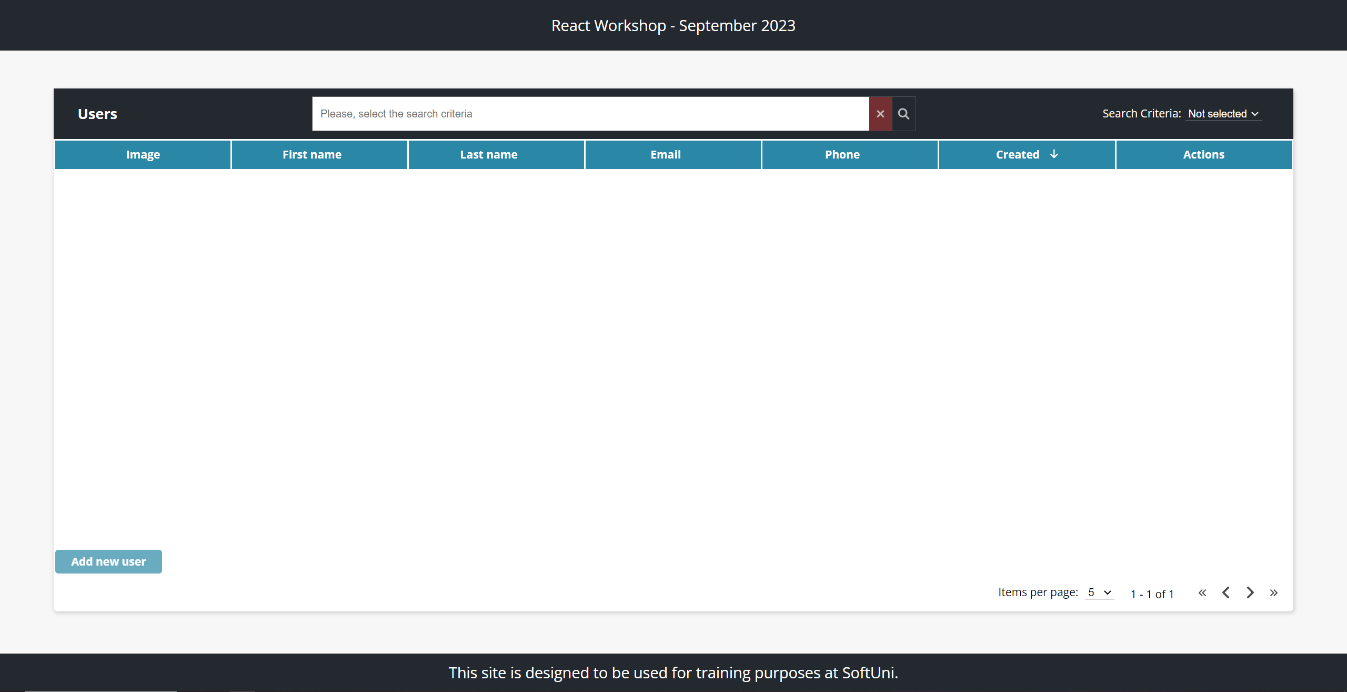
# ReactJS – User List Workshop

This is application was created as a workshop for the [ReactJS Course](https://softuni.bg/opencourses/react-js) @ [Software University](https://softuni.bg)

