

---

# Software Requirements Specification

for

**<MálaCar>**

Version 1.0 approved

Prepared by <Cergan Radu-Mihai>  
<Plaisanu Alexandru-Ciprian>  
<Micu Cristian-Mihai>

<Faculty of Automatics, Computers and Electronics>

<05.03.2021>

# Table of Contents

<b>Table of Contents</b>	<b>ii</b>
<b>Revision History</b>	<b>ii</b>
<b>1. Introduction</b>	<b>1</b>
1.1 Purpose	1
1.2 Document Conventions	1
1.3 Intended Audience and Reading Suggestions	1
1.4 Product Scope	1
1.5 References	1
<b>2. Overall Description</b>	<b>2</b>
2.1 Product Perspective	2
2.2 Product Functions	2
2.3 User Classes and Characteristics	2
2.4 Operating Environment	2
2.5 Design and Implementation Constraints	2
2.6 User Documentation	2
2.7 Assumptions and Dependencies	3
<b>3. External Interface Requirements</b>	<b>3</b>
3.1 User Interfaces	3
3.2 Hardware Interfaces	3
3.3 Software Interfaces	3
3.4 Communications Interfaces	3
<b>4. System Features</b>	<b>4</b>
4.1 System Feature 1	4
4.2 System Feature 2 (and so on)	4
<b>5. Other Nonfunctional Requirements</b>	<b>4</b>
5.1 Performance Requirements	4
5.2 Safety Requirements	5
5.3 Security Requirements	5
5.4 Software Quality Attributes	5
5.5 Business Rules	5
<b>6. Other Requirements</b>	<b>5</b>
<b>Appendix A: Glossary</b>	<b>5</b>
<b>Appendix B: Analysis Models</b>	<b>5</b>
<b>Appendix C: To Be Determined List</b>	<b>6</b>

## **Revision History**

<b>Name</b>	<b>Date</b>	<b>Reason For Changes</b>	<b>Version</b>

# **1. Introduction**

## **1.1 Purpose**

The purpose of this document is to give a detailed description of the requirements for Car Rental software application. It will illustrate the purpose and complete declaration for the development of the system. It will also explain system constraints, interface and interactions with other external applications. This SRS document is primarily intended to be proposed to a customer for its approval and a reference for developing the first version of the system for the development team.

## **1.2 Document Conventions**

The abbreviations used in this document are:

- DB(Database)
- UI(User Interface)
- IDE(Integrated Development Environment)
- APP(Application)
- SRS(Software Requirement Specification)
- SA(Software Application)
- JVM(Java Virtual Machine)
- PC(Personal Computer)

## **1.3 Intended Audience and Reading Suggestion**

The SRS document is written for a more general audience, this document is intended for individuals directly involved in the development of Car Rental Application: MálaCar. This includes software developers, testers, project managers, teachers, users and team members. This document need not be read sequentially. Users are encouraged to jump to any section they find relevant.

## **1.4 Product Scope**

MálaCar is a software system that automates the process of managing the car rental activity for a company, which helps people to rent cars at any moment from anywhere based on some preferences like price, car type, car details and more. The website can be accessed by any person, but for renting a car, the access is made through an account, creating one or log in into an existing account.

An administrator uses the web-portal in order to administer the system and keep the information accurate(add a car, delete a car). The administrator can, for instance, verify rented cars and manage user information.

The website needs a good connection to the Internet to display the results. All system information is maintained in a database.

## 1.5 References

<https://www.javatpoint.com/spring-mvc-tutorial>  
<https://spring.io/guides/gs/serving-web-content>  
[https://www.tutorialspoint.com/spring/spring\\_web\\_mvc\\_framework.htm](https://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm)  
<https://docs.oracle.com/javase/tutorial>  
<https://www.w3schools.com/>  
<https://www.codecademy.com/>  
<https://javascript.info/>  
<https://maven.apache.org/>  
<https://wikipedia.org/>

## 2. Overall Description

### 2.1 Product Perspective

This system will consist of two parts: the software application and one web portal. The SA will be used to find cars for rental and view information about them while the web portal will be used for managing the information about the cars and the system as a whole.

Since this is a data-centric product it will need somewhere to store the data. For that, a DB will be used. Both the APP and web portal will communicate with the DB, however in slightly different ways. The SA will only use the DB to get data while the web portal will also add and modify data. All of the database communication will go over the Internet.

### 2.2 Product Functions

The APP has two account types: one as a user(after you register you become a user and you can rent a car or do some other features) and the other one as an admin(which can manage the cars, manage the rental of the cars and more). Also you can enter on the site without being a user, as a guest, but you have some restrictions(limited access).

The functionalities are presented below:

- guests can register
- all types of users can see the available cars for renting
- guests and users can see the details about a car(class, motorization, price, location, etc.)
- all types of users can search a car
- all types of users can set the preferences for listing the cars
- all types of users can access the best deals available
- all types of users can view the renting stations from each city
- all types of users can access the website information page(About Us section)
  
- users can rent a car
- users can see the history of his rented cars

- users can see the reservation details of appointment(date to pick/bring back the car, hour , location, etc.)
- users can be penalized if something happens to the car by his fault(this can lead to a three months suspension ban, by accumulating a number of penalization points)
- users and administrators can log in
- users and administrator can see his profile information
- users and administrator can change the profile information
- users and administrator can log out
- the administrator can see the rented cars
- the administrator can change the details about a car(motorization, price, location, etc.)
- the administrator can modify the preferences for listing the cars
- the administrator can modify the stations(add/delete a station)
- the administrator modify the website information page(About Us section)
- the administrator can add/delete a car

## **2.3 User Classes and Characteristics**

There are three types of users that interact with the system: guests, users and administrators. Each of these three types of users has different uses of the system so each of them has their own features.

