

# AUI LAB4 REPORT

Hang Liu s179216

1. View (HTML page and JavaScript code) presenting list of all categories. View should allow for removing selected category. It is enough to display single user friendly value representing category. (1 point)

```
*.js files are supported by IntelliJ IDEA Ultimate
1  import {
2      getParameterByName,
3      clearElementChildren,
4      createLinkCell,
5      createButtonCell,
6      createTextCell,
7      createImageCell,
8      setTextNode
9  } from '../js/dom_utils.js';
10 import {getBackendUrl} from '../js/configuration.js';
11
12 window.addEventListener('load', () => {
13     fetchAndDisplayUser();
14     fetchAndDisplayTeachers();
15 });
16
17 /**
18  * Fetches all users and modifies the DOM tree in order to display them.
19  */
20 function fetchAndDisplayTeachers() {
21     const xhttp = new XMLHttpRequest();
22     xhttp.onreadystatechange = function () {
23         if (this.readyState === 4 && this.status === 200) {
24             displayTeachers(JSON.parse(this.responseText))
25         }
26     };
27     xhttp.open("GET", getBackendUrl() + '/api/users/' + getParameterByName('user') + '/teachers', true);
28     xhttp.send();
29 }
```

\*.js files are supported by IntelliJ IDEA Ultimate

```
34 * @param {{teachers: {id: number, name:string}}} teachers
35 */
36 function displayTeachers(teachers) {
37     let tableBody = document.getElementById('tableBody');
38     clearElementChildren(tableBody);
39     teachers.teachers.forEach(teacher => {
40         tableBody.appendChild(createTableRow(teacher));
41     })
42     let add_teacher = document.getElementById('add_teacher');
43     add_teacher.href = '../teacher_add/teacher_add.html?user=' + getParameterByName('user');
44 }
45
46 /**
47  * Creates single table row for entity.
48  *
49  * @param {{id: number, name: string}} teacher
50  * @returns {HTMLTableRowElement}
51  */
52 function createTableRow(teacher) {
53     let tr = document.createElement('tr');
54     tr.appendChild(createImageCell(getBackendUrl() + '/api/users/' + getParameterByName('user') + '/teachers/'
55         + teacher.id + '/portrait'));
56     tr.appendChild(createTextCell(teacher.name));
57     tr.appendChild(createLinkCell('view', '../teacher_view/teacher_view.html?user='
58         + getParameterByName('user') + '&teacher=' + teacher.id));
59     tr.appendChild(createLinkCell('edit', '../teacher_edit/teacher_edit.html?user='
60         + getParameterByName('user') + '&teacher=' + teacher.id));
61     tr.appendChild(createButtonCell('delete', () => deleteCharacter(teacher.id)));
62     return tr;
63 }
```

```
62     return tr;
63 }
64
65 /**
66  * Deletes entity from backend and reloads table.
67  *
68  * @param {number} teacher to be deleted
69  */
70 function deleteCharacter(teacher) {
71     const xhttp = new XMLHttpRequest();
72     xhttp.onreadystatechange = function () {
73         if (this.readyState === 4 && this.status === 202) {
74             fetchAndDisplayTeachers();
75         }
76     };
77     xhttp.open("DELETE", getBackendUrl() + '/api/users/' + getParameterByName('user')
78         + '/teachers/' + teacher, true);
79     xhttp.send();
80 }
81
82 /**
83  * Fetches single user and modifies the DOM tree in order to display it.
84  */
85 function fetchAndDisplayUser() {
86     const xhttp = new XMLHttpRequest();
```

```

82
83  /**
84   * Fetches single user and modifies the DOM tree in order to display it.
85   */
86  function fetchAndDisplayUser() {
87      const xhttp = new XMLHttpRequest();
88      xhttp.onreadystatechange = function () {
89          if (this.readyState === 4 && this.status === 200) {
90              displayUser(JSON.parse(this.responseText))
91          }
92      };
93      xhttp.open("GET", getBackendUrl() + '/api/users/' + getParameterByName('user'), true);
94      xhttp.send();
95  }
96
97  /**
98   * Updates the DOM tree in order to display user.
99   *
100   * @param {{login: string, name: string, surname:string}} user
101   */
102  function displayUser(user) {
103      setTextNode('username', user.login);
104      setTextNode('name', user.name);
105      setTextNode('surname', user.surname);
106      setTextNode('birthDate', user.birthDate);
107      setTextNode('email', user.email);
108  }
109
110

```

← → ↻ ⓘ 127.0.0.1:8083/user\_list/user\_list.html

## University teacher management

main about users

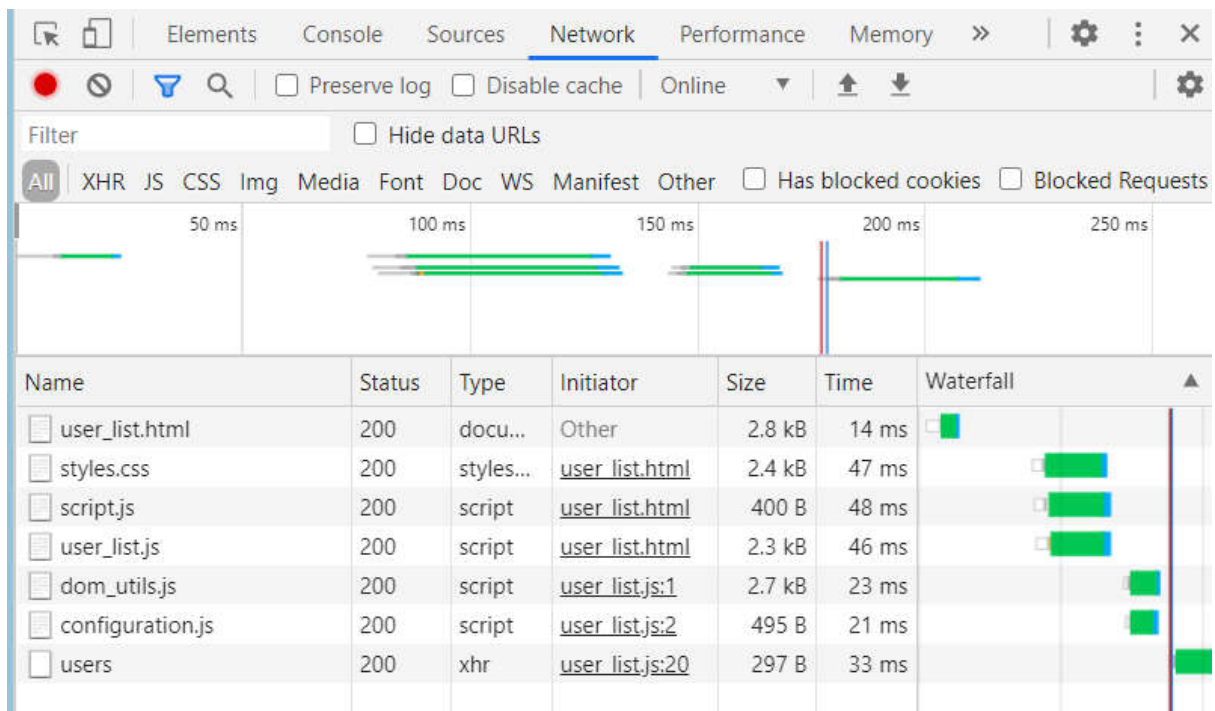
### Users

[ADD USER](#)

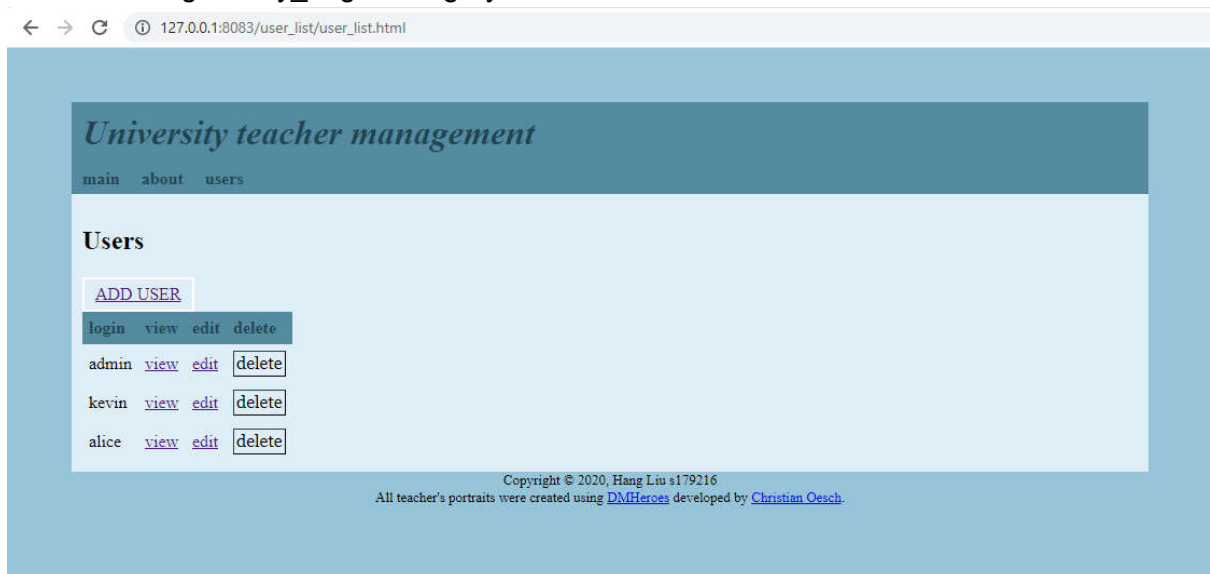
login	view	edit	delete
admin	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
kevin	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
alice	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
Dobby_Login	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>

Copyright © 2020, Hang Liu s179216  
All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#).











