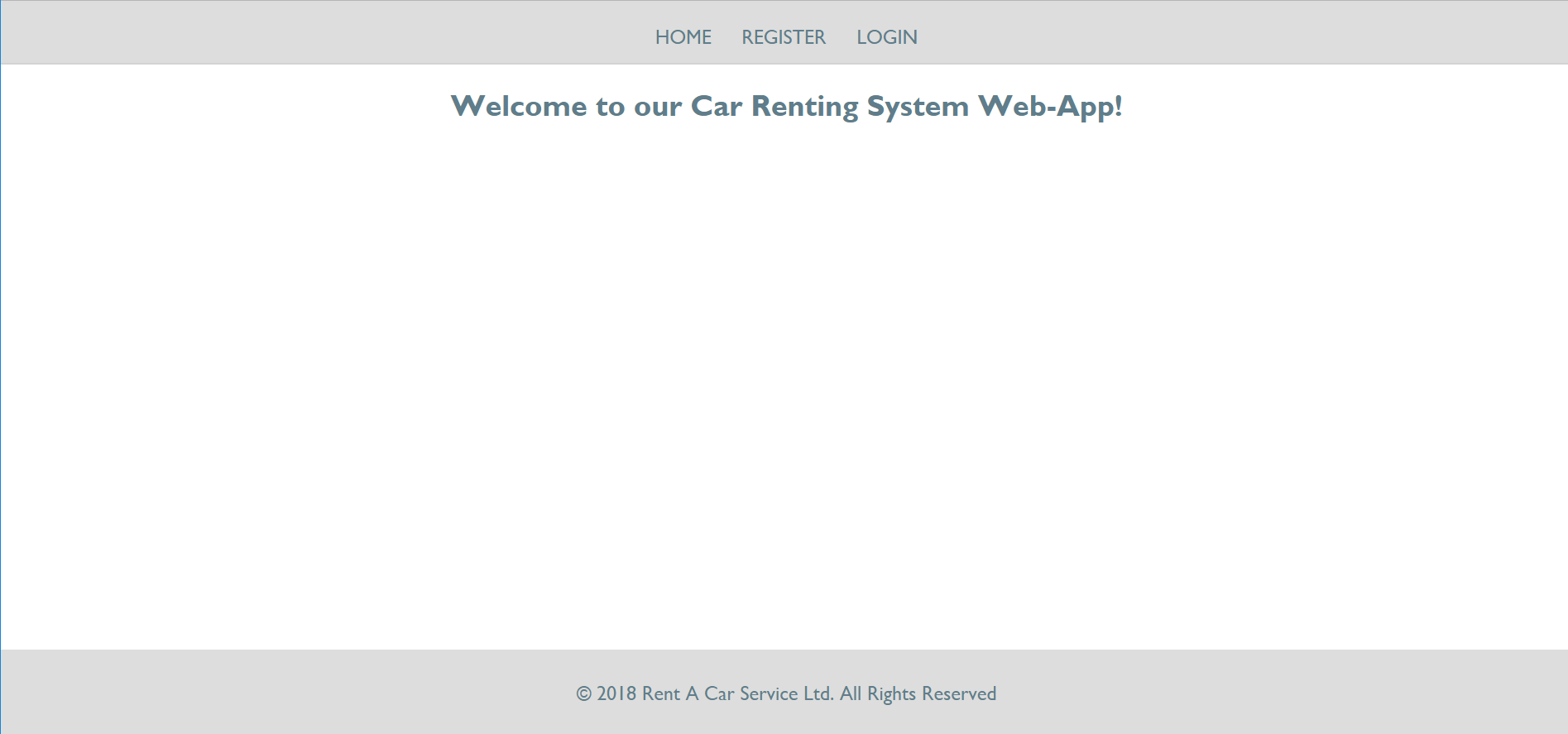
# Car Renting System Exercises

Problems for exercises and homework for the ["Node.js & Express.js Fundamentals" course @ SoftUni](https://softuni.bg/courses/express-js-fundamentals)"

## Create a car renting system

Prepare your project architecture for a **car renting system** and implement **user authentication** allowing administrator role.

