POLITECNICO DI MILANO
Computer Science and Engineering
Software Engineering 2 Project

Tesla Car Sharing

Requirements Analysis & Specification Document

Danchenko Elena, 874840
Cilloni Stefano, 880924

November 14th, 2016

Document version: 1.2

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to approach, from general to specific, the architecture of Tesla Car Sharing software system that is a project from 2016/2017 Software Engineering 2 class at Politecnico di Milano.
In this document we describe our technical decisions about the system taken in the context of the world it is interacting with.

1.2. Description of the given problem

This project aims to develop a system for electric car sharing service. This system allows clients to find the car near their current location or near a given address and reserve it for a defined amount of time. The system also allows company employees to manage data about cars and it also includes functionalities concerning inspiration of virtuous behaviors of clients. The system consists of two applications for Android and iOS and of a web application. All users are recognized thanks to the login procedure required to access to the system functionalities. Simple users can register to the system by themself through the mobile application or through the web application.

1.2.1. Actual system

There is no actual system present. It is a new company which wants to provide car sharing service and which needs an application for its business.

1.3. Goals

1.3.1. Users

1. Provide the possibility to register and to login to the system.
2. Provide the possibility to search for available cars.
3. Provide the possibility to reserve a car.
4. Provide the possibility to unlock the car when the user is near it.
5. Provide the mechanism to charge users for the ride.
6. Sensitize virtuous behaviours of users.

1.3.2. Employees

1. Provide the possibility for the administrator to manage employees accounts.
2. Provide the possibility for managers to manage car records.
3. Provide the possibility for managers to manage price for minute of the ride.
4. Provide the possibility for managers to define the set of areas which are safe to park.
5. Provide the possibility for operators to get information about cars requiring service.
6. Provide the possibility for operators to accomplish car service tasks.

1.4. Domain properties

We suppose that these properties hold in the analyzed world:

- The company only has cars equipped with a car control software that manage sensors in the car and that can provide to other systems the car position and information about its status.
- The car control software works properly without problems and in any conditions.
- GPS and control software in the car cannot be switched off.
- GPS system always gives the right position letting it the time to get accuracy.
- User cannot tamper the control software in the car.
- Operators correctly and on time insert in the system informations about cars status.
- Managers correctly and on time register new cars and remove the ones which cannot be used anymore.
- The number of passengers for ride could be only positive or zero.

1.5. Glossary

- *Car sharing*: service that provides people possibility to rent a car for one or more rides
- *User*: depending on the context, "user" can mean a generic actor of the system (such as an operator or a manager) that is interacting with the system or it can mean a client of the car sharing service that use functionalities intended to the clients of the service (simple user).
- *Administrator*: employee whose responsibility is the management of the software system and accounts
- *Manager*: employee who does management operations for the car sharing service. He/she takes care to register new cars and also to removes the ones that can't be used anymore. He/she works at the service back office helping operators with administrative operations.
- *Operator*: employee responsible for cars maintenance. He/she also moves cars from any place to safe areas.
- *Safe Area*: a polygon shape area over a map that defines where users can park and pick up cars. Safe areas also provide to users electric vehicle charging stations that let users put in charge cars.

- *Electric vehicle charging station*: small tower shaped element that supplies electric energy for the recharging of electric vehicles.
- *Ride*: it starts when the user unlock the car and ends when he/she leaves the car locking it (or the system lock it automatically)
- *Reservation*: user's ability to reserve a car for an amount of time in such a way that nobody else can reserve and use it. Reservation ends after an amount of time or when the user reaches the car to start the ride.
- *Mobile application (or mobile app)*: software application designed to run on mobile devices such as smartphones and tablets. In this document, "mobile application" is used to refer to both the Android and iOS applications.
- *Web application*: it is a client-server software application that provide to the user a graphical interface to access the software functionalities which runs in a web browser.
- *Car control software*: software that takes care to manage all car sensors, the display into the car and that provides car informations and user choices through API calls.
- *PaaS (Platform as a Service)*: as the name suggests, provides you computing platforms which typically includes operating system, programming language execution environment, database, web server, etc.

1.6. Text assumption

- There is an autonomous control system in any cars that is connected to internet and that is attainable from the Tesla Car Sharing software through APIs provided by the company that developed it.
- The mobile applications on user phones must be able to access to the device GPS services to obtain the user position.
- User location on the web application could not be enough accurated to search for nearby available cars because of the way in which desktop devices are localized. It is left to the user the choice to search available cars through the detected location or by a given address.
- To search for available cars users must be registered to the system.
- Given informations by the user (e.g. driver license code, CF, address, etc..) are supposed to be correct.
- Privacy policy terms are provided by the stakeholder of this project. We will provide to him or to the company that will take care of this all the required informations to understand how users data are managed.
- The web application properly only if it runs in browser up to date.

1.7. Constraints

1.7.1. Regulatory policies

- Since the system manages sensitive data (such as user details and payment informations), users must agree to the privacy policy during the sign up procedure to get access to the system.

- The system doesn't use notifications to users, such as email or mobile application notifications, to send them SPAM.
- The system needs access to the user location to work properly. The system asks to the user the permission to get the his/her location on first time that it is really necessary.
- The web application works properly only if it can store cookies to the user's browser. The user is informed about this and he/she must agree on cookies usage.

1.7.2. Hardware limitations

Mobile applications
Mobile phones must have GPS and internet connection turned on.
Mobile phones must have enough space to install the application.

Web application
User must have internet connection on his computer.
User must have a browser up to date.

1.7.3. Usage constraints

Operations of managers and administrator are available only through the web application interface. Instead, users are able to access car sharing service both from the web application and from the mobile applications. For operators is available only the mobile application inasmuch it is supposed that they are always moving around the city or they're working on cars in mechanic workshops so it's supposed that they're more comfortable to access to the system by a mobile application. The mobile application that the operator installs is the same that a simple user installs. However once the operator is logged in, he/she has special functionalities that are hidden to the simple user. These functionalities let the operator receives notifications from the system and manage car stuff.

	Web application	Mobile application
User	✔	✘
Administrator	✔	✘
Manager	✔	✘
Operator	✘	✔

Table 1: Applications accessibility according to users types.

1.7.4. Interfaces to other applications

Car control system
The system will interact with the car control software through the APIs provided by the car software developers.
It is assumed that the control software of the provides APIs to:

- Obtain current car location.
- Obtain current battery percentage.
- Obtain informations about number of passengers and driver presence.
- Obtain informations about accident sensors.
- Obtain informations about the car status to understand if maintenance is required.
- Obtain informations if the car requires repairs.
- Send to it the charges for a ride in order to show them to the user on the screen on board.
- Send to it the car lock and car unlock commands.

Payment platform
The only payment method expected is credit card. The system will interact with the payment platform through the APIs provided by the bank.

Google maps
Google Maps APIs are used whenever a map to locate cars is needed in the system. Google Maps services are used both in the web application and in the mobile applications.

1.7.5. Parallel operation

As the most popular computer systems the Tesla Car Sharing software supports parallel operations done by multiple users, operators and managers connected at the same time through the web application or the mobile application.

1.8. Proposed system

1.8.1 Client side

We propose a web application and two mobile applications for the most popular mobile operating systems: Android and iOS.

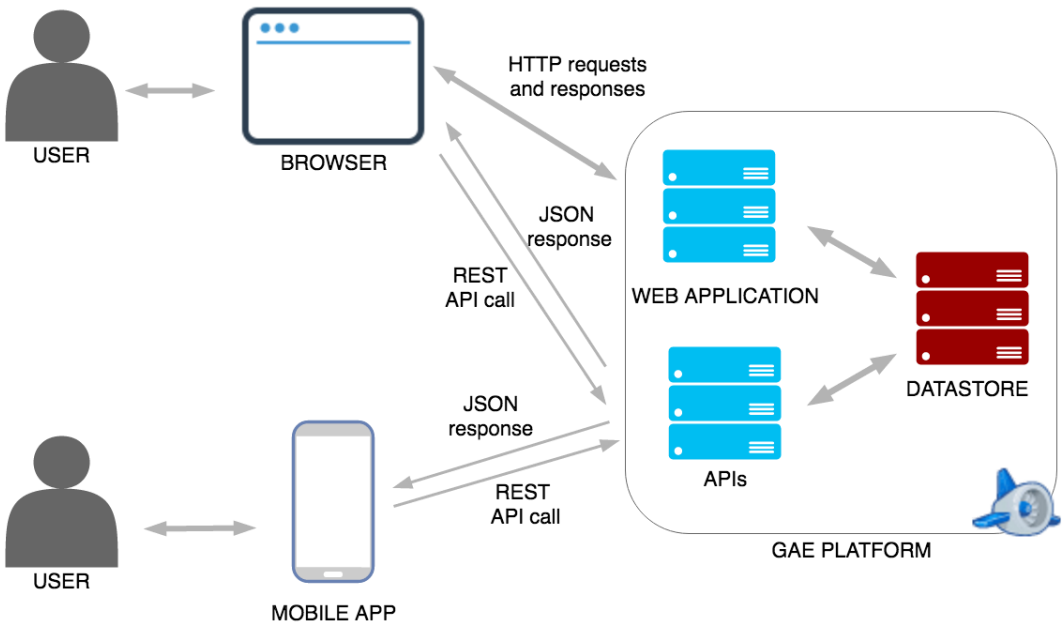


Image 1: Draft of the system architecture

The web application and mobile applications change functionalities and graphical interfaces depending on the type of user that is logged in. By the type of user the system rebuild the graphical interfaces showing or hiding sections and functionalities. For example the simple user can reserve a car and cannot manage accounts as the administrator does.
Furthermore, the web application also interact with server by API calls to allow some operations to be done without refreshing the page.
All APIs developed to let mobile applications and web application connect to the main system will be REST-compliant in such a way that the system can growth in complexity but preserving its maintainability.
Both Android and iOS application are developed with the native SDK and for this reason both follow the MVC design pattern.

Client side components:

- Android mobile app
- iOS mobile app
- Web application

1.8.2 Server side

The application server is the main core of the Tesla Car Sharing software. The application server serves to the users all web application pages. One of its parts is fully dedicated to implement APIs in order to provides to the mobile applications data access and functionalities.
Whole the application server and APIs will be build on the PaaS software environment called: Google App Engine (GAE).
To store data we will use the provided solution that is commonly used with the GAE environment: the Google Datastore noSQL database.

Server side components:

- Application server on Google App Engine platform
- Google Datastore

1.9. Identifying stakeholders

Only one main stakeholder is been identified: the board of directors of Tesla Car Sharing company.

1.10. Reference documents

- Specification document: Assignments AA 2016-2017.pdf
- IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.
- Google App Engine Documentation
- Examples document: RASD sample from Oct. 20 lecture.pdf

2. Actors identifying

In our system we have the following actors.

