Exercise: Data and Authentication

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

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

Write a JS program that records and displays messages. The user can post a message, supplying a name and content and retrieve all currently recorded messages.

The url for the requests - **http://localhost:3030/jsonstore/messenger**

When [**Send**] **button** is clicked you should create a **new object** and send a **post request** to the given url. Use the following message structure:

```
{
  author: authorName,
  content: msgText,
}
```

If you click over [**Refresh**] **button** you should **get all** messages with **GET request** and display them into the textarea. Use the following message format:
"**{author}: {message}**"

Examples

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

2. Phonebook

Write a JS program that can load, create and delete entries from a Phonebook. You will be given an HTML template to which you must bind the needed functionality.

When the **[Load]** button is clicked, a **GET** request should be made to the server to get all phonebook entries. Each received entry should be in a **li** inside the **ul** with **id="phonebook"** in the following format with text **"<person>: <phone> "** and a **[Delete]** button attached. Pressing the **[Delete]** button should send a **DELETE** request to the server and delete the entry. The received response will be an object in the following format:
{<key>:{person:<person>, phone:<phone>}, <key2>:{person:<person2>, phone:<phone2>,…} where **<key>** is an unique key given by the server and **<person>** and **<phone>** are the actual values.

When the **[Create]** button is clicked, a new **POST** request should be made to the server with the information from the Person and Phone textboxes, the Person and Phone textboxes should be cleared and the Phonebook should be automatically reloaded (like if the **[Load]** button was pressed).

The data sent on a POST **request should be a valid JSON object, containing properties** person **and** phone. **Example format:**

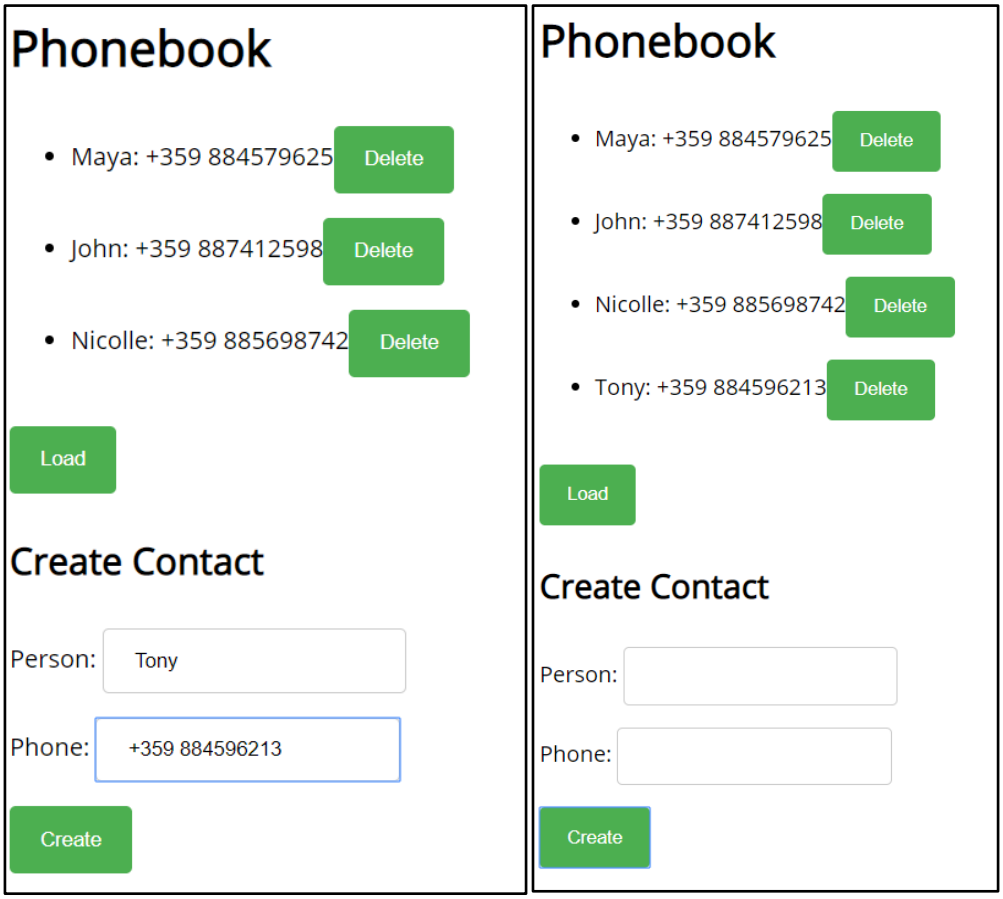
```
{
  "person": "<person>",
  "phone": "<phone>"
}
```

The **url** to which your program should make requests is:

http://localhost:3030/jsonstore/phonebook
GET and **POST** requests should go to **http://localhost:3030/jsonstore/phonebook**, while **DELETE** requests should go to **http://localhost:3030/jsonstore/phonebook/:key>** , where **:key** is the unique key of the entry (you can find out the **key** from the key property in the **GET** request)

SoftUni
© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

Screenshots:



3. Students

Your task is to implement functionality for creating and listing students from a database. Create a new collection called "**students**",

Each student has:

- **`FirstName`** - **`string`**, non-empty
- **`LastName`** - **`string`**, non-empty
- **`FacultyNumber`** - **`string of numbers`**, non-empty
- **`Grade`** - **`number`**, non-empty