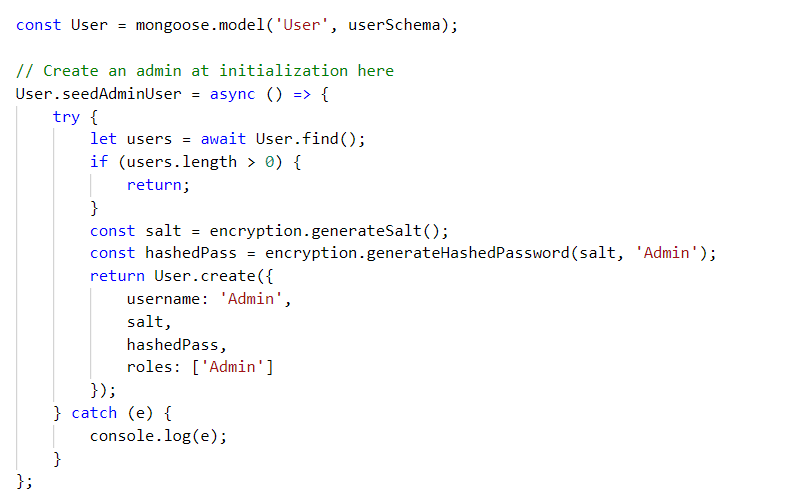


## Initial Setup

You will be provided with a **skeleton**, which contains all the views (handlebars templates) and some configurations. However, there are some things that you need to implement in the configurations

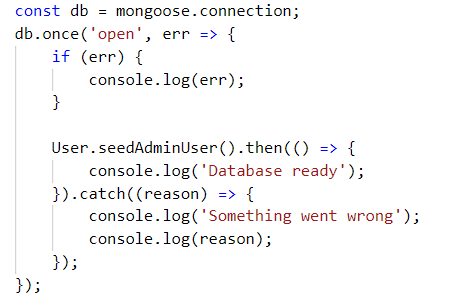
### User.js

At project initialization you should seed a user into the database. Use the code down below:



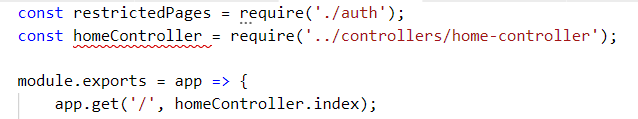
Database.js

At **config/database.js** invoke the seed admin user function the moment our database connection is **opened**



### Routes.js

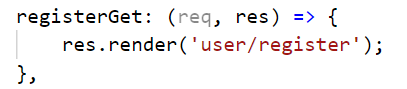
Here you place all the routes that uses a given action from the right controller (home, user, cars). You should also apply **middleware** to block routes that need authentication.



### User-controller.js

#### Register GET

To load the register page just render the needed template:

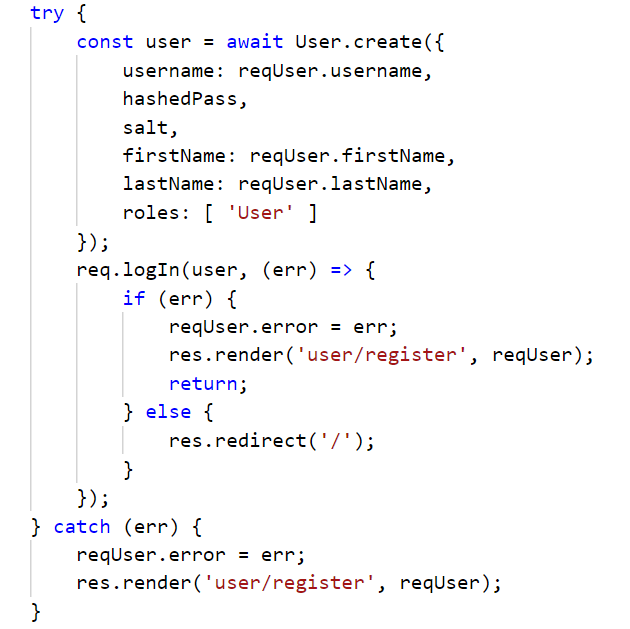


#### Register POST

We first have to validate the password and repeat password fields. If they are **empty strings** attach an **error** property and send it back as context to handlebars. After that we generate **salt** and use it to create a **hashed password** for better security. Use the **encryption** module from **util/encryption.js**.



After that we use the mongoose Model to create a user and add a **default** role. We use the **passport** function "**logIn"** to automatically login the user after successful registration.

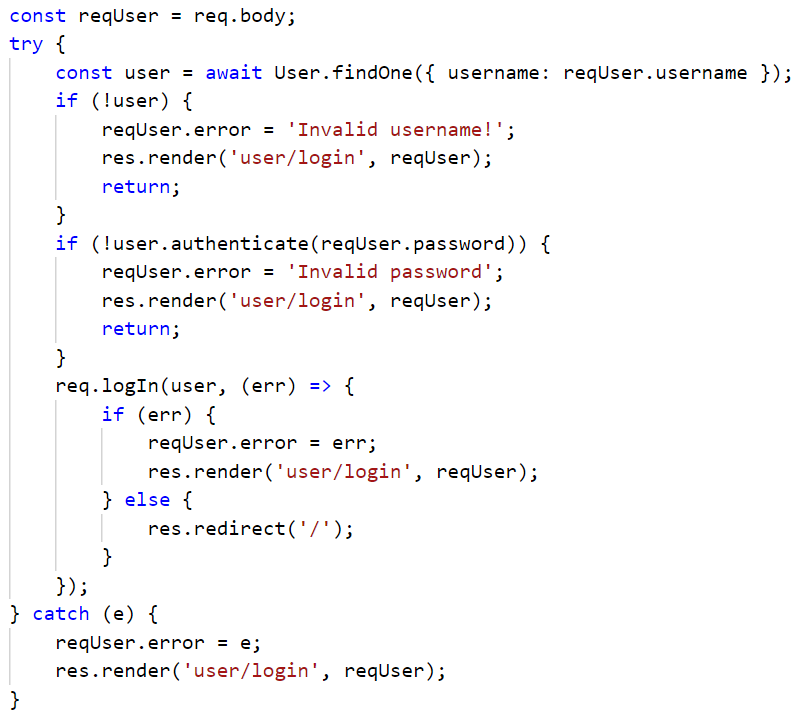


#### Login GET

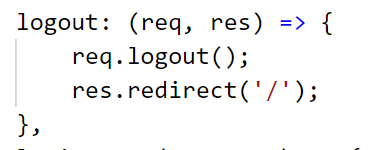
We render the login page the same way as register

#### Login POST

We **find** the correct user by username. Validate if the **username** and **password** are **both correct** and finally authorize.



#### Logout



## Car Model

Create a Car model. It should have **at least** the following properties:

* Model – string, required
* Image – string, required
* Price per Day - number, required
* Is Rented – boolean, required, default: false