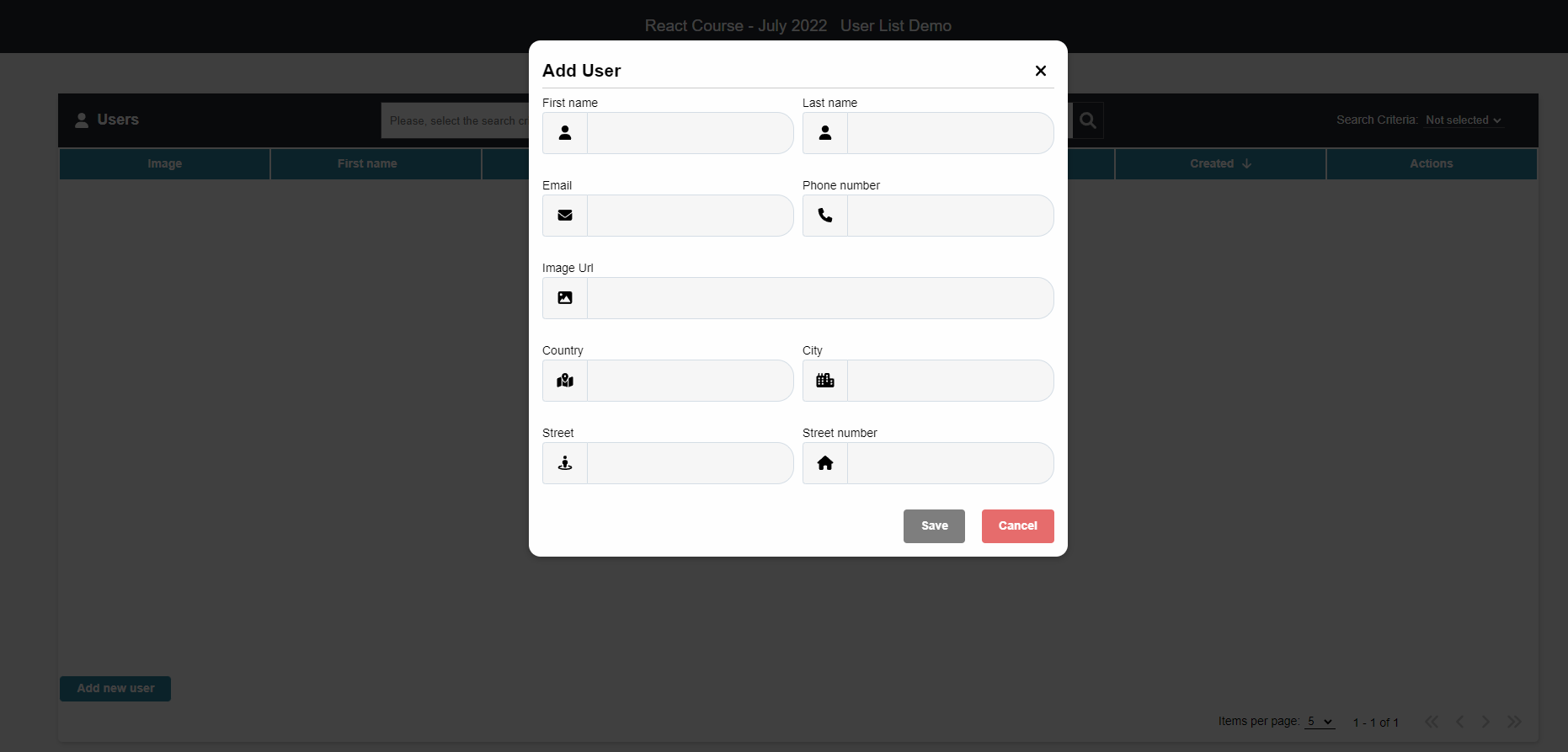
## Task Requirements



Use the provided **html** **structure** and **styles** to **create** a page with the following functionalities:

### Create user

Clicking on the **Add new user** button should display a form for creating users.



Send the following **request**:

Method: POST

URL: /jsonstore/users

The service expects a body with the following shape:

{

firstName,

lastName,

email,

imageUrl,

phoneNumber,

createdAt,

updatedAt,

address: {

country,

city,

street,

streetNumber

}

}

Then the service will return the newly created record.

Картина, която съдържа текст

Описанието е генерирано автоматично

### User info

Clicking the **[i]** button in the Actions section should display a window containing the information for the selected user.