- Users
- Administrator
- Managers
- Operators
- System
- Car control software

3. Requirements

3.1. Functional requirements

3.1.1. Clients

1. User must be able to register to the system.
    1.1. To register users must provide credentials.
        1.1.1. Credentials to be provided by the user are:
            - Name
            - Surname
            - Email
            - Birthdate
            - CF
            - Country of birth
            - City of residence
            - Street
            - House number
            - CAP
            - Identity document type and number
            - Number of driver licence and its expiration date
    1.2. To register users must provide payment informations.
        1.2.1. User must provides credit card holder name, credit card number, expiration date and security code.
    1.3. Upon registration users receive back their password.
        1.3.1. Users receive back a password on the web page or in the mobile app.
        Rationale: both web page and mobile applications will use HTTPS connections that protect informations with encryption. Send password through email could be a weakness in the system security.
2. Registered users must be able to login into the system.
    2.1. To login to the system users must use email and password
3. Registered users must be able to recover their password
    3.1. The system asks to the user the email address to identify him/her. Then the system asks the user to provide credit card informations so it can check the user identity and give him/her back a new password.
4. Registered users must be able to find the locations of available cars.
    4.1. Registered users must be able to find the locations of available cars within a certain distance from their current location. This distance is defined by managers from the management console of the system (It is a part of the web application).

5. Registered users must be able to find the locations of available cars from a specified address.
    5.1. Address can be specified by province, city, street, and house number.
6. Users must be able to reserve a car among the available cars in a certain geographical region.
    6.1. Users must be able to reserve a single car at time.
    6.2. Users must be able to reserve a car up to one hour before they pick it up.
7. After one hour of reservation, if the user doesn't pick up the car, reservation expires making the car available again.
    7.1. Upon the expiration of the reservation, the system makes the car available again.
    7.2. If the reservation expires the user pays a fee.
        7.2.1. The fee that user pays is 1 EUR.
8. A user who reaches a reserved car must be able to tell to the system that he/she is within the distance defined by the system to get the car unlocked.
    8.1. A user must be able to tell the system that he/she is within the distance defined by the system through the sharing of his/her location in geolocation services of the mobile device.
    8.2. The system must unlock the car after checking that the location of the machine and the one sent by the user coincide within an approximation.
    8.3. The system must send a code for engine ignition after the check done on car and user locations.
9. As soon as the engine ignites the system starts charging the user.
    9.1. The system charges the user for the amount of money per minute specified by the administrator in the management console of the software.
    9.2. The system sends current charges for the ride to the car control software so the user can seen it on the screen on board.
10. After the car is parked in one of the safe areas defined by the system and user exits the car the system stops charging the user.
    10.1. Users can lock the car autonomously but if it doesn’t happen the system must lock the car after it is parked and the user exited the car.
    10.2. When the user moves away from the car him/her receives a notification through the mobile app.
    Rationale: user must move away from the car so the system can understand if him/her put the car on charge or not.
        10.2.1. The notification includes information about price of the ride, applied discounts and fees.
11. If the car is left outside of safe area the system applies the fee defined by the managers through the management console.
12. The set of safe areas for parking cars is pre-defined into the management console by managers.
13. If the system detects the user took at least two other passengers onto the car, the system applies a discount of 10% on the last ride.
    13.1. Discount is applied if passengers have spent in the car at least half of the ride.
14. 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.

15. If a car is left at special parking areas where they can be recharged and the user takes care of plugging the car into the power grid, the system applies a discount of 30% on the last ride.
16. if a car is left at more than 3 KM from the nearest power grid station, the system charges 30% more on the last ride.
17. if a car is left with more than 80% of the battery empty, the system charges 30% more on the last ride.
18. If a car get involved in a crash the system is automatically notified by the car control software.
    18.1. The ride of the user ends immediately after the crash and even the receipt is delivered as a notification to the user immediately.
    18.2. The system puts the car in the list of accidents so operators can reach them to handle the situation.

3.1.2. Employees

1. Administrator must be able to login to the system.
    1.1. To login to the system administrator must use login and password
    1.2. For the first login administrator uses login “admin” and password “passwd”
    1.3. After administrator is logged in for the first time he is obliged to change password.
2. Administrator must be able to manage manager accounts.
    2.1. Administrator must be able to create new accounts with the manager name, surname and his/her e-mail. During the creation procedure the system generates an employee id (to identify uniquely him/her) and a temporary password that the manager will be obliged to change on the first login.
    2.2. Administrator must be able to modify manager accounts.
    2.3. Administrator must be able to delete an existing manager account.
3. Administrator must be able to manage operator accounts.
    3.1. Administrator must be able to create new accounts with the operator name, surname and his/her e-mail. During the creation procedure the system generates an employee id (to identify uniquely him/her) and a temporary password that the operator will be obliged to change on the first login.
    3.2. Administrator must be able to modify operator accounts.
    3.3. Administrator must be able to delete an existing operator account.
4. Managers must be able to login to the system.
    4.1. To login to the system managers must use email and password. If it is the first login with the password provided by the administrator the system forces him/her to set a new password.
5. Managers must be able to change the price per minute for the ride.
6. Managers must be able to define, within a certain range, the distance in which users can search (by their location or by address) for available cars.
7. Managers must be able to define the distance within the system consider a user near to one car.
8. Managers must be able to define the applied fee if the car is left outside of safe areas.

9. Managers must be able to define where the safe areas are, which shape they have and if they are provided with electric charging stations or not.
10. Managers must be able to register new cars in the system.
11. Managers must be able to modify information about cars in the system.
12. Managers must be able to remove cars from the system.
13. Operators must be able to login to the system.
    13.1. To login to the system operators must use email and password. If it is the first login with the password provided by the administrator the system forces him/her to set a new password.
14. Operators must be able to get informations about cars locations when cars are not used by the users or cars are involved in crashes.
15. Operators must be able to receive information about cars requiring services.
    15.1. Operators must be able to receive information about cars left outside of safe areas.
    15.2. Operators must be able to receive information about cars left without sufficient amount of charge.
    15.3. Operators must be able to receive information about cars requiring maintenance.
    15.4. Operators must be able to receive information about cars requiring repairs.
16. Operators must be able to receive instant notifications on the mobile app about crashed cars. The first free operator must take charge of the accident and then must reach the place to handle the situation. The operator must be able to accept the accident task so other operators can know that the task is already accepted by someone.
17. Operators must be able to choose a task to do.
18. Operators must be able to waive a chosen task.
19. Operators must be able to notify the system about successful task service completion.
20. After service being done system must mark the car as available

3.2. Non-functional requirements

The web application will provide an user interface built on a simple and intuitive design. The user will benefit of the best usage experience with as little effort as possible.

3.2.1. Web application interfaces

Login page

The first page of the system is the registration/login page. Unregistered users can start the signup procedure by the button “Sign Up”. If a user chooses for the signup procedure he/she is redirect redirected to another page showing the appropriate form with all the necessary informations to insert. Instead, if an already registered user wants to access to the system he/she has only to fill the login form with email and password and press "Log In".
We create this mock graphical user interface to give an example.

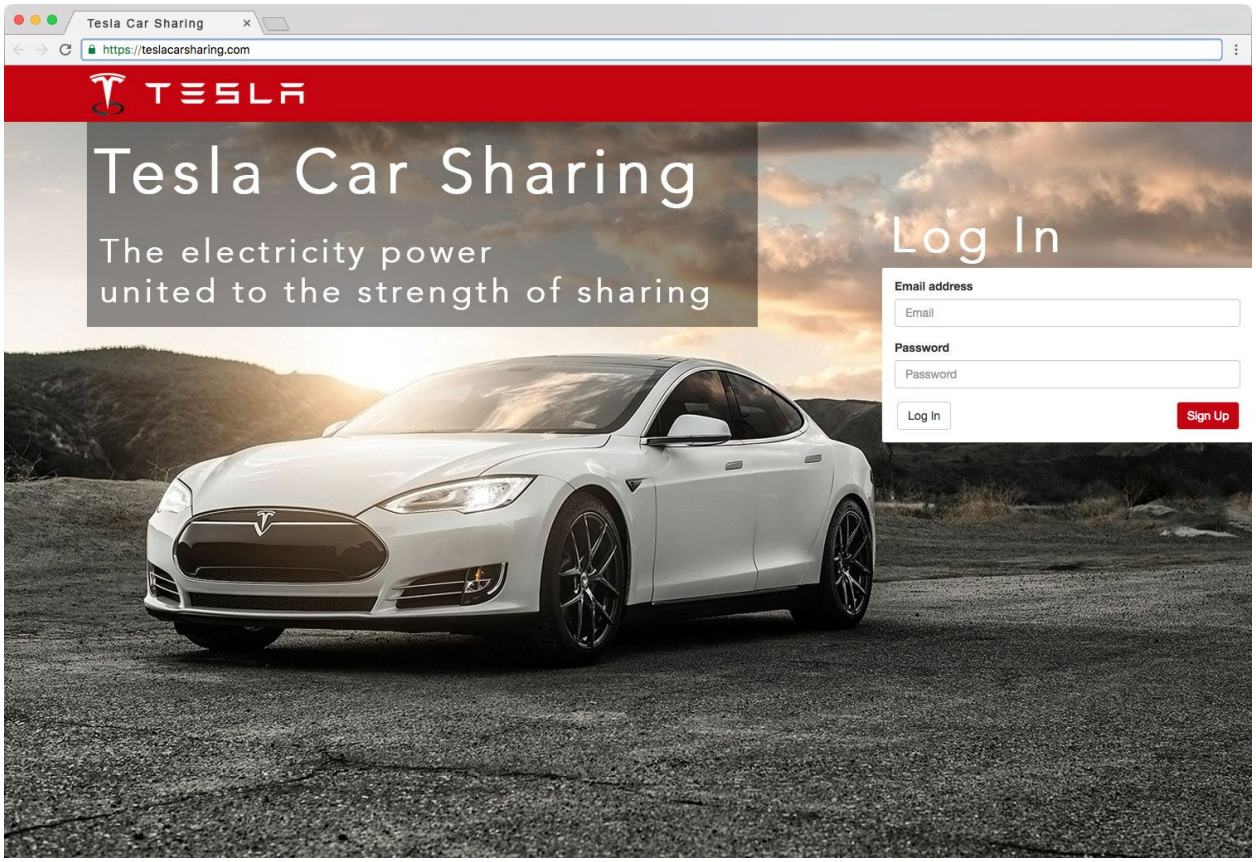


Image 2: Example of the graphical user interface of the login page.

Cars search page

The search page lets the user to search for available cars. He/she can search by an address putting it in the search bar or even by his/her current location by pressing the location button at the bottom right corner of the map.

Search results will appear on the left as a list of available cars. For each car some useful informations are reported. For example, the car model, the battery percentage and the car status.

