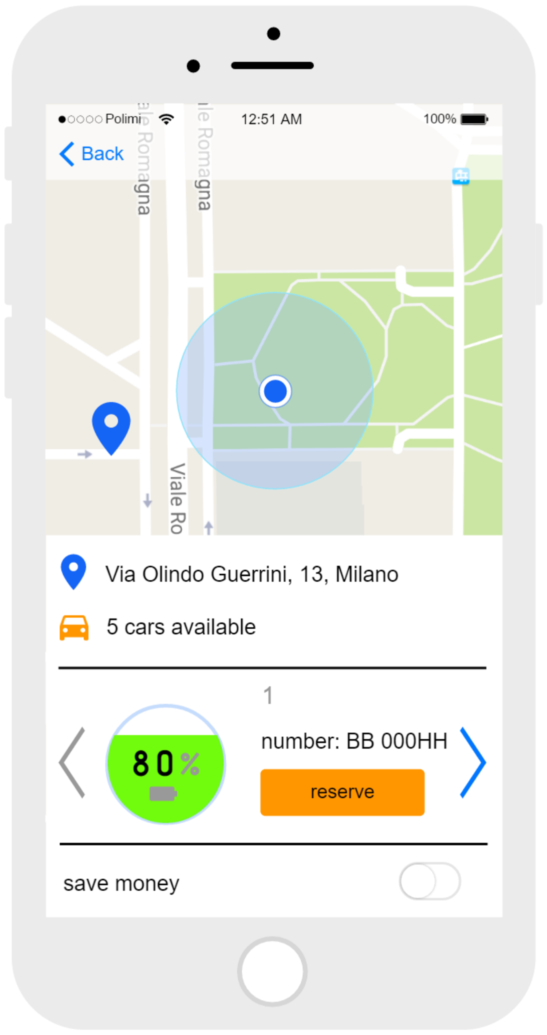
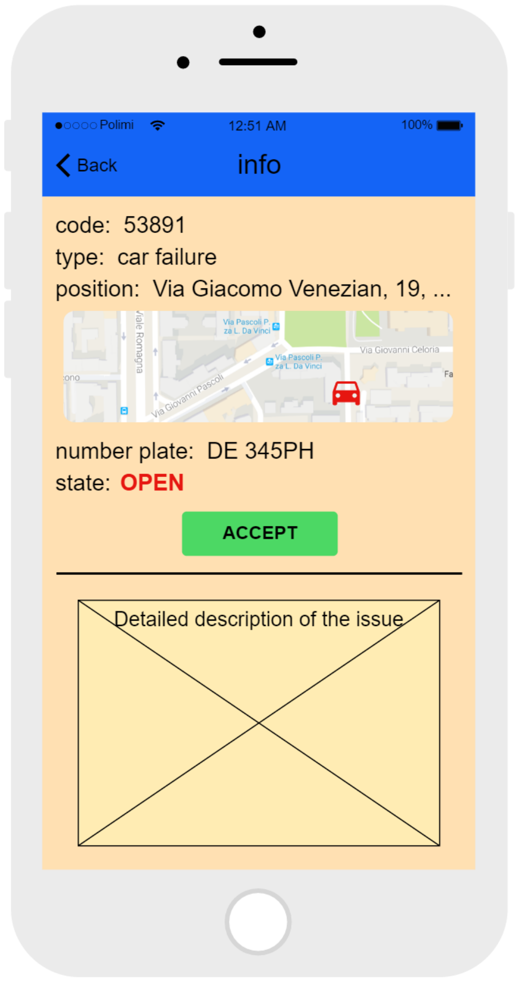


Figure 11: *Pick a safe area* view



●○○○○ Polimi 12:51 AM 100%

Back info

code: 53891

type: car failure

position: Via Giacomo Venezian, 19, ...

number plate: DE 345PH

state: **OPEN**

Figure 12: *Failure detailed description* for Road Operator view

Part III
Appendices

9 Scenarios

9.1 Alice registers to the service

Alice is interested in using the PowerEnjoy service and so she downloads and installs the app on her smartphone.

She fills in all the required fields of the credentials form: first and last name, date and place of birth, gender, residence, fiscal code, ID card number and driving license number. Once she has inserted all the required credentials, she inserts her e-mail address and a password to access the personal area. Alice then provides payment information inserting her MasterCard credit card number, the security code and the expiration date. Then Alice checks the box of the Terms of service in order to agree with them. Finally she clicks on the *Register* button.

After verifying the correctness of the inserted data the system sends Alice an e-mail with a link that she clicks on to confirm her registration. Thereafter, the system sends Alice another e-mail containing the PIN she will later use to unlock the cars she will rent through the PowerEnJoy service.

9.2 Alice experiences a plain ride

Alice needs a car to drive to a restaurant in Milan and she decides to rent a car. Therefore, she enters the PowerEnJoy app and logs into the service with her e-mail address and account password. Within the app, Alice can see cars available nearby, each characterized by the license plate, the battery charge level and the max estimated traveling distance. Alice can filter nearby cars by choosing her desired destination so that the app will show her only those having enough power charge. She rents a car located 200m from her with a 80% battery level and a max travel estimated to 50 km; the system reserves her the car for 1 hour. Alice walks towards the car for 5 minutes and she unlocks it through the app by tapping on the *Unlock Car* button and typing her *PIN*. Alice enters the car and presses a button to ignite the engine. Alice drives for a 20km trip, after which she arrives at a safe area, with no power grids, close to her destination and parks the car. On the monitor, Alice can see the fee she has to pay, aside from any discount or additional charges, which is proportional to the rental duration. Therefore Alice closes the left window she had opened before, switches off the engine, taps the *End Rental* button on the monitor and exits the car. The system locks the car and makes it available to other users. Alice left the car with a 40% battery level in a safe area 2km far from the closest power grid safe area, so she does't get any discount or additional charge. Alice gets charged for the rental fee.

9.3 Alice doesn't retire the rented car in time

Alice rents a car through the PowerEnJoy app and the system reserves it for an hour. However, Alice can't get to the car in time because of an unexpected event, thus the system re-locks the car and makes it available to other users. The system charges Alice a fee of 1€ for reserving the car and not using it.

9.4 Alice plugs in the car after parking

Alice rents a car through the app as usual and, getting nearby her destination, she decides to close the rent. Instead of parking the car in a common safe area, she decides to drive the car to be closest safe area equipped with a power grid. Arrived there, Alice parks the car and, after closing all the doors and windows and exiting from the car, she plugs the car into the power grid to recharge it. Therefore, Alice gets an additional 30% discount on the charged amount for the rent.