Картина, която съдържа текст

Описанието е генерирано автоматично

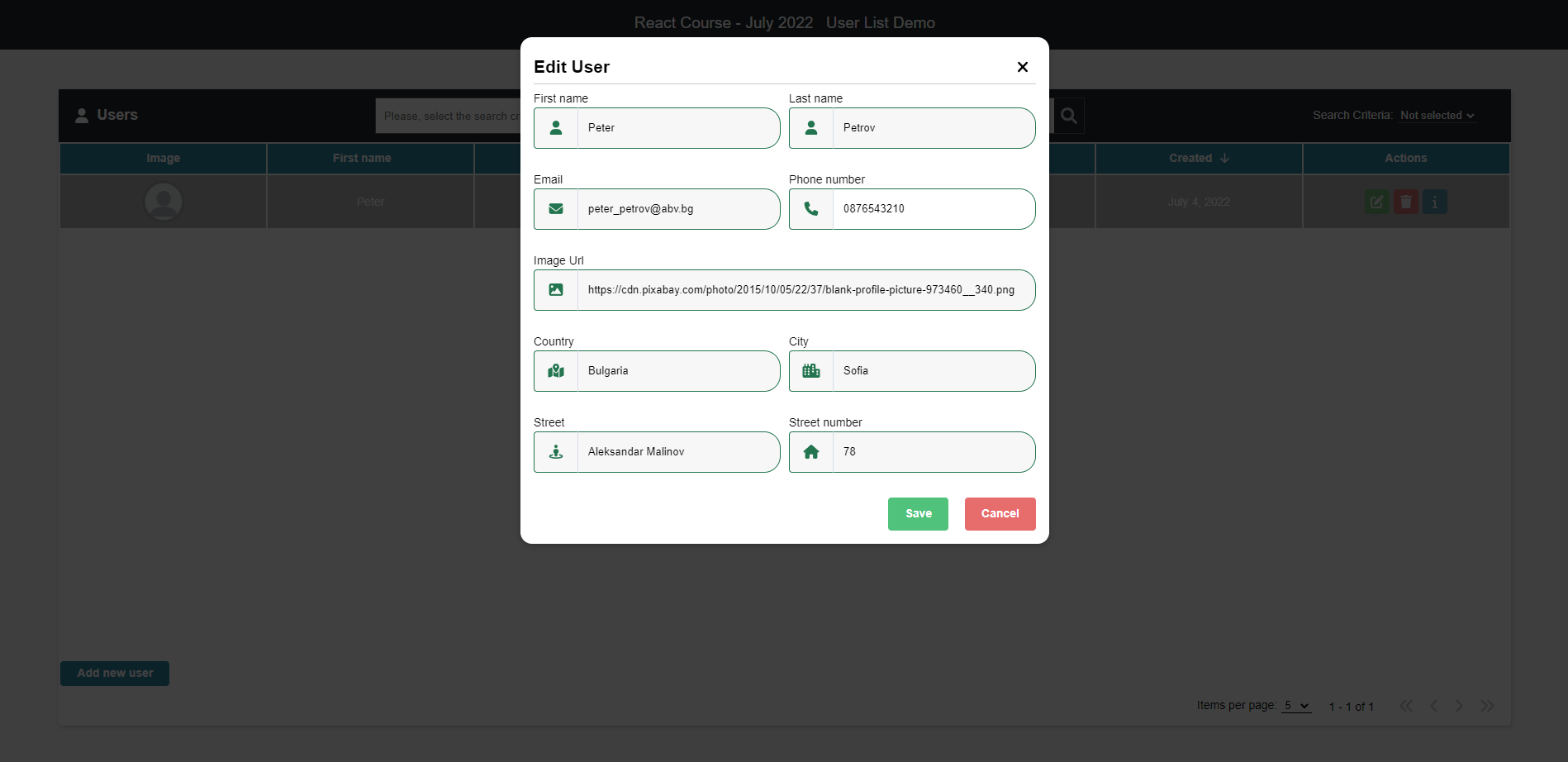
To retrieve the information, send the following **request**:

Method: GET

URL: /jsonstore/users/{userId}

### Edit user

Clicking the **[Edit]** button in the Actions section should display a form for editing the user information.



If the form is filled correctly send the following request:

Method: PUT

URL: /jsonstore/users/{userId}

The service expects a body with the following shape:

{

firstName,

lastName,

email,

imageUrl,

phoneNumber,

createdAt,

updatedAt,

address: {

country,

city,

street,

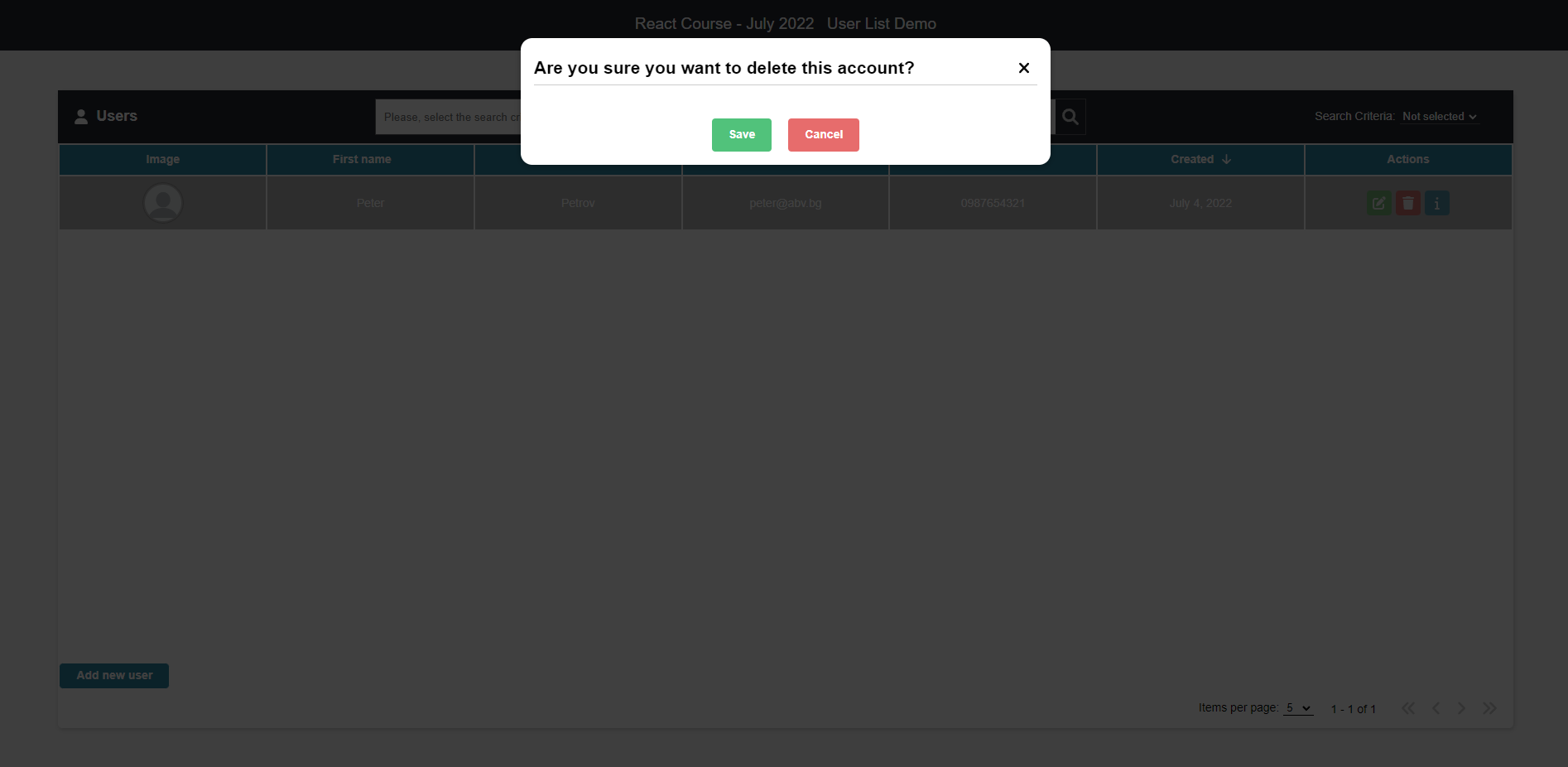
streetNumber

}

}

Then the service will return the newly modified record.

### Delete user

Clicking the **[Delete]** button in the Actions section should display a confirmation window.   
‘

Clicking the **[ Save ]** button should send the following request:

Method: DELETE

URL: /jsonstore/users/{userId}

Then the record with the given id will be deleted.

### Search

By selecting a **search criterion** and typing in the search bar the **criteria**, only the matching records should be visible

Картина, която съдържа текст, екранна снимка, монитор, закрито

Описанието е генерирано автоматично

### Pagination\*

Selecting how many items a page must include, implement a pagination

Картина, която съдържа маса

Описанието е генерирано автоматично

If there are more records than the maximum items per page, a second page must become active.

Картина, която съдържа текст, екранна снимка, монитор

Описанието е генерирано автоматично

### Sorting\*

Clicking on some of the main fields should **sort** the records in an **ascending** or **descending** order, depending on how many times it is clicked.

