# Lab: Data and Authentication

Problems for in-class lab for the ["JavaScript Applications" course @ SoftUni](https://softuni.bg/trainings/3847/js-applications-october-2022)

**Working with Remote Data**

For the solution of some of the following tasks, you will need to use an up-to-date version of the **local REST service**, provided in the lesson’s resources archive. You can [read the documentation here](https://github.com/softuni-practice-server/softuni-practice-server).

## 1. Cookbook – Part 2

The resources for this task are available in the following GitHub repository:

<https://github.com/viktorpts/js-apps-workshop>

You may check-out the repository or download the files via the green button labeled “Code” in the upper-right corner. Use the files located in **lesson-03/base** to begin the task. Before starting, make sure you have the most recent version of the repository. To see the solution, check the files inside **lesson-03/finished**.

Now you will extend the application from previous exercise. Write a JS functionality for register, login and logout. Also logged in users can create new recipies. You are again **provided with skeleton** (**HTML & CSS**) for this task, and you can use the same **server**.

The application now has a navigation bar, with links to other pages. You need to update the visibility of the buttons in the header, depending on whether there is a logged-in user or not. This is how the header should look like for a guest user:



When there is a logged-in user, the header should look like this:



To achieve this, change the visibility of the corresponding <div> element.

### Data

For this version of the app, the data source will be changed since we need to implement user profiles. Use the following endpoints:

* To obtain a list of all recipes:

GET http://localhost:3030/data/recipes?select=\_id%2Cname%2Cimg

* To obtain details about the selected recipe:

GET http://localhost:3030/data/recipes/:id

* To create a new recipe:

POST http://localhost:3030/data/recipes

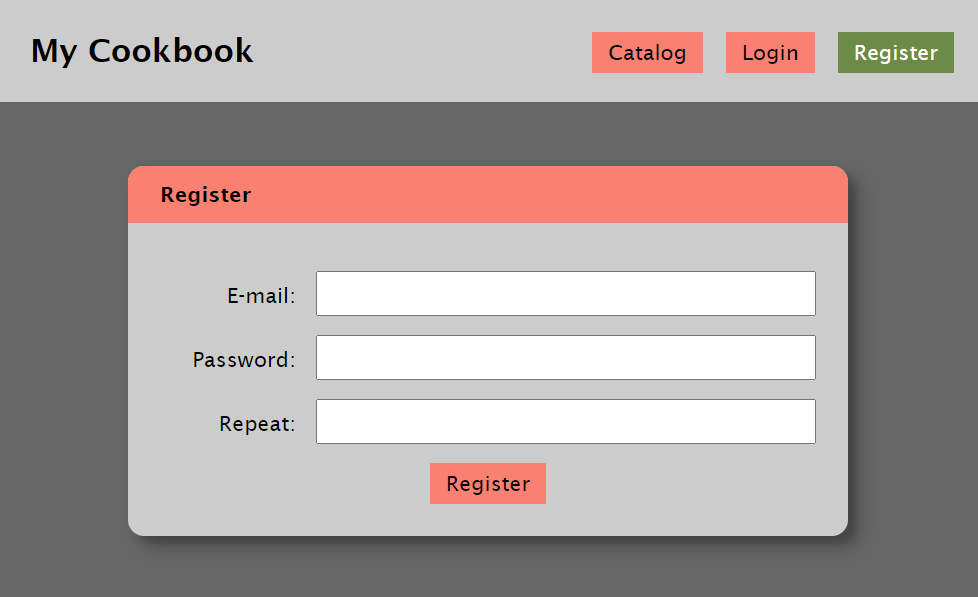
Note that for the last request (recipe creation) you will need to provide **user credentials** via the following request header:

X-Authorization: *{token}*

Read the following sections, to see how the token is obtained for a registered user.