Before removing selected category, the developer tools show:



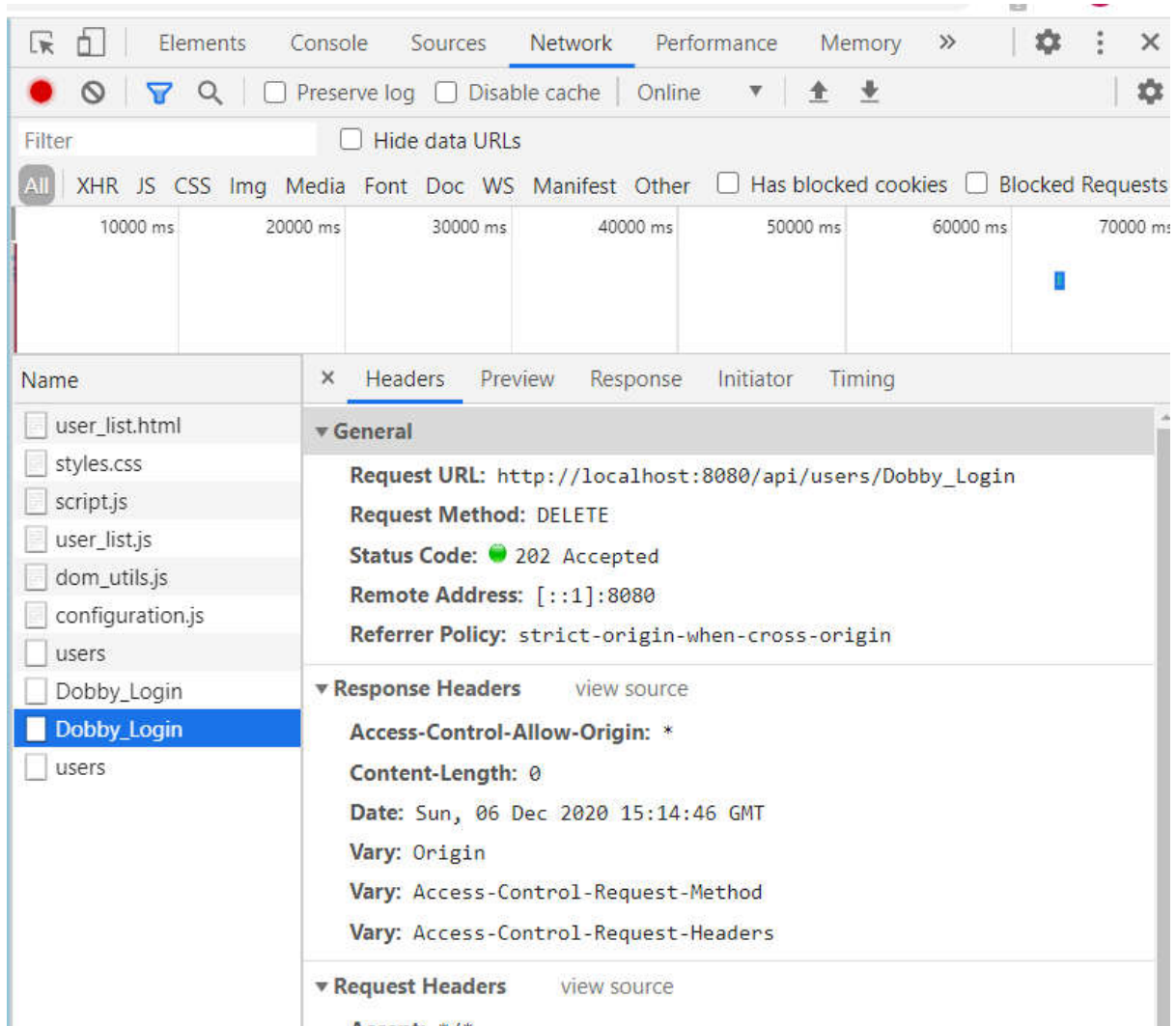
After removing Dobby\_Login category:



And the developer tool show:

<div> <div> <div> <div></div> <div></div> </div> <div> <div>Elements</div> <div>Console</div> <div>Sources</div> <div>Network</div> <div>Performance</div> <div>Memory</div> <div>»</div> </div> <div> <div>⚙</div> <div>⋮</div> <div>➤</div> </div> </div> <div> <div> <div>⛔</div> <div>🔍</div> </div> <div> <input type="checkbox"/> Preserve log <input type="checkbox"/> Disable cache <div>Online</div> <div>⬆</div> <div>⬇</div> </div> </div> <div> <div>Filter</div> <input type="checkbox"/> Hide data URLs </div> <div> <div>All</div> <div>XHR</div> <div>JS</div> <div>CSS</div> <div>Img</div> <div>Media</div> <div>Font</div> <div>Doc</div> <div>WS</div> <div>Manifest</div> <div>Other</div> <div><input type="checkbox"/> Has blocked cookies</div> <div><input type="checkbox"/> Blocked Reques</div> </div> </div>						
<div> <div>10000 ms</div> <div>20000 ms</div> <div>30000 ms</div> <div>40000 ms</div> <div>50000 ms</div> <div>60000 ms</div> <div>70000</div> </div>						
Name	Status	Type	Initiator	Size	Time	Waterfall
 user_list.html	200	docu...	Other	2.8 kB	14 ms	
 styles.css	200	styles...	<a href="#">user_list.html</a>	2.4 kB	47 ms	
 script.js	200	script	<a href="#">user_list.html</a>	400 B	48 ms	
 user_list.js	200	script	<a href="#">user_list.html</a>	2.3 kB	46 ms	
 dom_utils.js	200	script	<a href="#">user_list.js:1</a>	2.7 kB	23 ms	
 configuration.js	200	script	<a href="#">user_list.js:2</a>	495 B	21 ms	
 users	200	xhr	<a href="#">user_list.js:20</a>	297 B	33 ms	
 Dobby_Login	200		Other	0 B	5 ms	
 Dobby_Login	202	xhr	<a href="#">user_list.js:64</a>	202 B	53 ms	
 users	200	xhr	<a href="#">user_list.js:20</a>	283 B	26 ms	

I will select Dobby\_Login to have a further look.



Here are the snippets about the code, in html there is a button:

```
<section class="main-section">
  <article class="text--justified">
    <header><h2>Users</h2></header>
    <a class="custom-button" href="../user_add/user_add.html">ADD USER</a>
    <table class="data-table">
      <thead>
        <tr>
          <th>login</th>
          <th>view</th>
          <th>edit</th>
          <th>delete</th>
        </tr>
      </thead>
      <tbody id="tableBody">
        <tr>
          <td>name</td>
          <td><a href="#">view</a></td>
          <td><a href="#">edit</a></td>
          <td>
            <button class="ui-button ui-control">delete</button>
          </td>
        </tr>
      </tbody>
    </table>
  </article>
</section>
```

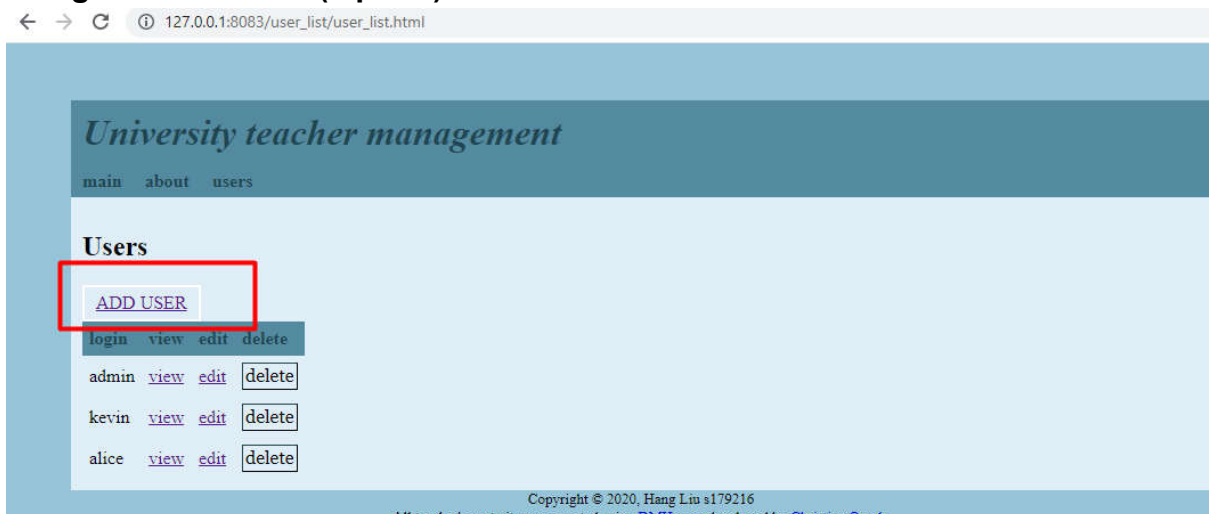


And we use JS to replace that delete value. And when we click it, it will jump to our AJAX function. It will perform a deleting request.

```
function createTableRow(user) {
    let tr = document.createElement('tr');
    tr.appendChild(createTextCell(user));
    tr.appendChild(createLinkCell('view', '../user_view/user_view.html?user=' + user));
    tr.appendChild(createLinkCell('edit', '../user_edit/user_edit.html?user=' + user));
    tr.appendChild(createButtonCell('delete', () => deleteUser(user)));
    return tr;
}

/**
 * Deletes entity from backend and reloads table.
 *
 * @param {string } user to be deleted
 */
function deleteUser(user) {
    const xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState === 4 && this.status === 202) {
            fetchAndDisplayUsers();
        }
    };
    xhttp.open("DELETE", getBackendUrl() + '/api/users/' + user, true);
    xhttp.send();
}
```

## 2. View for adding new category (form). View should be accessible from categories list view. (1 point)



Above picture shows adding a new category by element ADD\_USER. And when I click it, it will jump to user\_add.html

University teacher management

main about users

### Creating new user

Login:

Name:

Surname:

BirthDate:

Password:

Email:

Copyright © 2020, Hang Liu s179216  
All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#).