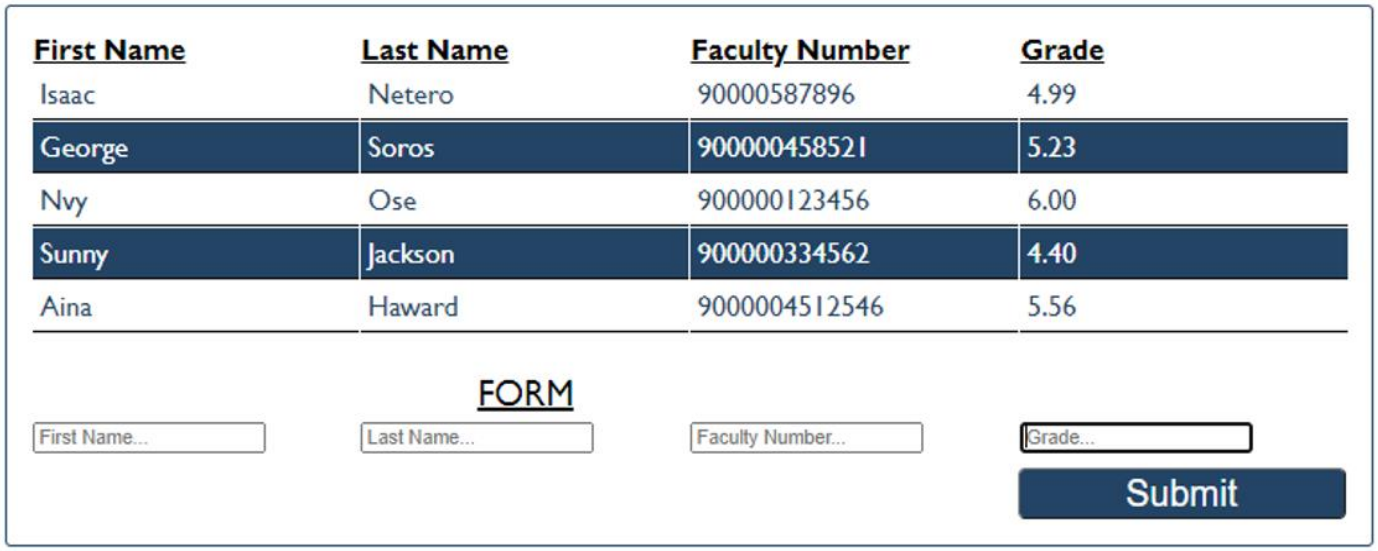
You need to write functionality for creating students. When creating a new student, make sure you name the properties accordingly.

You will also need to extract students. You will be given an **`HTML template`** with a table in it. Create an **`AJAX request`** that extracts all the students.

URL for this task: **http://localhost:3030/jsonstore/collections/students**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Screenshots

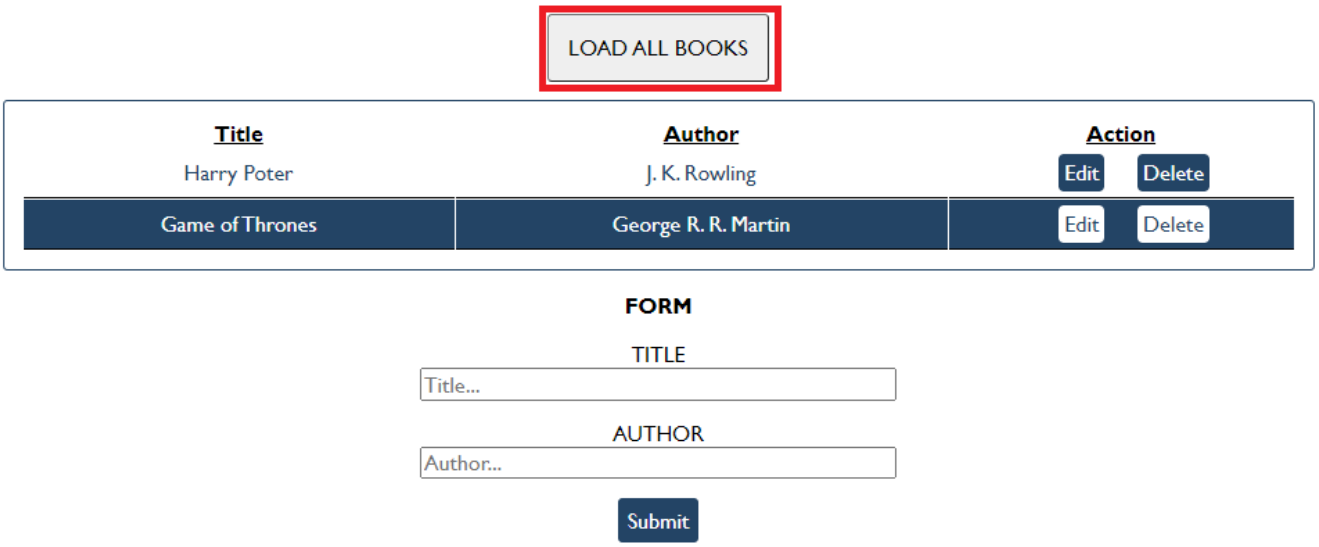


4. Book Library

First task is to "**GET**" all books. To consume the request with **POSTMAN** your **url** should be the **following**: **http://localhost:3030/jsonstore/collections/books**

Using the provided skeleton, write the missing functionalities.

Load all books by clicking the button "LOAD ALL BOOKS"



Get Book

This functionality is not needed in this task, but you can try it with postman by sending request to "GET" the Book with id:" d953e5fb-a585-4d6b-92d3-ee90697398a0". Send a GET request to this URL:

http://localhost:3030/jsonstore/collections/books/:id

SoftUni
© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

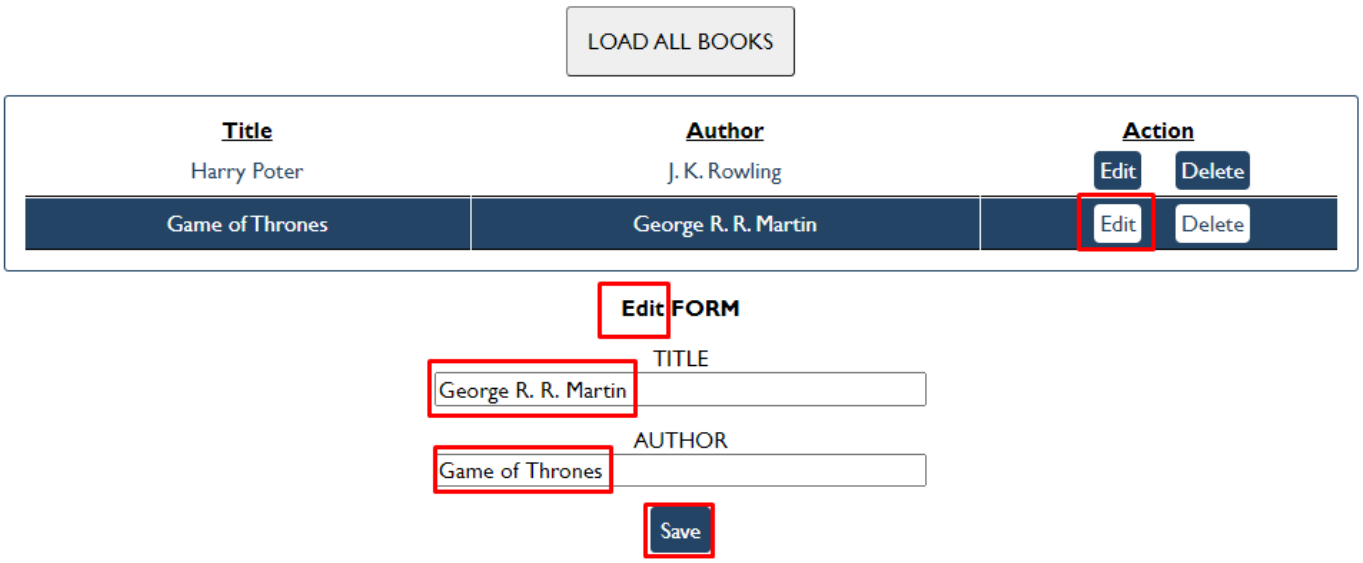
Create Book

Write functionality to create a new book, when the submit button is clicked. Before sending the request be sure the fields are not empty (make validation of the input). To **create** a book, you have to send a "**POST**" request and the JSON body should be in the **following** format:

```
{
  "author": "New Author",
  "title": "New Title"
}
```

Update Book

By clicking the edit button of a book, change the form like this:



The HTTP command "**PUT**" **modifies** an existing HTTP **resource**. The URL is:

http://localhost:3030/jsonstore/collections/books/:id

The JSON body should be in the **following** format:

```
{
  "author": "Changed Author",
  "title": "Changed Title"
}
```

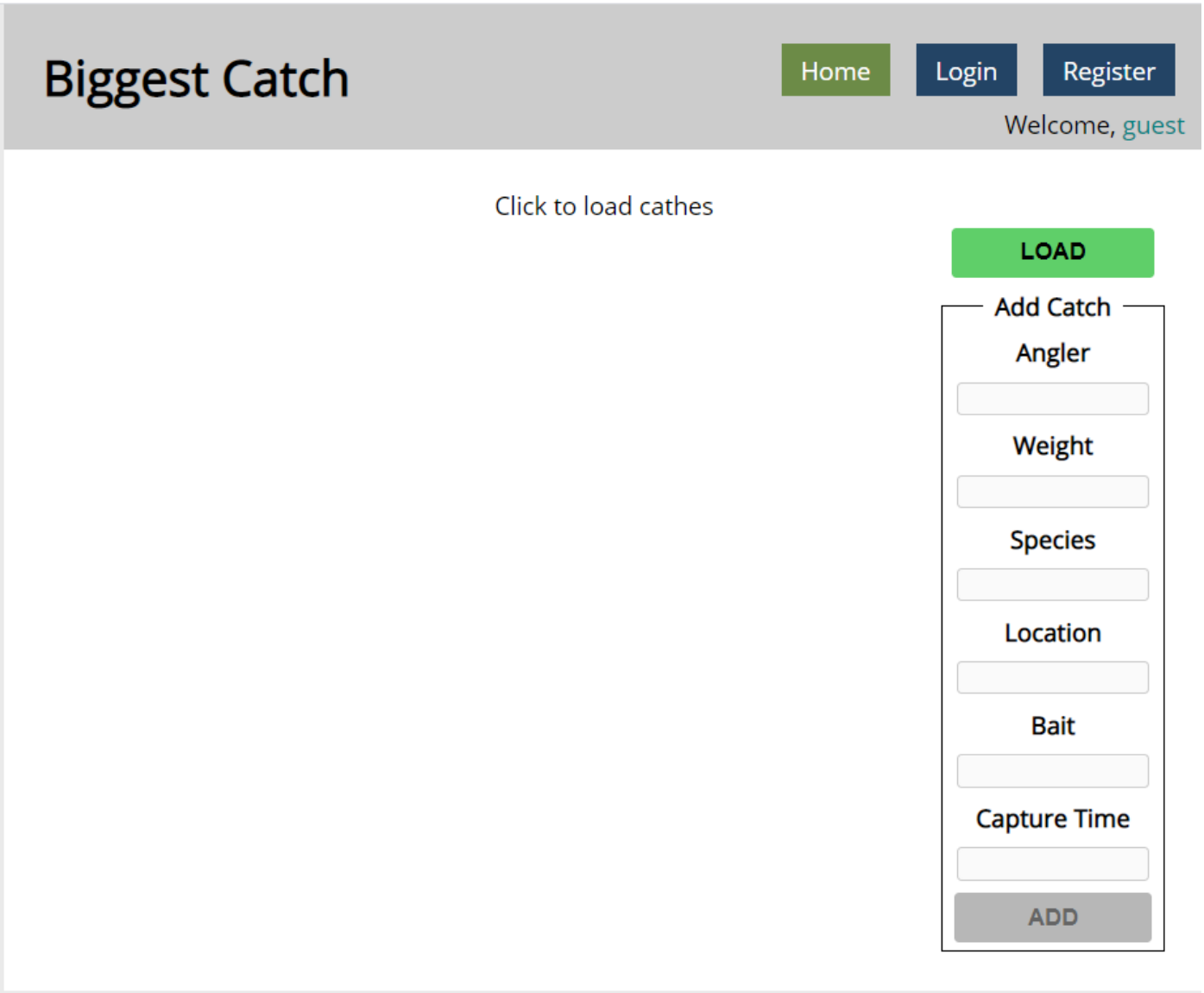
Delete Book

By clicking the delete button you have to delete the book, without any confirmation. To delete a book use "**DELETE**" command and send **REQUEST**: **http://localhost:3030/jsonstore/collections/books/:id**

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

5. Fisher Game

Use the provided skeleton and the server.