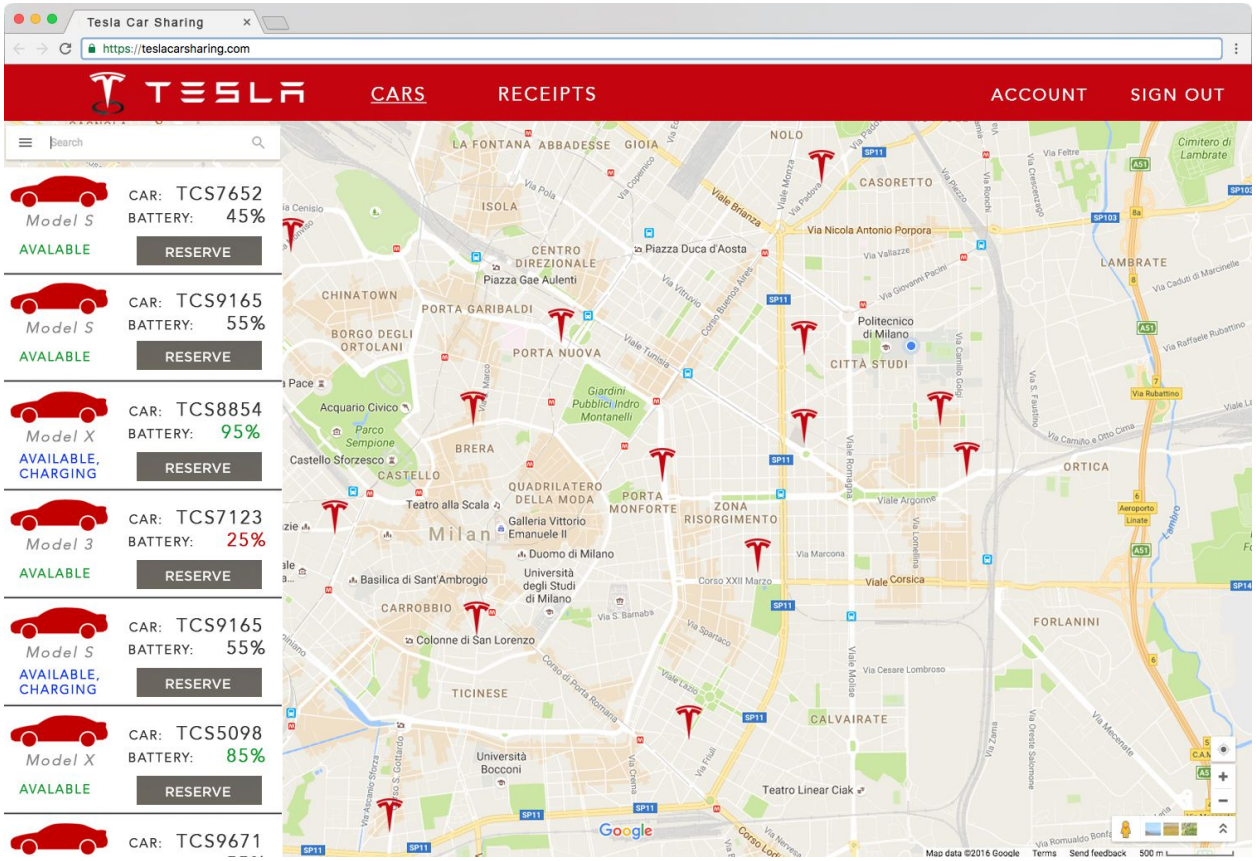


Image 3: Example of the graphical user interface of the car search page.

3.2.2. Mobile application interfaces

Login section

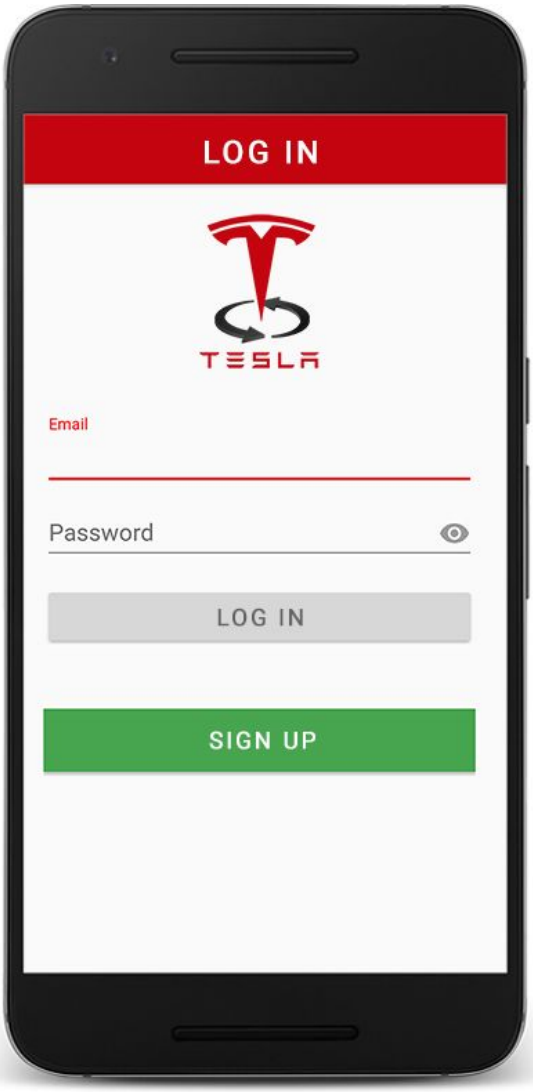


Image 4: Example of the mobile graphical user interface of the login section.

Cars search section

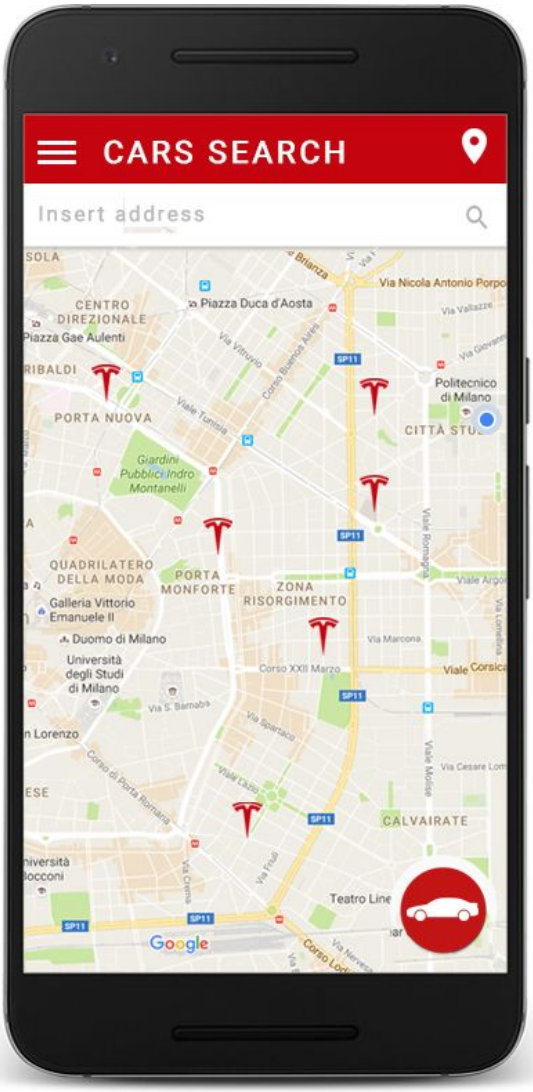


Image 5: Example of the mobile graphical user interface of the car search section.

Car reservation section

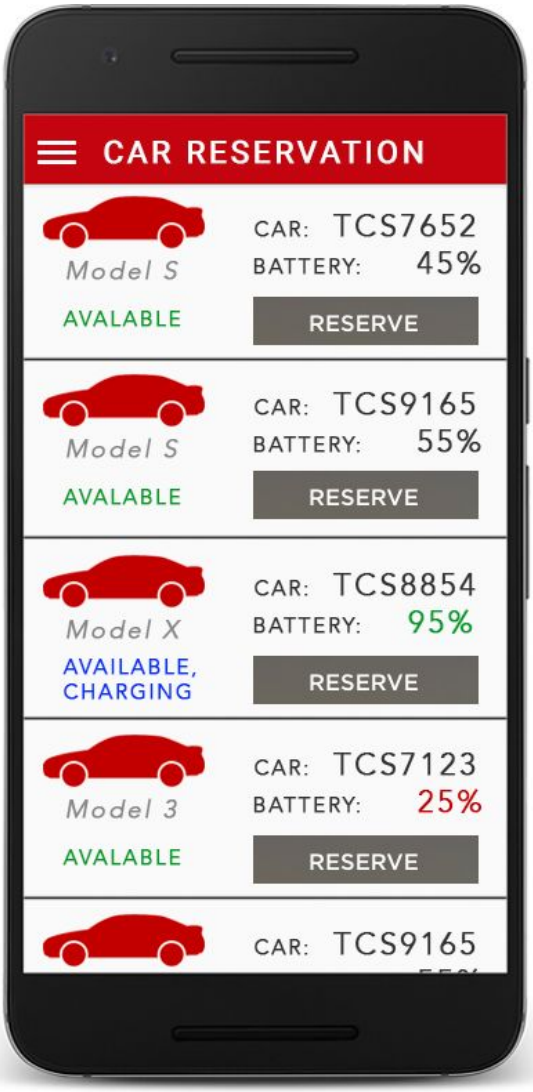


Image 6: Example of the mobile graphical user interface of the car reservation section.

3.2.3. Documentation

The following documents are drafted in order to achieve a well-organized plan of work in such a way to reduce development time and do in a fewer time the best work as possible:

- RASD: Requirement Analysis and Specification Document, to avoid misunderstanding between stakeholder ideas and functionalities that will be build. This document analyze in a detailed way goals and how to reach them by the definition of precise requirements and specifications.
- DD: Design Document, to define the real structure and implementation of the final system. It describes all the application tiers and how they interacts each other and with third-party applications.

3.2.4. Architectural Consideration

About the core of the Tesla Car Sharing application software we want to suggest a scalable solution like the cloud environment Google App Engine. This platform is recommended for softwares that are built from the scratch and that potentially can reach a workload of millions of operations per second. Since it is a cloud software environment it gives a lot of benefits in terms of scalability, fault tolerance, reliability and modularity.
Both the application server core and API section will be implemented on this PaaS environment.
About the database component we suggest to use the non relational database that is commonly used on Google App Engine environment: the Google Cloud Datastore. Its non relational structure allows also an high level of flexibility that is useful to develop software that can require to evolve during their life.

3.2.5. Used technologies

We will use the following technologies.

Server

- Flask micro web framework
- Jinja2 templating framework
- Google App Engine environment
- Google Cloud Datastore
- RESTful standard as a guideline to build great APIs
- SSL standard security to protect HTTP connections
- JSON standard to even communications between application parts

Client: Web application

- Bootstrap frontend CSS and Javascript framework
- JQuery javascript framework
- Sass CSS language
- AJAX development technique

Client: Mobile applications

- Java language for Android development
- Swift language for iOS development
- Original SDKs for respectively both the platforms

3.2.6. Performance consideration

Since the main part the software will run on a cloud environment, the computing power necessary to keep it with high performances will be provided on request. Many instances of the software will be automatically deployed by the platform to handle high workload. The same instances will be turned off when the workload decrease and this mechanism helps to save unused computing resources.
No problems about server reliability will be expected. All server possible issues are completely handled by the cloud platform. (e.g. blackout, server failure, backups to be done, etc..)

4. Scenario identifying

Following are some possible scenarios of usage.