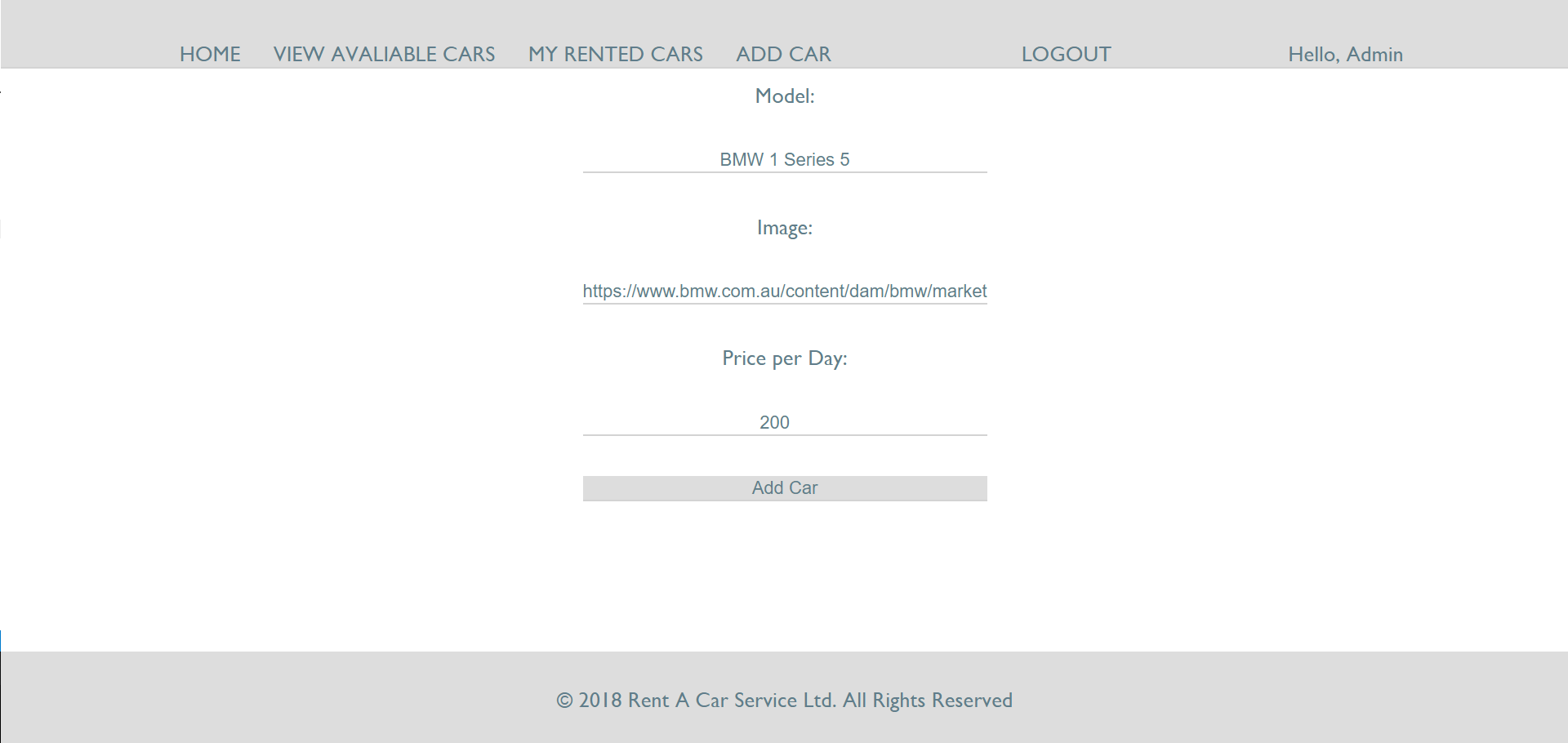
Create a Rent model. It should have the following properties:

* Days – number, required
* Car – object ID, required, reference to Car model
* Owner – object ID, required, reference to User Model