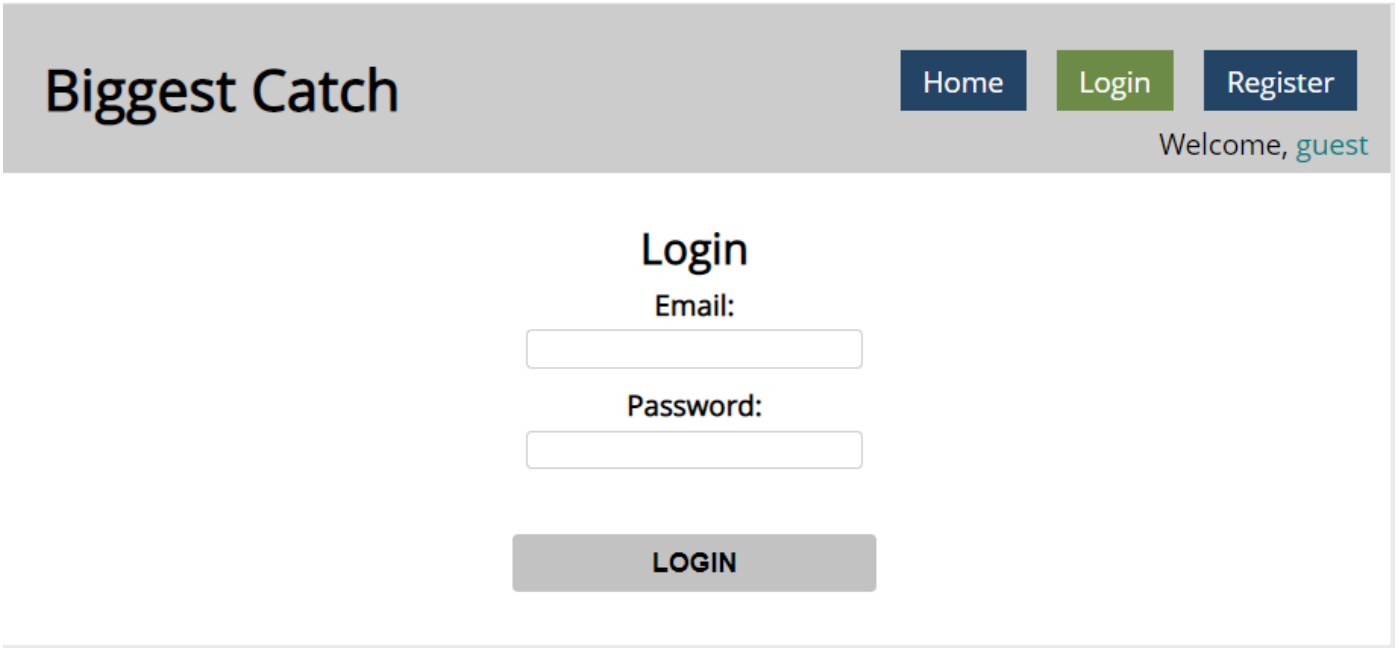


Login User

The **Login** page contains a form for existing user authentication. By given **username** and **password,** the app should login an existing user.

- After a **successful login** the **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**.
- Get request: **http://localhost:3030/users/login**

SoftUni
© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

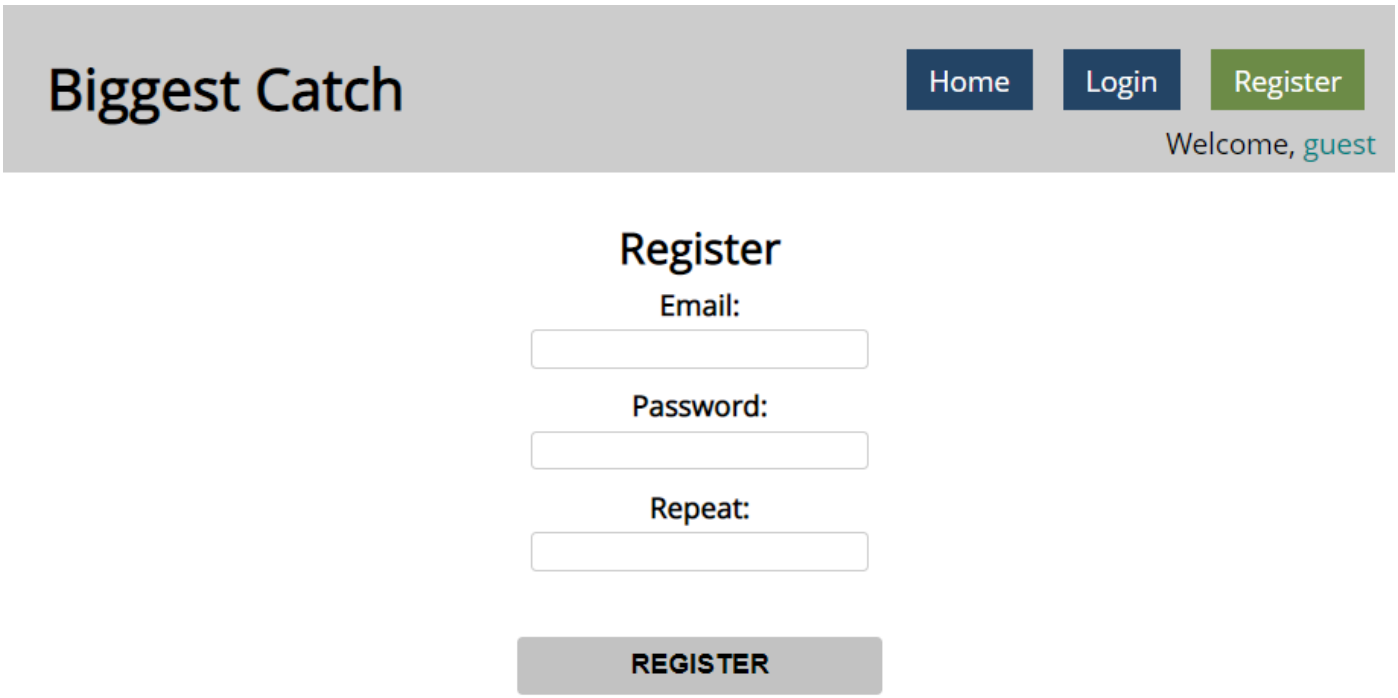


If the user is not logged in, all the buttons should be disabled except the "LOAD" button.

