

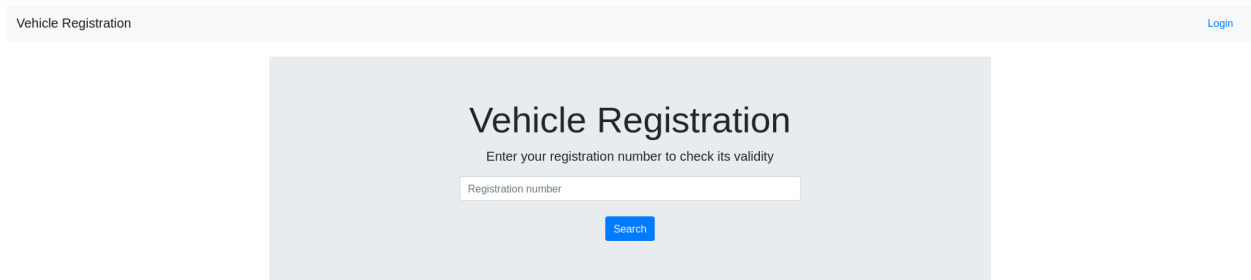
## Challenge - PHP PDO

An application for managing registered vehicles.

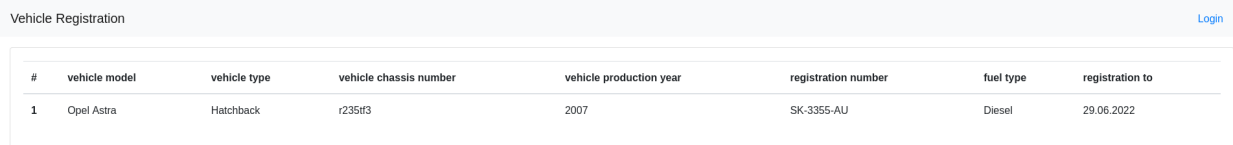
For this challenge we need to create an application for managing vehicle licensing. It has 2 parts, one for all visitors and one for authenticated users (admins).

### Part 1.

The home page, looks like on the screenshot below.



There is an input for entering a license plate number of a vehicle. If the user enters a valid license plate number (one added by the admin in part 2), all information about that car should be displayed in a table.



#	vehicle model	vehicle type	vehicle chassis number	vehicle production year	registration number	fuel type	registration to
1	Opel Astra	Hatchback	r235d3	2007	SK-3355-AU	Diesel	29.06.2022

If the license plate does not exist, a message that there is no such record should be printed out on screen.

Add a navbar with a Login button in the right corner. There is no need to write logic for registering users as the admins will be hardcoded into the database and no new users will be allowed to register (this is a closed system).

## Part 2.

Implement the login functionality using session. The users are stored in a database table and the passwords are hashed.

After login, the admin should be presented with the following view:

Vehicle Registration
Logout

### Vehicle Registration

Vehicle Model  
Default select

Vehicle chassis number

Vehicle registration number

Registration to  
mm / dd / yyyy

Vehicle Type  
Default select

Vehicle production year  
mm / dd / yyyy

Fuel type  
Default select

Submit

Search...
Search

This page consists of 2 parts:

### 1. A Vehicle Registration Form

The form should have the following inputs:

- vehicle model (dropdown, options fetched from database),
- vehicle type (dropdown, options fetched from database),
- vehicle chassis number (text),
- vehicle production year (date),
- registration number (text),
- fuel type (dropdown, options fetched from database),
- registration to (date)

**Vehicle model**, **vehicle type** and **fuel type** should all be resources coming from database tables. You can name these tables as the inputs themselves, only in plural, for example "vehicle\_models". For **vehicle\_type** the options can be hardcoded into a database, there is no need to create dynamic CRUD for that. The available options are: sedan, coupe, hatchback, suv, minivan. Same applies for the **fuel\_type** table, where available options are: gasoline, diesel and electric.

For the vehicle\_models table, a CRUD needs to be created and the admin should be able to insert a new vehicle model if it doesn't exist, before licensing it.

All the information from the vehicle registration form is stored in one table called **registrations**. Make a validation that prevents the admin from adding two records in the database with the same chassis number.

## 2. A table with all the licensed vehicles

The table should display all the information entered from the form.

Make a logic to check the registration\_to date and color the row according to its value. Display that table row in yellow color if the registration is one month before expiration. If the registration has expired, display that row in red.

In the action column besides the "delete" and "edit" buttons, add an "extend" button for editing only the "registration\_to" date for those records that are expired or about to expire (yellow or red).

Add an input in the right corner of the table with a "search" button. When you click on the search button, only the vehicles matching the search criteria should be shown. The text from the input text field is used to search the vehicles by model, license plate number or chassis number. Here you should use the LIKE and OR operators.

Don't leave room for SQL Injection attacks.

It would be great if you can create all of these operations using the OOP programming model.

---

Deadline:

**One** week after the day of presentation (23:59).

---