Once I fill the form, I will click the create button.

University teacher management

main about users

### Creating new user

Login:

Name:

Surname:

BirthDate:

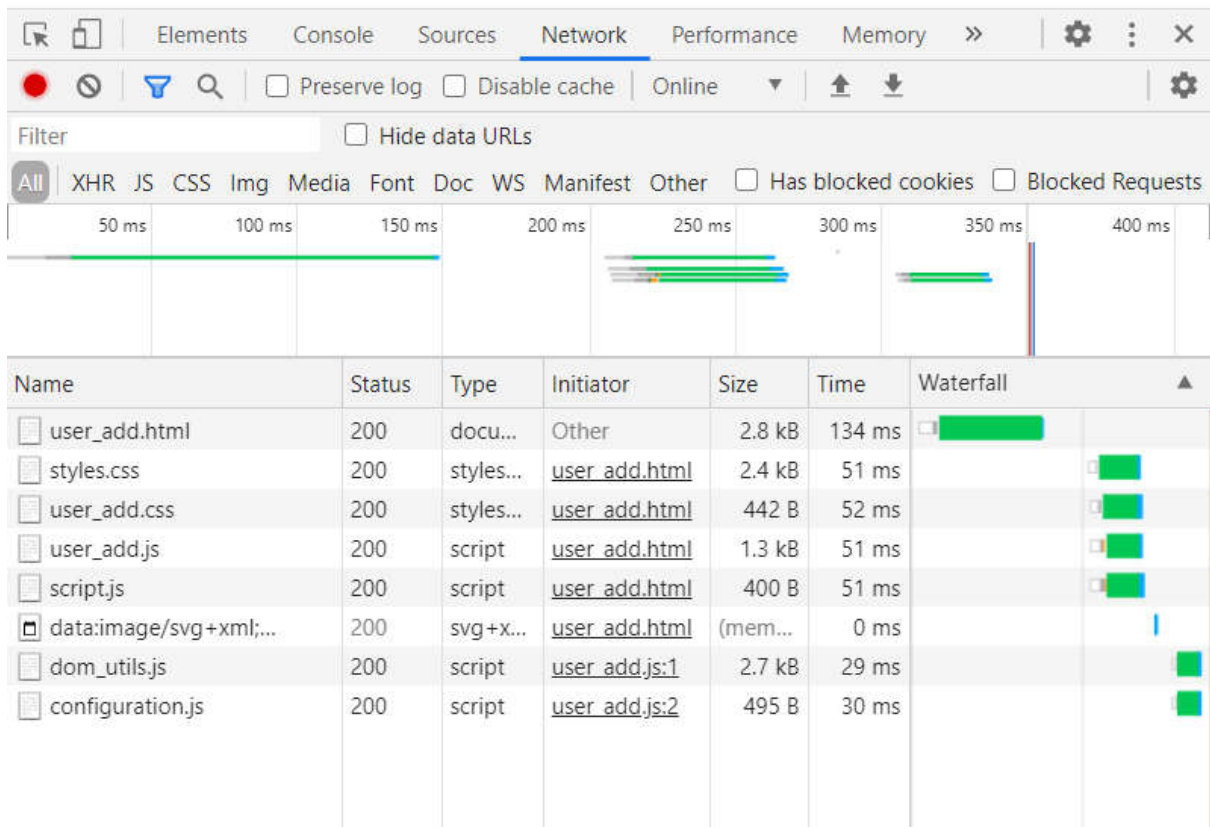
Password:

Email:

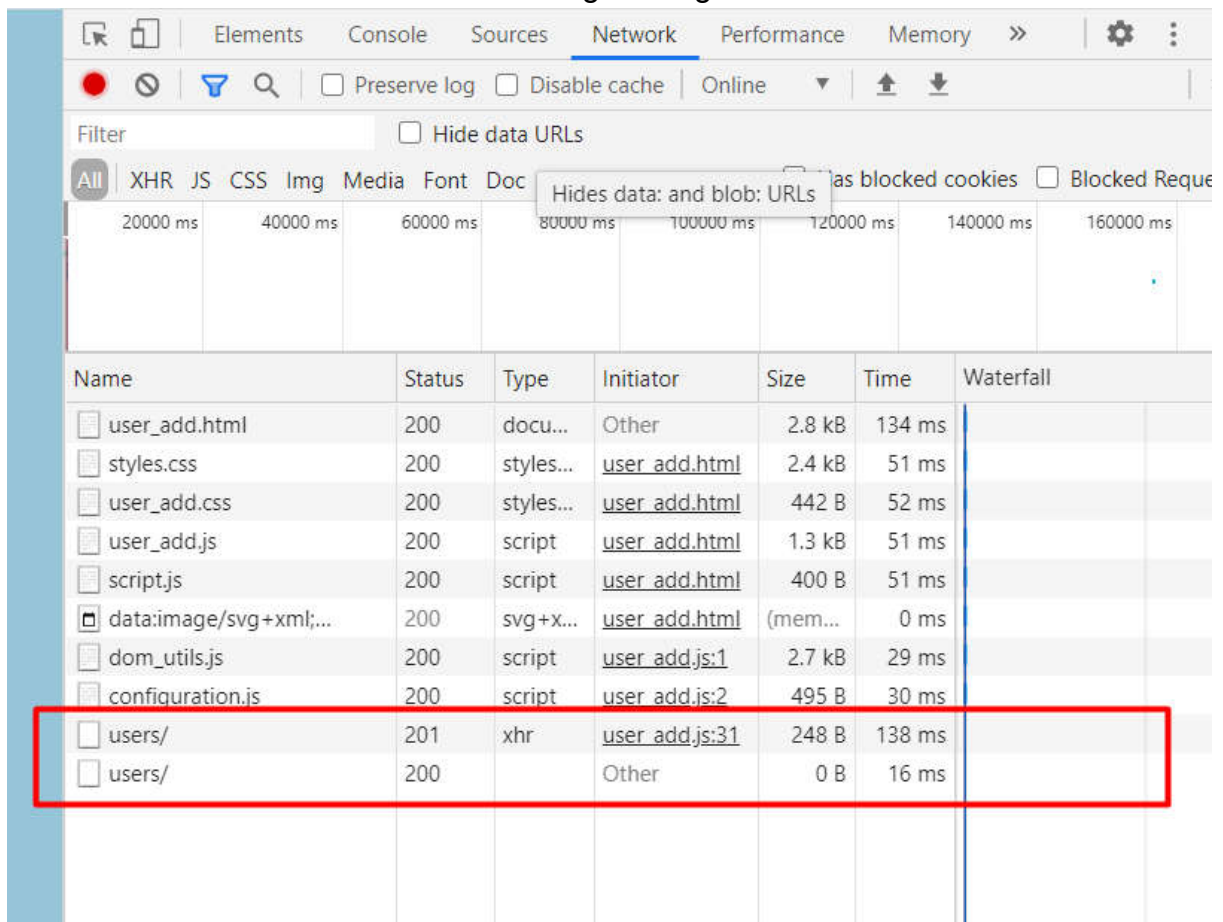
Copyright © 2020, Hang Liu s179216  
All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#).