Register User

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

- 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** the **home page should be displayed**.
- Post request: **http://localhost:3030/users/register**



© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

Logout

The logout action is available to **logged-in users**. Send the following **request** to perform logout:

- Get: **http://localhost:3030/users/logout**

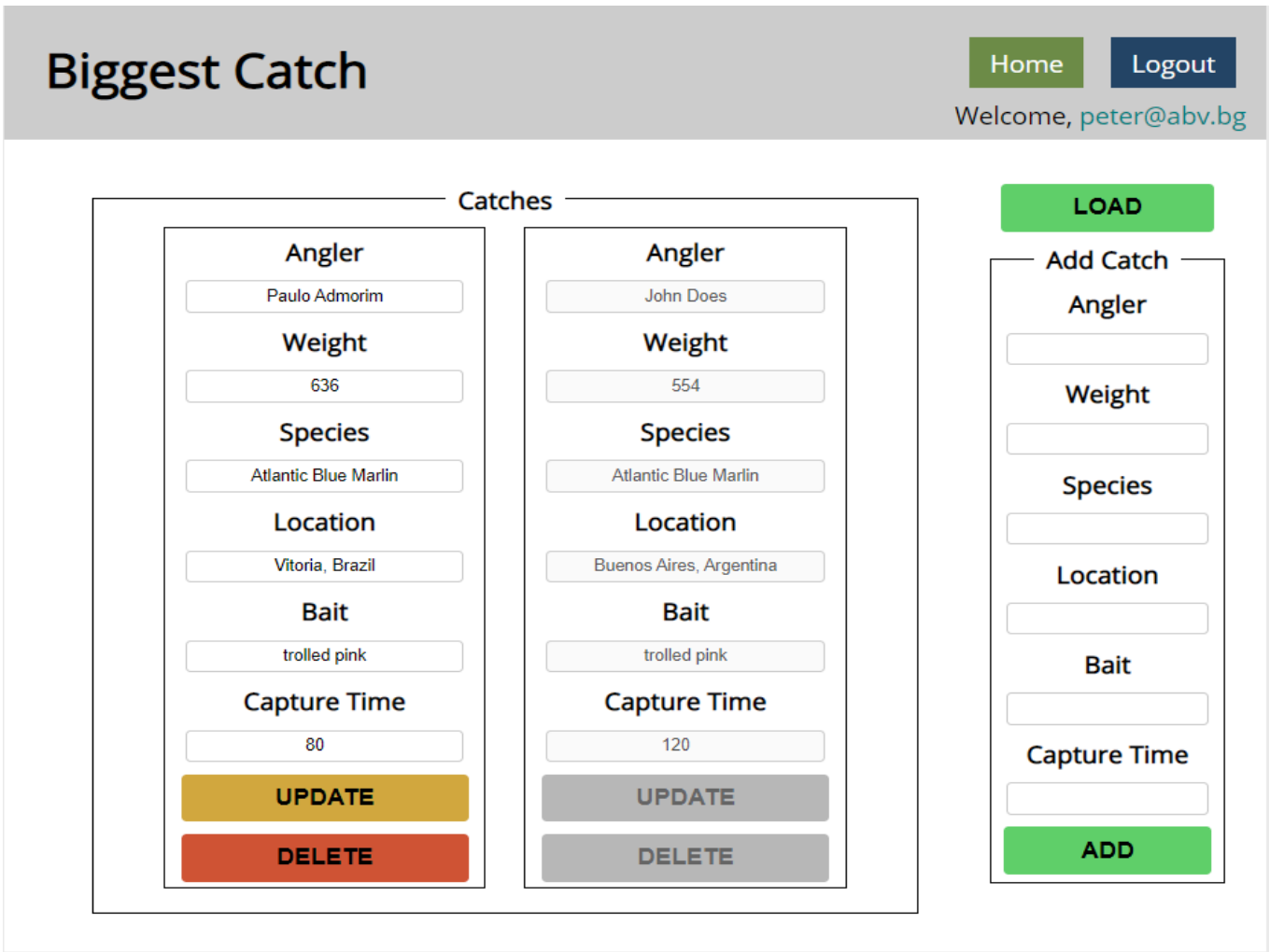
Required **headers** are described in the documentation. Upon success, the **REST service** will return an **empty response**. Clear any session information you've stored in browser storage.

If the logout was successful, **redirect** the user to the **Home** page and change the button in navigation.

Load catches

By clicking it you have to load all the catches from the server and render them like on the picture:

- Pressing the **[Load]** button should **list all** catches. (For all users)
- Pressing the **[Update]** button should send a **PUT** request, updating the catch in **http://localhost:3030/data/catches/:id**. (**Only for the creator of the catch**)
- Pressing the **[Delete]** button should delete the catch from **http://localhost:3030/data/catches/:id**. (**Only for the creator of the catch**)
- Pressing the **`[Add]`** button should submit a new catch with the values of the inputs in the fieldset with **`id="addFrom"`**. (**Only for logged in users**)
- Button **[Add]** should be **disabled** in there are no **logged in user**.
- Buttons **[Update]** and **[Delete]** should be **disabled** if the currently logged-in user is not the **author** of the catch.



© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.
Follow us:

Each catch should have:

- **`angler`** - **`string`** representing the name of the person who caught the fish
- **`weight`** - **`floating-point number`** representing the weight of the fish in kilograms
- **`species`** - **`string`** representing the name of the fish species
- **`location`** - **`string`** representing the location where the fish was caught
- **`bait`** - **`string`** representing the bait used to catch the fish
- **`captureTime`** - **`integer number`** representing the time needed to catch the fish in minutes

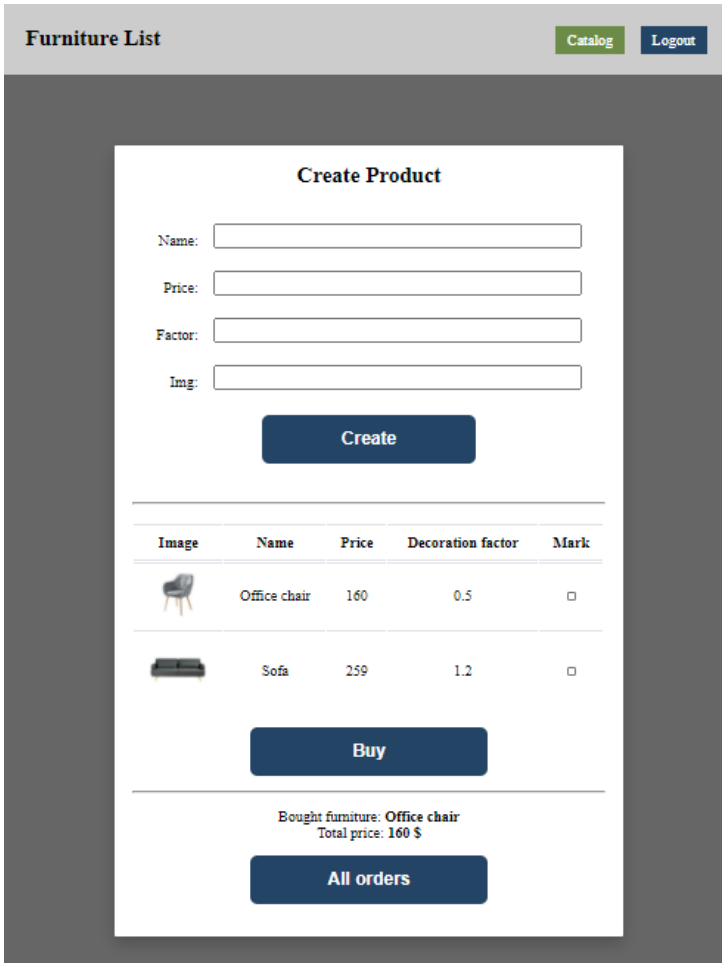
Use the following requests to access your data:

- **List All Catches**
  - Endpoint - **http://localhost:3030/data/catches**
  - Method: **GET**
- **Create a New Catch**
  - Endpoint: **http://localhost:3030/data/catches**
  - Method: **POST**
  - Request body (JSON): **`{"angler":"…", "weight":…, "species":"…", "location":"…", "bait":"…", "captureTime":…}`**
- **Update a Catch**
  - Endpoint: **http://localhost:3030/data/catches/:catchId**
  - Method: **PUT**
  - Request body (JSON): **`{"angler":"…", "weight":…, "species":"…", "location":"…", "bait":"…", "captureTime":…}`**
- **Delete a Catch**
  - Endpoint: **http://localhost:3030/data/catches /:catchId**
  - Method: **DELETE**

6. Furniture *

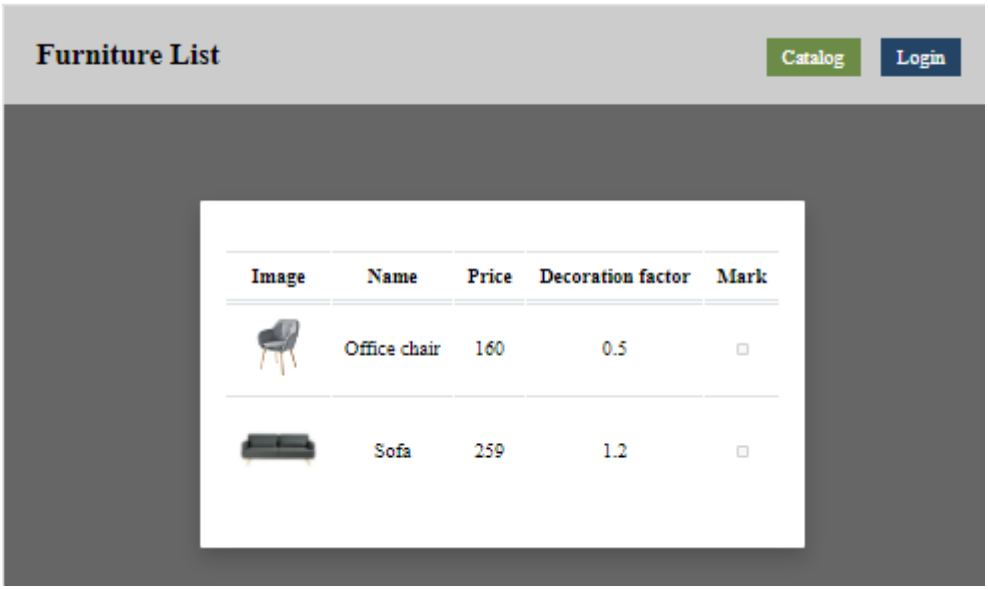
Your task is to write the functionality of app, which shows list of furniture. By logged in user there is a possibility to buy furniture and list the bought products of the logged user. Also logged user can create new products (offers).

© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.



Home page (not logged)

When the page is loaded the app should list all the furnitures in a table:



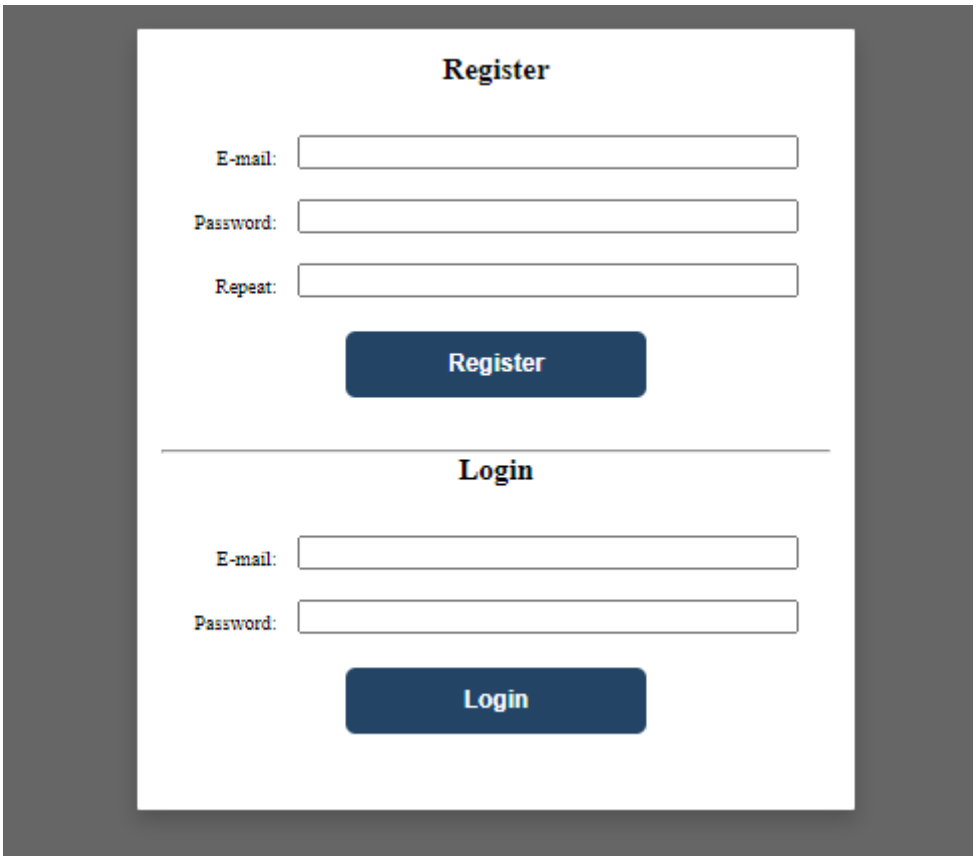
The checkbox should be disabled. You can send GET request on the URL:
http://localhost:3030/data/furniture