The guest and users can only use the application to find a car for renting.

This means that they have to be able to search for a car, choose a car from that search and then rent it(option available only for users). In order for the users to get a relevant search result there are multiple preferences they can specify and all results match all of those.

In order to become a user, a guest must register or log in into his account.

The administrators also only interact with the web portal. They are managing the overall system so there is no incorrect information within it. The administrator can manage the information for each car, station as well as for APP users .

## **2.4 Operating Environment**

The SA can run on different operating systems, the access can be made through a browser. Also you must need the java virtual machine(JVM) in order to convert Java bytecode into machine languages. The version 11.0.8 needs to be supported to access the APP.

For Windows it can be accessed from Windows 7 to the Windows 10 by using the next browsers:

- Chrome(at least version 48.0.2564)
- Firefox (at least version 62.0)
- Opera (at least 15.0)
- Internet Explorer(at least version 8.0)

For macOS it can be accessed from macOS X 10.10 to the macOS 10.13 by using the next browsers:

- Chrome(at least version 48.0.2564)
- Firefox (at least version 62.0)
- Opera (at least 15.0)
- Safari (at least version 8.0)
- 

For Linux it can be accessed from /\*64-bit Ubuntu 14.04+, Debian 8+, Fedora Linux 24+\*/ by using the next browsers:

- Chrome
- Firefox
- Opera

For Smartphones it can be accessed from Android 5.0 and iOS 8.0 but it will be displayed in desktop mode, by using the next browsers:

- Chrome
- Firefox
- Opera
- Safari(for iOS)

## **2.5 Design and Implementation Constraints**

IntelliJ IDE Ultimate license

Adobe Dreamweaver license

Microsoft Teams add-on licenses

## **2.6 User Documentation**

At the end of the project we will provide to the customer a SA, which incorporates DB, the SRS document and if it is needed we can provide help via Microsoft Teams and also we can be contacted on the email.

## **2.7 Assumptions and Dependencies**

The Internet connection is a primary key in order to use the SA. Since the APP fetches data from the database over the Internet, it is crucial that there is an Internet connection for the application to function.

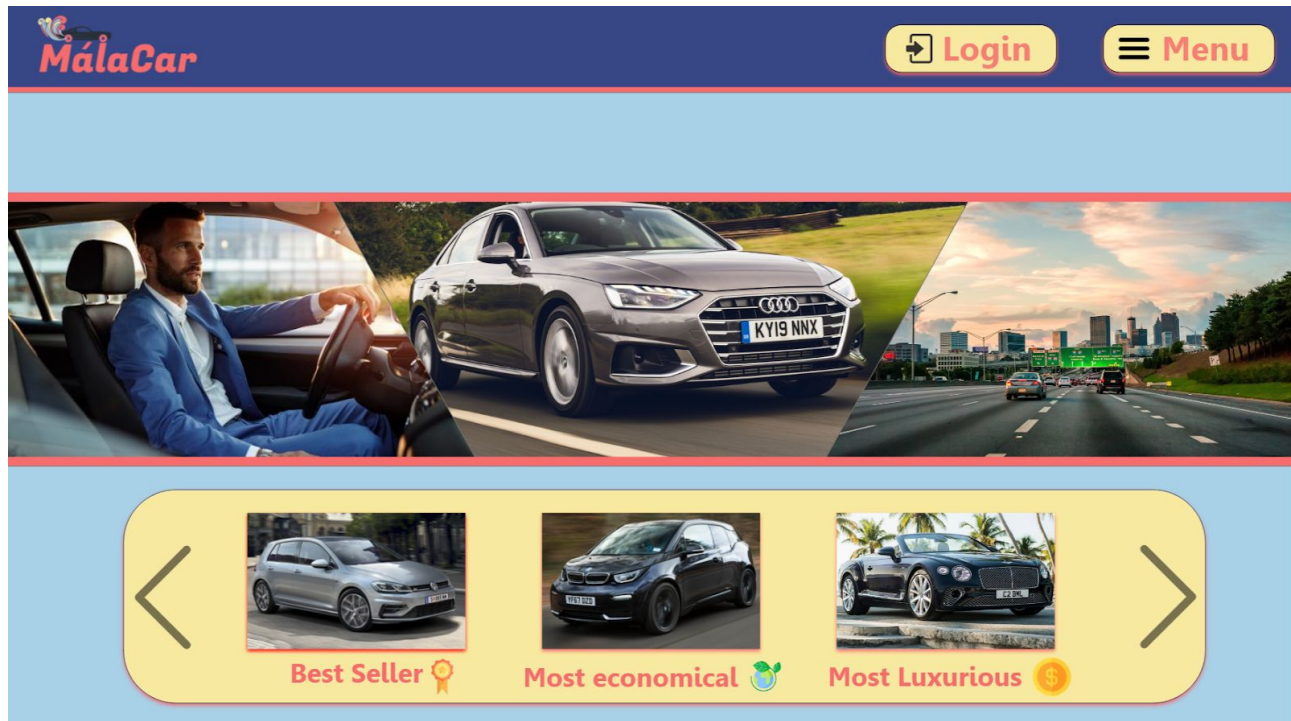
Another assumption about the product is that it can be used on mobile phones and desktops that have enough performance for supporting browsers needed to access the SA and also a fully equipped PC(keyboard, mouse or touchpad in case of laptop).

