Assignment 4 (6 points total)

In this assignment we will learn how to securely check a user's password for authentication, learn about HTTP authentication, try client-side rendering and style our application with more advanced CSS.

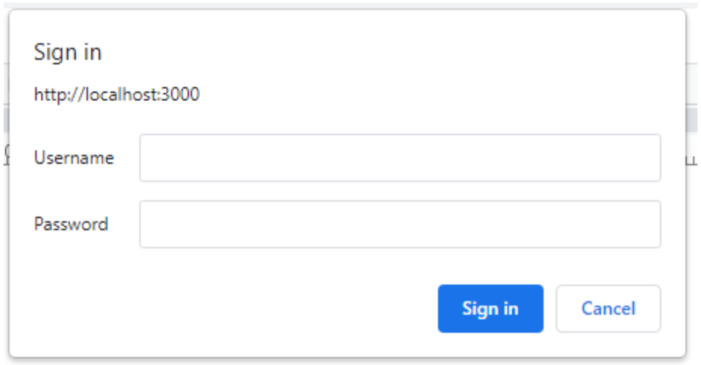
Prerequisites:
- Have Node and Express (with cookie and body parsers) installed

Task 1

In this task you will need to create an application for storing messages that can only be viewed when the user has entered correct credentials. It should works as follows:

1. The user logs in with their username and password
2. They save a message to view later
3. The user might log out to let others leave their messages
4. The user logs back in and sees their message. If they provide invalid password, an error is displayed and no message is shown.

It is unsafe to store user's passwords as plain text in an application. If a database is leaked, attackers will get access to the accounts of all users in said database. In this task you **must use PBKDF2** to avoid storing user's passwords in your application. You must also use HTTP authentication for this task. If implemented correctly, you should be able to see a login prompt in your browser:



Your page should also have a logout button.

Hints:
- Use a middleware to check if a user is already authenticated
- Store a key generated using PBKDF2 to check whether the provided password is correct
- You can return a page with status 401 Unauthorized to log the user out

Task 2

In this task you will create and style a web interface for your car application. Create at least two pages: a page with a list of cars and a page for adding a new car to the app. Right-clicking on a car element should delete the car after a confirmation. You can store the cars in a file or in a database. In this task your server is required to send only static HTML files. All rendering must be done on the client-side with AJAX. In addition, your server API must be changed to use JSON for both request and response data. Lastly, add appropriate CSS styles so that your web pages fit these criteria:

- All pages must have the same header with a title, link to the car list and a link to add a new car. All elements must be centered vertically and have left and right paddings of at least 1em (or equivalent size in other units)
- All pages must have responsive design (that is, have elements properly scaled on small screens)

Page with a car list:

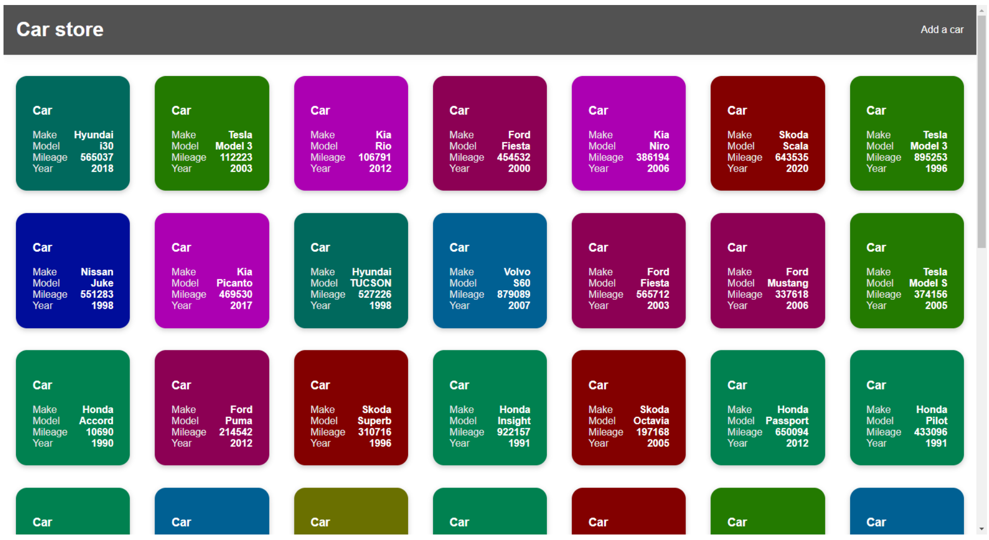
- The borders of the car elements should be clearly visible or alternatively they can have a visually different background from the page itself
- Car elements must all have the same dimensions
- The car elements must have both a padding and margin on all sides of at least 1ch (or equivalent size in other units)
- The car elements must have their properties displayed with the names aligned to the left and the values aligned to the right. There must be some space between names and values
- The car elements must have a box shadow
- The car elements must fill the entire available width of their container, wrapping to a new line if necessary
- Each line of car elements must be centered horizontally

Page for adding a new car:

- Form used for adding a new car must be centered horizontally
- Labels must use a bold font
- Inputs must have only their bottom border visible
- Inputs must have a bigger font size than labels

Example style:

Internet of Things TX00CI63-3006



Car store

Add a car

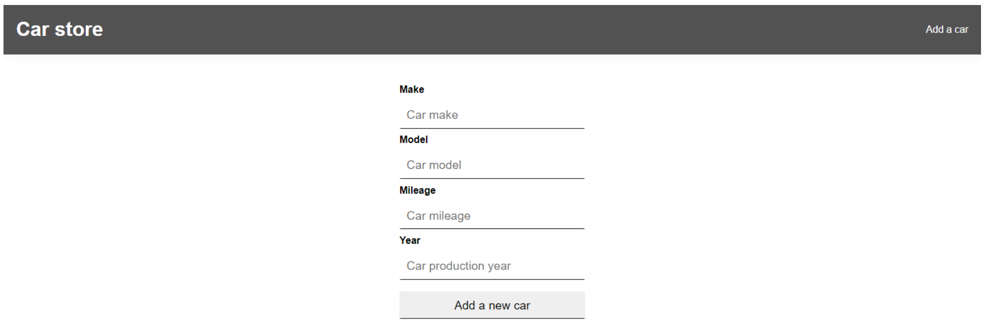
	Car		Car		Car		Car		Car		Car		Car
Make	Hyundai	Make	Tesla	Make	Kia	Make	Ford	Make	Kia	Make	Skoda	Make	Tesla
Model	i30	Model	Model 3	Model	Rio	Model	Fiesta	Model	Niro	Model	Scala	Model	Model 3
Mileage	565037	Mileage	112223	Mileage	106791	Mileage	454532	Mileage	386194	Mileage	643535	Mileage	895253
Year	2018	Year	2003	Year	2012	Year	2000	Year	2006	Year	2020	Year	1996

	Car		Car		Car		Car		Car		Car		Car
Make	Nissan	Make	Kia	Make	Hyundai	Make	Volvo	Make	Ford	Make	Ford	Make	Tesla
Model	Juke	Model	Picanto	Model	TUCSON	Model	S60	Model	Fiesta	Model	Mustang	Model	Model S
Mileage	551283	Mileage	469530	Mileage	527226	Mileage	879089	Mileage	565712	Mileage	337618	Mileage	374156
Year	1998	Year	2017	Year	1998	Year	2007	Year	2003	Year	2006	Year	2005

	Car		Car		Car		Car		Car		Car		Car
Make	Honda	Make	Ford	Make	Skoda	Make	Honda	Make	Skoda	Make	Honda	Make	Honda
Model	Accord	Model	Puma	Model	Superb	Model	Insight	Model	Octavia	Model	Passport	Model	Pilot
Mileage	10690	Mileage	214542	Mileage	310716	Mileage	922157	Mileage	197168	Mileage	650094	Mileage	433096
Year	1990	Year	2012	Year	1996	Year	1991	Year	2005	Year	2012	Year	1991

Car Car Car Car Car Car Car

Main page

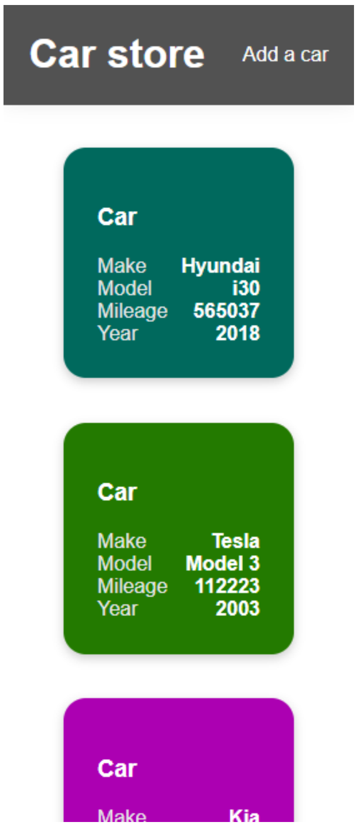


Car store

Add a car

Make	Car make
Model	Car model
Mileage	Car mileage
Year	Car production year

New car page



Car

Make Hyundai
Model i30
Mileage 565037
Year 2018

Car

Make Tesla
Model Model 3
Mileage 112223
Year 2003

Car

Make Kia

Small screen layout

Hints:

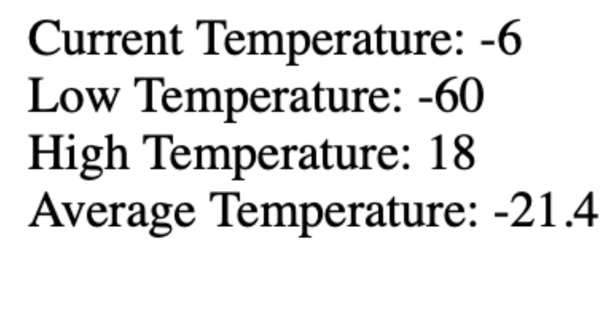
- Use AJAX to fetch the list of cars when the page is loaded
- Use Flexbox to position car elements
- Use `Content-Type` header to indicate that you are sending content in JSON format

Task 3

In this task you will use UART to collect temperature data. If you are a member of the Embedded systems programming course, you can use your Lab exercise from this week to store persistent temperature data when you press the button on your LPCxpresso. If you are not a member of embedded systems programming, a firmware has been made for you that will be executed on the signal capture board. The device will simulate giving you temperature through UART at a random interval. In both cases the format of the data is in JSON and the data should be stored upon receipt in a method of your choosing. I chose to use split, but you are indeed free to use mongoDB or a file. The data will arrive in the following format:

```
{
      "samplenr": 3536,
      "timestamp": 3539372,
      "temperature": -53
}
```

When a request is made from the homepage, your server will get the following data from wherever it is stored and display it on the page.



Current Temperature: -6
Low Temperature: -60
High Temperature: 18
Average Temperature: -21.4

Current temperature is the most recent temperature measurement, Low temperature is the lowest stored temperature, high is the highest, and average is the average temperature of all measurements rounded to one decimal place.

The hex file can be found in Oma with this assignment. You can use a PSoC programmer to flash the signal capture board with the appropriate firmware.

IMPORTANT INFORMATION

Show all of your sources to the teacher before the deadline for full points. Assignments submitted even a moment after the deadline will only be able to get half points.
!!!There is no exception to this rule!!!