## Functionality

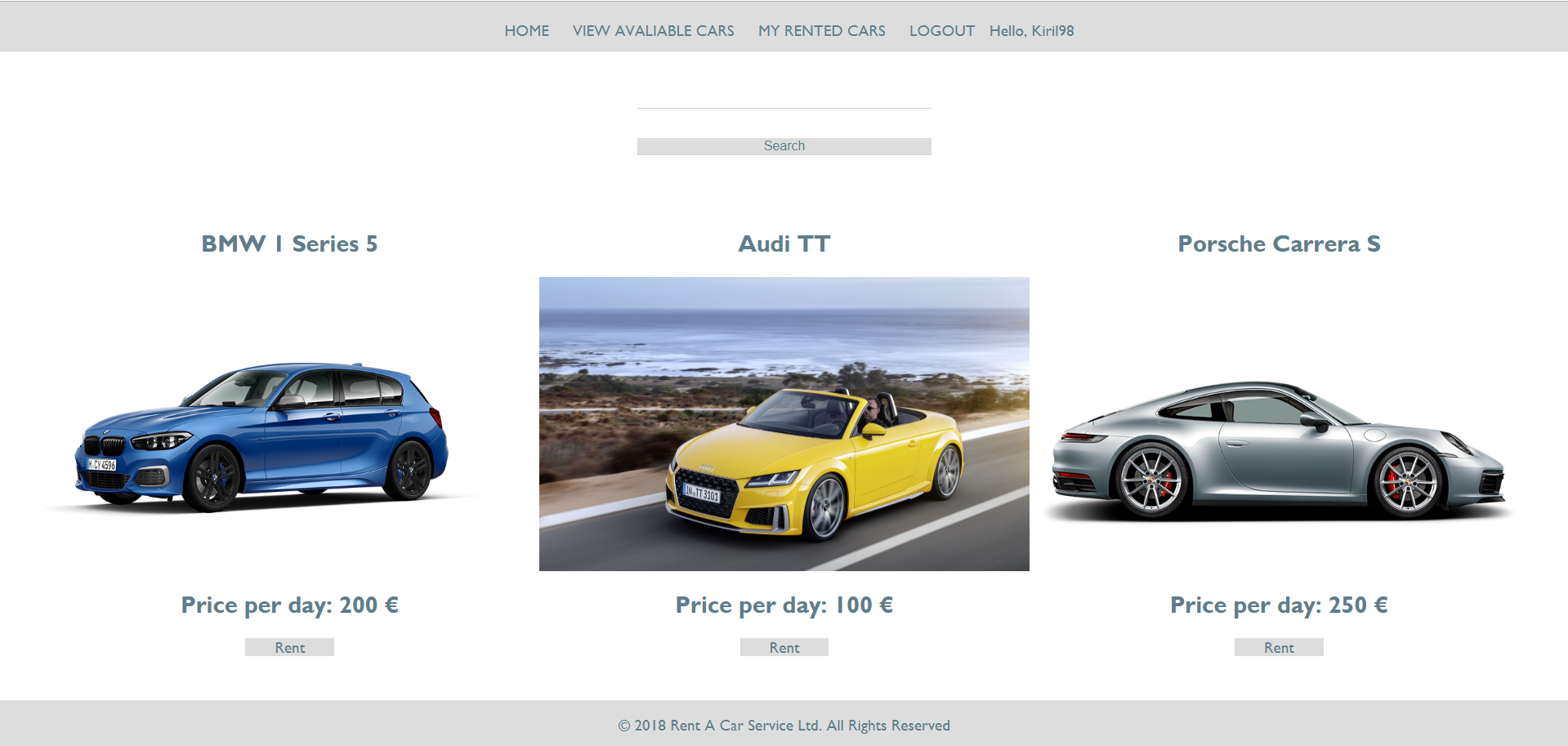
### Add /car/add route

This route should be available only for **administrators.** It should display a form with which a car can be added to the system. Choose what properties each car should have but make sure you save **at least an image**, **model** and **price** **per day** for every data entry. After successfully adding a car, redirect to **'cars/all'**