9.5 Alice shares the car with two friends

Alice has rented a car and she is driving around Milan. Alice stops at her friend Bob's house to pick him and his brother Charlie. Through sensors inside the car, the system acknowledges that there are two passengers inside the car. When they arrive to destination, Alice parks the car leaving it with a 60% battery charge in a safe area 2km away from the closest power grid-equipped safe area; therefore, Alice gets a 20% discount on the charged amount. Also, because she shared the ride with her two friends, she gets an additional discount of 10% of the non-discounted amount; the two discounts are additive, so she gets an overall discount of 30% of the rent fee.

9.6 Alice runs out of battery

Alice is driving the car she rented but, unfortunately, she is not paying attention to the battery status of the car. After reaching the 10% threshold she hears a warning sound and the car's monitor displays advice to stop at the closest safe area, of whom are shown on the map the directions to reach it. Relentlessly, Alice keeps driving towards her destination when, suddenly, the car stops working. Alice is stuck in the middle of the road. She exits the car and contacts the assistance service through the app. The system locks the car if it can establish a communication with it. The car is made unavailable to be rented by other users. Alice then pays a fee of 500€, additionally to the amount due for the duration of the travel before the car ran out of charge.

9.7 David recovers a car involved in an accident

David works as a road operator for PowerEnJoy. On the smartphone provided by the company, he gets notified through their road assistance app about failures and accidents involving the cars. He can locate all the cars in his assigned zone and see their current battery level and if they need any maintenance. While he is waiting for some work to do at his zone's repair center, he receives a notification about a car involved in an accident 1.5km away from his position. Assisted by the directions provided by the road assistance app, he gets to the car with his tow truck to recover it and take it at the repair center to be fixed.

9.8 David takes a car to the closest power grid-equipped safe area

David gets notified through the road assistance app that a car has a low battery level ($<$10%) and hasn't been picked up by any user for some time, hence the system has made it unavailable to rent. Therefore, he locates the car and gets to its position with his company car. Once there, he unlocks the low battery car through the app, then drives it to the closest power grid-equipped safe area with an available plug, as indicated by the directions provided inside the app. Once arrived, David exits the car and plugs it into the power grid; the system locks the car and makes it available once again to rent.

9.9 Internal diagnostic system detects a failure

Alice is driving the car she rented but suddenly the engine stop working and the check engine light is illuminated. The car signals the failure to the Assistance Service. An operator of the assistance department dispatches a road operator to take the car to the closest repair center. Alice is notified on the screen that an operator has been dispatched. Then she has to leave the car, which gets locked by the system.

9.10 Alice temporarily parks the car during the rent

Alice has rented a car and is driving it around the town. She has to get groceries for dinner, so she gets to the closest safe area to the supermarket and parks the car in available spot. Alice presses the *Temporary stop* button on the touch monitor inside the car and exits the car. The car gets locked but the rent is still ongoing. When the shopping is over, Alice returns to the car, unlocks it through the app and resumes the ride. At the end of the rental, Alice will also be charged for the time the car was parked while she was at the supermarket.

9.11 Alice parks in area more than 3 km away from the closest power grid-equipped safe area

Alice is driving the car she rented and now needs to park it. She picks the nearest safe area to her but unfortunately this area is far more than 3 km to the closest special parking area. The system detects the safe area where Alice has parked the car and charges a 30% additional amount on the ride.

9.12 Alice activates the Money Saving Option

Alice has activated the Money Saving Option. She decides to rent a car and so she inserts her destination in the app. The app provides Alice a specific parking area near to the destination she has chosen. Alice drives the car and parks it in the parking area suggested by the app, so she gets a 30% discount on the ride fee.

9.13 Alice leaves a car with more than 80% battery empty in a non grid-equipped safe area

Alice is driving the car she rented and now needs to park it. She picks the nearest safe area to her because she is in a rush but unfortunately this area isn't equipped with a grid to plug the car into. The system detects that the car has a 10% remaining charge and it is not plugged in, hence it charges Alice a 30% additional amount on the ride.

10 Use cases

10.1 [U.1] Register to the service

Participating actors: Customer

Entry conditions: A potential *customer* hasn't yet registered to the service

Flow of events:

- The *customer* registers to the service through the website or the mobile app. They compile a form with the requested data:
  - Personal data
    - First name and last name
    - Date and place of birth
    - Gender
    - Residence
    - Fiscal code
    - ID card number
    - Driving license number
    - Mobile phone number
  - Data to access the personal area
    - Email address
    - Password
  - Payment details
    - Credit card number
    - Security code
    - Expiration date
- The *customer* accepts the ToS (Terms of service) agreement checking the related box
- The *customer* clicks on the *Register* button.
- The system sends the *customer* an email to confirm the registration
- The *customer* clicks on the link contained in the email

Exit conditions: The *customer* receives an e-mail containing a PIN that is unique and allows to recognize them.

Exceptions:

- Some information are not valid (e.g. the e-mail address or the credit card number), hence the system asks the *customer* to insert again these data.

- The *customer* exits the registration process while in the middle of it, and the system doesn't keep a copy of the data.
- The *customer* doesn't confirm the registration clicking on the link received via e-mail within 7 days, hence the link expires and the user has to repeat the procedure from the beginning.

10.2 [U.2] Reserve a car

Participating actors: Customer

Flow of events:

- The *customer* launches the PowerEnJoy mobile app on their device and logs in typing their e-mail address and account password and tapping *Login* button.
- The system will validate the input credentials. In case they match the mobile app will display a view of the city map, customer current location and available cars geographical position. The *customer* can choose a car in the following ways
  - by tapping on the desired car icon showed on the map
  - by switching to a list view displaying all the available cars in a list, sortable for distance from customer location or battery charge level
  - by filtering the available cars to those with enough battery charge to reach the desired destination.
- After a car is chosen, the app will show details about license plate, battery charge level, number of km available and distance from *customer* location. The *customer* taps *Rent* button and the system reserves the chosen car to the *customer* for 1 hour. The *customer* will have to pick the car up within 1 hour.

Exit condition:

- The car is reserved to the *customer*
- The car is no longer available for reservation for at least 1 hour

Exceptions:

- The *customer* forgot their credentials. Consider a password recovery use case.
- The *customer* does not pick the car up within 1 hour. The reservation expires. They are charged a fee for the expired reservation and the car is made available again.

Special requirements:

- The reservation system availability must equal to 99,999%. A *customer* cannot reserve a car if and only if no free cars are available.

