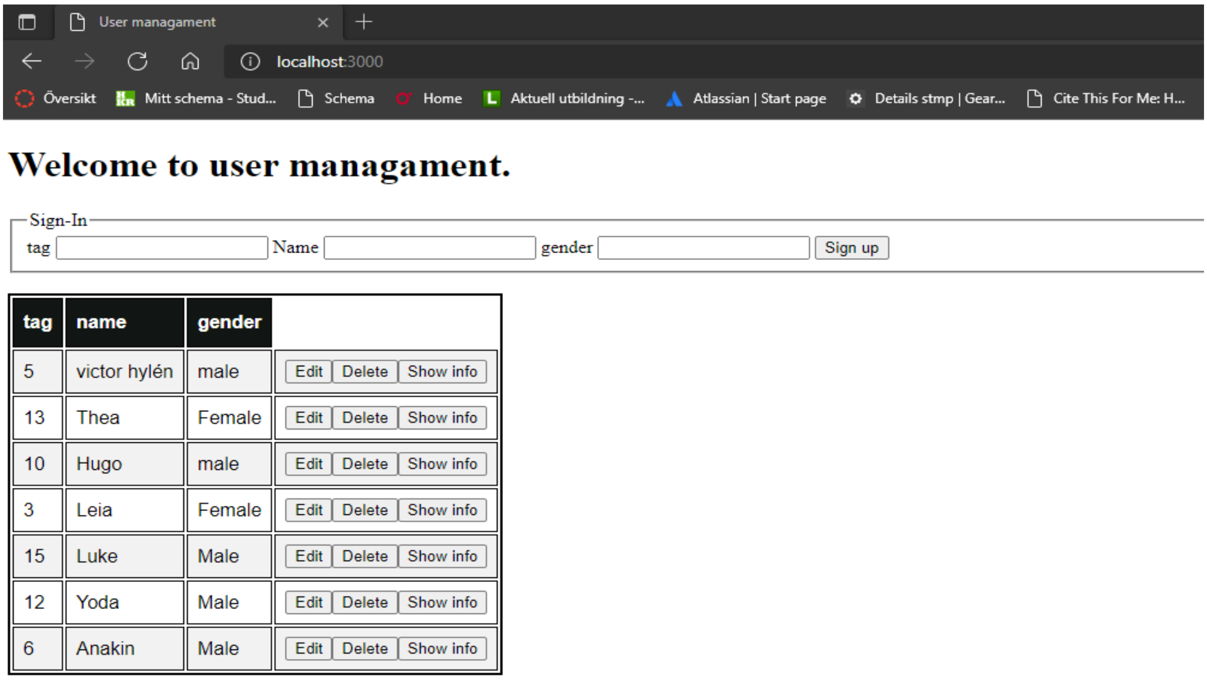
Lab 1

Task 1.

Here the goal was to create a Node.js server. When we started our server, we should see a home page called “Index.html”. The given URL were to be “http://localhost:3000/”. When we then started our server, this was the following homepage. How this was possible were threw Express framework. With Express can we for example define routing table, which we use when we want to perform different things when it depends on the http methods and all different URLS.



Welcome to user managament.