### Register

You need to write the functionality for registration of new user. By clicking the "Register" button you have to load the registration form. When the "Register" button of the form is clicked you need to send a post request to this URL: **http://localhost:3030/users/register**

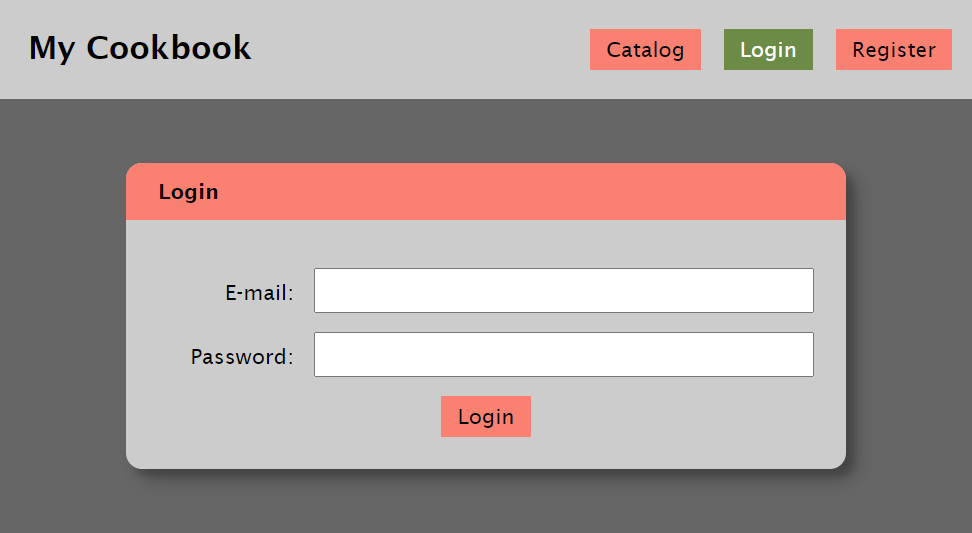


If the registration is successful you can redirect to catalog page. The service will respond with an **authorization token**, that you must save in the browser, using sessionStorage. With every following request, send the token in the authorization header.

### Login

If the user has already registration, the user can login by using the login form. After successful login the user should be redirected to the catalog page. Save the returned token in the session storage, and send it with every request.

The **URL** for login is: **http://localhost:3030/users/login**

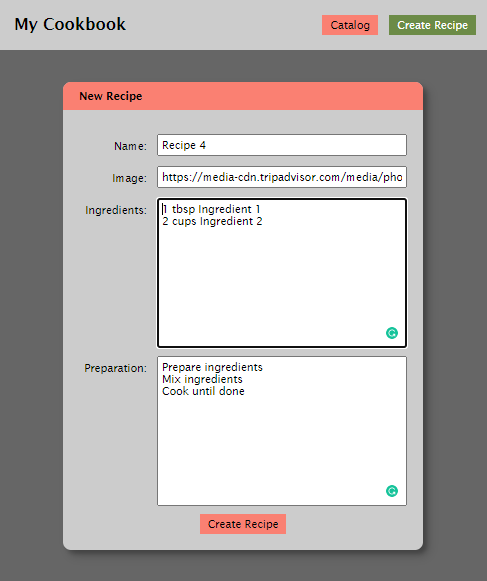


### Logout

The logged in user can be logged out by clicking the logout button. Write the functionality for this action. You can use this URL for logout (send GET request): **http://localhost:3030/users/logout**

### Create Recipe

The logged in user can also create new recipes. In the form you have to fill the following fields: name, img, ingredients and steps. The ingredients and the steps should be written each on new line, and then converted to an array splitting the text on line-breaks (use the line-break character "/n").



After successful creation of the recipe, you should redirect to the catalog page, where the new recipe should appear.

## Cookbook – Part 3

The resources for this task are available in the following GitHub repository:

<https://github.com/viktorpts/js-apps-workshop>

You may check-out the repository or download the files via the green button labeled “Code” in the upper-right corner. Use the files located in **lesson-04/base** to begin the task. Before starting, make sure you have the most recent version of the repository. To see the solution, check the files inside **lesson-04/finished**.

Let's write more functionalities for this task and extend the application. In this task we will write the functionalities for editing and deleting recipies.

### Code Refactoring

Rewrite the application, so that it is a single-page application, using modules with import/export statements. There will be a single HTML file, containing all resources and the views will be dynamically generated via JavaScript.

### Recipe Details

Instead of expanding in the Catalog view, clicking on a recipe preview changes the view to a details page.



### Edit Recipe (only for the creator)

By clicking on the recipie you can see the details for it, and if you are the creator of this recipe you have to be able to edit and delete it (the two buttons bottom-right are visible)

On click the "Edit" button you are redirected to the edit page and all the details of the recipe should be filled up in the input fields.