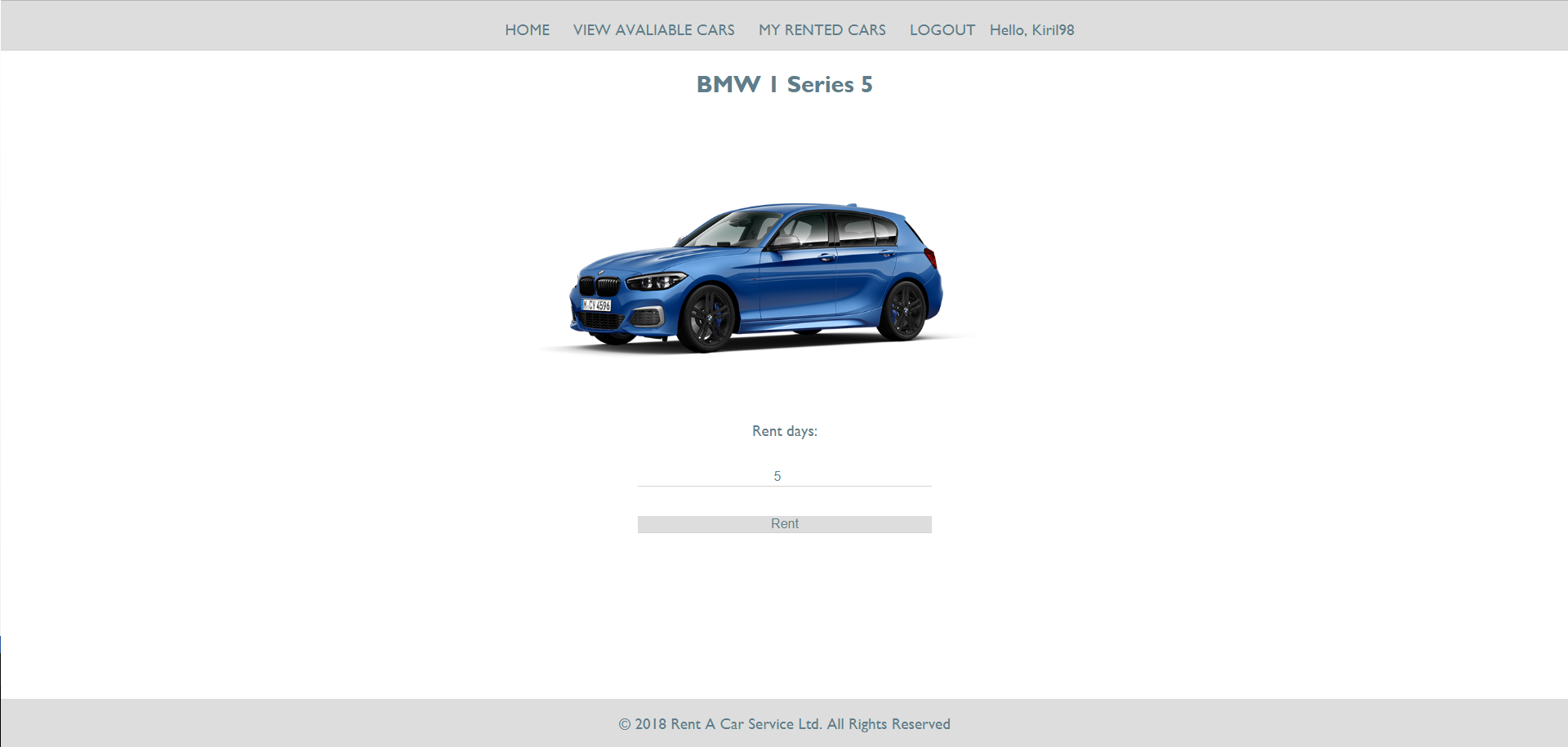
### Add /car/all route

This route should be **available for all**. It should display **all available cars** for rent. Keep in mind that **currently rented cars should not be displayed here**. Add **searching** by a **property** of your choice.