10.3 [U.3] Car reservation expires

Participating actors: Customer

Entry conditions: The *customer* has reserved a car through the app and hasn't picked it up in less than 1 hour

Flow of events:

- The system cancels the *customer*'s reservation
- The system notifies the *customer* that they haven't unlocked the car in time and their reservation is canceled
- The system charges the *customer* a fee

Exit␣conditions: The system makes the car available for rent to other *customers*

10.4 [U.4] Drive a rented car

Participating actors: Customer

Entry conditions: The *customer* has reserved a car for no longer than 1 hour

Flow of events:

- The *customer* walks to the reserved car and launches the PowerEnJoy mobile app. Since they have already logged in, the app will show a rental view. Because the *customer* is in proximity of the car, the *Unlock Car* button in the app is enabled and the *customer* taps it.
- The *customer* is prompted to type in the PIN. The system checks whether the PIN matches the *customer* one, they do and the car unlocks.
- The *customer* enters the car, removes the keys from their spot located next to the monitor and plugs them in and ignites the engine. From now on the system starts charging the *customer*. They drive to the destination.
- During the travel the monitor shows the fees the *customer* will pay and, in a city map view, the car and safe areas locations. The *customer* parks in a safe area close to their destination.
- In order to end the rental, the *customer* checks whether all the windows are close and switches off the engine. Then they tap *End Rental* button on the monitor and exit the car. The ride is over. The system is acknowledged that the car is not plugged in, the battery level is lower than 50% but higher than 20%, therefore no discount or overcharge are applied to the *customer*. The system contacts the payment service provider to charge the *customer* for rent.

Exceptions:

- The *customer* cannot unlock the car through the mobile app, due to internet connectivity or app issues. They call the assistance service to have the car unlocked remotely.
- The *customer* forgot their PIN.
- The *customer* exits the car without going through the rental ending procedure. If the system detects no presence inside the car, the stop mode is not toggled and the car engine is switched off from 5 minutes, the rental ending procedure is automatically done. The *customer* is charged as usual and 5 minutes are added to their actual traveling time.

Special requirements:

- The car must unlock within 10 seconds from the *customer*'s PIN input
- The rental must end within 5 seconds from monitor *End Rental* button tap
- The car must lock after 15 seconds from the door closing.

10.5 [U.5] Park and plug the car in

Participating actors: Customer

Entry conditions: A *customer* has rented a car and wants to park it in a special parking area

Flow of events:

- The *customer* parks the car in an available spot of the special parking area
- The *customer* presses the *End rental* button on the touch screen monitor
- The *customer* exits the car
- The car locks itself and closes all the windows
- The *customer* plugs the car into the power grid

Exit conditions:

- The system makes the car available to rent by *customers*
- The *customer* gets a discount on the charged amount for the rent

Exceptions:

- If there aren't any available plugs in the special parking area, the *customer* has to look after another special parking area if they want to get a discount
- If the car is left in the special parking area without plugging it, the *customer* doesn't get any discount

10.6 [U.6] Temporarily park the car

Participating actors: Customer

Entry conditions: A *customer* has rented a car and has to stop while keeping the car for later use

Flow of events:

- The *customer* parks the car in an available spot of a safe area
- The *customer* presses the *Temporary stop* button on the touch monitor inside the car
- The *customer* exits the car
- The car locks itself and closes all the windows
- The system keeps the car unavailable to rent by other users
- After some time, the *customer* is back at the car and have unlocked it

Exit conditions:

- The *customer* resumes their ride
- At the end of the rent, the *customer* gets charged also for the time the car was parked during the stop

10.7 [U.7] Share a ride

Participating actors: Customer

Entry conditions: The *customer* has rented a car and is driving it around

Flow of events:

- During the ride, the *customer* stops the car to take aboard two other people, at the same time or in two different times
- The car senses that there are passengers aboard, and how many there are in the car
- The car constantly informs the system about the number of passengers aboard

Exit conditions:

- At the end of the rent, the *customer* will get a discount on the charged amount for sharing the ride with at least two people

10.8 [U.8] Manage a failure

Participating actors: Customer, Assistance service operator, Road Operator

Entry conditions: A fault happens to the car

Flow of events:

- The car notifies the *customer* about the failure, giving them information about what went wrong through the touch monitor
- The *customer* exits the car
- The car locks itself and closes all the windows, if possible
- The system makes the car unavailable to rent
- The car also notifies an *assistance service operator* about the failure, sending a complete failure report and the last recorded position in case the GPS is not working any more
- The *assistance service operator*, after having tried to interact with the customer, opens an *assistance request*
- Once an *assistance request* has been opened the system notifies the *road operator* nearest to the position of the car
- The *road operator* accepts the assistance request and can see detailed information about the accident and the exact position of the car
- The *road operator* picks up his tow truck and gets to the provided position following the app's directions
- The *road operator* tows the car to the repair center

Exit conditions:

- The *customer* gets charged a fee for the rent time until the failure

Exceptions:

- The car has a failure but it hasn't been detected, hence the *customer* calls the assistance service in order to receive support.

10.9 [U.9] Recover a car involved in an accident

Participating actors: Road operator, assistance service operator

Entry conditions: A car has been involved in an accident

Flow of events:

- The *assistance service operator* is notified by the system that a car has been involved in an accident
- The *assistance service operator*, after having tried to interact with the customer, opens an *assistance request*
- Once an *assistance request* has been opened the system notifies the *road operator* nearest to the position of the car
- The *road operator* accepts the assistance request and can see detailed information about the accident and the exact position of the car
- The *road operator* picks up his tow truck and gets to the provided position following the app's directions
- The *road operator* arrives on location

Exit conditions: The *road operator* tows the car to the repair center to get it fixed

Exceptions:

- If the *road operator* first notified by the system doesn't accept the *assistance request* in 1 minute the system sends another notification to another *road operator* and so on. Once a road operator accepts the *assistance request* all other notifications are deleted
- If the car isn't present at the provided location, the *road operator* contacts back the *assistance service*
- If the *road operator* determines that the car hasn't any problem, he tows the car to the closest safe area with available spot

10.10 [U.10] Recharge a car with low battery

Participating actors: Road operator

Entry conditions: A car has been left with less than 10% of charge in a safe area without power plugs and hasn't been used for more than 4 hours

Flow of events:

- The system makes the car unavailable to rent
- The nearest *road operator* gets notified about a car he has to get to a special parking area
- The *road operator* see the position of the car through the app
- The *road operator* picks up his company car and gets to the provided position following the app's directions