4.1. Scenario 1

Matt has to go to pick up his two sons after the football training. He's currently at work and in the morning he took the bus to come to the office so he doesn't have with him his car. Matt also has to hurry up because children will finish the football training soon. He heard about the Tesla Car Sharing service and he wants to try it. He already is registered to the service and he is ready to reserve a car. He opens the mobile app on his smartphone and in the car search section he just searches for a car nearby his location. Fortunately there's a car 500 meters far and he decides to immediately reserve it. He knows that the reservation is up to one hour but just in a quarter he will end at work. After Matt reaches the car, he opens the mobile application and asks for the ignition code. The system verifies that Matt is near the car thus it unlocks the car and sends the notification with the ignition code. Matt uses the ignition code to ignite the car engine and then he goes to pick up his sons. Since there's a charging station near to the Matt house he puts the car in charge. When Matt is far from the car enough, the system sends a notification to the Matt phones with the receipt. There are reported: the discount for the two passengers ride (that's applied because the football field is more close to Matt's office than his house), the discount for putting the car on charge and the total amount of money that he is charged.

4.2. Scenario 2

Jane, Laura and Lucy are three friends that love do shopping together. Jane already uses the Tesla Car Sharing service from two weeks and she's really happy of how it works. She decides to invite her friends to buy a new dress for their night in the club and she proposes them to go to the mall with the car sharing service. Jane opens the app on her phone and after the login procedure she searches for a car near her home. A car is available not so far and she reserves it. The three girls arrive to the mall and they find out that unfortunately there aren't any safe areas near. When Jane moves away from the car the system send her the receipt with the applied discount for the ride with two passengers and with the 30% additional charge because of the car parked outside a safe area.

4.3. Scenario 3

Paul travels a lot for work. Today he is arrived at the trains station and he already had decided to try the Tesla Car Sharing to to reach the place of the work appointment. Still during the travel on train he searched a car through the mobile app. Since he wasn't already at the station he searched the car with the search by address functionality. After Paul had inserted the station address the list of available cars appeared on the app. Many cars were available near the station and Paul reserved one of them. The system notified him of the time limit for the reservation but he had planned to arrive on time at the car. After Paul arrives to the car he asks with the app to unlock it and the system, after the check on car and user locations, gets unlocked the car. Then Paul receives the ignition code so he can start his ride.
After the ride, Paul decides to put the car in charge because he had read about the discount available through this simple action. Afterwards when he moves away from the car the system notify him with the receipt that contains the summary of the ride. There's the the total amount of money that he has been charged and the applied discount for put the car on charge.

4.4. Scenario 4

Kate is usual to use Tesla Car Sharing service to go work and also to come back from it. Today as many others she takes the nearest available car to do her trip. Unfortunately during a turn in a road intersection someone collides with the car that Kate is driving. The damages are not heavy and the car system notifies the central system about the accident. The ride is terminated immediately and Kate receives the receipt with the amount of money that she is charged. Later an operator of the car sharing service arrives on the place and takes care to talk with authorities and bring back the car to the headquarter of the company.

4.5. Scenario 5

Robert is Sam's brother and he suggested him to try the Tesla Car Sharing service. Robert is usual to do some trips with the service and it is really useful in his opinion. Since that, Sam wants to try the service too and he decides to register to the system through the website. The system asks him about his personal information such as: name, surname, email and some other details. After this first step he goes ahead and the website asks him for the payment informations. Sam must provide his credit card details so that the system will be able to charge him for provided services. Right after the end of registration Sam takes an overall look to the service functionalities for his next trip.

4.6. Scenario 6

Bob wants to reach some friends to the party in suburb of the city. He knows that the car sharing service is cheaper than call a taxi and he also likes the idea to help the environment using an electric car. He rapidly search for a car by his location and then he reserves it. When he is near the car he asks

to unlock the car. The system verifies his location and the one of the car and then unlocks the car. After that Bob receives on the mobile app the code to ignite the car. Bob drives a lot to reach the suburb of the city and when he arrives to the party the car battery has only the 15% of the charge. He parks the car in a near Safe Area and he puts it on charge too. Afterwards when he moves away from the car he receives the receipt as a notification on the mobile app. The system charges him with the cost for ride and it also adds a more 30% additional charge for the battery that has been left quite empty. The system also applies the discount because Bob has put the car on charge.

4.7. Scenario 7

The car sharing service is growing fast. By the analysis of service usage it is noticed that a lot of users try to reserve an available car that immediately becomes reserved. The CEO of the Tesla affiliated company that holds the sharing service wants to add 10 more cars to improve the number of available cars in the city. After the cars have been configured and installed to work properly with the car sharing system they need to be recorded in the system to start with the service. Managers are the ones who can do this. James, a manager that worked for the Tesla Car Sharing service since the beginning, starts the recording procedure through the web application. He opens the dedicated section on the management console and then he fills the form with all the required information. Operator who helps him verifies that the system can connect to the car control software so that they can communicate. After the final revision of the inserted data the car becomes available to be reserved. Finally, it is brought to a random safe area in the city by the operator that helps James in the recording procedure of new cars.

5. Use cases

5.1. Registration

Actors	● User ● System
Enter conditions	User can connect to the system through the mobile app or the web application.
Flow of events	1. User starts signup procedure. 2. System asks for user credentials. 3. User inserts his/her credentials. 4. System asks for payment information. 5. User inserts payment information. 6. System notifies user of successful registration and gives him/her back a password.
Exit conditions	● User account is created. ● User receives back a password.
Exceptions	● If user inserted an information which do not fit to the given format the system prompts shows an error messages about it and user cannot proceed with registration until correct information is not inserted. ● If user inserted wrong payment informations the system shows an error messages about it and user cannot proceed with registration until correct information is not inserted. ● If email is already registered the system prompts an error message about it and suggests to the user to recover the password through the dedicated section.

5.2. User login

Actors	● User ● System
Enter conditions	● User is already registered. ● Connection to the system.
Flow of events	1. User opens the login section. 2. System asks for login and password. 3. User inserts login and password. 4. System logs the user in.
Exit conditions	User is logged in so he/she can access to the system functionalities.
Exceptions	If user inserted wrong email or password system shows an error message about it and suggests to try to log in once again or to recover the password through the dedicated section.

5.3. Search for available cars by the current location

Actors	● User ● System
Enter conditions	● User is logged in. ● Connection to the system.
Flow of events	1. User opens search section and searches for cars by his/her current location. 2. System shows the results.
Exit conditions	User gets the list of available cars and their location
Exceptions	If user doesn't have internet connection or the GPS doesn't work properly the system shows the related error messages.

5.4. Search for available cars by a given address

Actors	● User ● System
Enter conditions	● User is logged in. ● Connection to the system.
Flow of events	1. User opens search section and searches for cars by a given a address. 2. System lets the user insert the address. 3. User inserts the address and does the search. 4. System shows the results.
Exit conditions	User gets the list of available cars and their location.
Exceptions	● If user doesn't have internet connection the system shows an error message. ● If the address could not be found system shows an error message about it and suggests the user to reenter it.

5.5. Reservation

Actors	● User ● System
Enter conditions	● User searched for available cars. ● There's at least one available car. ● Connection to the system
Flow of events	1. User chooses one of the cars provided by the search. 2. Users reserves the car and the systems asks him to confirm the reservation operation. 3. The user confirms the intention to reserve the car. 4. The system notifies the user of successful reservation.
Exit conditions	● The car is reserved. ● User gets the notification of successful reservation.

Exceptions	● If the car chosen by user is taken before he could reserve it, the system shows an error message about it and suggests the user to choose another car. Reservation is not done.

5.6. Car unlocking

Actors	● User ● System
Enter conditions	● User reserved the car within one hour. ● The user is within the distance defined to unlock the car. ● Connection to the system.
Flow of events	1. User communicates his/her location to the system. 2. The system unlocks the car and sends to the user the car engine ignition code.
Exit conditions	● The system unlocks the car. ● The system sends to the user a notification with the car engine ignition code.
Exceptions	If user is too far from the car, the system shows an error message about it and asks to the user to come closer to the car. Car is not being unlocked.

5.7. Car engine ignition

Actors	● User ● System
Enter conditions	● The user is inside the car (seen by car control software). ● The user knows the car engine ignition code received by the system.
Flow of events	1. User opens the engine ignition code section on the car display. 2. The car control software asks for engine ignition code. 3. User inserts the ignition code. 4. System ignites the engine.
Exit conditions	Car’s engine is ignited.

Exceptions	If user inserted wrong code system shows an error message about it and doesn't ignite the engine. The user can retry the code insertion.

5.8. Employee login

Actors	● Employee ● System
Enter conditions	● Employee's account exists. ● Connection to the system.
Flow of events	1. Employee opens the login section. 2. System asks for login and password. 3. Employee inserts login and password. 4. System logs user in.
Exit conditions	Employee is logged in.
Exceptions	If employee inserted wrong email or password system shows a message about it and suggests to try to log in once again.

5.9. Employee account creation

Actors	● Administrator ● System
Enter conditions	● Administrator is logged in. ● Connection to the system.
Flow of events	1. Administrator opens the account management section. 2. Administrator chooses the account creation operation. 3. Administrator inserts information about the new account. 4. System asks for confirmation. 5. Administrator confirms the creation. 6. System notifies administrator about successful account creation.
Exit conditions	● New employee account is created. ● Administrator gets the notification of successful account creation.

Exceptions	If administrator inserted information about new account which does not fit to given format, the system shows an error message about it and doesn't let administrator proceed with account creation until correct information is not inserted.

5.10. Employee account modification

Actors	● Administrator ● System
Enter conditions	● Administrator is logged in. ● Connection to the system.
Flow of events	1. Administrator opens the account management section. 2. Administrator chooses the account modification operation. 3. Administrator chooses the account to be modified. 4. Administrator modifies the account data. 5. System asks for confirmation. 6. Administrator confirms the modification. 7. System notifies administrator about successful account modification.
Exit conditions	● Employee account is modified. ● Administrator gets the notification of successful account modification.
Exceptions	If administrator changed information to values, which do not fit to given format, the system shows error messages related to them and doesn't let administrator proceed with account modification until correct values are not inserted.

5.11. Employee account removal

Actors	● Administrator ● System
Enter conditions	● Administrator is logged in. ● Connection to the system.
Flow of events	1. Administrator opens the account management section. 2. Administrator chooses the account removal operation. 3. Administrator chooses account to be removed. 4. System asks for confirmation. 5. Administrator confirms the removal. 6. System notifies administrator about successful account removal.
Exit conditions	● Employee account is removed. ● Administrator gets the notification of successful account removal.
Exceptions	If the employee is logged in when the administrator try to delete his/her account the system shows an error messages and the account deletion fails.

5.12. Car registration

Actors	● Manager ● System
Enter conditions	● Manager is logged in. ● Connection to the system
Flow of events	1. Manager opens the car management section. 2. Manager chooses the car registration operation. 3. Manager inserts information about the new car. 4. System asks for confirmation. 5. Manager confirms the registration. 6. System notifies manager about successful car registration.
Exit conditions	● New car record is added. ● Manager gets the notification of successful car registration.

Exceptions	If manager inserted information about new car, which does not fit to given format, the system shows an error message about it and doesn't let manager proceed with car record creation until correct information is not inserted.