### 3. External Interface Requirements

#### 3.1 User Interfaces

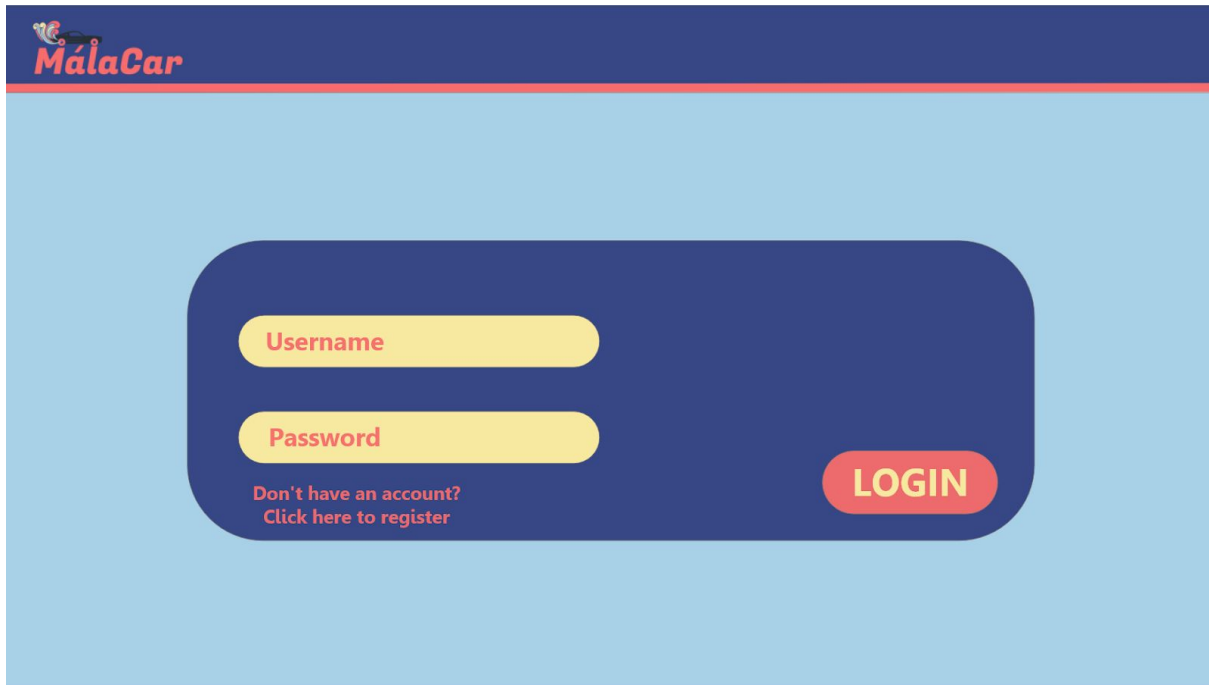
When opening the website, the guest can see the start-up page. This page should act like a welcome page. The guest can see the login button and the menu button, which is a pop-up list of other buttons such as “Rent a Car”, “About us”, etc. If logged in as a user, instead of the login button should be the “My profile” button. In the first page is also presented a set of “Best Deals” that should act like a slider in JS, there are located cars sorted by different criterias such as a best selling car.

Note ! The website’s logo located at the top-left corner of the tab should act like a “Home” button on every page related to our site.



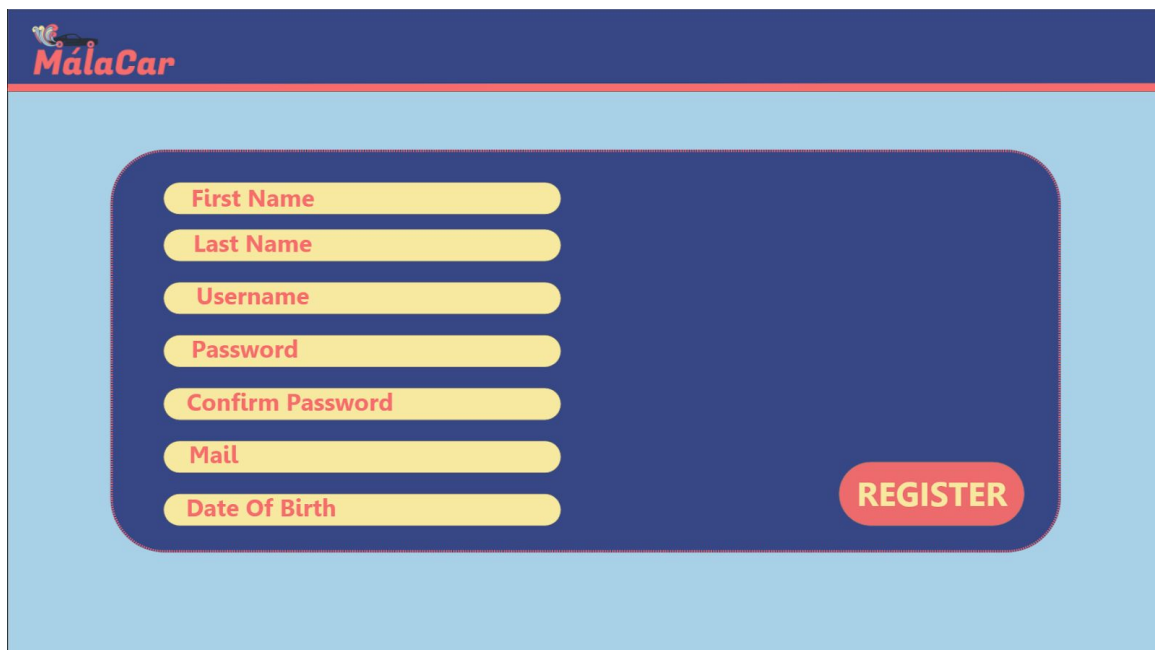


After pressing the “Login” button from the homepage, the following page should appear. If the guest doesn’t have an account, he should press “Don’t have an account? Click here to register”.