- The *road operator* arrives on location
- The *road operator* unlocks the car
- The *road operator* drives the car to the closest special parking area with available plugs
- Once arrived at the special parking area, the *road operator* exits the car, locks it and plugs it into the power grid

Exit conditions: The system makes the car available to rent by *customers*

Exceptions:

- If the car can't reach the closest special parking area with available plugs because it has a too low level of charge, then the *road operator* tows it there with his tow truck

11 Traceability Matrix

Raw ID	Goal ID	Use case ID
r.1	G.1	U.1
r.2	G.1	U.2
r.3	G.1	U.4
r.4	G.2	U.2
r.5	G.3	U.2
r.6	G.4	U.3
r.7	G.5	U.5
r.8	G.6	U.7
r.9	G.7	U.6
r.10	G.8	U.4
r.11	G.8	U.5
r.12	G.8	U.6
r.13	G.9	U.2
r.14	G.10	U.10
r.15	G.11	U.8
r.16	G.11	U.9
r.17	G.12	U.8
r.18	G.12	U.9

12 UML Diagrams

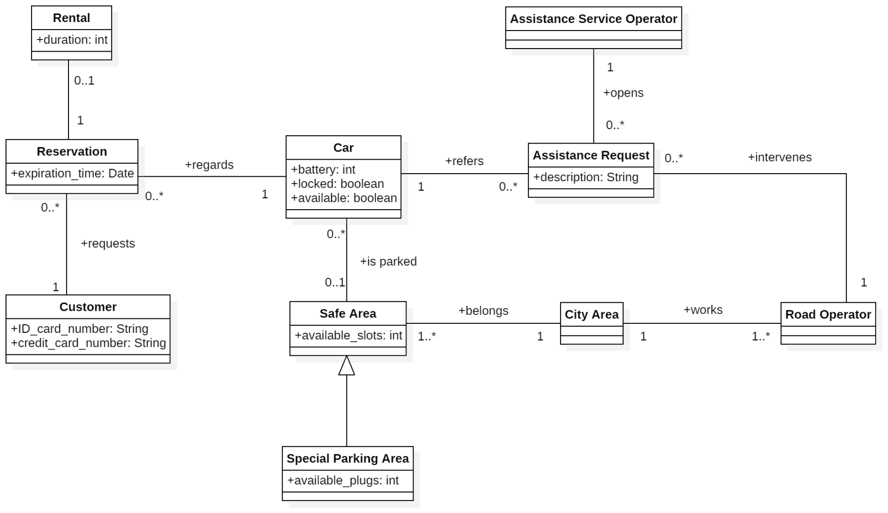


Figure 13: *World model* class diagram

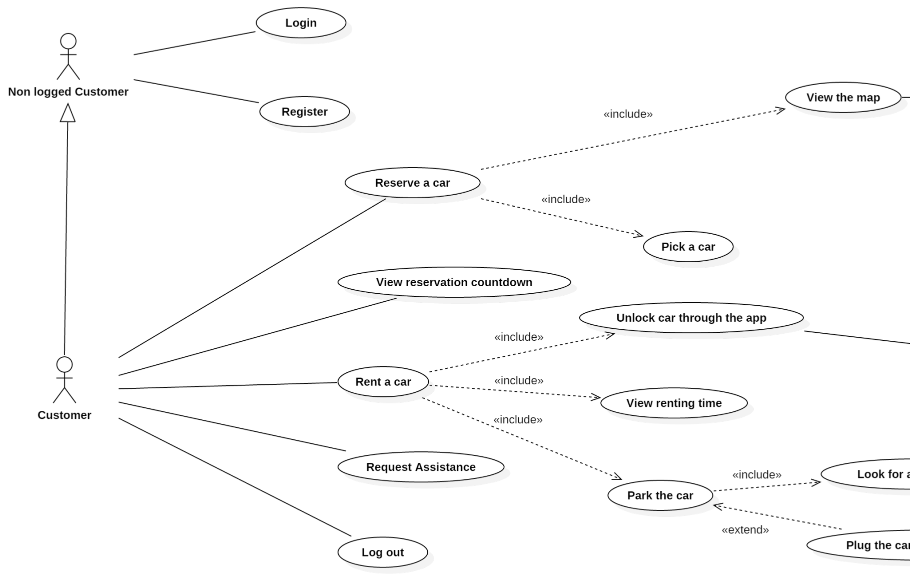
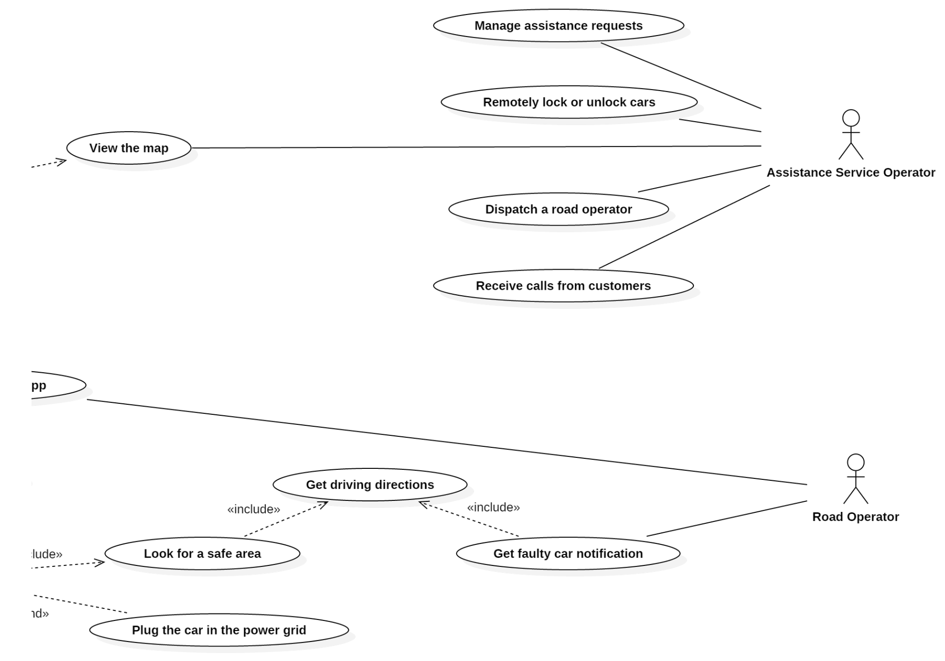


Figure 14: *Use case* diagram - 1



Manage assistance requests
Remotely lock or unlock cars
View the map
Assistance Service Operator
Dispatch a road operator
Receive calls from customers
pp
Get driving directions
«include»
«include»
Road Operator
lude»
Look for a safe area
Get faulty car notification
nd»
Plug the car in the power grid

