Exercise: Architecture and Testing

Problems for exercises and homework for the "JavaScript Applications" course @ SoftUni Global.

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.

1. Messenger – Testing

Your task is to **write tests** for the **functionality** of the "Messenger" problem from the previous lessons. The solution must be obtained prior to implementing test. This app doesn't need any registration. Every user is able to write a new message.

Testing: load messages

First you need to **test** if **all the messages are loaded** and **showed** on the webpage **by clicking** the "Refresh" **button**.

Testing: send message

You need to **test** also the **functionality** of the **sending a new message**. Test if **by clicking the "Send" button** a **request** to the database **is send** with the right parameters.

2. Book Library – Testing

This task is to **write tests for the functionality** of the Book Library problem from the previous lessons. The solution must be obtained prior to implementing test. This app doesn't need any registration. Every user is able to read,write, edit and delete the book details.

Testing: load books

First you need to **test** if all the **books are loaded** and **showed** on the webpage by clicking the "Load all books" button.

Testing: add book

The second thing you need to **test** is the functionality of **the adding a new book**. Test the validation of the input fields (no empty input fields are allowed by creating a new book). Test if the **right request** with the correct parameters **is send** to the back-end. Note there are two forms on the page!

Testing: edit book

The next thing for testing is the **edit functionality**. When the "Edit" button is clicked, the correct form should be made visible (note there are two forms in the page) and it's input fields must show the title and author of the selected book. By **clicking** the "**Save**" **button** a **PUT request** with the right parameters **should be sent** to the back-end.

Testing: delete book

The last testing is the **delete functionality**. By **clicking** the "**Delete**" **button**, a confirmation dialog should open. Confirming the dialog should send the correct **request** to the back-end.

3. SoftTerest *

This task has automated tests, included in the resources. To run the tests, install dependencies with `npm install`, then run in two **separate** terminals the following commands:

```
npm run start
```

```
npm run test
```

There is **no need to submit** this task for manual peer-assessment.

You are assigned to implement a **Single Web Application** (SPA), such that it passes all provided tests. The app keeps **users** and **ideas**. **Guests** should be able to **register** and **login**. Users should be able to view **all ideas**, **create ideas**, see **details** about an **idea** and **logout**. Users should also be able to **delete** the ideas **they have created**.

Navigation Bar

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

- Clicking on the links in the **NavBar** should display the view behind the link (views are represented as sections in the HTML code).
- 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: **`Icon`** (**`icon.jpg`**) which is a **link** to the **Home page**, [**`Dashboard`**], [**`Create`**], [**`Logout`**].
    - The Icon should be a link that navigates to the **currently logged in user's profile**.

- The guest users navbar should contain the following elements: : **`Icon`** (**`icon.jpg`**) which is a **link** to the **`Home page, [Dashboard],`** [**`Register`**] and [**`Login`**].

Home Page

The initial page (view) should display the **navigation bar** ("**`Home`**" (icon), "**`Dashboard`**", "**`Register`**"and "**`Login`**") + **`Home Page`** + **`Footer`**.

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** should be **at least 3 characters** long
    - The **password** should be **at least 3 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 email/password), the user should be able to **try** to register again.
- Keep the user session data in the browser's **local storage**.
- **POST** request **- http://localhost:3030/users/register**