5.13. Car record modification

Actors	● Manager ● System
Enter conditions	● Manager is logged in. ● Connection to the system.
Flow of events	1. Manager opens the car management section. 2. Manager chooses the car record modification operation. 3. Manager chooses the record to be modified. 4. Manager modifies car data. 5. System asks for confirmation. 6. Manager confirms the modification. 7. System notifies manager about successful record modification.
Exit conditions	● Car record is modified. ● Manager gets the notification of successful car record modification.
Exceptions	If manager changed information to values, which do not fit to given format, system prompts error message about it and doesn't let manager proceed with car record modification until correct values are not inserted.

5.14. Car record removal

Actors	● Manager ● System
Enter conditions	● Manager is logged in. ● Connection to the system
Flow of events	1. Manager opens the car management section. 2. Manager chooses the car record removal operation. 3. Manager chooses record to be removed. 4. System asks for confirmation. 5. Manager confirms the removal. 6. System notifies manager about successful record removal.
Exit conditions	● Car record is removed. ● Manager gets the notification of successful record removal.
Exceptions	If the car is reserved or is doing a ride when the manager tries to remove its record, the system shows an error message about it and doesn't let manager proceed with car record removal until car is not checked to be available.

5.15. Change of price per minute

Actors	● Manager ● System
Enter conditions	● Manager is logged in. ● Connection to the system
Flow of events	1. Manager opens the price per minute management section. 2. Manager chooses the price change operation. 3. Manager inserts the new price. 4. System asks for confirmation. 5. Manager confirms the change. 6. System notifies the manager about successful price change.
Exit conditions	● Price per minute is changed. ● Manager gets the notification of successful price change.

Exceptions	If manager changed price to a value which do not fit to given format, the system shows an error message about it and doesn't let the manager proceed with price change operation until a correct value is not inserted.

5.16. Car service task acceptance

Actors	● Operator ● System
Enter conditions	● Operator is logged in. ● Connection to the system
Flow of events	1. Operator opens the car service tasks section. 2. System shows the set of tasks for cars requiring service. 3. Operator chooses one task. 4. System asks for confirmation. 5. Operator confirms the task. 6. System notifies operator about successful task linkage and removes the task from available tasks set.
Exit conditions	● Task is linked to operator. ● Operator gets the notification of task assignment.
Exceptions	If the task chosen by the operator is taken before he/she could ask to do it, the system shows an error message about this situation and suggests the operator to choose another task to do. Task linkage is not done.

5.17. Car service task completion confirmation

Actors	● Operator ● System ● Car control software
Enter conditions	● Operator is logged in. ● The car task is linked to the operator ● Connection to the system

Flow of events	1. Operator opens the service tasks section. 2. Operator asks to filter on assigned tasks to him/her. 3. System shows the set of tasks assigned to the operator. 4. Operator chooses the task that his/her completed. 5. Operator set that the task has been completed. 6. System asks for confirmation. 7. Operator confirms the completion of the task. 8. System connects to the car control software to check if everything work well and if the car is in good status. 9. Car control software confirms the good status of the car. 10. System marks the car as available and marks the task as completed removing it from the assigned tasks set of the operator.
Exit conditions	● Car is marked as available. ● Task is marked as completed. ● Task is removed from assigned tasks set of the operator.
Exceptions	If car control software doesn't confirm a good car status, system sends a notification to the operator which includes car conditions description and says to the operator that the task is not yet finished.

5.18. Car service task waiver

Actors	● Operator ● System
Enter conditions	● Operator is logged in. ● At least one task is assigned to the operator. ● Connection to the system
Flow of events	1. Operator opens the service tasks section. 2. Operator asks to filter on assigned tasks to him/her. 3. System shows the set of tasks assigned to the operator. 4. Operator chooses the task that he/she wants to waive. 5. Operator waives the assigned task. 6. System asks for confirmation. 7. Operator confirms the waiver of the task. 8. The system notifies operator about successful task waiver and makes the task available to other operators.

Exit conditions	• The task is unlinked to the operator and it is removed from assigned tasks set of the operator. • The task comes back available again to other operators.
Exceptions	There are no exceptional situations.

6. UML models

6.1. Use case diagrams

6.1.1. User

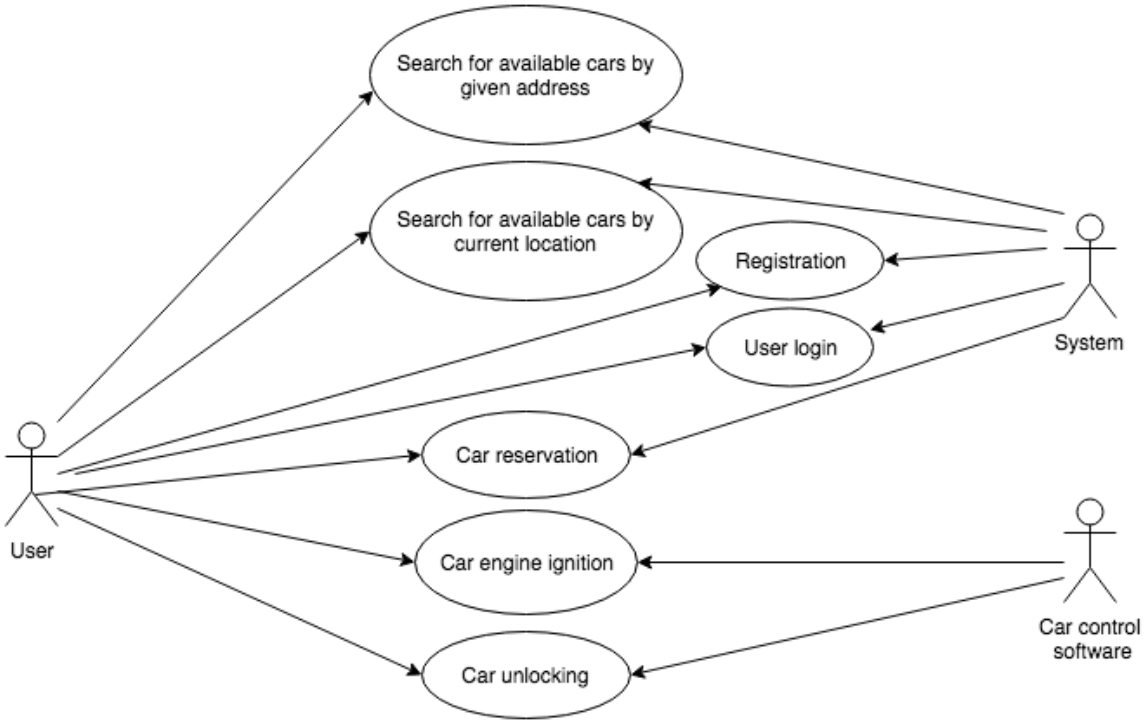


Image 7: User use case diagram.

6.1.2. Administrator

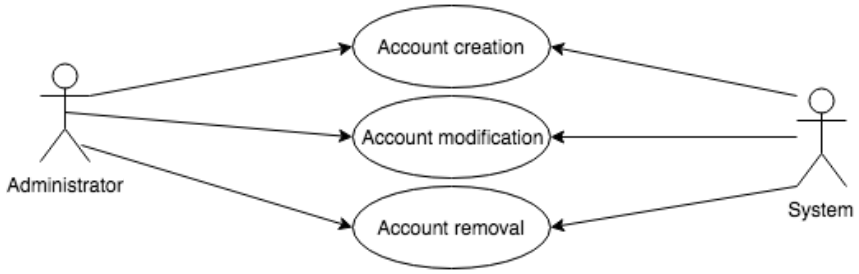


Image 8: Administrator use case diagram.

6.1.3. Manager

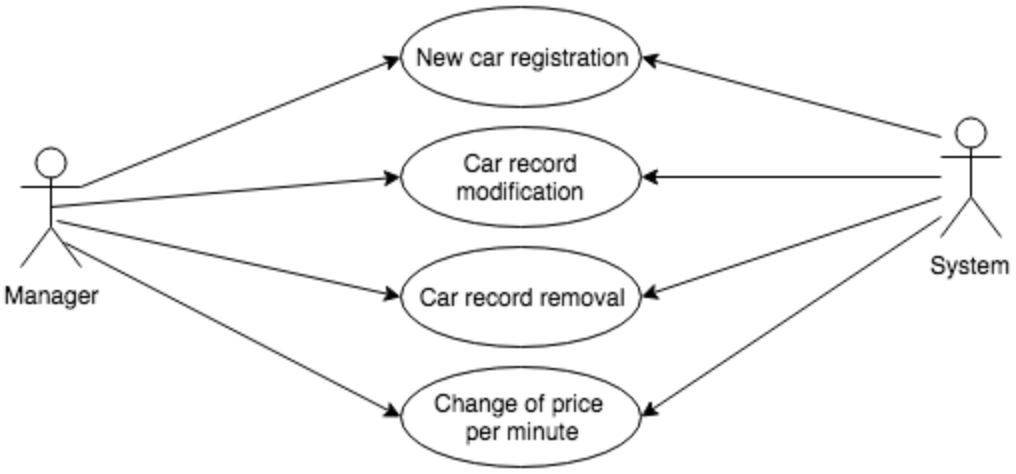


Image 9: Manager use case diagram.

6.1.4. Operator

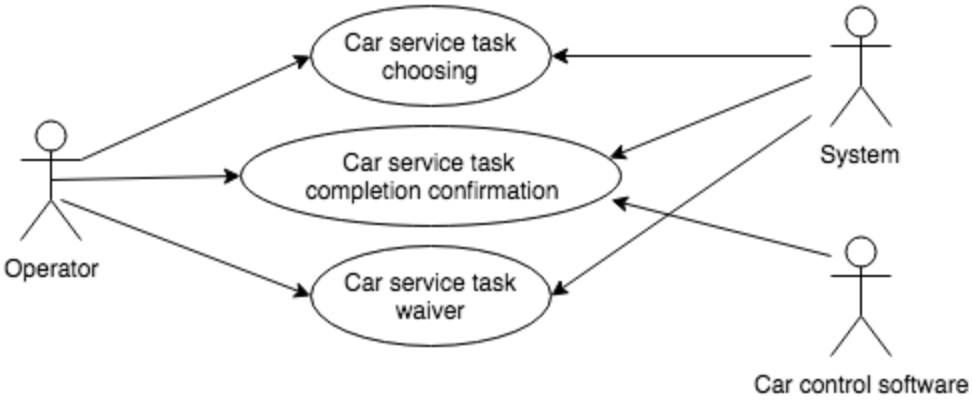


Image 10: Operator use case diagram.

6.3. Class diagram

The following class diagram represent the first draw of the system classes structure.