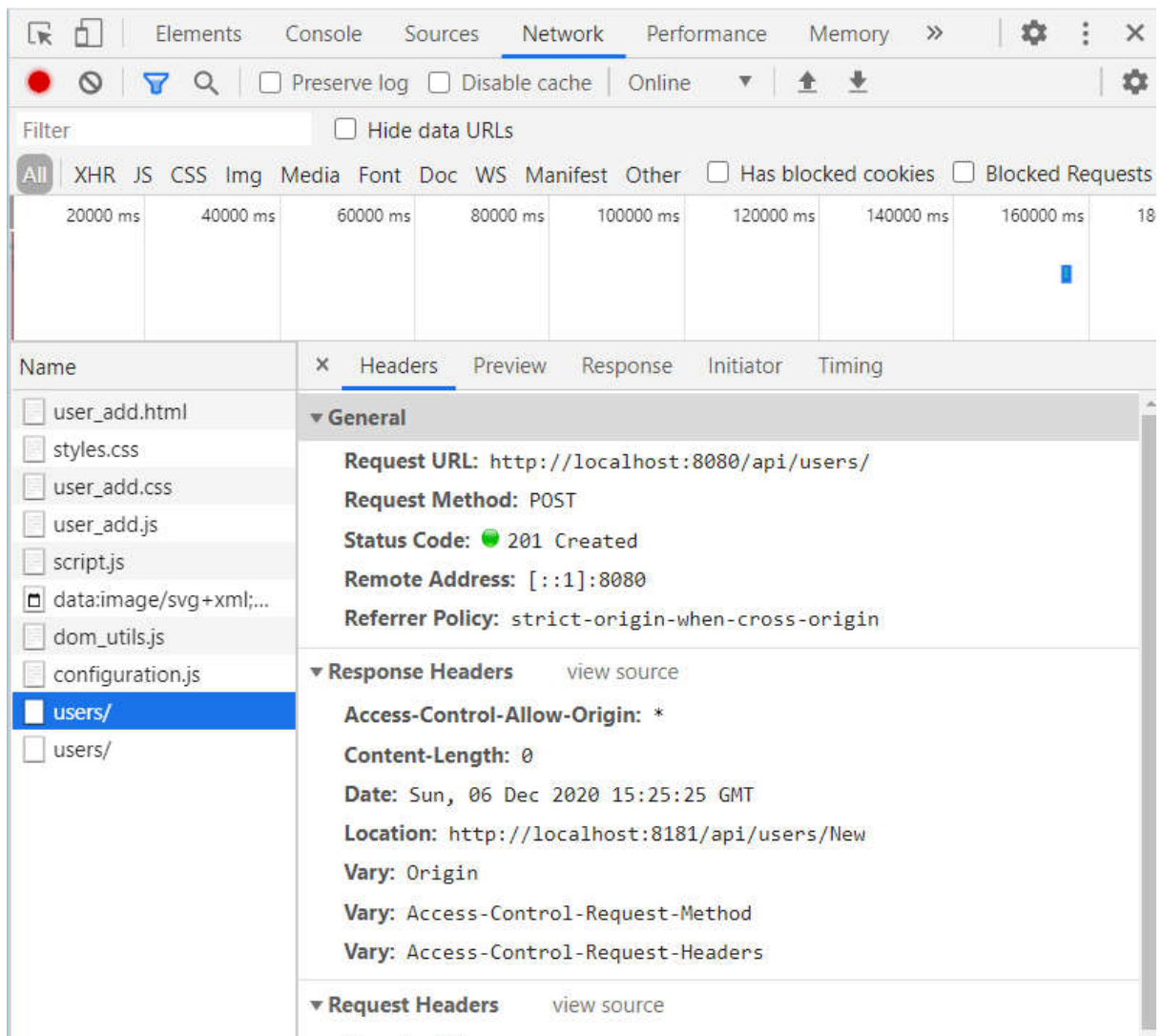
And before we move on, I will show you how the network panel looks in the developer tool.





Now we click the Create button. The things change here.





So far so good, I will show you our new category in the category list.



And the code for implementing this:

We are doing this by form.

```

27
28 <section class="main-section">
29   <article class="text--justified">
30     <header><h2>Creating new user</h2></header>
31     <form class="teacher__form" id="newUserForm">
32       <label for="login">Login:</label>
33       <input class="ui-control ui-input" id="login" name="login" type="text"/>
34       <label for="name">Name:</label>
35       <input class="ui-control ui-input" id="name" name="name" type="text"/>
36       <label for="surname">Surname:</label>
37       <input class="ui-control ui-input" id="surname" name="surname" type="text"/>
38       <label for="birthDate">BirthDate:</label>
39       <input class="ui-control ui-input" id="birthDate" name="birthDate" type="date"/>
40       <label for="password">Password:</label>
41       <input class="ui-control ui-input" id="password" name="password" type="password"/>
42       <label for="email">Email:</label>
43       <input class="ui-control ui-input" id="email" name="email" type="email"/>
44       <input class="ui-control ui-button" type="submit" value="Create"/>
45     </form>
46   </article>
47 </section>
48

```

And in JS, we add this form to event listener:

```

3
4
5 window.addEventListener('load', () => {
6   const newUserForm = document.getElementById('newUserForm');
7
8   newUserForm.addEventListener('submit', event => createUser(event));
9
10
11 });
12

```

Once the user click Create, which type equals to 'submit',

```

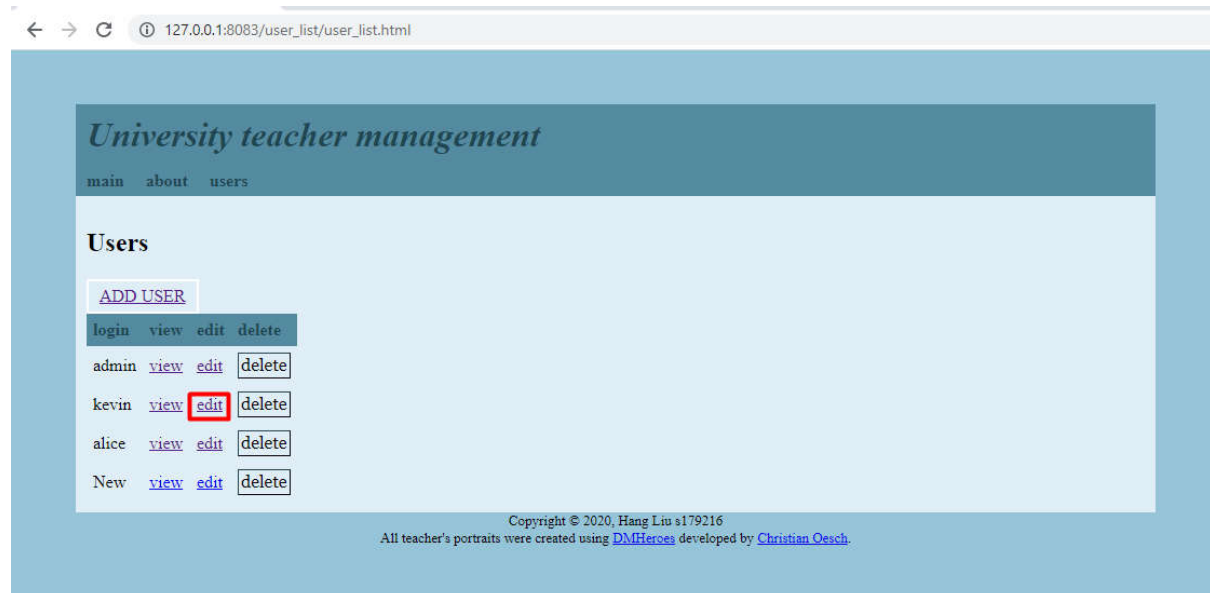
10
11 });
12
13 function createUser(event) {
14   event.preventDefault();
15
16   const xhttp = new XMLHttpRequest();
17
18   xhttp.open("POST", getBackendUrl() + '/api/users/', true);
19
20   const request = {
21     'login': document.getElementById('login').value,
22     'name': document.getElementById('name').value,
23     'surname': document.getElementById('surname').value,
24     'birthDate': document.getElementById('birthDate').value,
25     'password': document.getElementById('password').value,
26     'email': parseInt(document.getElementById('email').value)
27   };
28
29   xhttp.setRequestHeader('Content-Type', 'application/json');
30
31   xhttp.send(JSON.stringify(request));
32

```

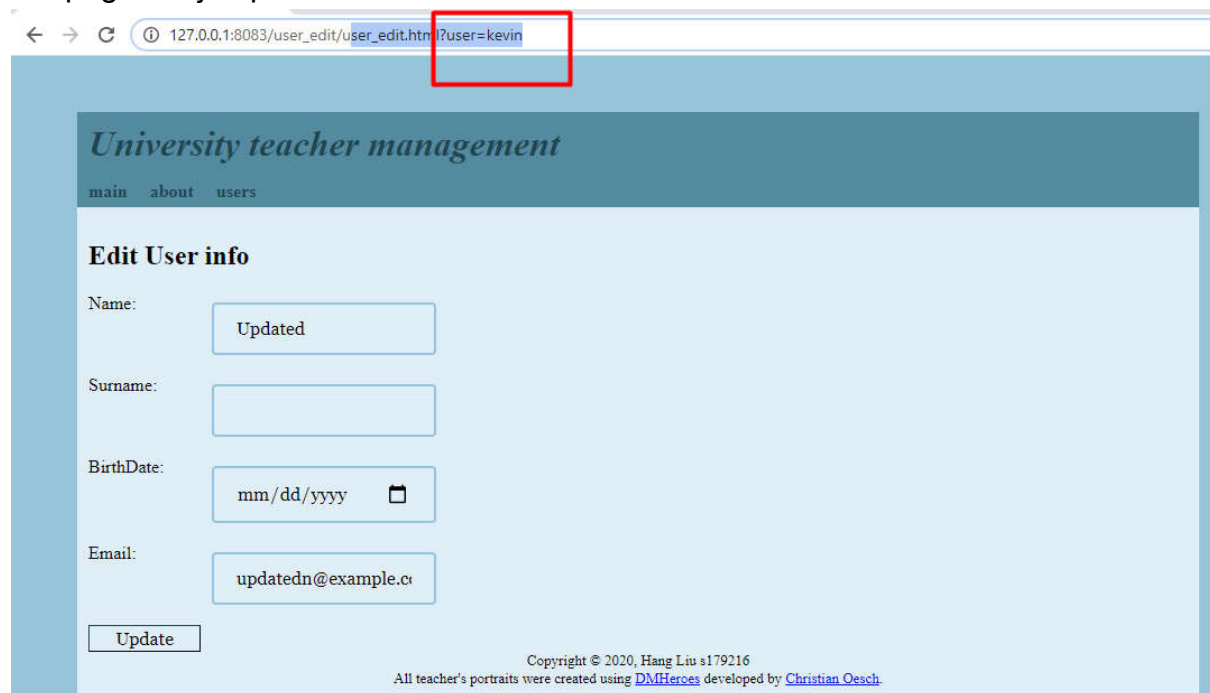
The createUser function will be invoked.

