# Exercise: Single Page Application

Problems for exercises and homework for the ["JavaScript Apps" course @ SoftUni.](https://softuni.bg/trainings/3708/js-applications-june-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).

## Movies – SPA

You are assigned to implement a **Single** **Page** **Web Application**. The app keeps **users** and **movies**. Logged-in users should be able to view **add** **movies** and **like movies**. Logged-in users should also be able to **edit** or **delete** the movies **they have added**.

**REST Service**

You can use the provided server to store your data. Use the following endpoints:

* Get all movies: **/data/movies** (GET)
* Create movie: **/data/movies** (POST)
* Update movie: **/data/movies/:id** (PUT)
* Delete movie: **/data/movies/:id** (DELETE)
* Get number of likes for a movie: **/data/likes?where=movieId%3D%22{movieId}%22&distinct=\_ownerId&count** (GET)
* Get like for a movie from specific user: **/data/likes?where=movieId%3D%22{movieId}%22%20and%20\_ownerId%3D%22{userId}%22** (GET)
* Add a like: **/data/likes** (POST)
* Revoke a like: **/data/likes/:id** (DELETE)

Note that the provided service automatically limits the number of results from every request to 10. You may need to use pagination options, to get all entries.

**HTML and CSS**

You have been given the web design of the application as **HTML** + **CSS** files. Initially all views and forms are shown by the HTML. Your application may **hide**/**show elements** by CSS (display: none) or **delete**/**reattach** from and to the DOM all unneeded elements, or just display the views it needs to display.

**Important**: Don’t change the elements’ **class names** and **ids**. Don’t rename form fields/link/ids. You are **allowed** to add **data attributes** to any elements. You may modify href **attributes** of links and add action**/**method **attributes** to **forms**.

### Navigation Bar

Navigation links should correctly change the current page (view).

* Clicking on the links in the **NavBar** should display the view behind the navigation link.
* Your application may **hide**/**show elements** by CSS (display: none) or **delete**/**reattach** from and to the DOM all unneeded elements, or just display the views it needs to display.
* **The Logged-in** user navbar should contain the following elements:[**Movies**] a **link** to the Home page, the user caption ("Welcome, {email}"), [Logout].Each link navigates to the named page.



* The guest users navbar should contain the following elements: : [**Movies**] which is a **link** to the Home page and [Login], [**Register**].



### Register User

By given **email** and **password,** the app should register a new user in the system.

* The following validations should be made:
  + The **email** input must be **filled**
  + The **password** should be **at least 6 characters** long
  + The **repeat password** should be **equal to the password**
* After a **successful registration** the app should **redirect** to the **home page with the right navbar**.
* In case of **error** (eg. invalid username/password), an appropriate error **message** should be displayed, and the user should be able to **try** to register again.
* Keep the user data in the browser's **session or local storage**.
* After a **successful registration** redirect to **Home page.**



### Login User

By given **username** and **password,** the app should login an existing user.

* After a **successful login** the user **home page should be displayed**.
* In case of **error**, an appropriate error **message** should be displayed and the user should be able to fill in the login form again.
* Keep the user data in the browser's **session or locale storage**.
* After a **successful login** redirect to **Home page.**



### Logout

Successfully logged in users should be able to **logout** from the app.

* After a **successful** logout the **anonymous screen** should be shown
* The **"logout" REST service** at the back-end **must** be called at logout
* All local information in the browser (**user session data**) about the current user should be deleted
* After a **successful logout** redirect to **Login page.**

### Home Page

Users should be welcomed by the Homepage. They should be able to see all added movies:

[**Add Movie**] **button** should refer to the **add** **movie** **form**. It should only be visible if there is a logged-in user.

### Add Movie

Logged-in users should be able to **add movie**.

Clicking the [Add Movie] **button** in the **Home page** should **display** the Add Movie page.

* The form should contain the following validations:
  + The **title, description and image** shouldn’t be **empty**.
  + After a **successful** movie adding the Home pageshould be shown.
* The newly added movie should be stored in the database collection "movies".



Trying to submit empty feelds should **NOT** be possible.

### Edit Movie

Logged-in users should be able to **edit** movies, added by them.

Clicking the [Edit] **button** of a **particular movie** on the Movie Details page should **display** the Edit Movie page inserting the additional information of the movie in the input feelds:



* After a **successful edit** the user should be redirected to the current movie Details page.

### Like Movie

Logged-in users should be able to **like** movie, added by **other** **user**.

**NOTE**: A user should **NOT** be able to like a **movie**, created by **himself**.

Clicking the [Like] **button** of an **movie** should **make a POST request to the following URL: http://localhost:3030/data/likes with body { movieId: id } ( id – this is the id to the current movie ) .** After **successfully like**:

* [Like] **button** changes to [Liked {number of likes}] **span so** users can‘t **like a movie** multiple times**.**



### Delete Movie

Logged-in users should be able to **delete their** movies.

Clicking the [Delete] **button** of an **movie** (on the Movie Details page) should **delete** the **movie**.

After **successful** **movie delete** the Home pageshould be **shown**