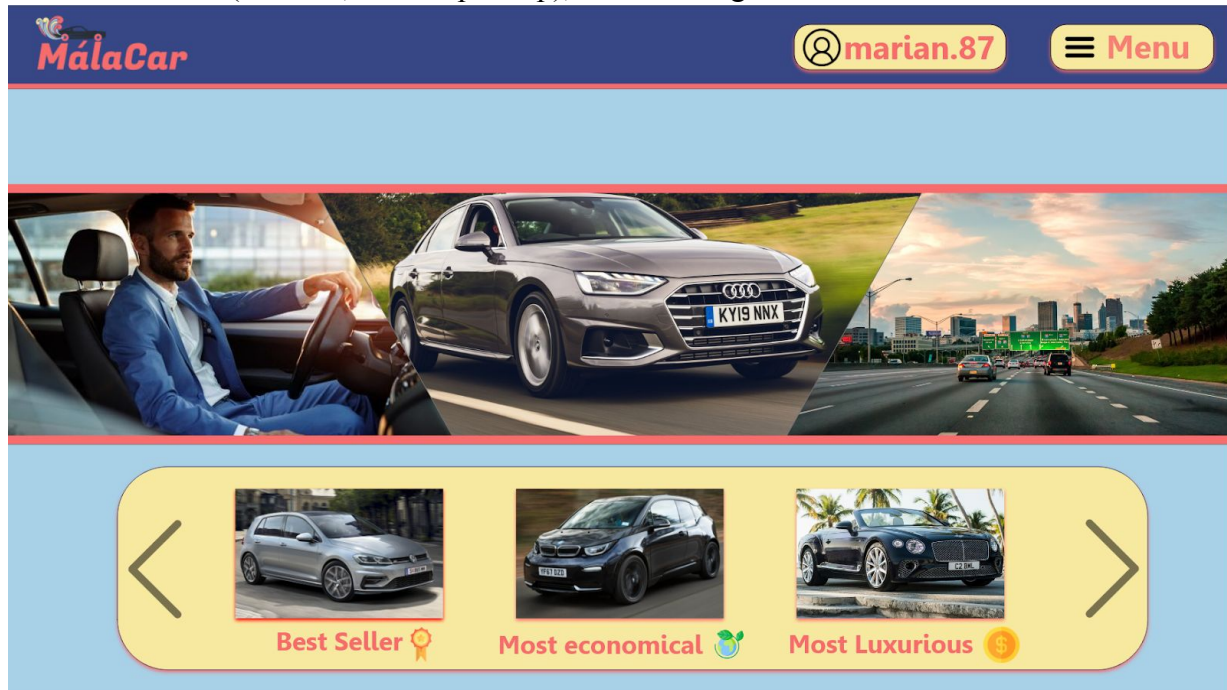
The login form is displayed on a light blue background with a dark blue header containing the MálaCar logo. The form itself is a dark blue rounded rectangle. It contains two yellow input fields labeled 'Username' and 'Password'. To the right of these fields is a red 'LOGIN' button. Below the 'Password' field, there is a link that says 'Don't have an account? Click here to register'.

When the guest presses the “Don’t have an account? Click here to register” button, he will be redirected to the following page, where he must complete all the fields in order to advance. Note that the field “Password” must match with “Confirm Password”. After the registration completes, the guest can log in as a user.