3. View for editing existing category (pre-populated form). View should be accessible from categories list view. View form should be populated with original values. Category should be determined by the query parameter. (1 point)

Once I click edit,



the page will jump to



And one more example

This shows you how query parameters determine the different categories. And also in the view of editing existing categories, you can see all forms are populated with original values.

Here is how original values displaying things work.

Firstly, in the event listener, I fetched the categories' values.

```
import {getCategories} from '../api/categories.js';

window.addEventListener('load', () => {
  const editUserForm = document.getElementById('editUserForm');

  editUserForm.addEventListener('submit', event => updateUserAction(event));

  fetchAndDisplayUser();
});

/**
 * Fetches currently logged user's name and updates edit form
```



```

/**
 * Fetches currently logged user's users and updates edit form.
 */
function fetchAndDisplayUser() {
  const xhttp = new XMLHttpRequest();
  xhttp.onreadystatechange = function () {
    if (this.readyState === 4 && this.status === 200) {
      let response = JSON.parse(this.responseText);
      for (const [key, value] of Object.entries(response)) {
        let input = document.getElementById(key);
        if (input) {
          input.value = value;
        }
      }
    }
  };
  xhttp.open("GET", getBackendUrl() + '/api/users/' + getParameterByName('user'), true);
  // xhttp.open("http://localhost:8181/api/users/kevin", true);
  xhttp.send();
}

function updateUserAction(event) {
  event.preventDefault();

```

And here is how the query parameter implements.  
If you remember I click edit on the page edit.html

```

      <th>delete</th>
    </tr>
  </thead>
  <tbody id="tableBody">
    <tr>
      <td></td>
      <td>name</td>
      <td><a href="#">view</a></td>
      <td><a href="#">edit</a></td>
      <td>
        <button class="ui-button ui-control">delete</button>
      </td>
    </tr>
  </tbody>
</table>

```

And the JS replaces this edit text data because of the event listener.

```

* @returns {HTMLTableRowElement}
*/
function createTableRow(user) {
  let tr = document.createElement('tr');
  tr.appendChild(createTextCell(user));
  tr.appendChild(createLinkCell('view', '../user_view/user_view.html?user=' + user));
  tr.appendChild(createLinkCell('edit', '../user_edit/user_edit.html?user=' + user));
  tr.appendChild(createButtonCell('delete', () => deleteUser(user)));
  return tr;
}

/**
 * Deletes entity from backend and reloads table.

```




**4. View with category details and elements list. View should present all details**

considering selected category and list of all elements from that category. It is enough to display single user-friendly value representing element. Category should be determined by the query parameter. View should allow for removing selected element. (1 point)

The screenshot shows a web application titled "University teacher management" with a navigation bar containing "main", "about", and "users". The main content area displays the details for a user named "kevin". Under the "Details" section, the following information is shown:

- Name: Updated
- Surname: null
- BirthDate: null
- Email: updatedn@example.com

Below the details is a section titled "Teachers" with an "ADD TEACHER" button. A table lists three teachers:

portrait	name	view	edit	delete
	Bob	<a href="#">view</a>	<a href="#">edit</a>	<button>delete</button>
	sadasd	<a href="#">view</a>	<a href="#">edit</a>	<button>delete</button>
	TeacherforKevin	<a href="#">view</a>	<a href="#">edit</a>	<button>delete</button>

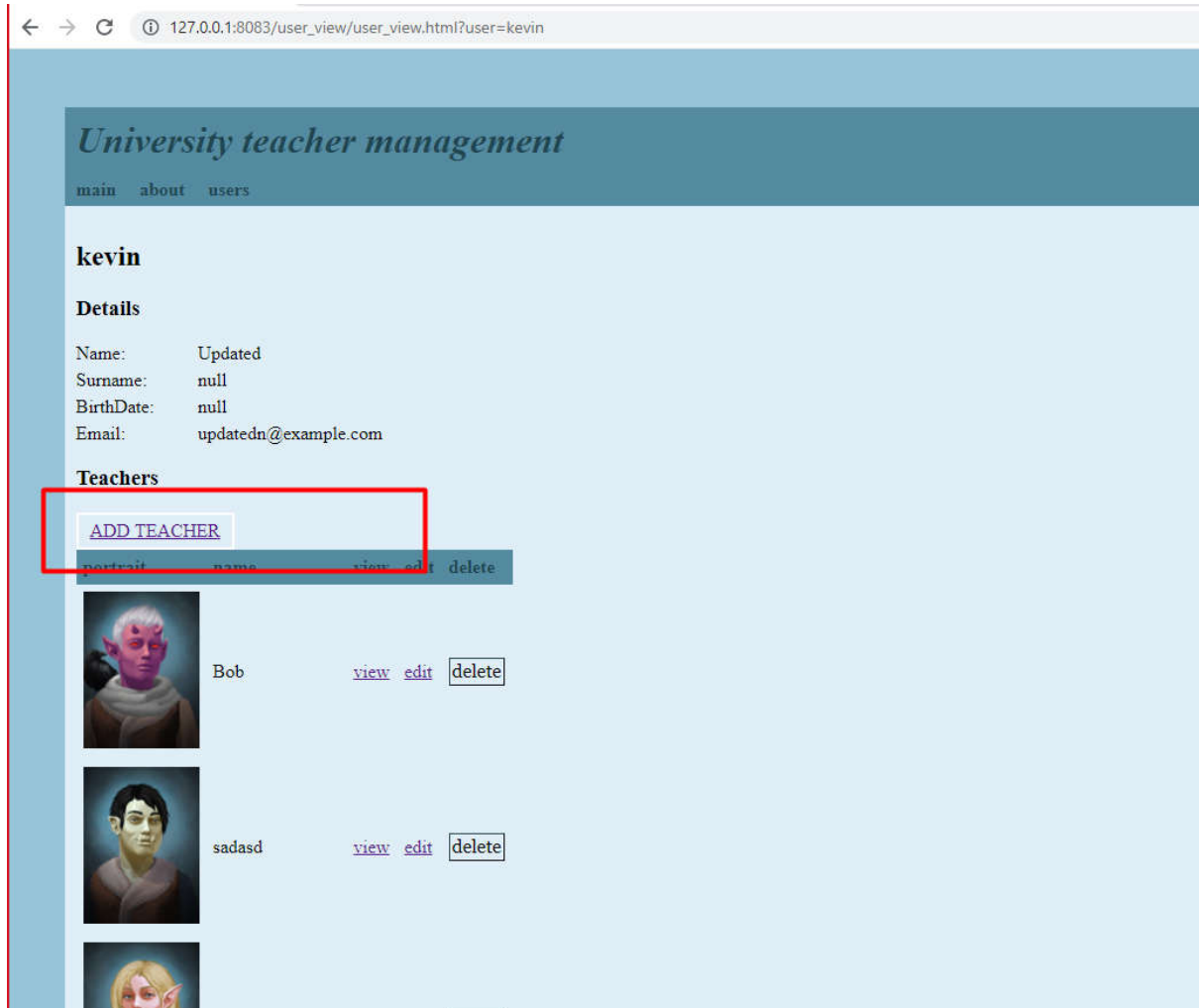
At the bottom of the page, there is a copyright notice: "Copyright © 2020, Hang Liu s179216. All teacher's portraits were created using DMHeroes developed by Christian Giesch."

The screenshot shows a web browser with the URL "127.0.0.1:8083/user\_view/user\_view.html?user=kevin" highlighted in a red box. Below the browser, a code editor displays the following JavaScript code:

```
function createTableRow(user) {  
  let tr = document.createElement('tr');  
  tr.appendChild(createTextCell(user));  
  tr.appendChild(createLinkCell('view', '../user_view/user_view.html?user=' + user));  
  tr.appendChild(createLinkCell('edit', '../user_edit/user_edit.html?user=' + user));  
  tr.appendChild(createButtonCell('delete', () => deleteUser(user)));  
  return tr;  
}
```



5. View for adding new element (form). View should be accessible from category details view. Category of the new element should be determined by the query parameter. (1 point)