Figure 15: *Use case* diagram - 2

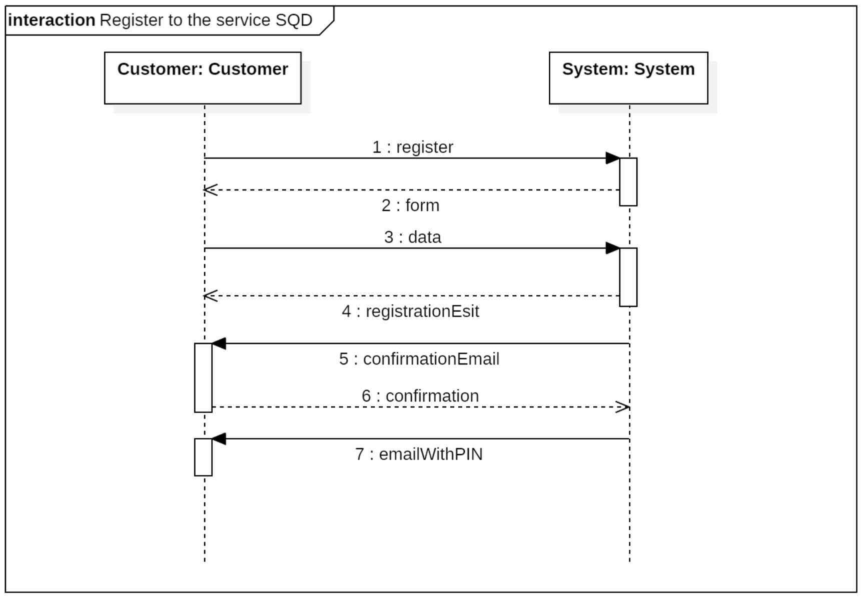


Figure 16: *Register to the service* sequence diagram

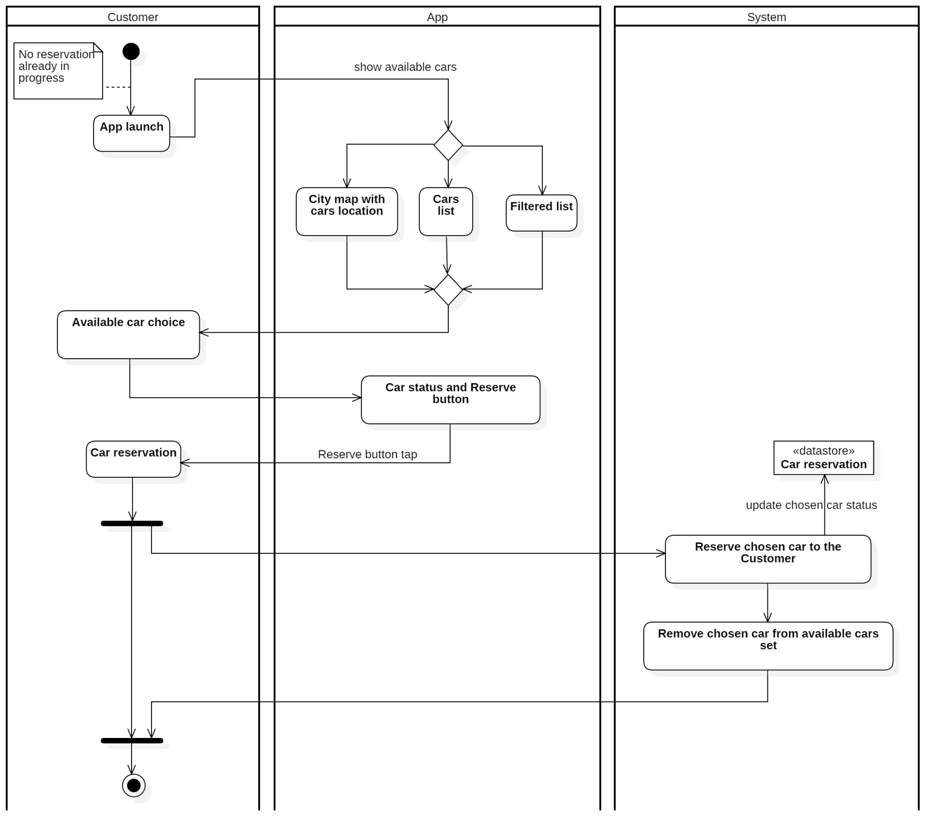


Figure 17: Activity diagram of *Reserve a car* use case

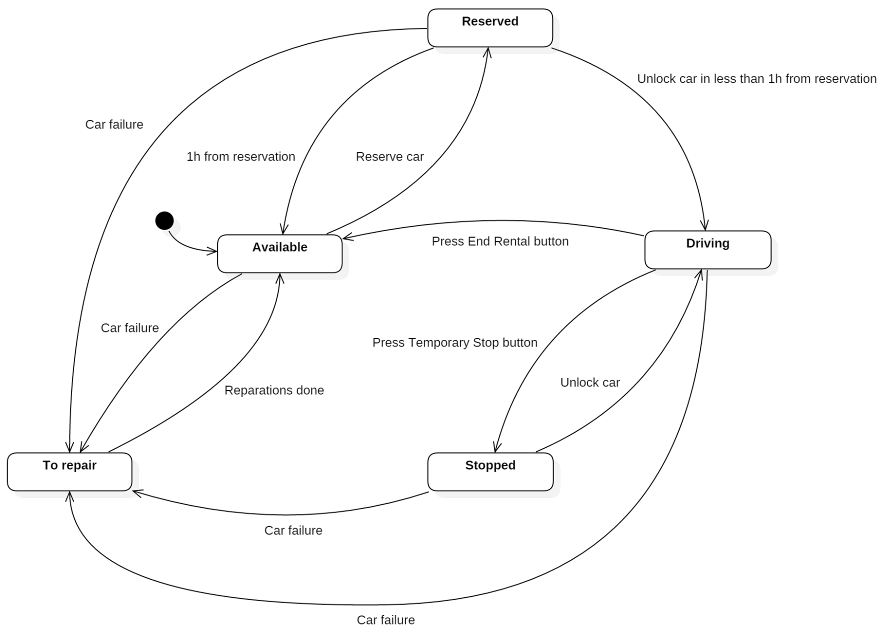


Figure 18: *Car status* state chart diagram

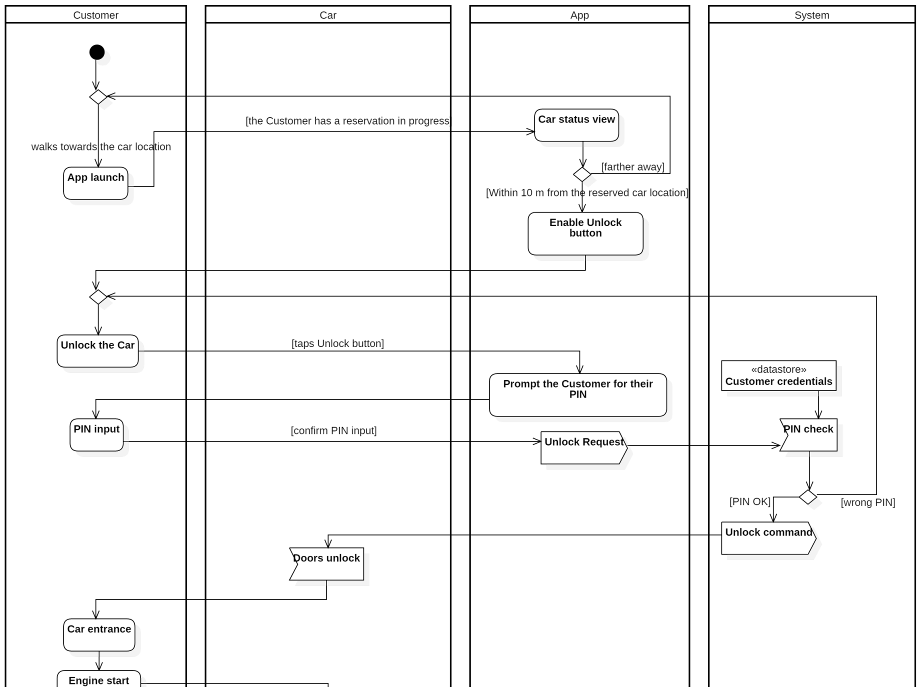


Figure 19: Activity diagram of *Drive a rented car* use case - 1