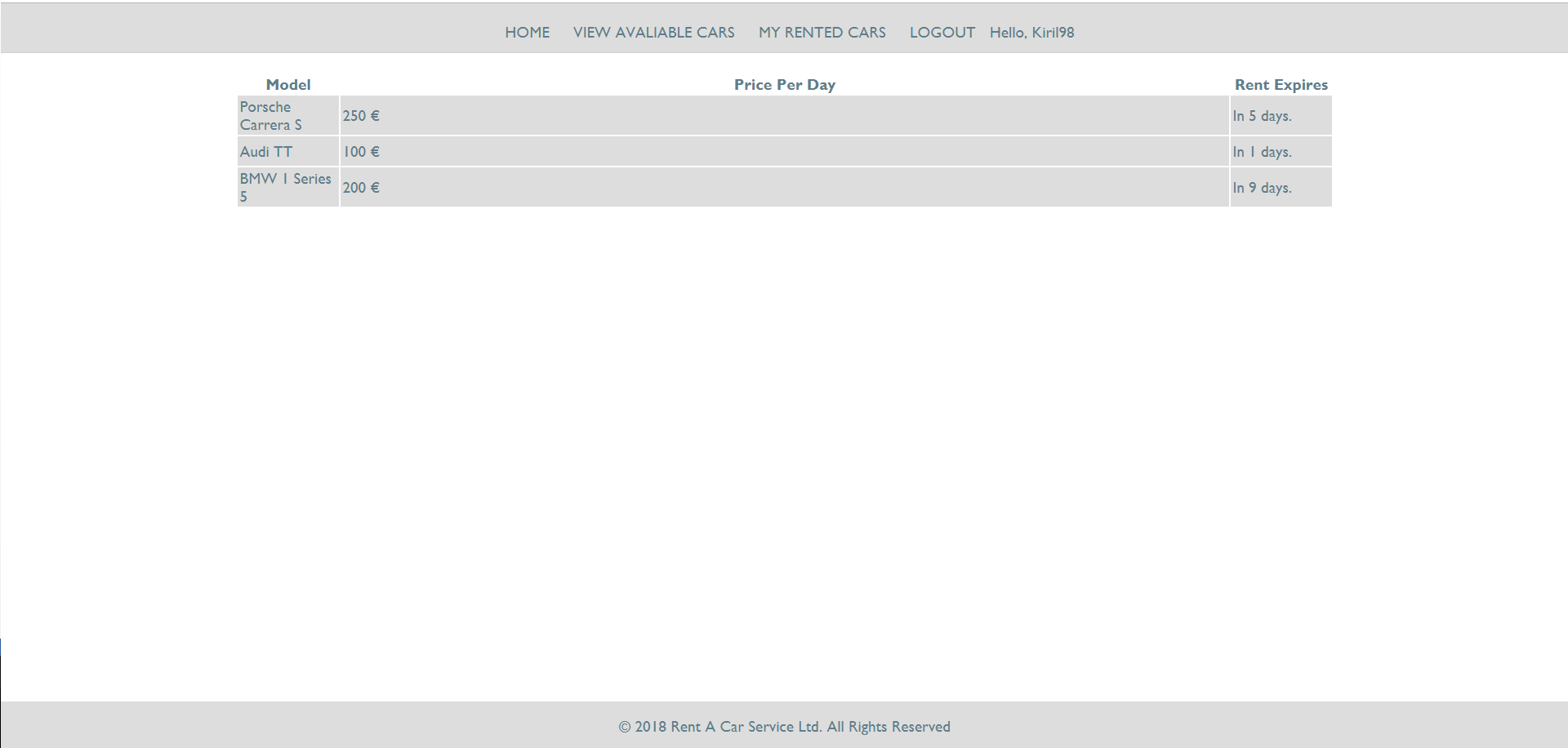
### Add /car/rent/:id route

This route should be available for all **registered users**. Each user should be able to choose **number of days** for which to rent the car with the provided Id. **Each rented car** should **not** be shown on the **/cars/all route**.