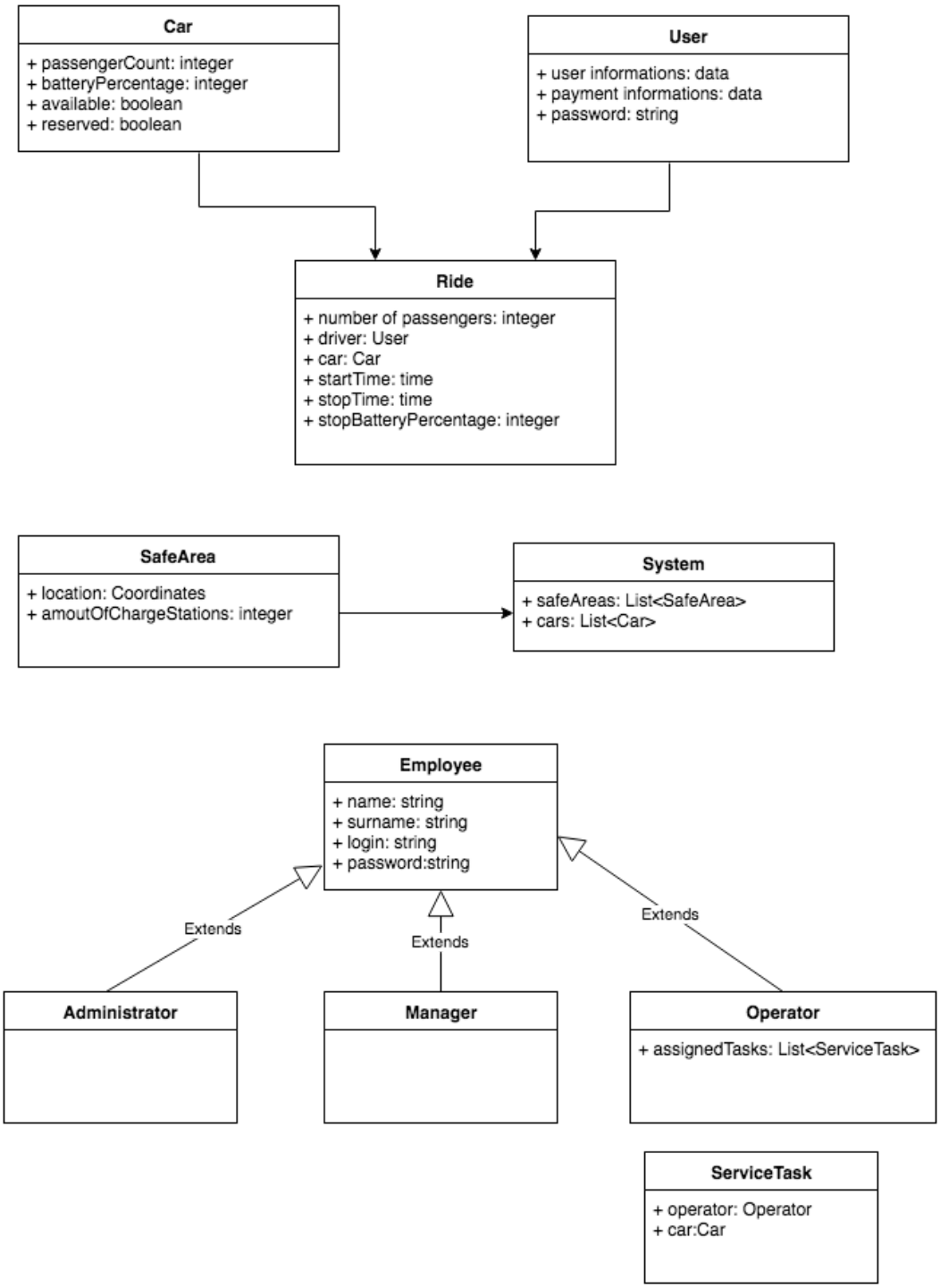


Image 11: System class diagram.

6.4. Sequence diagrams

6.4.1. Car unlocking

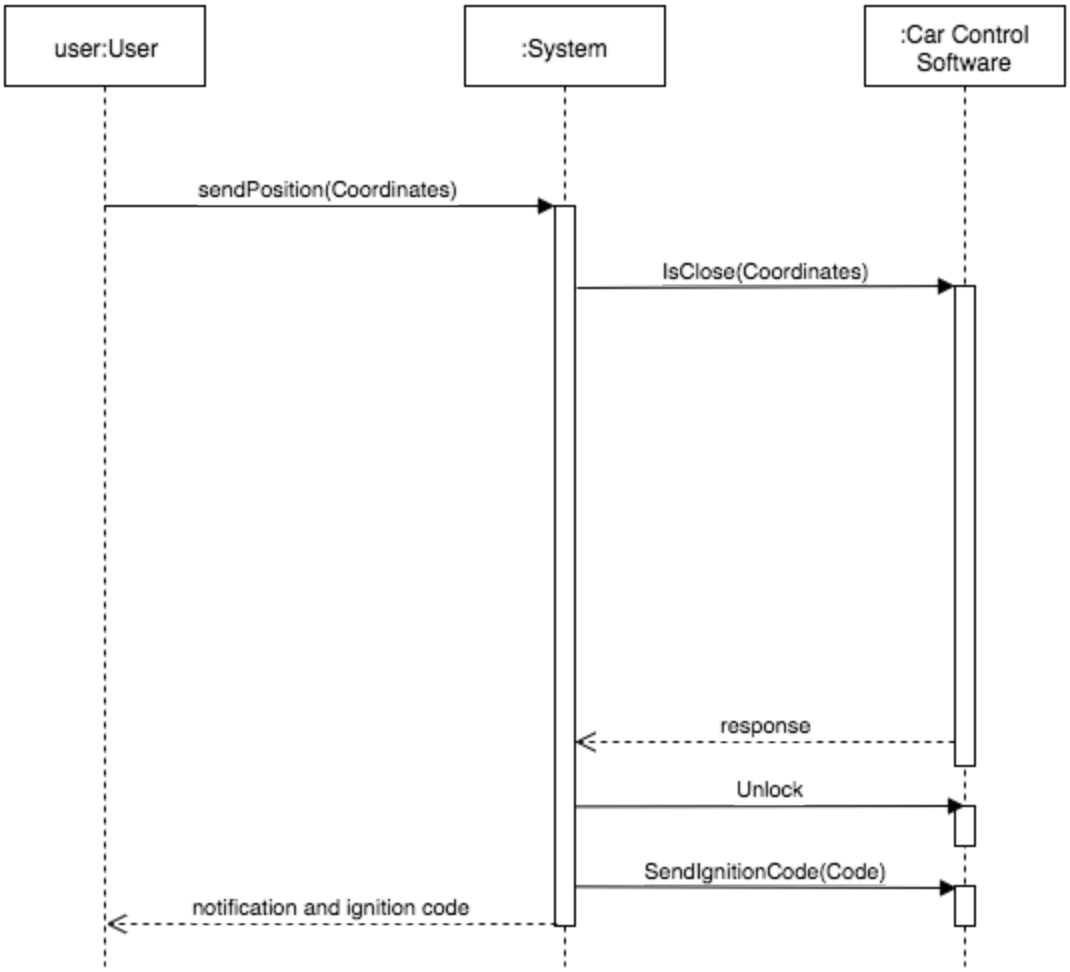


Image 12: Car unlocking sequence diagram.

6.4.2. Car service completion confirmation

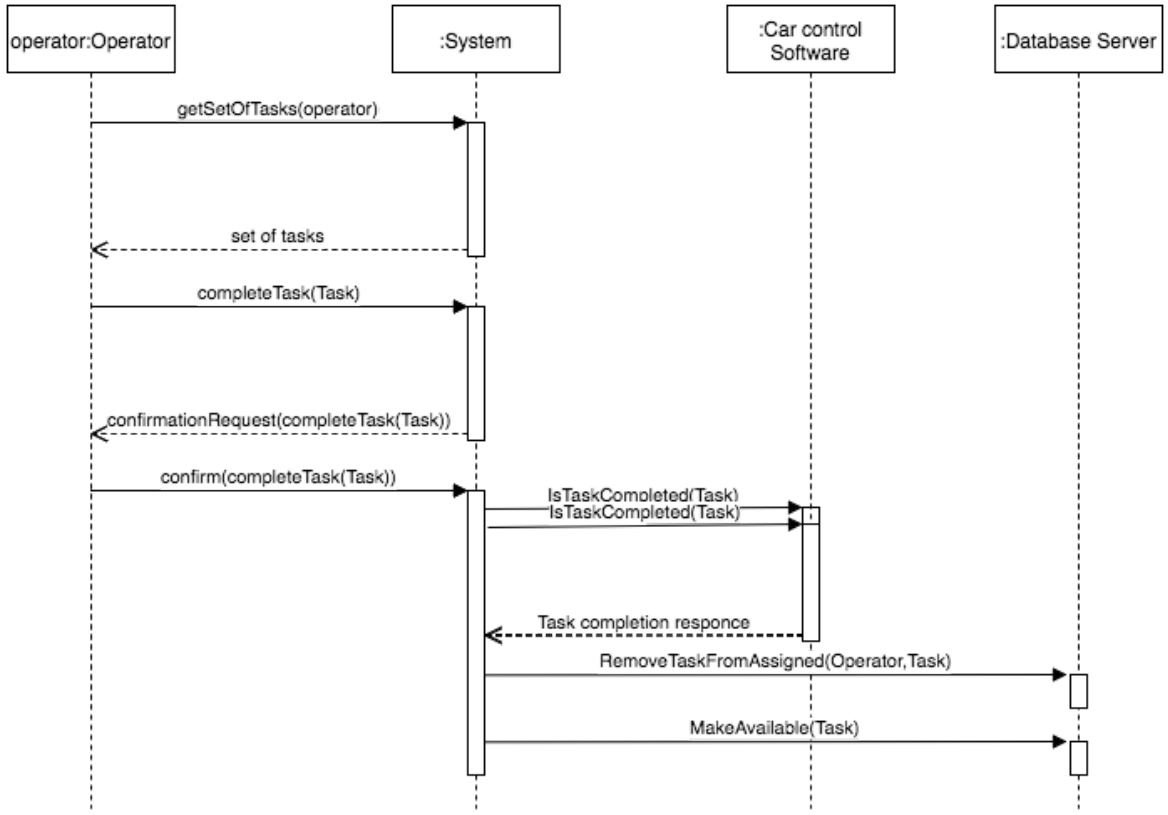


Image 13: Car service completion confirmation sequence diagram.

6.5. Activity diagrams

6.5.1. Search for available cars by given address

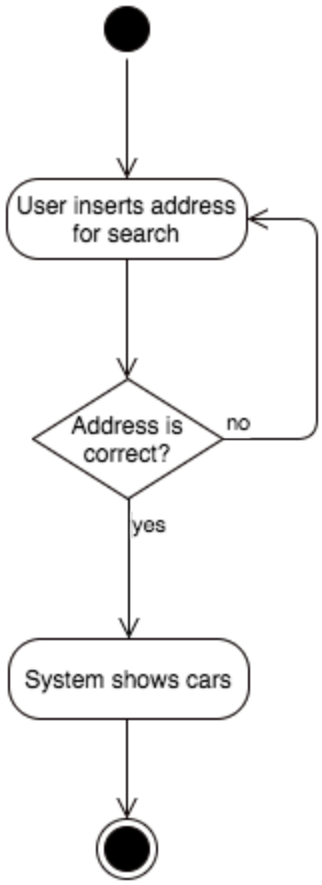


Image 14: Search for available cars by given address activity diagram.