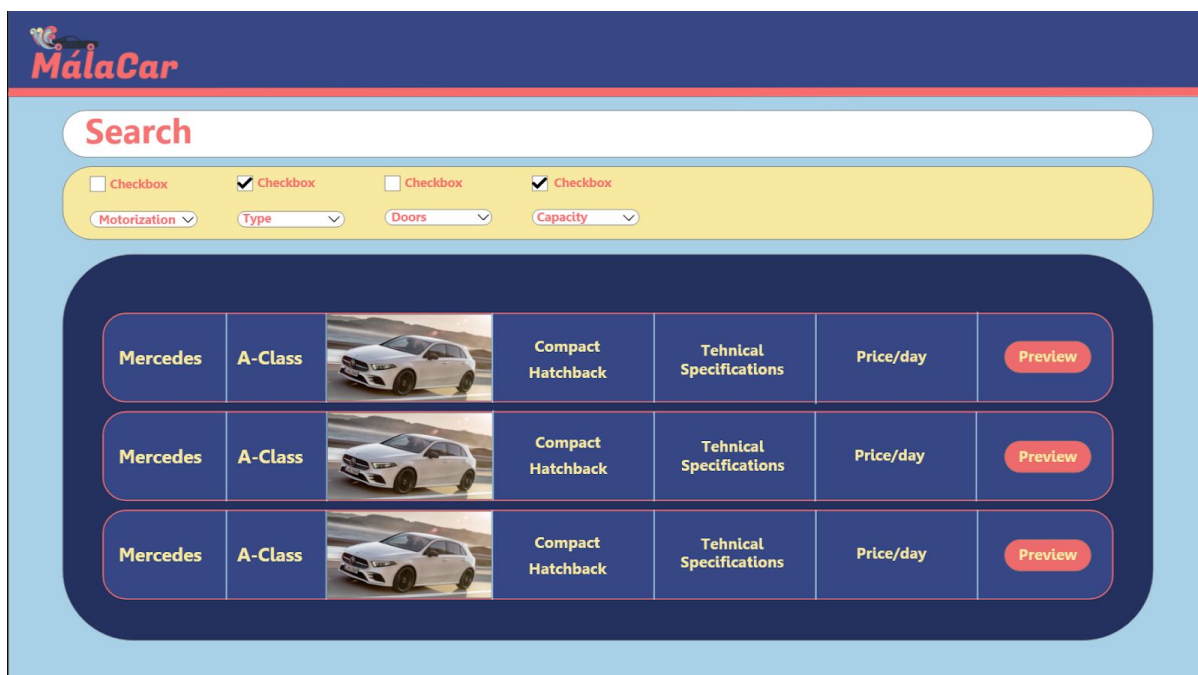


The registration form is displayed on a light blue background with a dark blue header containing the MálaCar logo. The form is a dark blue rounded rectangle. It contains seven yellow input fields labeled 'First Name', 'Last Name', 'Username', 'Password', 'Confirm Password', 'Mail', and 'Date Of Birth'. To the right of these fields is a red 'REGISTER' button.

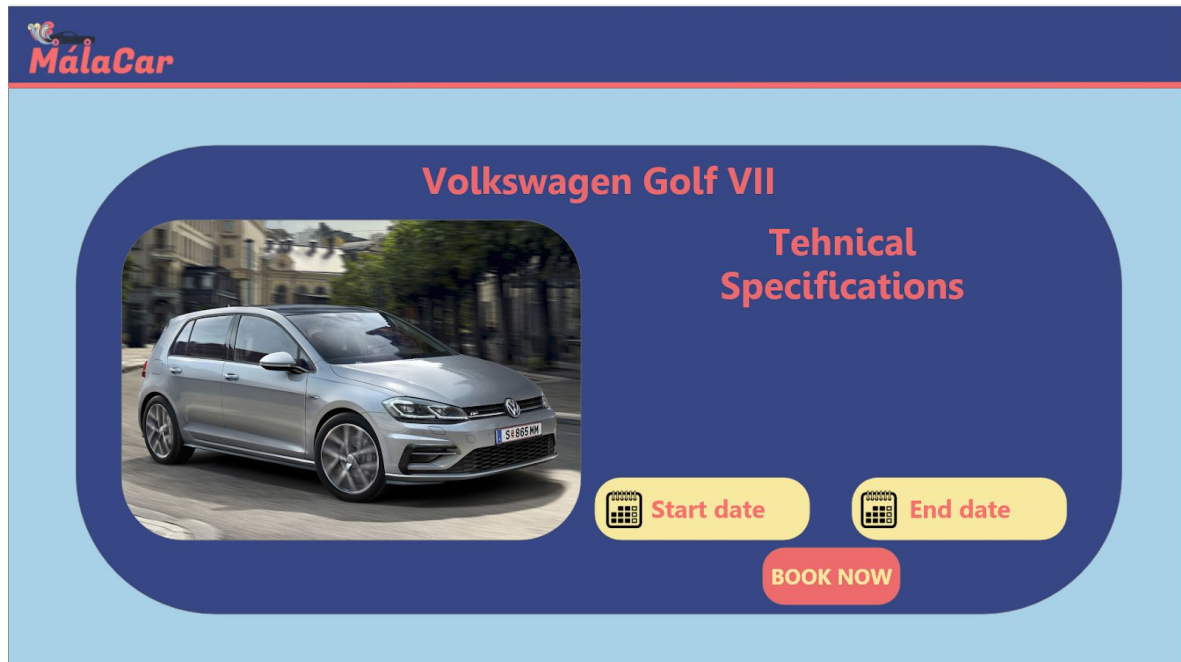
After a successful login, the username should be displayed at the top-right. This should also act as a button too, showing a pop-up list of “My profile” - this leads to the information of the account, where we can change the password, mail, etc. , “History” - where the user can check his renting history and the penalization points, “Reservation” - where the user can see the information about the car he wants to rent(location, hour to pick up), and the “Logout” button.