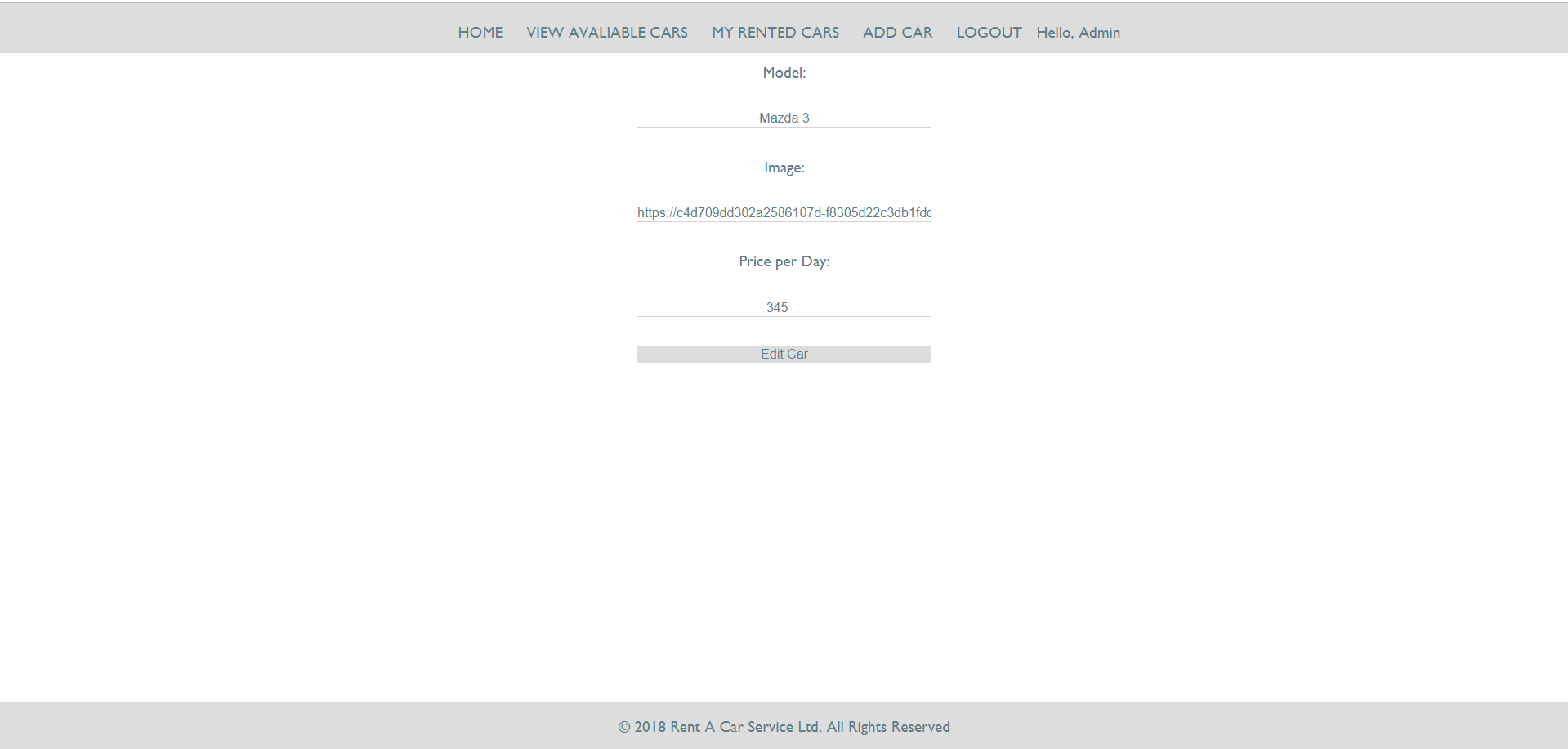
### Add /user/rents route

This route should be available **only for registered users**. It should show for the currently logged in user all **his/her rent history in a table**.



### Add /car/edit/:id route

This route should be available only **for administrators**. It should allow the user to edit the car with the provided Id.



## \* Sky is the limit

Add functionality to the system as you see fit – you may add statistics, image uploads, nice design, etc. ☺