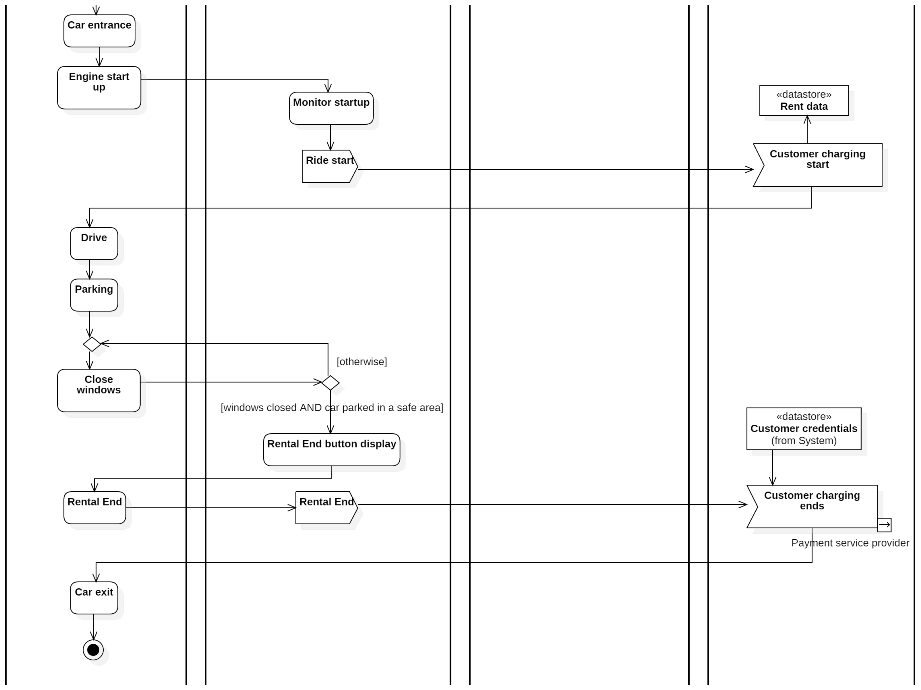


Figure 20: Activity diagram of *Drive a rented car* use case - 2

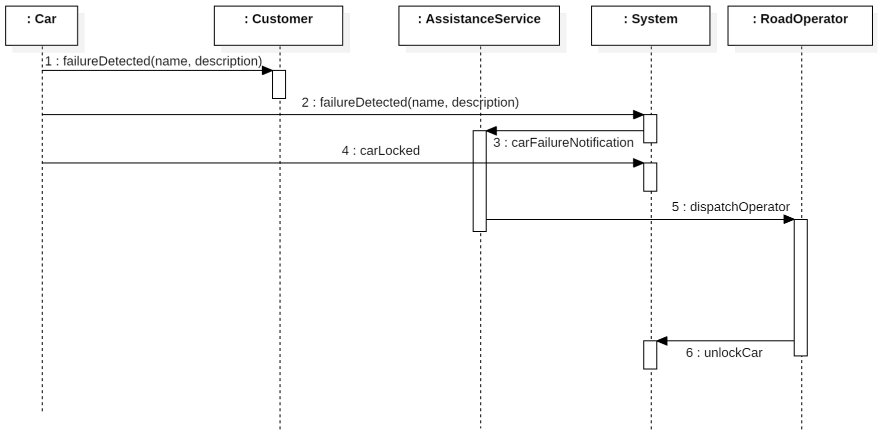

: Car
: Customer
: AssistanceService
: System
: RoadOperator
1 : failureDetected(name, description)
2 : failureDetected(name, description)
3 : carFailureNotification
4 : carLocked
5 : dispatchOperator
6 : unlockCar