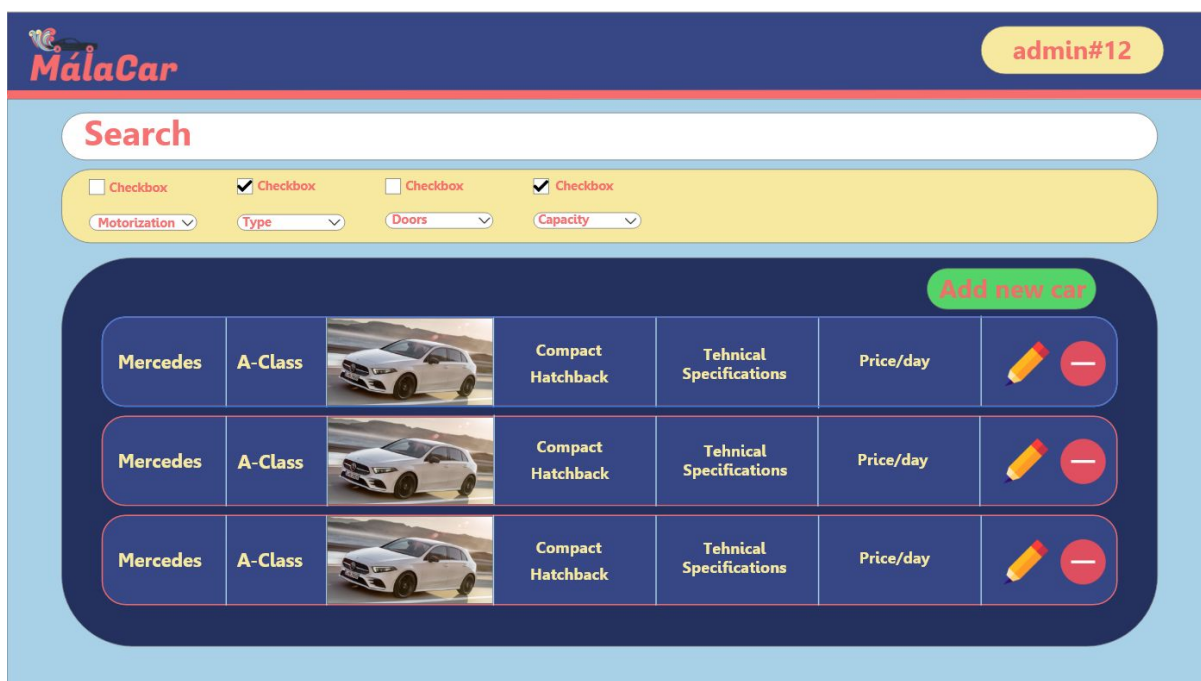
The following page is the “brain” of the website. The user should be able to search for a car regarding some preferences such as type, brand, model, etc. . There will also be listed all the cars, before entering any selecting criteria.



By pressing the “Preview” button from the previous screen, the user should be redirected to another page, where he should be able to see all the features about the car he selected. After completing the two mandatory fields of “start date” and “end date”, the user can press “Book Now” for advancing into the payment section.



Let's not forget that the admin can add, delete or modify the listed cars.



## 3.2 Hardware Interfaces

Not Applicable.

## 3.3 Software Interfaces

The software application communicates with the database in order to get the information about the cars, their status and other specifications.

The communication between the database and the web portal consists of operation concerning both reading, modifying and deleting the data, while the communication between the database and the software application consists of only reading operations.

Also in some cases can show up some messages like username/password incorrect, car rented already, this username/email already exists, date of birth is under 18 years old, the introduced specification doesn't match with any of our cars, the required fields are not completed.

- UserAccount getUserAccount() - get user account from db
- void setUserAccount(UserAccount userAccount) - set user account in db
- Car getCar() - get car from DB
- Car getCar(String type) - get car by type from db
- Car getCar(String model) - get car by model from db
- Car getCar(double price) - get car by price from db
- Car getCar(String motorization) - get car by motorization from db
- Car getCar(int numberOfDoors) - get car by numberOfDoors
- Car getCar(String class) - get car by class
- void setCar(Car car) - set car in db
- Station getStation() - get station from db
- Station getStation(String city) - get station by city from db
- void setStation(Station station) - set station in db

## 3.4 Communications Interfaces

Not Applicable.

# 4. System Features

## 4.1 User Class 1 - The Guest

### 4.1.1 Register

#### 4.1.1.1 Description and Priority

In order to rent a car is mandatory to register through the SA by completing the registration .

High Priority.

#### 4.1.1.2 Stimulus/Response Sequences

The guest must complete the mandatory fields(First name, Last name, Username, Password, Confirm password, E-mail, Date of Birth) in order to register. In case the user or the email already exists, it will display an error message(this username/email already exists).

Another error message will be shown if the introduced Date of Birth is under 18 years old(date of birth is under 18 years old). Also, if the guest does not complete all obligatory fields, he is informed to fill the missing ones.

After the fields are filled up guests must press the “Register” button in order to complete the registration.

#### **4.1.1.3 Functional Requirements**

REQ - 1: After the guest clicked on sign button, will be read from interface 3 parameters(username, email, date of birth);

REQ - 2: The requirements(username, email) will be compared with information from DB;

REQ - 3: If username/email is found in DB, the creation of the account will be denied;

REQ - 4: If date of birth is under 18, the access will be denied;

REQ - 5: If the mandatory fields are not completed, an exception will be thrown up;

REQ - 6: If username and email are not found in the DB, also the date of birth will meet the imposed criteria and the mandatory fields are completed, the account is successfully created;

REQ - 7: Also the entered password must be confirmed in a field below;

REQ - 8: These two need to be matched in order to complete the registration;

REQ - 9: If they do not match an exception will be thrown;

### **4.1.1 Log in**

#### **4.2.1.1 Description and Priority**

In order to use the site for what it is intended(to rent a car), the guest must login.  
High Priority.

#### **4.2.1.2 Stimulus/Response Sequences**

The user must introduce the username and password in order to be logged in into the website. In case one of the required fields is introduced incorrectly, a message will be displayed(username/password incorrect or does not exist).

To complete the action the user must press the button “Log in”.

#### **4.2.1.3 Functional Requirements**

REQ - 1: After the button is pressed, two parameters(username, password) should be read from interface;

REQ - 2: The requirements should be compared with information from DB;

REQ - 3: If the username and password are not traced in the DB the login is not permitted, an exception will be thrown

REQ - 4: If the parameters are found, the access is granted

## 4.2 User Class 2 - The User

### 4.2.1 Search for a car

#### 4.2.1.1 Description and Priority

In order to be able to rent a car the user must navigate to the “Rent a car” page.