Sign-In

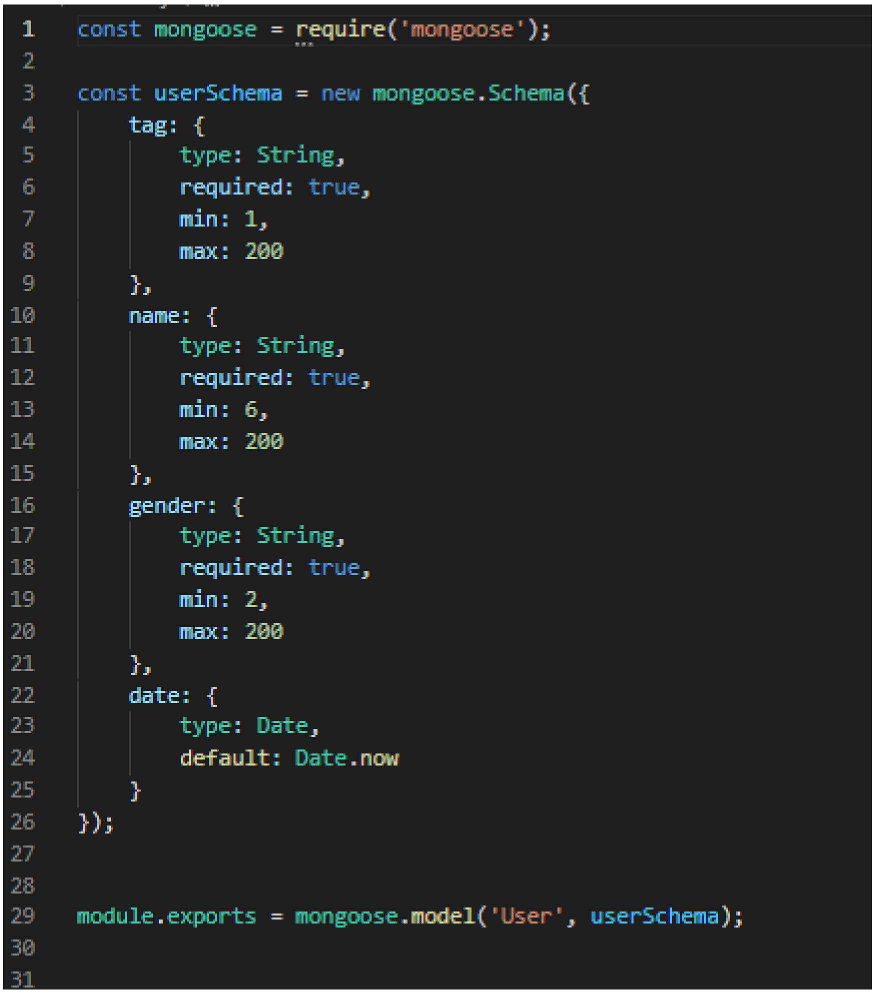
tag ______ Name ______ gender ______ Sign up

tag	name	gender	
5	victor hylén	male	Edit Delete Show info
13	Thea	Female	Edit Delete Show info
10	Hugo	male	Edit Delete Show info
3	Leia	Female	Edit Delete Show info
15	Luke	Male	Edit Delete Show info
12	Yoda	Male	Edit Delete Show info
6	Anakin	Male	Edit Delete Show info

Figure 1.

Task 2.

No, we wanted to create a database and add some data (users) to our database. In this work we used MongoDB. When we wanted to have some users, we had to make a table with some columns. We can see on figure 1 with some data in the table and following figure with the table rules,