Картина, която съдържа текст, електроника, екранна снимка, софтуер

Описанието е генерирано автоматично

When "Update recipe" button is clicked, the app sends PUT request to the back-end and updates the recipe. After that, redirect to details page.

### Delete Recipe (only for the creator)

By **clicking on "Delete"** button the app needs first confirmation for deleting, then sends DELETE request to the back-end and deletes the recipe. Then the app shows message, that the recipe is deleted successful.

## My Cookbook – Testing

The resources for this task are available in the following GitHub repository:

<https://github.com/viktorpts/js-apps-workshop>

You may check-out the repository or download the files via the green button labeled “Code” in the upper-right corner. Use the files located in **lesson-05/base** to begin the task. Before starting, make sure you have the most recent version of the repository. To see the solution, check the files inside **lesson-05/finished**.

This task is to write tests for the functionality of the "My Cookbook" app. You are provided with the working app and need to test the following:

### Testing: Catalog

* The catalog page should load and render the content of the API
* Displays the recipe details

### Testing: Authentication

* "register" makes correct API call
* "login" makes correct API call

### Testing: CRUD operations

* "create" makes correct API call for logged in user
* The author can see the "Edit" and "Delete" button
* "edit" loads the correct article data for logged in user
* "edit" makes correct API call for logged in user
* "delete" makes correct API call for logged in user

## My Cookbook – Refactoring

Refactor the application, so separation of ceoncerns and other best practices are followed. The following functionality can be abstracted into it’s own module, so that the views only contain business logic:

* Requests to the server
* User authentication
* Navigation and view switching
* DOM rendering

The tests that you created in Task 2 will be very helpful – as you change the code, by running the tests you will always know if everything works as expected or if a bug has been introduced.

## My Cookbook – Part 4

### Home View

Create **automated tests** for this functionality. It’s your choice whether to create the tests first (**Test-Driven Development**) or to write them during or after the implementation of the functionality.

The home page contains a welcome message and a preview of the three most recently added recipes, from the newest to the oldest. The application now starts in this view, instead of the Catalog. Clicking on the application title (top left) takes the user back to the home view.

Картина, която съдържа текст, екранна снимка, храна

Описанието е генерирано автоматично

* Get the three most recent recipes: /data/recipes?select=\_id%2Cname%2Cimg&sortBy=\_createdOn%20desc&pageSize=3 (GET)

### Catalog Pagination

Create **automated tests** for this functionality. It’s your choice whether to create the tests first (**Test-Driven Development**) or to write them during or after the implementation of the functionality.

Implement pagination for the catalog. Each page must hold 5 recipes. Display page controls at the top and at the bottom of the page.

Картина, която съдържа текст, екранна снимка, Бързо хранене, храна

Описанието е генерирано автоматично

* Get a page of recipes (**offset** is the number of recipes to skip):  
  /data/recipes?select=\_id%2Cname%2Cimg&offset={offset}&pageSize=5 (GET)
* Get the total number of recipes: /data/recipes?count (GET)

## My Cookbook – Refactoring

The resources for this task are available in the following GitHub repository:

<https://github.com/viktorpts/js-apps-workshop>

You may check-out the repository or download the files via the green button labeled “Code” in the upper-right corner. Use the files located in **lesson-06/base** to begin the task. Before starting, make sure you have the most recent version of the repository. To see the solution, check the files inside **lesson-06/finished**.

Refactor the existing code so that it uses **templates** to render each section ad its contents, instead of static HTML derived from a file. You may use the **tests** from the **previous lesson** to continuously check if everything is implemented correctly.

## My Cookbook – Part 5

Add **comments** functionality to the application. Unregistered users only see the list of comments:

Картина, която съдържа текст, екранна снимка, храна

Описанието е генерирано автоматично

Logged-in users see a **button**, that **displays a form** for adding a new comment Any registered user can leave a comment under any recipe.  
Картина, която съдържа текст, екранна снимка, Шрифт, Правоъгълник

Описанието е генерирано автоматично

Картина, която съдържа текст, екранна снимка, Правоъгълник, Шрифт

Описанието е генерирано автоматично

Use the following endpoints:

* To load all comments for a recipe: **data/comments?where=recipeId%3D%22{recipeId}%22** (GET)
* To create a new comment: **/data/comments** (POST)

Each comment has the following structure:

{

recipeId,

content

}

Where recipeId is the Id of the recipe, which the comment is associated with, and content is the comment’s text content.