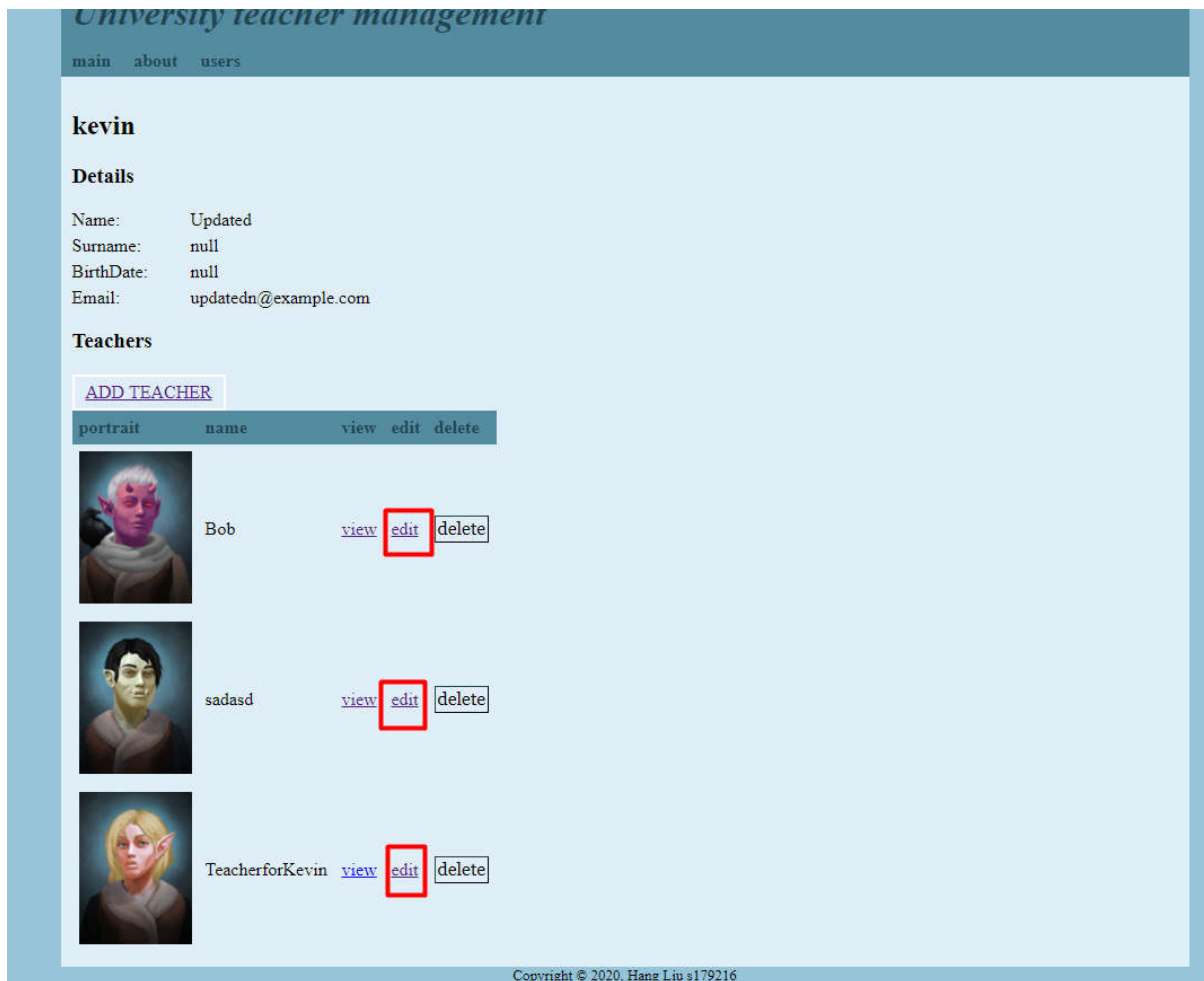
A new view for adding new element with well performed query parameter.

The image displays two screenshots of a web application titled "University teacher management". The application has a navigation bar with links for "main", "about", and "users". The main content area is titled "Creating new teacher" and contains a form with four input fields: "Name:", "Age:", "Background:", and "Direction:". Below the form is a "Create" button. The footer of the page includes copyright information: "Copyright © 2020, Hang Liu s179216" and "All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#)".

The top screenshot shows the browser address bar with the URL `127.0.0.1:8083/teacher_add/teacher_add.html?user=kevin`. The bottom screenshot shows the browser address bar with the URL `127.0.0.1:8083/teacher_add/teacher_add.html?user=alice`.

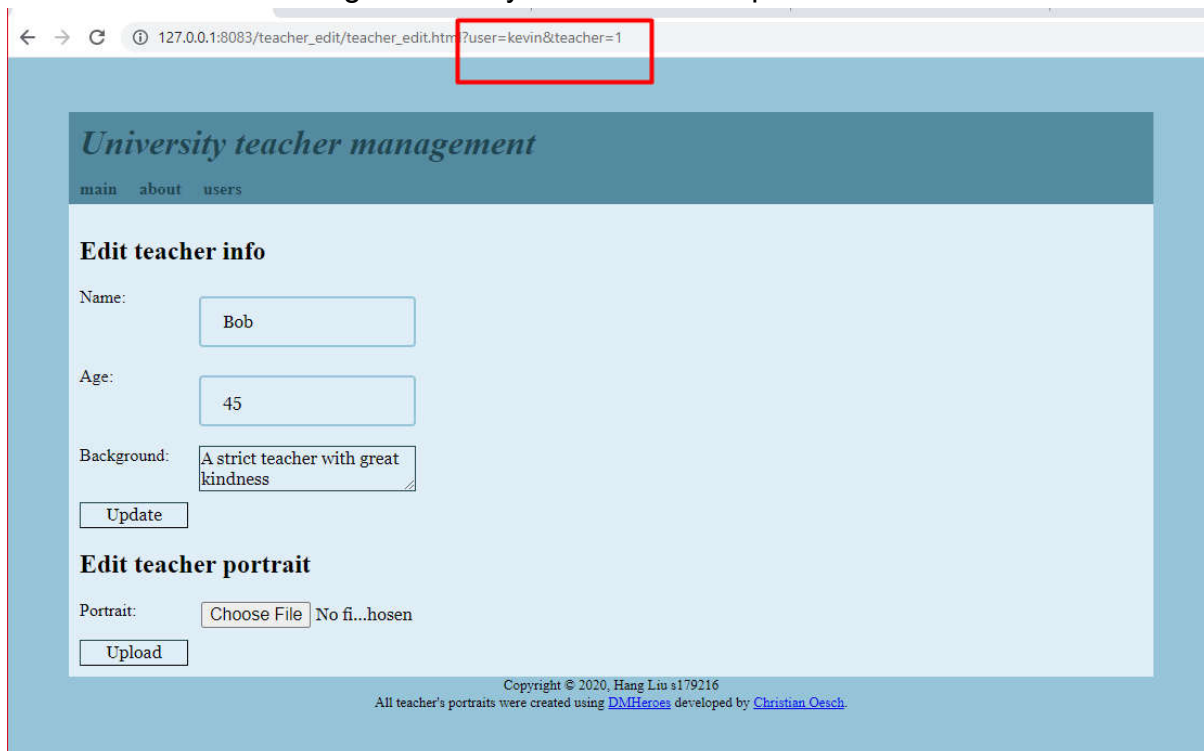
I also implemented this by form technique.





Copyright © 2020, Hang Liu s179216

And the view for editing existing element are all populated with original values. The mechanism for achieving this is very similar to checkpoint 3.



And you can find the codes:

```

import {getParameterByName} from '../js/dom_utils.js';
import {getBackendUrl} from '../js/configuration.js';

window.addEventListener('load', () => {
  const infoForm = document.getElementById('infoForm');
  const portraitForm = document.getElementById('portraitForm');

  infoForm.addEventListener('submit', event => updateInfoAction(event));
  portraitForm.addEventListener('submit', event => uploadPortraitAction(event));

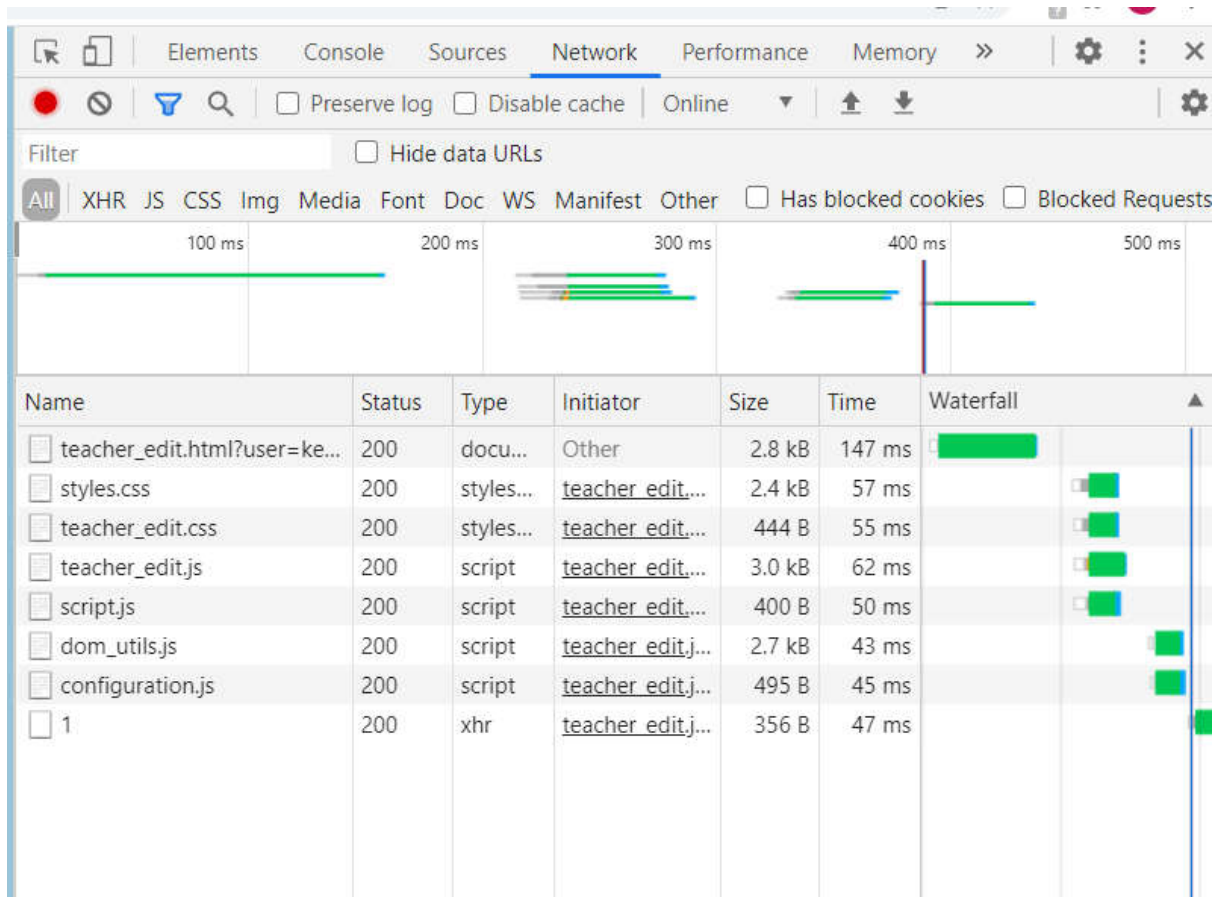
  fetchAndDisplayTeacher();
});

/**
 * Fetches currently logged user's teachers and updates edit form.
 */

14
15
16 /**
17  * Fetches currently logged user's teachers and updates edit form.
18  */
19 function fetchAndDisplayTeacher() {
20   const xhttp = new XMLHttpRequest();
21   xhttp.onreadystatechange = function () {
22     if (this.readyState === 4 && this.status === 200) {
23       let response = JSON.parse(this.responseText);
24       for (const [key, value] of Object.entries(response)) {
25         let input = document.getElementById(key);
26         if (input) {
27           input.value = value;
28         }
29       }
30     }
31   };
32   xhttp.open("GET", getBackendUrl() + '/api/users/' + getParameterByName('user') + '/teachers/'
33     + getParameterByName('teacher'), true);
34   xhttp.send();
35 }