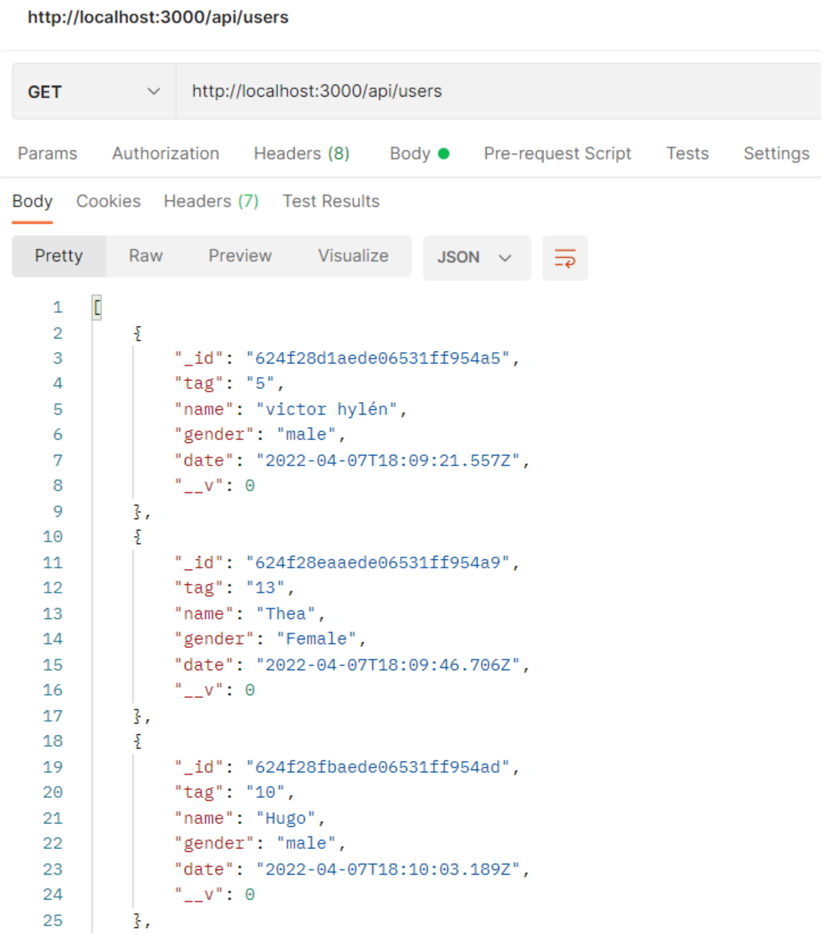
```
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
    tag: {
        type: String,
        required: true,
        min: 1,
        max: 200
    },
    name: {
        type: String,
        required: true,
        min: 6,
        max: 200
    },
    gender: {
        type: String,
        required: true,
        min: 2,
        max: 200
    },
    date: {
        type: Date,
        default: Date.now
    }
});


module.exports = mongoose.model('User', userSchema);
```

Figure 2.

Task 3.

Now we gas given to create rest API. The goal here was to use the http-methods like POST, PATCH, GET and DELETE.

i) We wanted to make it possible to show all our users in our database with following URL, "http://localhost:3000/api/users". The following HTTP methos was GET.



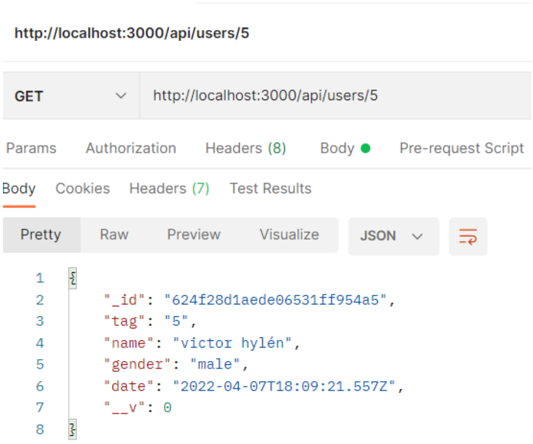
http://localhost:3000/api/users

GET http://localhost:3000/api/users

```
[
    {
        "_id": "624f28d1aede06531ff954a5",
        "tag": "5",
        "name": "victor hylén",
        "gender": "male",
        "date": "2022-04-07T18:09:21.557Z",
        "__v": 0
    },
    {
        "_id": "624f28eaaede06531ff954a9",
        "tag": "13",
        "name": "Thea",
        "gender": "Female",
        "date": "2022-04-07T18:09:46.706Z",
        "__v": 0
    },
    {
        "_id": "624f28fbaede06531ff954ad",
        "tag": "10",
        "name": "Hugo",
        "gender": "male",
        "date": "2022-04-07T18:10:03.189Z",
        "__v": 0
    },
```

Figure 3.

ii) Then we wanted to make it possible to show only one specific user. The URL had to be "http://localhost:3000/api/users/:id" with http method GET. For example, if we used tag (id) 5, It returned following,

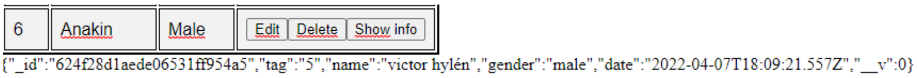


http://localhost:3000/api/users/5

GET http://localhost:3000/api/users/5

```
{
    "_id": "624f28d1aede06531ff954a5",
    "tag": "5",
    "name": "victor hylén",
    "gender": "male",
    "date": "2022-04-07T18:09:21.557Z",
    "__v": 0
}
```

Figure 4.