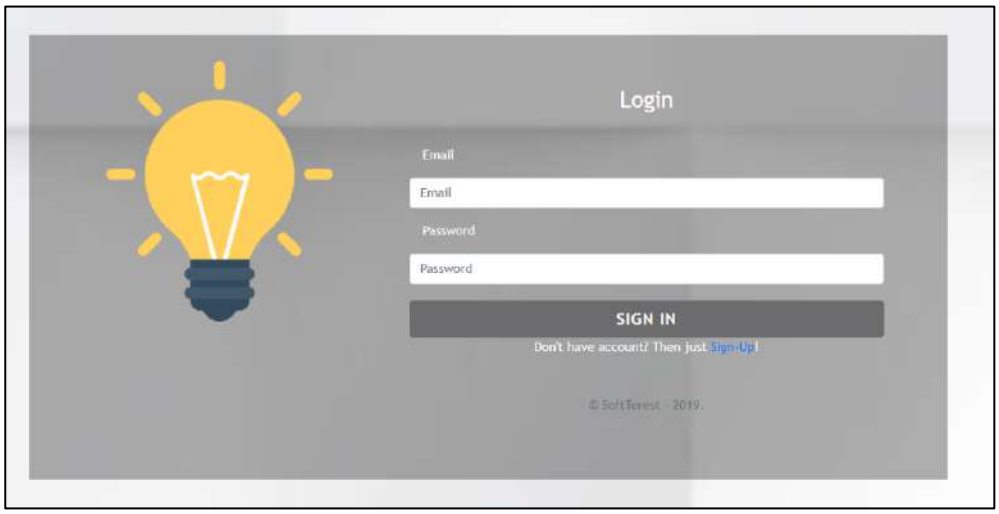
Register once and create/Like awesome ideas!

Login User

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

- After a **successful login** and the user home screen should be displayed.
- In case of **error**, the user should be able to fill in the login form again.
- Keep the user session data in the browser's **local storage**.
- Clear **all** input fields after a **successful** login.
- **POST** request - **http://localhost:3030/users/login**

You are one step away from awesome ideas!



Logout

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

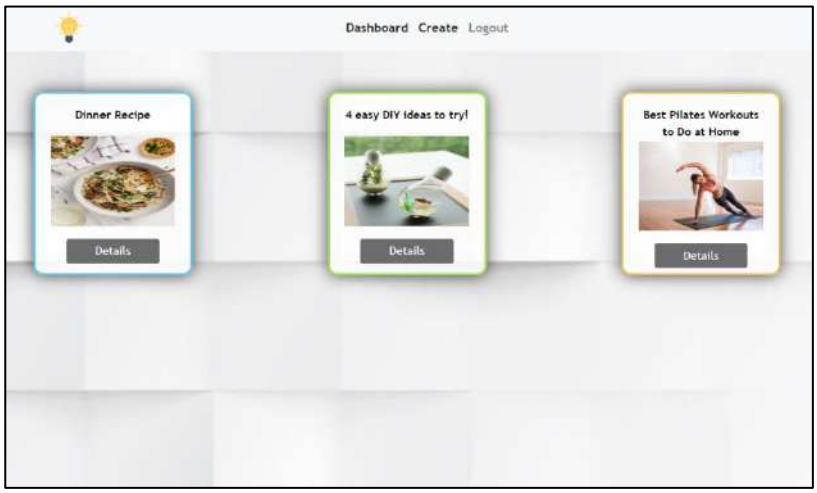
- After a **successful** logot the **anonymous screen** should be shown

- 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
- **GET** request - **http://localhost:3030/users/logout**

Dashboard

All users should be able to see the **`Dashboard`**. They should be able to see all created ideas.

GET request - **http://localhost:3030/data/ideas?select=_id%2Ctitle%2Cimg&sortBy=_createdOn%20desc**



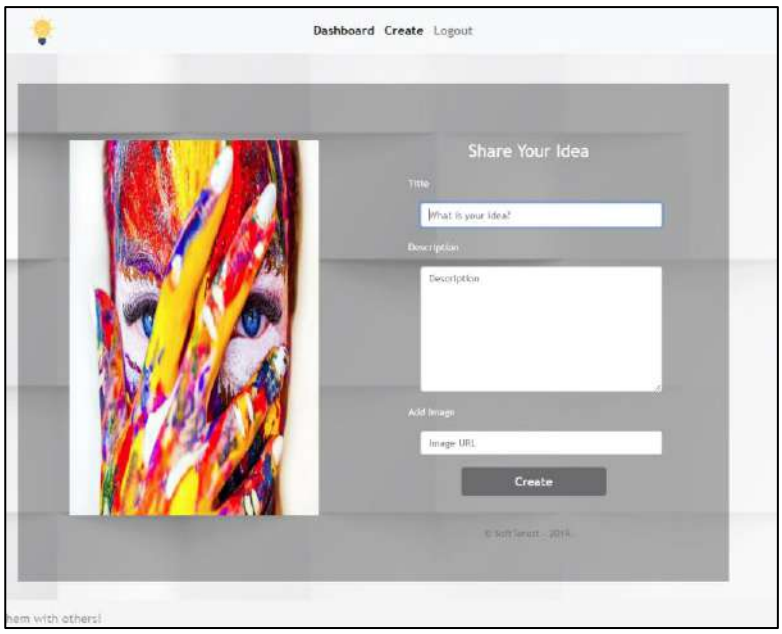
If there are **NO** such ideas, the following view should be displayed:

Create

Logged-in users should be able to **Create** ideas.

Clicking the [**`Create`**] **link** in the **NavBar** should **display** the **`Create page`**.

- The form should contain the following validations:
  - The **title** should be **at least 6 characters** long.
  - The **description** should be **at least 10 characters** long.
  - The **image** should be **at least 5 characters** long.
  - After a **successful** idea creation the **`Dashboard`** should be shown.
- The inputs fields in the form **should be cleared.**
- **POST** request - **http://localhost:3030/data/ideas**



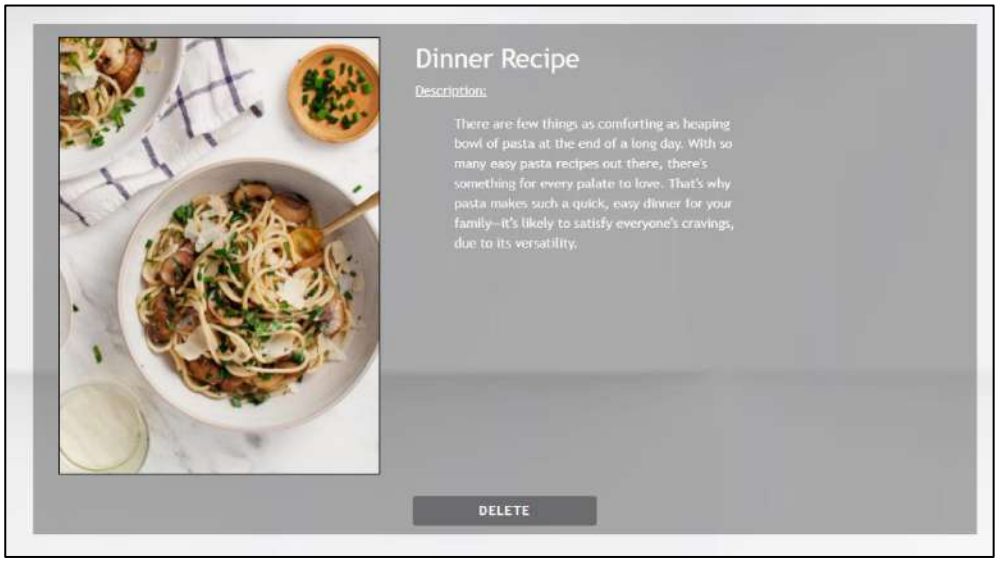
Idea Details

All users should be able to **view details** about ideas.

Clicking the [`Details`] **link** in of a **particular idea** should **display** the **`Idea Details` page**.

- If the currently logged-in user is the organizer of the idea, [`Delete`] **button** should be set to **visible.**

GET request - **http://localhost:3030/data/ideas/:id –** for idea details



Delete Idea

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

Clicking the [`Delete`] **link** of an **idea** (on the **`Idea Details` page**) should **delete** the **idea**.

- After **successful idea delete** the **`Dashboard`** should be **shown**
- **DELETE** request - **http://localhost:3030/data/ideas/:id**

Submitting Your Solution

Place in a **ZIP** file the content of the given resources including your solution. Exclude the **`node_modules`** folder if there is one. Upload the archive to Judge.