```

Before we updating the new data, the developer tool shows like:



And after updating,

## University teacher management

main about users

### Edit teacher info

Name:

Age:

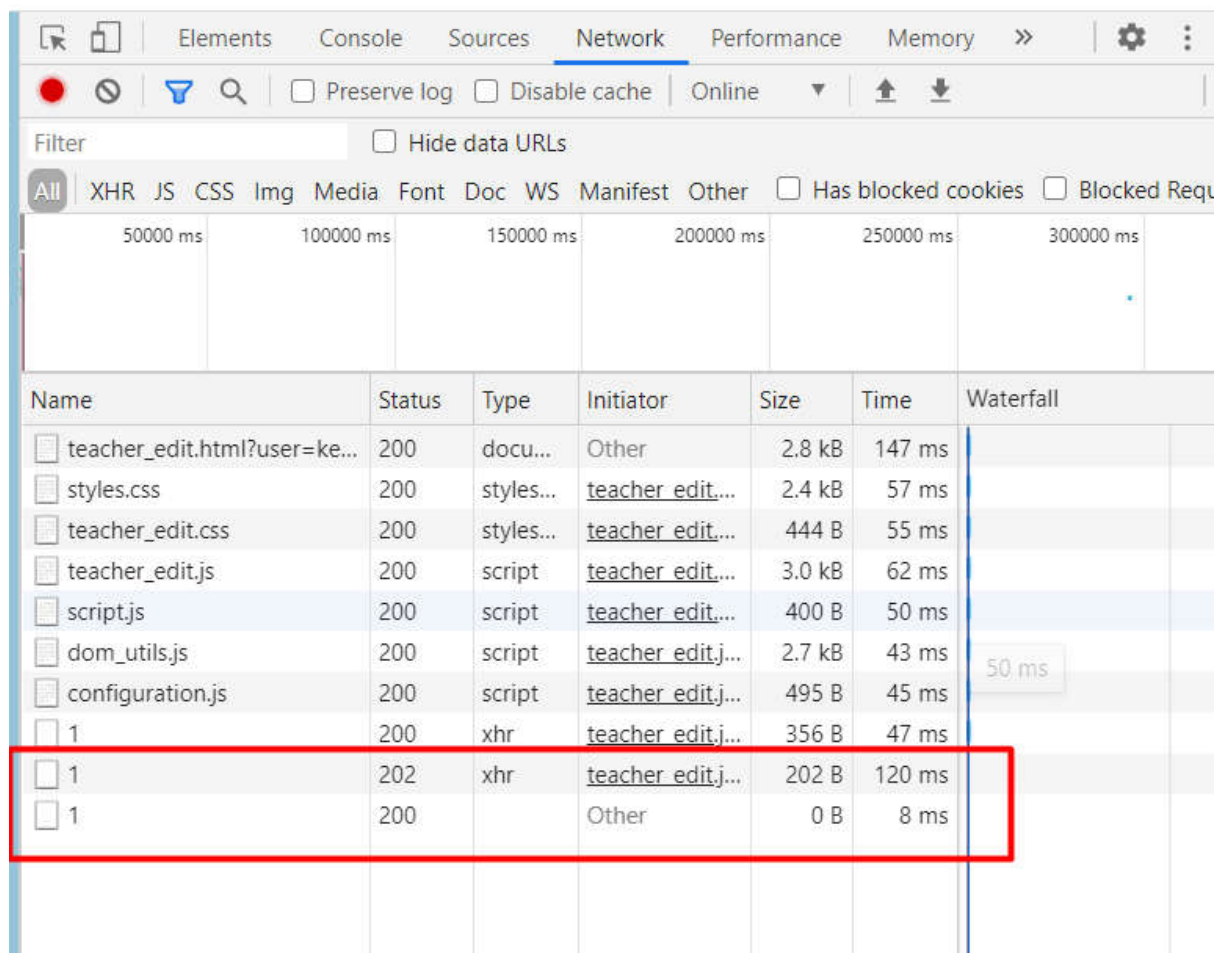
Background:

### Edit teacher portrait

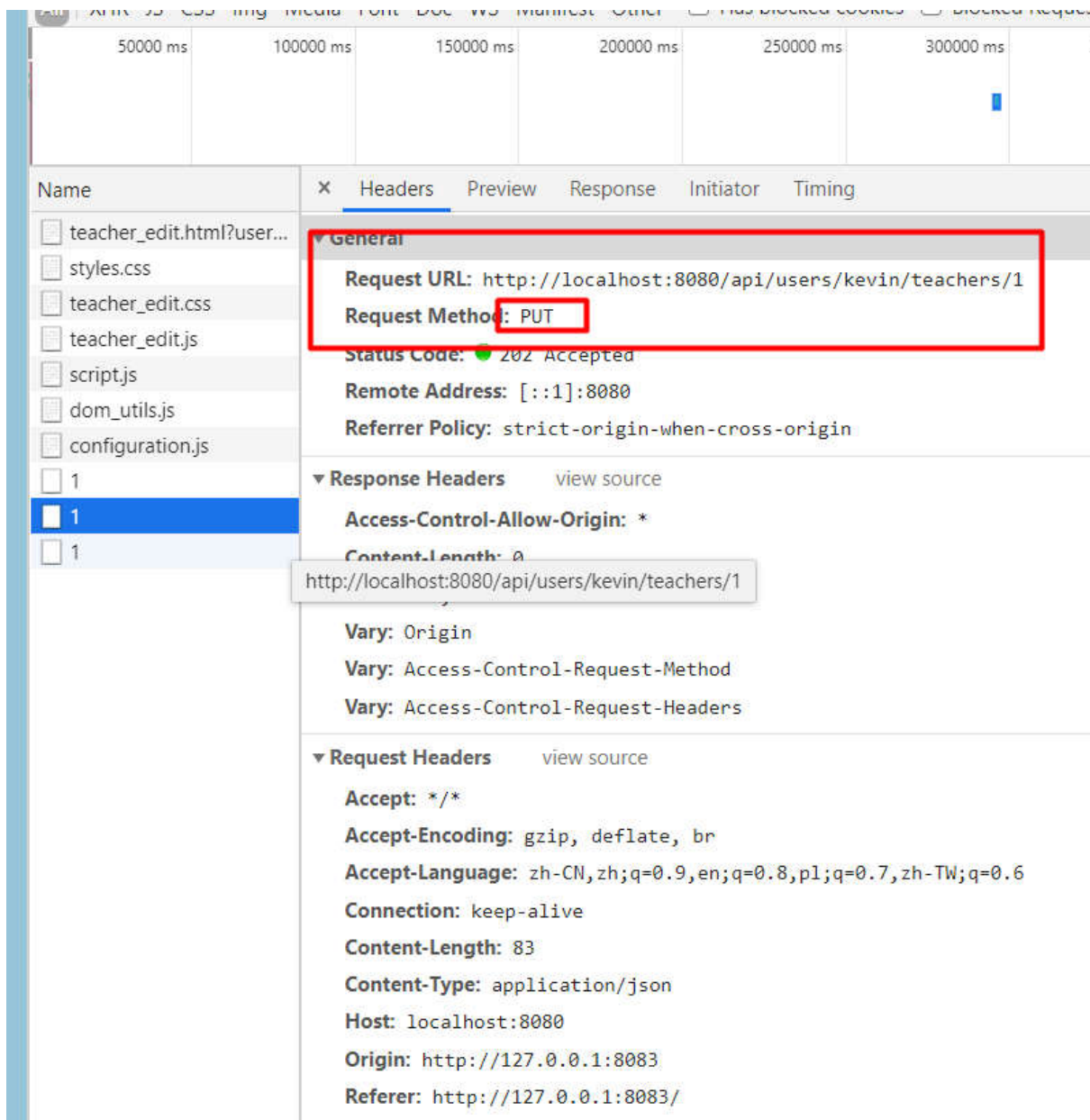
Portrait:  No fi...hosen

Copyright © 2020, Hang Liu s179216  
All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#).

The developer tool contents will have changes.