Figure 21: *Managing a failure* sequence diagram

13 Alloy Model

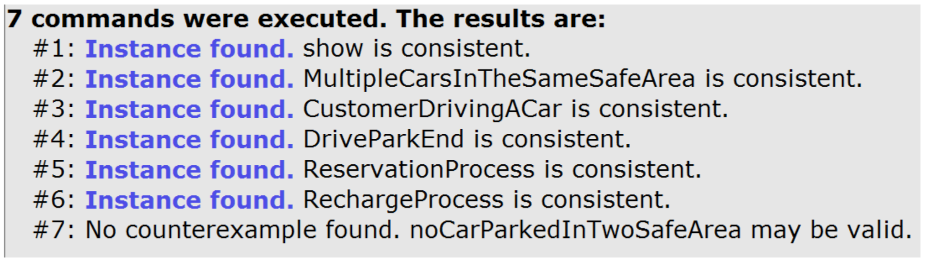
```
/**********************
 * Utility Signatures *
 **********************/
sig Chars {}
sig Code {}
sig LicencePlateCode {}
abstract sig Bool {}
one sig True extends Bool {}
one sig False extends Bool {}

/********************
 * World signatures *
 ********************/
sig ParkingSpot {}

sig Plug {}

// Inteded as fault classes
sig Fault {}
one sig EmptyBatteryFault extends Fault { }

sig SafeArea {
    parkedCars: set Car,
    availableSpots: set ParkingSpot
} {
    #parkedCars <= #availableSpots
}

sig SpecialParkingArea extends SafeArea {
    pluggedCars: set Car,
    availablePlugs: set Plug,
} {
    #pluggedCars <= #availablePlugs
    #pluggedCars <= #parkedCars
    // A Safe Area is a Special Parking Area
    // if provides at least 1 Plug
    #availablePlugs > 0
    all c: Car | c in pluggedCars implies
        c in parkedCars
}

// Battery levels are thought as intervals,
// as single percentage values are not interesting
// in the model
```

```
abstract sig Battery { }
one sig Greater50 extends Battery { }
one sig Greater20Less50 extends Battery { }
one sig Less20 extends Battery { }
one sig LowBattery extends Battery { }

sig Car {
    licencePlate: one LicencePlateCode,
    currentSafeArea: lone SafeArea,
    available: one Bool,
    batteryLevel: one Battery,
    pluggedIn: lone SpecialParkingArea,
    fault: lone Fault
} {
    // If the car is plugged in a special parking
    // area, it cannot be parked somewhere else
    some pluggedIn iff
        ( some spa: SpecialParkingArea |
            this in spa.pluggedCars and pluggedIn = spa )

    // If any Customer has this car reserved or it's
    // faulty, it shall not be available
    available = True iff
        (all c: Customer | c.reservedCar != this and
            fault = none) and batteryLevel != LowBattery

    // Battery level is considered empty only if the car
    // is not plugged in. The (thin) time lapse between
    // a empty battery and, let's say, 1% of charge is
    // not interesting in the model
    batteryLevel = LowBattery implies pluggedIn = none

    // If battery level is low the car is no longer
    // available and it's marked as faulty
    batteryLevel = LowBattery iff fault = EmptyBatteryFault
}

// A person interested in PowerEnJoy service
abstract sig User {}

// A user not-yet registered who is involved
// in a registration process
sig UnregisteredUser extends User {}

// A registered user who can access PowerEnJoy
// service
```

```
sig Customer extends User {
    username: one Chars,
    password: one Chars,
    pin: one Code,
    reservedCar: lone Car,
    previousReservations: set Car
} { }

sig RoadOperator {
    available: one Bool,
    currentTask: Car -> Fault
} {
    // Max one task at the time
    #currentTask <= 1
    // Current Task relates a Car to its fault only (if any task)
    all c: Car | c.currentTask = c.fault or c.currentTask = none
    // If a Road Operator has a current task then they are busy
    available = True iff #currentTask = 0
}

// Possible variations on the final rental fees
abstract sig PaymentVariation { }
one sig MultiplePassengersDiscount extends PaymentVariation { }
one sig HalfBatteryDiscount extends PaymentVariation { }
one sig CarRechargeDiscount extends PaymentVariation { }
one sig Less20BatteryOvercharge extends PaymentVariation { }
one sig MoneySavingDiscount extends PaymentVariation { }

sig Rental {
    reservation:  Customer -> Car,
    paymentVariations: set PaymentVariation
} {
    // Rental exists only if there is
    // a relation between a Customer and a Car
    #reservation = 1
    all c: Customer | (c.reservation in c.previousReservations) or
    c.reservation = none

    // Payment variation rules
    all c: Customer | c.reservation != none implies (
        (HalfBatteryDiscount in paymentVariations iff (
            (c.reservation.batteryLevel = Greater50) and
            (c.reservation.currentSafeArea != none)
        )) and
        (CarRechargeDiscount in paymentVariations iff
            c.reservation.pluggedIn != none) and
        (Less20BatteryOvercharge in paymentVariations iff
                ((c.reservation.batteryLevel = Less20) and
                (CarRechargeDiscount not in paymentVariations)
                and (c.reservation.currentSafeArea != none))
        ))
}

one sig System {
    availableCars: set Car,
    customers: set Customer,
    safeAreas: set SafeArea,
    faultyCars: set Car,
    rentals: set Rental
} {
    all c: Car | c.available = True iff c in availableCars
    // All registered users must be in system's customers list
    all c: Customer | c in customers
    all sa: SafeArea | sa in safeAreas
    all c: Car | c.fault != none iff c in faultyCars
    all r: Rental | r in rentals
}

/*********
 * Facts *
 *********/

fact ReservationRules {
    // A car cannot be reserved to more
    // than one Customer
    all c1, c2: Customer | c1 != c2 and
        some c1.reservedCar implies
            c1.reservedCar != c2.reservedCar

    // A car reserved to the user is added
    // to the previous reservations set
    all c: Customer, car: Car |
        c.reservedCar = car implies car in c.previousReservations

    // A faulty car cannot be reserved to any Customer
    all c: Customer, car: Car |
        not (c.reservedCar = car and car.fault != none)
}

fact ParkingRules {
    // If a car is parked there exists a
    // SafeArea in which that car is parked
    all c: Car | one c.currentSafeArea implies
        (one sa: SafeArea |
            c.currentSafeArea = sa and c in sa.parkedCars )

    // A safe area has a car parked in if
    // and only if the car is parked in
    // that safe area
    all c: Car, sa: SafeArea |
            c.currentSafeArea = sa iff c in sa.parkedCars

    // A plug cannot be shared between
    // different SpecialParkingAreas
    all p: Plug, disjoint spa1, spa2: SpecialParkingArea |
        p in spa1.availablePlugs implies
            p not in spa2.availablePlugs
}

fact AssistanceRules {
    // Only one road operator fixing a faulty car
    all disjoint ro1, ro2: RoadOperator, c: Car, f: Fault |
        (c->f) in ro1.currentTask implies (c->f) not in ro2.currentTask
}

/*************
 * Predicates *
 **************/

pred show() { }

// Test whether it is possibile to
// have more cars parked in the same
// safe area
pred MultipleCarsInTheSameSafeArea
[c1, c2: Car, sa: SafeArea] {
    c1 != c2 and
    c1.currentSafeArea = c2.currentSafeArea and
    c1.currentSafeArea = sa
}

// Test whether it is possible to have
// a car reserved to a Customer but not in any
// safe area, i.e. the Customer is driving
pred CustomerDrivingACar
[c: Customer, car: Car] {
    c.reservedCar = car and
    all sa: SafeArea | not (car in sa.parkedCars)
```

```
}

// Equality on previousReservations relations
// is omitted because two Customer
// could be the same even if they have not driven
// the same car in the past,
// e.g: right before and after a car reservation
pred SameCustomer
[c1, c2: Customer] {
    c1.username = c2.username and
    c1.password = c2.password and
    c1.pin = c2.pin
}

pred SameCar
[c1, c2: Car] {
    c1.licencePlate = c2.licencePlate
}

// Test a drive-park-end_rental process
pred DriveParkEnd
[c, c': Customer, car, car': Car, sa: SafeArea, r, r': Rental] {
    SameCustomer[c, c'] and
    SameCar[car, car'] and
    // If car is same to car', if car is
    // in Customer previous reservations,
    // car' must be aswell
    car in c.previousReservations and
    car in c'.previousReservations and
    car' in c.previousReservations and
    car' in c'.previousReservations and
    // c and c' have the same previous reservations in this case
    c.previousReservations = c'.previousReservations and
    c.reservedCar = car and
    car.currentSafeArea = none and
    c'.reservedCar = none and
    car'.currentSafeArea = sa and
    // The system has a rental for c and c'
    r.reservation = c -> car and
    r'.reservation = c' -> car'
}

// Test a car reservation process
pred ReservationProcess
[c, c': Customer, car, car': Car, r: Rental] {
    SameCustomer[c, c'] and
    SameCar[car, car'] and
    car.available = True and
    car'.available = False and
    c.reservedCar = none and
    c'.reservedCar = car' and
    // Same previous reservation but the one c' has reserved
    c.previousReservations = (c'.previousReservations - car') and
    // The system registers a new rental for c'
    r.reservation = c' -> car' and
    r.paymentVariations = none
}

// Test a emptyBattery-recharge process of a car
pred RechargeProcess
[car, car': Car, spa: SpecialParkingArea, ro: RoadOperator, r, r': Rental] {
    SameCar[car, car'] and
    car.currentSafeArea = car'.currentSafeArea and
    car.batteryLevel = LowBattery and
    car.pluggedIn = none and
    // A road operator is assigned to car to recharge it
    ro.currentTask = car -> car.fault and
    car'.pluggedIn = spa and
    car'.batteryLevel != LowBattery and
    car'.fault = none
}

/********
 * Runs *
 *********/

run show
run MultipleCarsInTheSameSafeArea
run CustomerDrivingACar
run DriveParkEnd
run ReservationProcess
run RechargeProcess

/***********
 * Asserts *
 ***********/

// There are no safe areas with the same car parked in
assert noCarParkedInTwoSafeArea {
    no c: Car, s1, s2: SafeArea |
        s1 != s2 and c in s1.parkedCars and
        c in s2.parkedCars
```

```
}

check noCarParkedInTwoSafeArea
```



```
7 commands were executed. The results are:
  #1: Instance found. show is consistent.
  #2: Instance found. MultipleCarsInTheSameSafeArea is consistent.
  #3: Instance found. CustomerDrivingACar is consistent.
  #4: Instance found. DriveParkEnd is consistent.
  #5: Instance found. ReservationProcess is consistent.
  #6: Instance found. RechargeProcess is consistent.
  #7: No counterexample found. noCarParkedInTwoSafeArea may be valid.
```