Auth page

When "Login" is clicked, the app should redirect to "Login page". There are two possibilities:

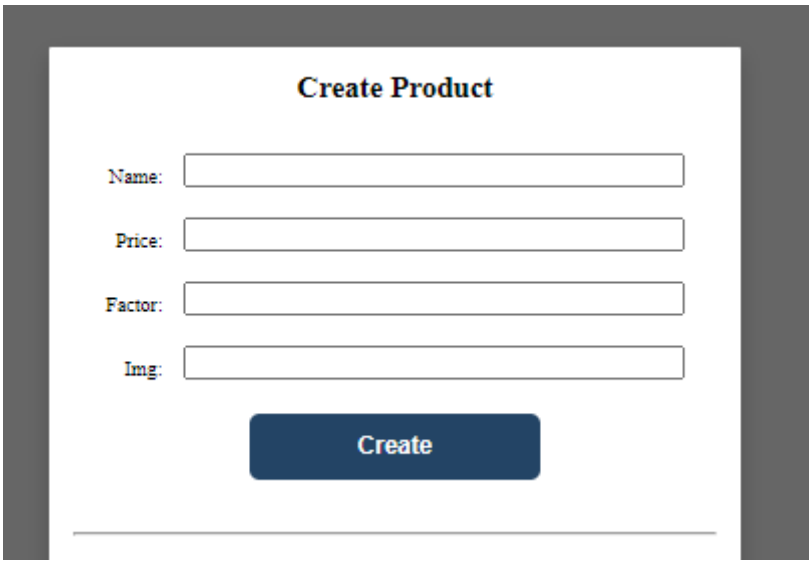
- to register a new user, send a POST request to the URL: **http://localhost:3030/users/register**

- to login, send a POST request to the URL: **http://localhost:3030/users/login**



Home page (logged in)

When the **"Create" button is clicked**, add a **new row to the table** for each piece of furniture with **name, price, factor and img**. Send POST request to: **http://localhost:3030/data/furniture**



When the **"Buy"** button is clicked, get all **checkboxes that are marked** and save the information for these orders on the server. Make POST request to: **http://localhost:3030/data/orders**

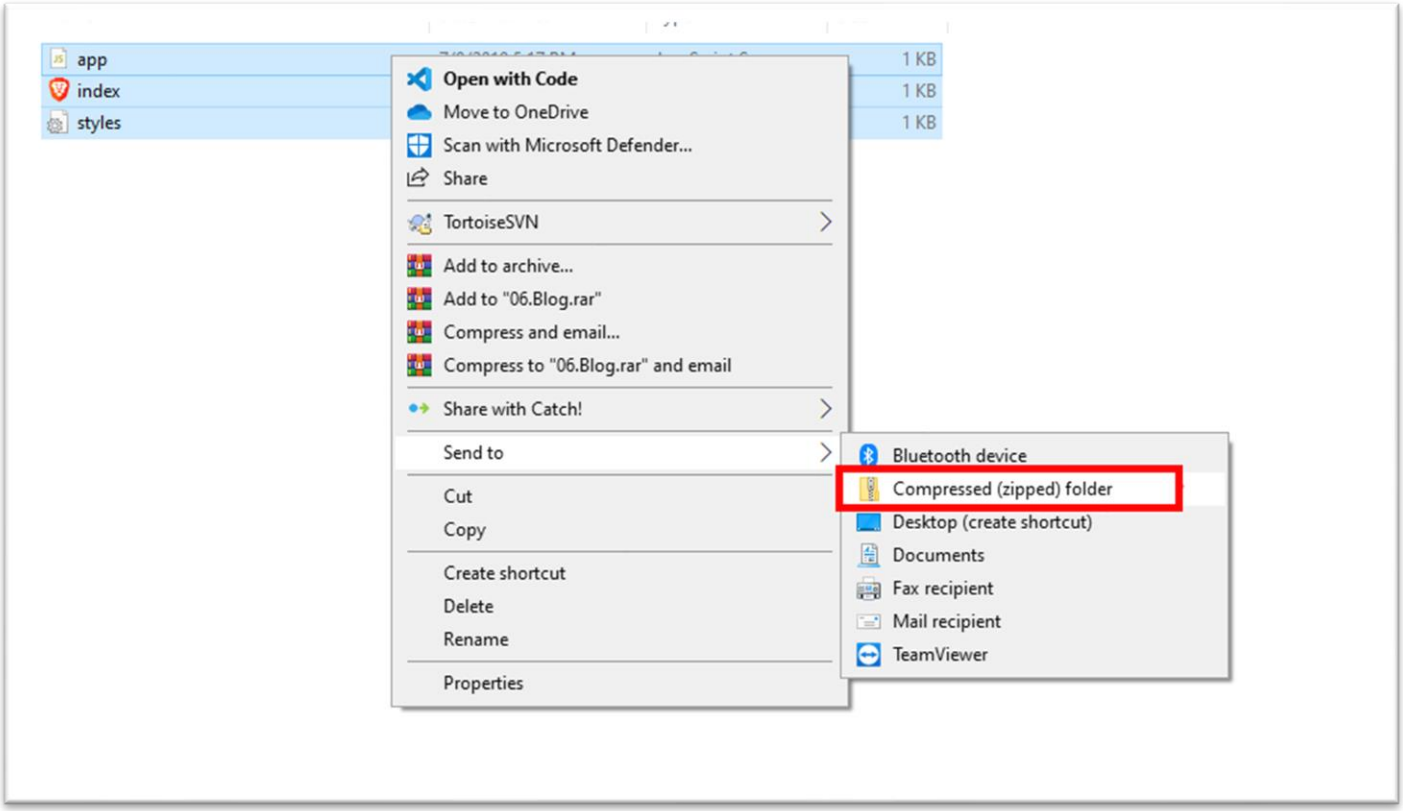
© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.