Ascending by ‘**Last name’**:

Картина, която съдържа текст, екранна снимка, монитор

Описанието е генерирано автоматично

Descending by ‘**Last name’**:

Картина, която съдържа маса

Описанието е генерирано автоматично

### Spinner\*

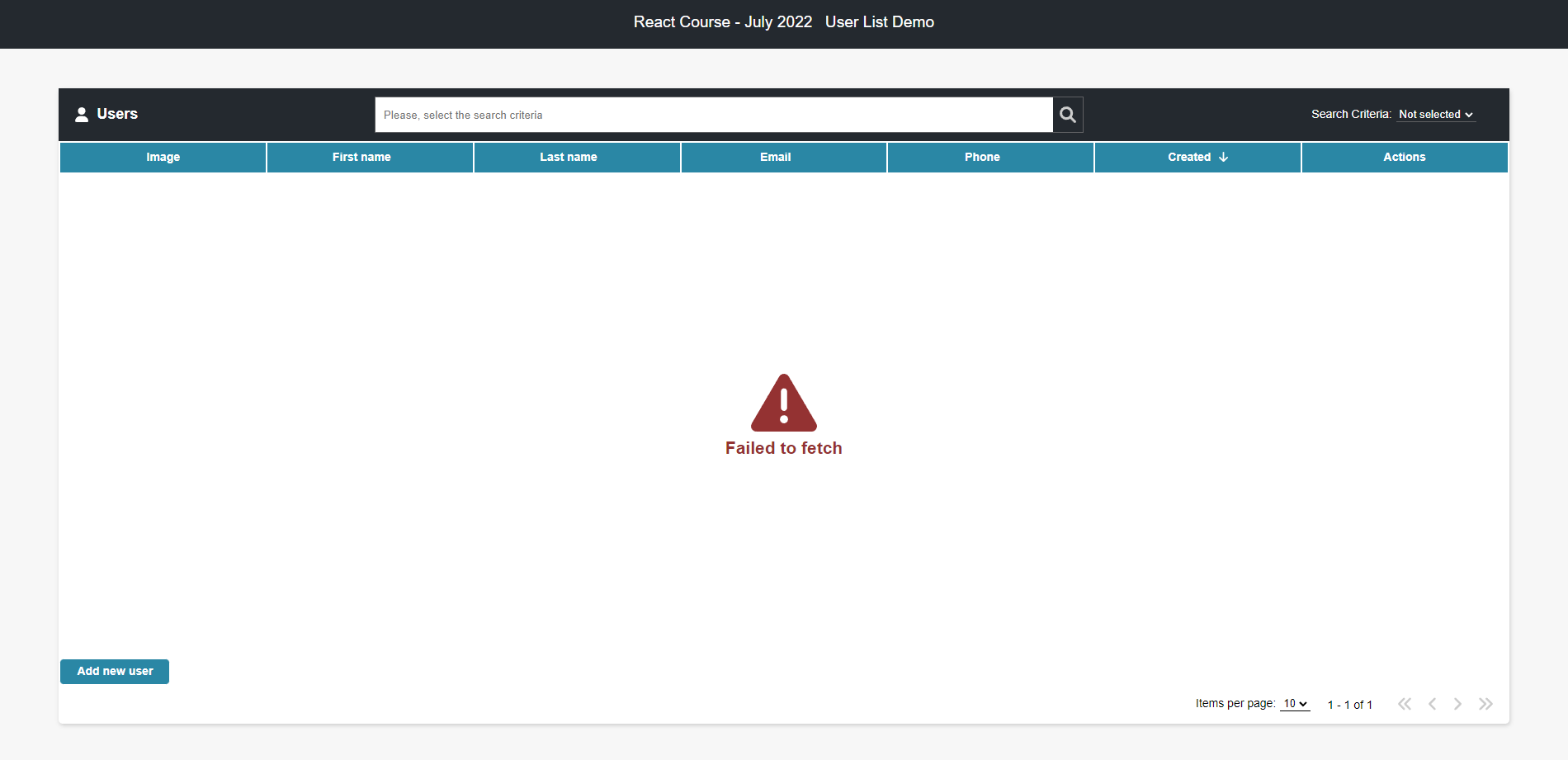
While waiting a response from a request, a spinner should be displayed:

Картина, която съдържа текст, екранна снимка, монитор

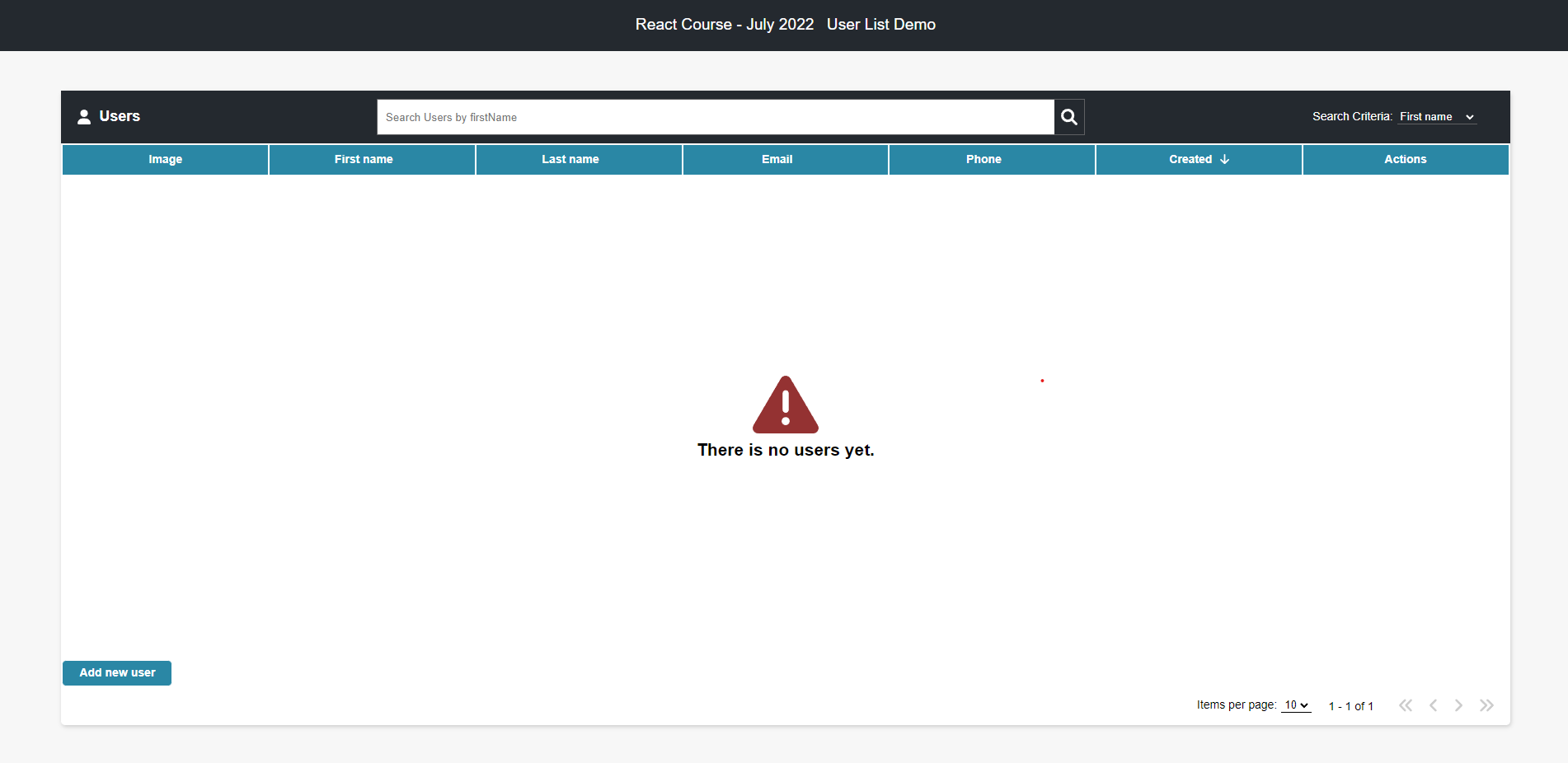
Описанието е генерирано автоматично

### Errors

If a request failed, the following error should be displayed:



If there are no records yet, the following error should be displayed:



If there is no record matching the given search criterion the following error should be displayed:

Картина, която съдържа текст

Описанието е генерирано автоматично

## Initialize the App

Use the **"npm create vite user-list-demo"** command to create a new React App.

**3. Using the Local REST Service**

**Starting the Service**

The REST service will be in a folder named “server” inside the provided resources archive. It has no dependencies and can be started by opening a terminal in its directory and executing:

**node server.js**

If everything initialized correctly, you should see a message about the **host address and port** on which the service will respond to requests.

**Sending Requests**

To send a request, use the **hostname** and **port**, shown in the initialization log and **resource address** and **method** as described in the **application requirements**. If data needs to be included in the request, it must be **JSON-encoded**, and the appropriate **Content-Type** **header** must be added. Similarly, if the service is to return data, it will be JSON-encoded. Note that **some requests do not return a body** and attempting to parse them will throw an exception.

**More Information**

You can find more details on the [GitHub repository of the service](https://github.com/softuni-practice-server/softuni-practice-server/blob/master/README.md).