7. View with element details. View should present all details considering selected element. Category and element should be determined by the query parameters. (1 point)

## University teacher management

[main](#) [about](#) [users](#)

### alice

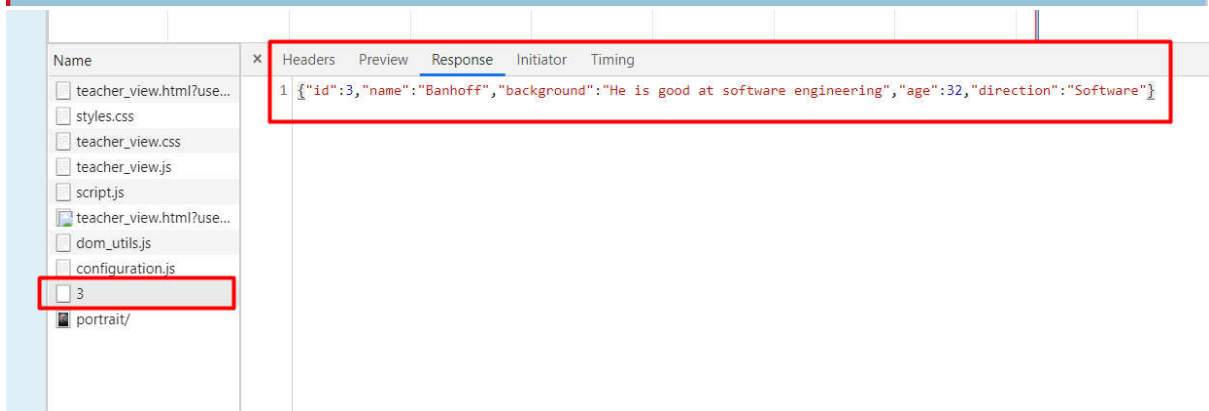
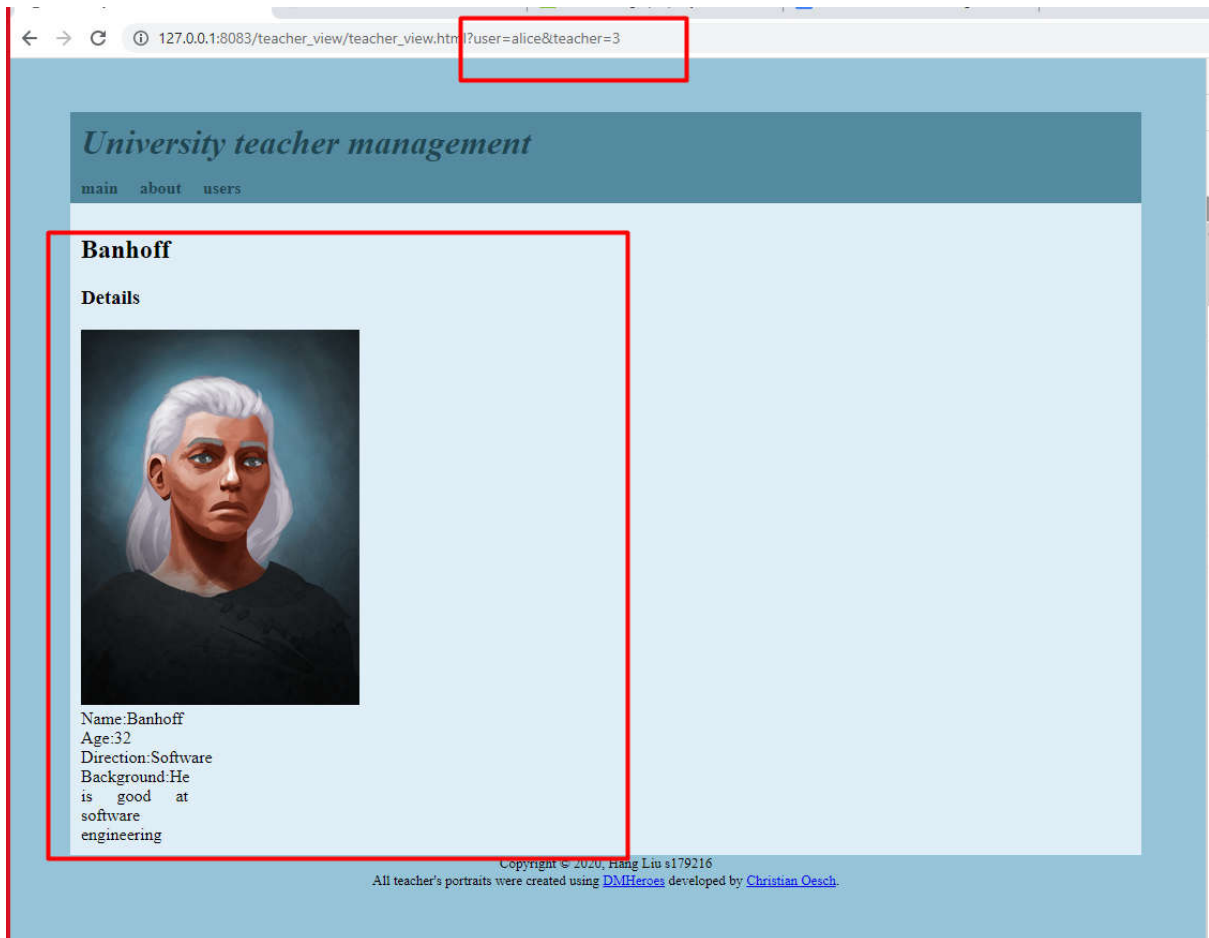
#### Details

Name: ne\_Alice  
Surname: Grape  
BirthDate: 2002-03-19  
Email: alice@example.com

#### Teachers

[ADD TEACHER](#)

portrait	name	view	edit	delete
	Banhoff	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
	Sebastian	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>



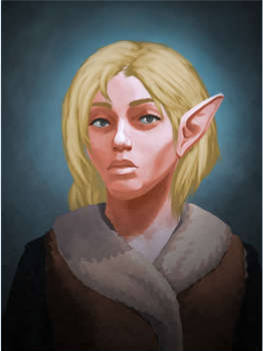
127.0.0.1:8083/teacher\_view/teacher\_view.html?user=kevin&teacher=5

# University teacher management

main about users

## TeacherforKevin

### Details



Name: TeacherforKevin  
Age: 50  
Direction: Electronics  
Background: A new guy

Copyright © 2020, Hang Liu s179216  
All teacher's portraits were created using [DMHeroes](#) developed by [Christian Oesch](#).

Blocked Requests

50 ms 100 ms 150 ms

Name

- teacher\_view.html?user=kevin&teache...
- styles.css
- teacher\_view.css
- teacher\_view.js
- script.js
- teacher\_view.html?user=kevin&teache...
- dom\_utils.js
- configuration.js
- 5
- portrait/

Name	Headers	Preview	Response	Initiator	Timing
teacher_view.html?use...	1		{ "id": 5, "name": "TeacherforKevin", "background": "A new guy", "age": 50, "direction": "Electronics" }		
styles.css					
teacher_view.css					
teacher_view.js					
script.js					
teacher_view.html?use...		http://127.0.0.1:8083/teacher_view/teacher_view.js			
dom_utils.js					
configuration.js					
5					
portrait/					

We can also delete element, for instance we will delete element sadasd from the category kevin.

## University teacher management

[main](#) [about](#) [users](#)

### Kevin

#### Details

Name: Updated  
Username: null  
BirthDate: null  
Email: updatedn@example.com

#### Teachers

[ADD TEACHER](#)

portrait	name	view	edit	delete
	updated_Bob	<a href="#">view</a>	<a href="#">edit</a>	<input type="button" value="delete"/>
	sadasd	<a href="#">view</a>	<a href="#">edit</a>	<input type="button" value="delete"/>
	TeacherforKevin	<a href="#">view</a>	<a href="#">edit</a>	<input type="button" value="delete"/>

## University teacher management

[main](#) [about](#) [users](#)

### kevin

#### Details

Name: Updated  
Surname: null  
BirthDate: null  
Email: updatedn@example.com

#### Teachers

[ADD TEACHER](#)

portrait	name	view	edit	delete
----------	------	------	------	--------



updated_Bob	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
-------------	----------------------	----------------------	------------------------



TeacherforKevin	<a href="#">view</a>	<a href="#">edit</a>	<a href="#">delete</a>
-----------------	----------------------	----------------------	------------------------

Network tab view showing a DELETE request to `http://localhost:8080/api/users/kevin/teachers/2`. The request method is `DELETE`, status is `202 Accepted`, and the response headers include `Access-Control-Allow-Origin: *`.

**Request Details:**

- Request URL:** `http://localhost:8080/api/users/kevin/teachers/2`
- Request Method:** `DELETE`
- Status Code:** `202 Accepted`
- Remote Address:** `[::1]:8080`
- Referrer Policy:** `strict-origin-when-cross-origin`

**Response Headers:**

- `Access-Control-Allow-Origin: *`
- `Content-Length: 0`
- `Date: Sun, 06 Dec 2020 16:07:42 GMT`
- `Vary: Origin`
- `Vary: Access-Control-Request-Method`
- `Vary: Access-Control-Request-Headers`

**Request Headers:**

- `Accept: */*`
- `Accept-Encoding: gzip, deflate, br`
- `Accept-Language: zh-CN,zh;q=0.9,en;q=0.8,pl;q=0.7,zh-TW;q=0.6`
- `Connection: keep-alive`
- `Host: localhost:8080`
- `Origin: http://127.0.0.1:8083`
- `Referer: http://127.0.0.1:8083/`
- `Sec-Fetch-Dest: empty`
- `Sec-Fetch-Mode: cors`