High priority.

#### 4.2.1.2 Stimulus/Response Sequences

The user should see a list with available cars and a chooseable list of preferences will be displayed above the cars. In case of some preferences are selected may be a possibility to appear a message(No cars are available according to the selected preferences).

#### 4.2.1.3 Functional Requirements

REQ - 1: After the section “Rent a car” is selected, a list with available cars should appear(cars which are taken from the DB);

REQ - 2: Some preferences(class, brand, model, type, price, motorization, doors) could be selected;

REQ - 3: If there are no selected preferences, the list from REQ - 1 will be unchanged;

REQ - 4: If there are preferences selected will be taken from DB just the cars that meets those criteria;

REQ - 5: The button “Search” should be pressed in order to be displayed the list of cars;

REQ - 6: After pressing the button the list will be updated according to the selected specifications;

### 4.2.2 Rent a car

#### 4.2.2.1 Description and Priority

For renting a car, the user must select a car from the list, then will be redirected on a page, where details will be displayed about it.

High Priority.

#### 4.2.2.2 Stimulus/Response Sequences

Users will see details about the selected car and a button for “Book now”. If the car is not available in the period of time that he selected, it will show up a message(The car is rented already in that period).

#### 4.2.2.3 Functional Requirements

REQ - 1: A button “Preview” is shown and must be pressed

REQ - 2: After that a page will be opened with details about the car and two boxes for renting period time;

REQ - 3: The period of time should be selected for renting the car;

REQ - 4: The button “Book now” must be pressed;

REQ - 5: If the dates introduced are incorrect an exception will be thrown

REQ - 6: He will be redirected on the details about reservation;

REQ - 7: He should select the method of payment;

REQ - 8: After that the reservation is successfully complete;

REQ - 9: After the transaction is completed the rented car will be marked as rented in the DB;

REQ - 10: This means that the car should be removed from the search lists;

REQ - 11: The car should be available after the client will leave the car in any station;

REQ - 12: Another possibility for renting a car is from Home page in best deals slide;

REQ - 13: After accessing one of the best deals the REQ-2 - REQ-9 should be followed;

### **4.2.3 View History of rented cars**

#### **4.2.3.1 Description and Priority**

The user has the possibility to see his history of rented cars, where also will be shown the penalization points.

Medium Priority.

#### **4.2.3.2 Stimulus/Response Sequences**

Users can access the section “History” where they should be able to see the cars rented in the past and some details about that renting(car, price, location, date and penalization points). If he does not rent a car at that moment, a message will be displayed like “You have not rented cars yet”. Also if he reaches the limit of the penalization points he will be suspended for three months, a message will be displayed(You accumulated too many penalized points and you are suspended for three months from now) and when he would want to rent a car the button “Book now” will be blocked.

#### **4.2.3.3 Functional Requirements**

REQ - 1: A section “History” can be accessed from “Profile picture”;

REQ - 2: SA check if he has rented cars in the past in the DB;

REQ - 3: If there are no rented cars in DB for the user, an exception “You have not rented cars yet” will be displayed;

REQ - 4: If he has rented cars, a list will be shown where he can see details about the rented cars;

REQ - 5: He should be able to view his penalization points;

REQ - 6: If the user reaches the limit of penalization points he should be suspended for three months by renting a car;

### **4.2.4 View rental Stations**

#### **4.2.4.1 Description and Priority**

Users can access a page which will list all cities where a Station is located.

Medium - Low Priority.

#### **4.2.4.2 Stimulus/Response Sequences**

The user can view a list with all Stations, by accessing the section “Stations” , and he will be able to see from where he can pick up a car and where he can leave a car.(city, street and number).

#### **4.2.4.3 Functional Requirements**

REQ - 1: A section “Stations” can be accessed from “Menu”;

REQ - 2: A list with cities that contains the Station for renting will be displayed;

#### **4.2.5 Manage Profile**

##### **4.2.5.1 Description and Priority**

Users can access the section “My profile” where it will display details about him and can change some of them.

Low Priority.

##### **4.2.5.2 Stimulus/Response Sequences**

The users can see his profile where he can change some details about him(username, password).

##### **4.2.5.3 Functional Requirements**

REQ - 1: The list about the user will be shown up;

REQ - 2: Some fields can be changed;

REQ - 3: If there are made some changes the button “Save” must be pressed;

REQ - 4: After the changes the new fields will be updated in the DB;

#### **4.2.6 Log out**

##### **4.2.6.1 Description and Priority**

Users can access the section “Log out” from “Profile Picture” which will lead to being a guest.

Low Priority.

##### **4.2.6.2 Stimulus/Response Sequences**

The users can log out and become a guest which leads to being unable to rent a car.

##### **4.2.6.3 Functional Requirements**

REQ - 1: After pressing the button “Log out” the command will be applied and will appear a message(You have been log out from the account);

### **4.3 User Class 3 - The Administrator**

#### **4.3.1 Manage profile**

##### **4.3.1.1 Description and Priority**

Administrators can access the section “My profile” where it will display details about him and can change some of them.

Low Priority.

##### **4.3.1.2 Stimulus/Response Sequences**

The administrators can see his profile where he can change some details about him(username, password, card details for payment).