When the "**All orders**" button is clicked, get all bought furniture of the current user, and show their names and the total price, as shown on the picture:

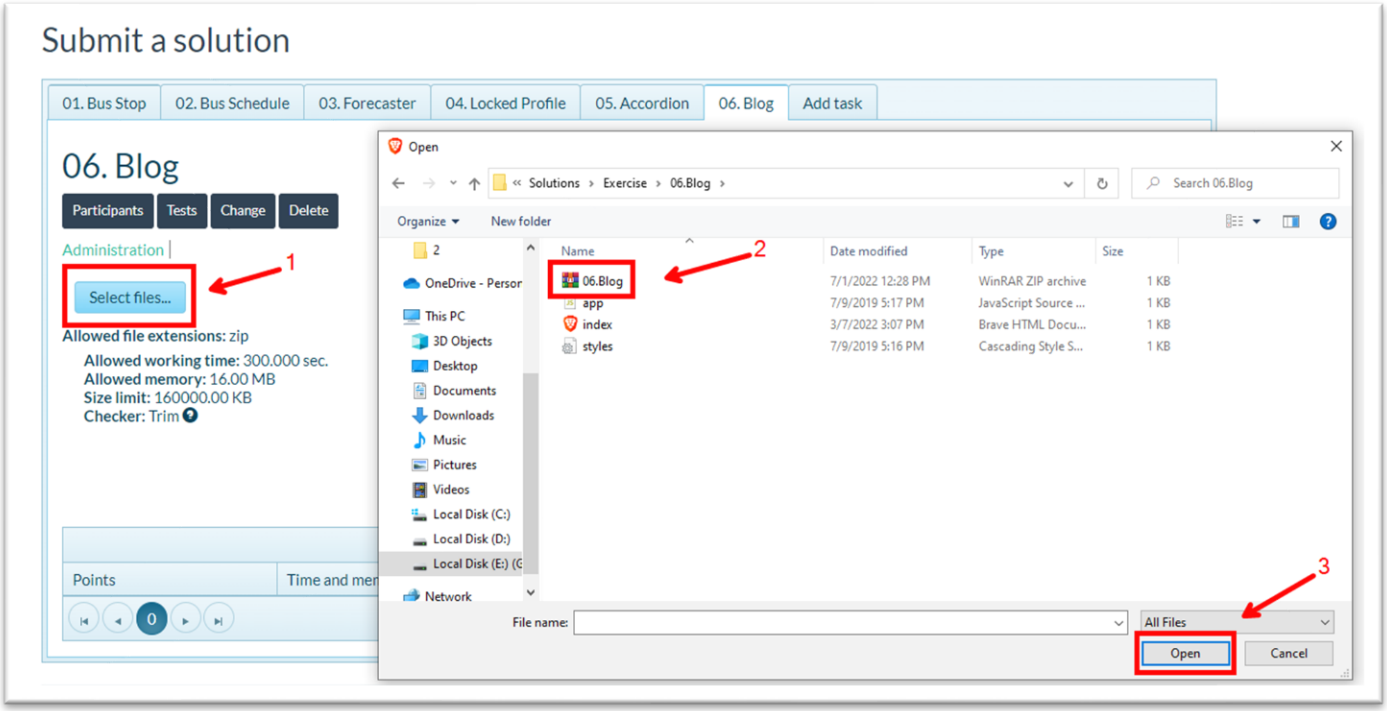
This could happen with GET request on this URL:
http://localhost:3030/data/orders?where=_ownerId%3D{userId}

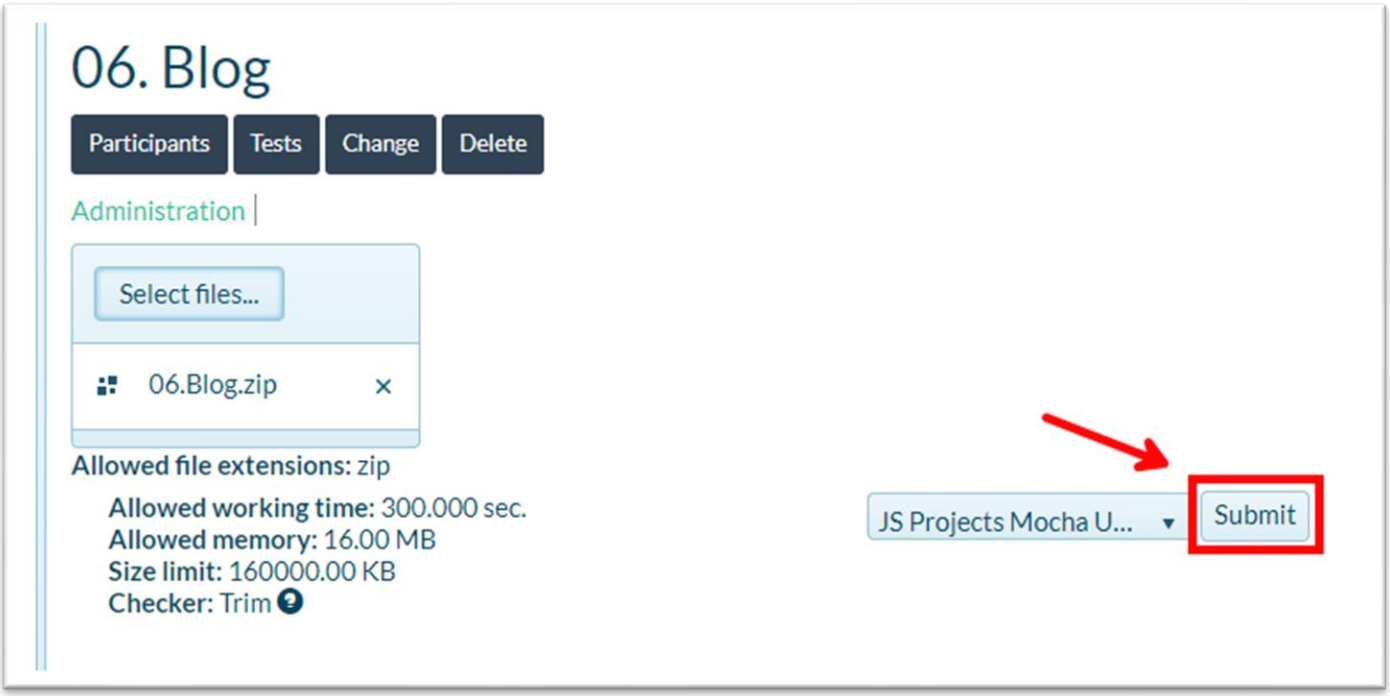
Submitting Your Solution

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.



© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.





© SoftUni – about.softuni.bg. Copyrighted document. Unauthorized copy, reproduction or use is not permitted.