6.5.2. Car reservation

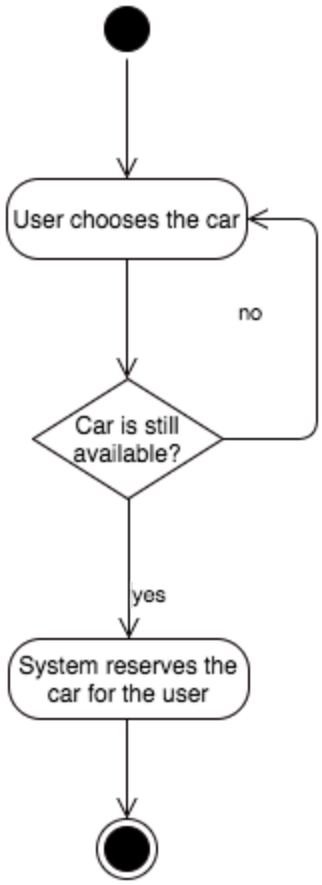


Image 15: Car reservation activity diagram.

6.6. State diagrams

6.6.1. Car state diagram

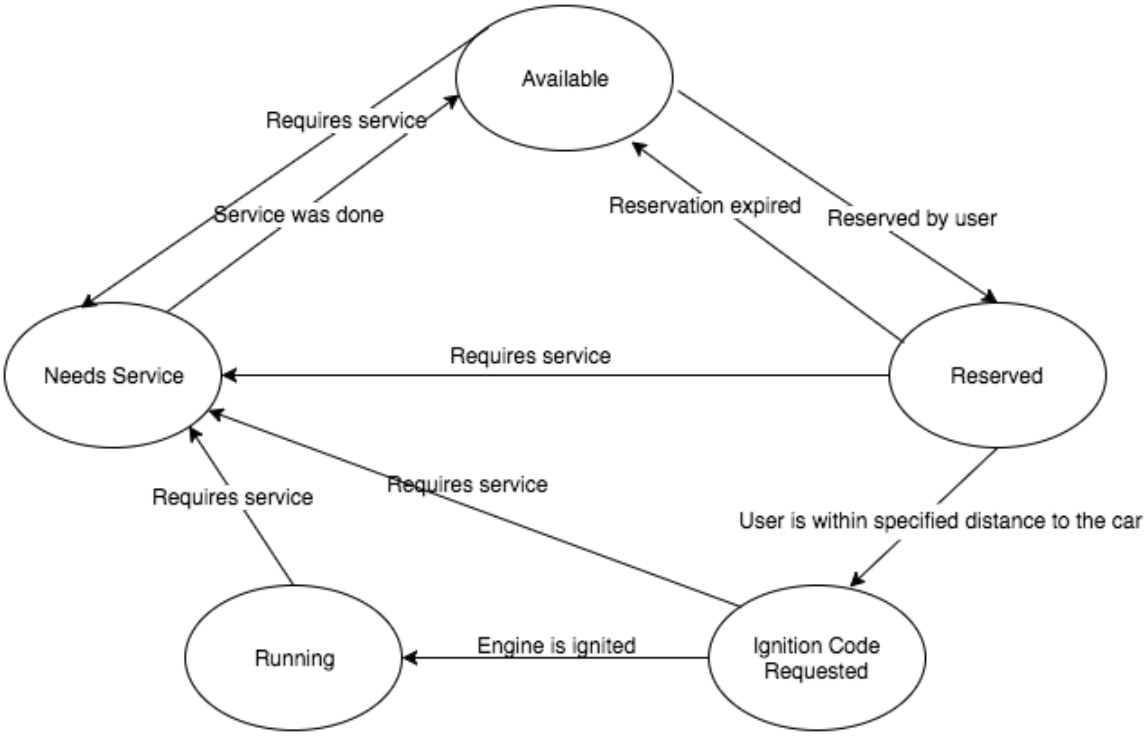


Image 16: Car state diagram.

7. Alloy modeling

Thanks to the alloy modelling language we can use the Alloy Analyzer to try to understand if the hypnotized classes can be consistent.
It is reported the code used to define the model and the

7.1. Model

```
//Helper signatures
enum Bool {True, False}

sig Position {
}

sig Plug {
}

sig Timestamp {
time:Int
}

//Actors
one sig ControlUnit {
contents: set SafeArea,
cars: set Car,
states: set State,
state: Car->one State,
statesCar:State->one Car,
users: set User,
pricePerMinute: Int
}{
state=~statesCar
}

fact {all u: User | u in ControlUnit.users
all c: Car | c in ControlUnit.cars
all st: State | st in ControlUnit.states
}
sig User {
```

```
userCode: Int
}{
userCode > 0
}

abstract sig Employee {
employeeId: Int,
}{
employeeId > 0
}

sig Manager extends Employee {
}

sig Operator extends Employee {
assignedtasks: set ServiceTask
}

one sig Administrator  extends Employee {
}

sig SafeArea {
borderPoints: set Position
}
sig PowerGridStation extends SafeArea{
}

sig Car {
carCode: Int
}{
carCode > 0
}

//Discounts and Overcharges
abstract sig PriceModifier{
}
abstract sig ChargePriceModifier extends PriceModifier{
}
one sig LowBatteryOvercharge extends ChargePriceModifier{
}
one sig FarFromPowerGridOvercharge extends PriceModifier{
}
```

```
one sig SufficientButteryDiscount extends ChargePriceModifier{
}
one sig CarLeftPlaggedDiscount extends ChargePriceModifier{
}
one sig PassagersDiscount extends PriceModifier{
}

//Fees
abstract sig Fee{
}
one sig NonSafeAreaParkingFee extends Fee{
}


//Car states
abstract sig State{
//car: one Car
}
sig ServiceTask extends State{
operator: lone Operator
}
sig Available extends State{
}
abstract sig Activity extends State{
user: one User
}

sig Reservation extends Activity{
startTime: one Timestamp
}

sig CodeRequest extends Activity{
//reservation:one Reservation
}

sig Ride extends Activity{
startTime: one Timestamp,
endTime: one Timestamp,
passQuantity: Int,
leftOutside:Bool,
PercentOfChargeLeft:one Int,
CarPluggedIn:one Bool,
Price:Int,
```

```
FeesToBeApplied: set Fee,
priceModifiersToBeApplied: set PriceModifier,
//ended: Bool
WithinSafeArea:lone SafeArea,
DistanceToPowerGridStation: Int//3 means 3 kilometers
}{
passQuantity>=0
Price>0
CarPluggedIn=True implies WithinSafeArea=PowerGridStation
}


//>>>>>Facts<<<<<

//Internal consistence constraints
fact {no disj a1,a2:Activity|a1.user=a2.user}
fact                        {all                          cu:ControlUnit,
c1,c2:Car|c1->(c1.(cu.state))=c2->(c2.(cu.state)) implies c1=c2}
fact {all r:Ride|r.startTime.time<r.endTime.time}
fact  {all  o:Operator,  st:ServiceTask|(st  in  o.assignedtasks)<=>
(st.operator=o)}
fact         {all         r:Ride|0<=r.PercentOfChargeLeft         and
r.PercentOfChargeLeft<=7}//mapping  of percents for alloy. 2 equals to
20 percent, 4 equals to 50 percent

//Identifiers uniqueness facts
fact userCodeUniqueness {
all u1,u2: User | (u1 != u2) => u1.userCode != u2.userCode
}

fact carCodeUniqueness {
all c1,c2: Car |(c1 != c2) => c1.carCode != c2.carCode
}

fact employeeIdUniqueness {
all e1,e2: Employee | (e1 != e2) => e1.employeeId != e2.employeeId
}

//Discounts, overcharges and fees facts. They will be applied if ride
ends at that moment
fact{all         r:Ride|         (r.PercentOfChargeLeft<=2         and
r.CarPluggedIn=False)<=>            LowBatteryOvercharge            in
r.priceModifiersToBeApplied}
```

```
fact{all       r:Ride|       (r.PercentOfChargeLeft>=4              and
r.CarPluggedIn=False)<=>SufficientButteryDiscount                   in
r.priceModifiersToBeApplied}
fact{all   r:Ride|   r.CarPluggedIn=True<=>CarLeftPlaggedDiscount   in
r.priceModifiersToBeApplied}
fact{all     r:Ride|     r.passQuantity>=2<=>PassagersDiscount      in
r.priceModifiersToBeApplied}
fact{all   r:Ride|   r.WithinSafeArea=none<=>NonSafeAreaParkingFee   in
r.FeesToBeApplied}
fact{all                                                        r:Ride|
r.DistanceToPowerGridStation>3<=>FarFromPowerGridOvercharge         in
r.priceModifiersToBeApplied}

//>>>>>Predicates<<<<<

// Predicate about changing state

pred changingState[cu, cu':ControlUnit, c: Car] {
c->(c.(cu.state))=c->(Available)   implies   cu'.state=   cu.state   -
c->(c.(cu.state))   +   c->(Reservation)   or   cu'.state=   cu.state   -
c->(c.(cu.state))  + c->(ServiceTask)
c->(c.(cu.state))=c->(Reservation)   implies   cu'.state=   cu.state   -
c->(c.(cu.state))   +   c->(Available)   or   cu'.state=   cu.state   -
c->(c.(cu.state))    +  c->(CodeRequest)  or  cu'.state=  cu.state  -
c->(c.(cu.state))  + c->(ServiceTask)
c->(c.(cu.state))=c->(CodeRequest)   implies   cu'.state=   cu.state   -
c->(c.(cu.state))   +   c->(Available)   or   cu'.state=   cu.state   -
c->(c.(cu.state))       +   c->(Ride)   or   cu'.state=   cu.state   -
c->(c.(cu.state))  + c->(ServiceTask)
c->(c.(cu.state))=c->(Ride)    implies    cu'.state=    cu.state    -
c->(c.(cu.state))   +   c->(Available)   or   cu'.state=   cu.state   -
c->(c.(cu.state))  + c->(ServiceTask)
c->(c.(cu.state))=c->(ServiceTask)   implies   cu'.state=   cu.state   -
c->(c.(cu.state)) + c->(Available)
}
```

```
//>>>>>Assertions<<<<<
assert NoDoubleReservation{
no disj r1,r2:Reservation|r1.user=r2.user
}
check NoDoubleReservation

assert NoMultipleChargePriceModifiers{
no r:Ride, cpm1,cpm2:ChargePriceModifier| cpm1 in
r.priceModifiersToBeApplied and cpm2 in r.priceModifiersToBeApplied
and cpm1!=cpm2
}
check NoMultipleChargePriceModifiers


pred show {
#Car = 5
#Ride= 2
some r: Ride | r.CarPluggedIn=True
}
run changingState for 5
run show for 5
```

7.2. Alloy result

The built model is consistent as the alloy results prove.