##### **4.3.1.3 Functional Requirements**

REQ - 1: The list about the administrator will be shown up;

REQ - 2: Some fields can be changed;

REQ - 3: If there are made some changes the button “Save” must be pressed;

REQ - 4: After the changes the new fields will be updated in the DB;

#### **4.3.2 Manage cars**

##### **4.3.2.1 Description and Priority**



The administrators can manage the cars by accessing the section “Manage cars”.

High - Medium Priority.

#### **4.3.2.2 Stimulus/Response Sequences**

Administrators can add, delete a car and also modify information about it.

#### **4.3.2.3 Functional Requirements**

REQ - 1: The list with the cars should be extracted from the DB;

REQ - 2: A button “Modify” is shown near each listed car;

REQ - 3: If pressed, a group of fields appears already containing the data about the car and also a “Save” button;

REQ - 4: After changing the desired fields, the button “Save” must be pressed;

REQ - 5: Once pressed, the new data is updated in DB;

REQ - 6: A button “Add new car” is shown;

REQ - 7: If pressed, a group of fields should be filled;

REQ - 8: When the fields are filled, a new entry should be created in DB;

REQ - 9: A button “Delete car” is shown near each listed car;

REQ - 10: If pressed, the car is removed from the DB;

### **4.3.3 Manage stations**

#### **4.3.3.1 Description and Priority**

The administrators can manage the stations by accessing the section “Manage stations”.

High - Medium Priority

#### **4.3.3.2 Stimulus/Response Sequences**

Administrators can add, delete a station and also modify information about it.

#### **4.3.3.3 Functional Requirements**

REQ - 1: The list with the stations should be extracted from the DB;

REQ - 2: A button “Modify” is shown near each listed station;

REQ - 3: If pressed, a group of fields appears already containing the data about the station and also a “Save” button;

REQ - 4: After changing the desired fields, the button “Save” must be pressed;

REQ - 5: Once pressed, the new data is updated in DB;

REQ - 6: A button “Add new station” is shown;

REQ - 7: If pressed, a group of fields should be filled;

REQ - 8: When the fields are filled, a new entry should be created in DB;

REQ - 9: A button “Delete station” is shown near each listed station;

REQ - 10: If pressed, the station is removed from the DB;

### **4.3.4 Log out**

#### **4.3.4.1 Description and Priority**

Administrators can access the section “Log out” from “Profile Picture” which will lead to being a guest.

Low Priority.

#### **4.2.6.2 Stimulus/Response Sequences**

The administrators can log out and become a guest which leads to being unable to rent a car.

#### 4.2.6.3 Functional Requirements

REQ - 1: After pressing the button “Log out” the command will be applied and will appear a message(You have been log out from the account);

## 5. Other Nonfunctional Requirements

### 5.1 Performance Requirements

The APP can be accessed with the minimum requirements for accessing a browser like Google Chrome, Firefox, Opera. So the minimum specification are:

- at least Windows 7, macOS X 10.10 or 64-bit Ubuntu 14.04+, Debian 8+, Fedora Linux 24+
- at least an Intel Pentium 4 processor, 128 MB of RAM (Firefox needs 512 MB of RAM)
- at least 100MB of hard disk space (Firefox needs 200MB free space)

The APP can be accessed also from a mobile device, the single requirement is to have at least Android 5.0 and at least iOS 8.0 .

### 5.2 Safety Requirements

Not Applicable.

### 5.3 Security Requirements

The website has a system based on a log in function. In order to rent a car, the guest must login with an username and a password to become a user. Additionally, a new field, named “Confirm password” must be completed at the registration.

### 5.4 Software Quality Attributes

- **Imports are ordered as follows:**

- ❑ All static imports in a single block
- ❑ All non-static imports in a single block

If there are both static and non-static imports, a single block line separates the two blocks. There are no other blank lines between import statements.

- **@Override : always used**

A method is marked with the `@Override` annotation whenever it is legal. This includes a class method overriding a superclass method, a class method implementing an interface method, and an interface method respecifying a superinterface method

**Exception:** `@Override` may be omitted when the parent method is `@Deprecated`

- **Finalizers: not used**

It is **extremely rare** to override `Object.finalize`.

- **Package names**

Package names are all lowercase, with consecutive words simply concatenated together (no underscores). For example, `com.example.deepspace`, not `com.example.deepSpace` or `com.example.deep_space`.

- **Non-constant field names**

Non-constant field names (static or otherwise) are written in lowerCamelCase.

These names are typically nouns or noun phrases. For example, `computedValues` or `index`.

- **Parameter names**

Parameter names are written in lowerCamelCase.

One-character parameter names in public methods should be avoided.

- **Local variable names**

Local variable names are written in lowerCamelCase.

Even when final and immutable, local variables are not considered to be constants, and should not be styled as constants.

- **Source file structure**

A source file consists of, in order:

- ☐ License or copyright information, if present
- ☐ Package statement
- ☐ Import statements
- ☐ Exactly one top-level class

Exactly one blank line separates each section that is present.

- **Braces are used where optional**

Braces are used with `if`, `else`, `for`, `do` and `while` statements, even when the body is empty or contains only a single statement.

- **Line-wrapping**

**Terminology Note:** When code that might otherwise legally occupy a single line is divided into multiple lines, this activity is called *line-wrapping*.

There is no comprehensive, deterministic formula showing *exactly* how to line-wrap in every situation. Very often there are several valid ways to line-wrap the same piece of code.

## **5.5 Business Rules**

Not Applicable.

## **6. Other Requirements**

**Appendix A: Glossary**

**Appendix B: Analysis Models**

**Appendix C: To Be Determined List**