Figure 22: Alloy model execution results

14 Domain assumptions

- The company is not responsible for driving infractions committed by the customers, and eventual fines and charges are taken in charge by the external payment service.
- All the payment transactions from the customer to the company are successful; if a customer's credit isn't enough to pay his fees or fines the external payment service will inform the company, which will ban the user from using the car-sharing service.
- The customers can't drive out of the city limits and agree to this when they accept the terms of service; if they don't respect this rule, they will be legally prosecuted by the company.
- In case of customer misbehavior the Car service state is considered *'to repair'*, hence a Road Operator is assigned to restore Car availability.
- The safe areas are hard-coded in the developed software, as they aren't expected to change often; a management console for updating safe areas may be added in the future.
- Car informative system is considered always up and running. No system halt or reboot could occur.
- Car 3G module is always stably connected to the System. No disconnection or message loss occur.

15 Working time

Arcari Leonardo

- 20/10/2016 - 2h
- 24/10/2016 - 1h
- 25/10/2016 - 1h
- 27/10/2016 - 3h
- 29/10/2016 - 3h
- 03/10/2016 - 1h
- 04/10/2016 - 5h
- 05/10/2016 - 2h
- 06/10/2016 - 1h
- 07/11/2016 - 2h

- 08/11/2016 - 2h
- 12/11/2016 - 2h
- 13/11/2016 - 2h

Bertoglio Riccardo

- 20/10/2016 - 2h
- 25/10/2016 - 1h
- 27/10/2016 - 3h
- 28/10/2016 - 1h
- 29/10/2016 - 1h
- 02/11/2016 - 1h
- 03/11/2016 - 30min
- 04/11/2016 - 3h
- 06/11/2016 - 2h
- 08/11/2016 - 2:30h
- 09/11/2016 - 1h
- 10/11/2016 - 2h
- 11/11/2016 - 2h
- 12/11/2016 - 2h

Galimberti Andrea

- 20/10 1h
- 24/10 1h
- 25/10 2h - early draft of scenarios
- 26/10 2h - user and road operator scenarios
- 27/10 2h - group work
- 29/10 1h - car reservation expiration scenario
- 30/10 2h - added 2 road operator use cases + fixes
- 01/11 1h - added 3 use cases

- 03/11 2h - fixes to use cases + first draft of class diagram
- 04/11 2h - class diagram 05/11 2h - update to use case diagram + added two scenarios
- 06/11 2h - first draft of the RASD + removed payment amounts from use cases
- 07/11 2h - removed the customer service operators from the project
- 08/11 2h - added car statechart diagram
- 09/11 1h - traceability matrix
- 12/11 3h - updated UMLs and document
- 13/11 1h - added domain assumptions to document

16 Changelog

- 02/12/2016 - Version 1.1
  - *Domain assumption on Car service status in case of Customer misbehavior*
  - *Domain assumptions on Car informative system and connection availability*
  - *Deeper description of sensors / modules available on the car*
  - *Security concerns*
- 14/11/2016 - Version 1.0.1
  - *Aesthetic fixes*
- 13/11/2016 - Version 1.0