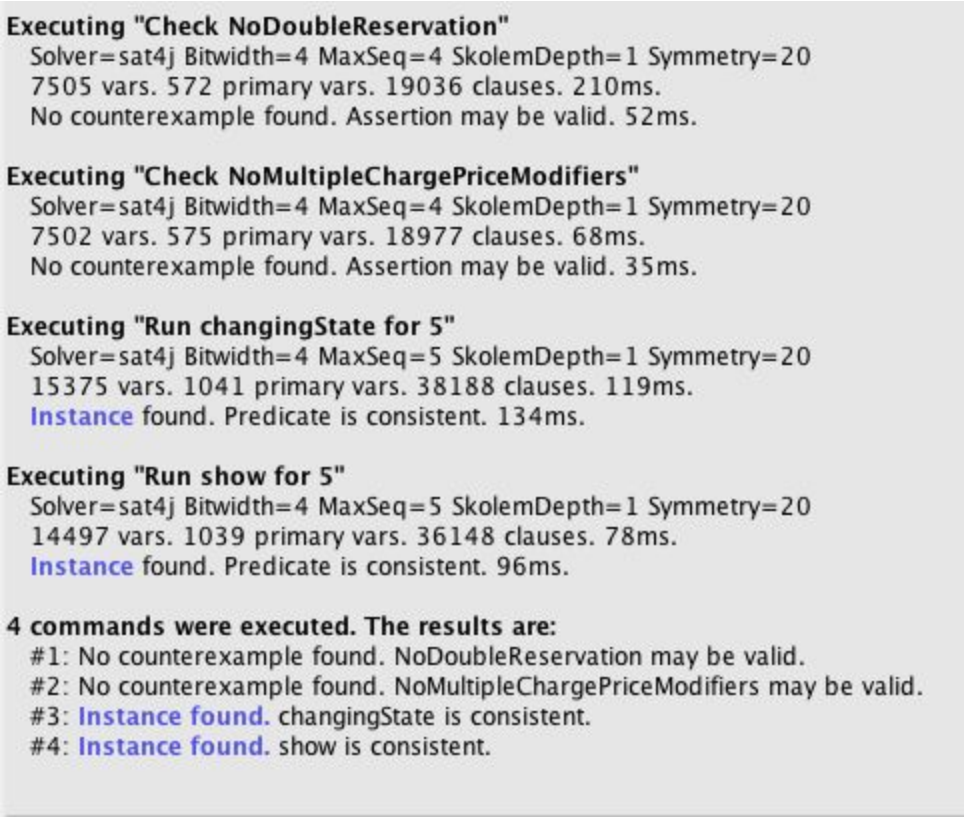
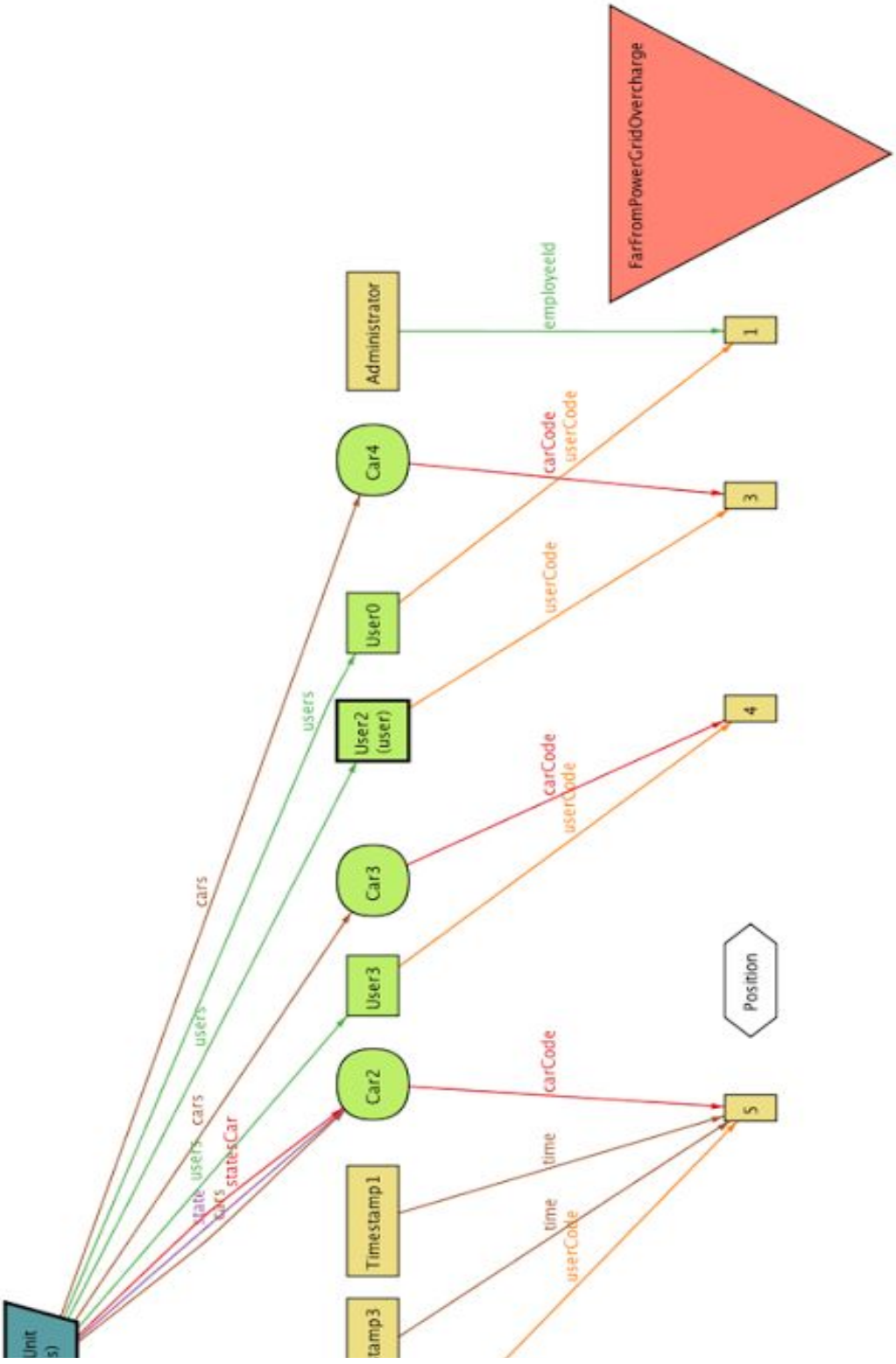


Image 17: Alloy results.

7.3. World generated



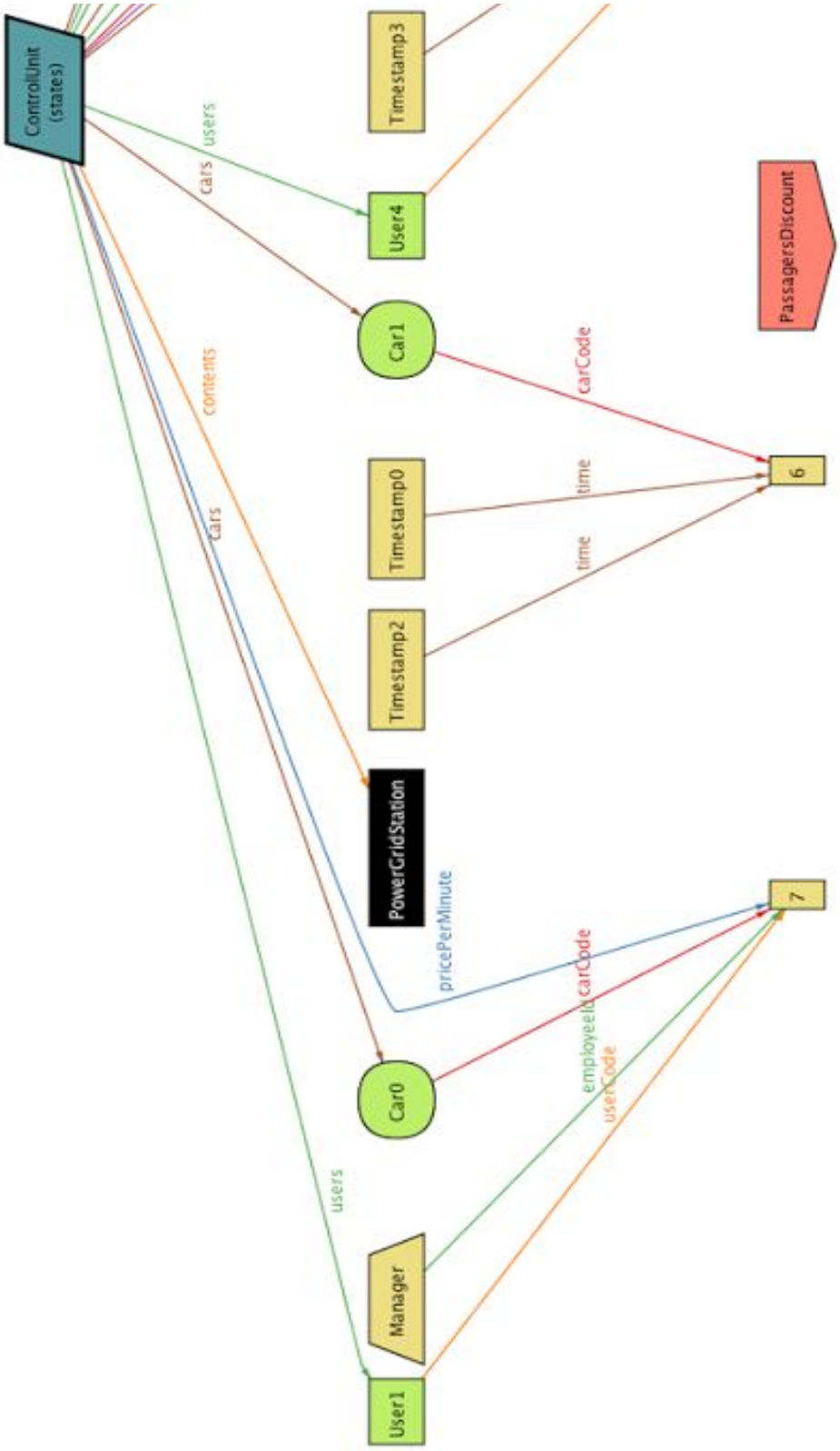


Image 18: Alloy generated world.

8. Future developments

The followings could be future developments.

- Provide payments via PayPal
- Let the user to park a car with a reservation so that he/she can comes back to the car and continue the ride without that other user can reserve and pick up the car.

9. Used tools

- GitHub: for version control
- Draw.io: to create diagrams
- Google Docs: to write the document
- Adobe Photoshop: for interfaces samples
- Alloy Analyzer 4.2: to prove the consistency of our model.

10. Hours of work

10.1. Elena Denchenko

- 21/10/2016: 3h
- 25/10/2016: 3h
- 27/10/2016: 1h
- 29/10/2016: 2h 30min
- 03/11/2016: 2h
- 06/11/2016: 3h
- 09/11/2016: 30 min
- 10/11/2016: 4h
- 12/11/2016: 2h

10.2. Stefano Cilloni

- 21/10/2016: 3h
- 25/10/2016: 3h
- 26/10/2016: 2h
- 29/10/2016: 2h 30min
- 04/11/2016: 1h
- 06/11/2016: 3h
- 09/11/2016: 30 min
- 10/11/2016: 4h
- 12/11/2016: 2h

11. Changelog

- v1.1
    - Restructured proposed system architecture
    - Improved regulatory policies

- v1.2
    - Alloy model improvements
    - User interfaces samples error correction