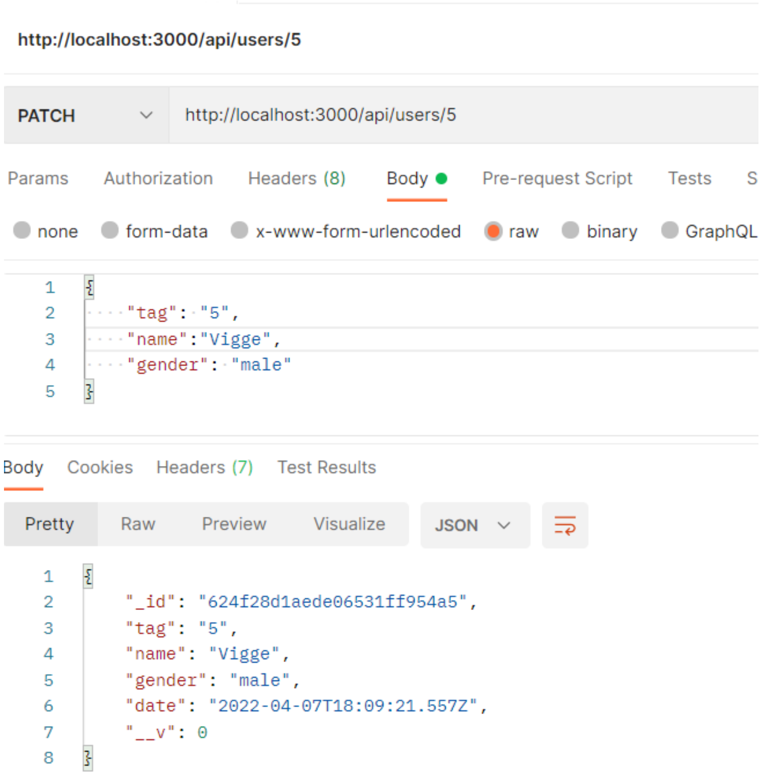


6	Anakin	Male	Edit Delete Show info

```
{"_id":"624f28d1aede06531ff954a5","tag":"5","name":"victor hylén","gender":"male","date":"2022-04-07T18:09:21.557Z","__v":0}
```

Figure 5.

iii) We were to be able to create new users as well with the HTTP-method PATCH. The following URL be "http://localhost:3000/api/users/:id". We look a figure 4 and then use PATCH, We will get following changes,



http://localhost:3000/api/users/5

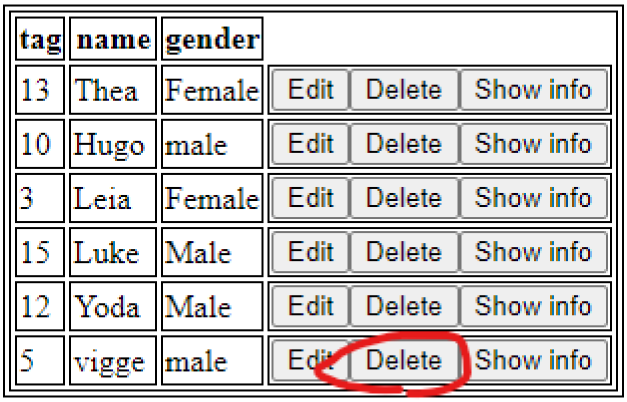
PATCH http://localhost:3000/api/users/5

```
{
    "tag": "5",
    "name":"Vigge",
    "gender": "male"
}
```

```
{
    "_id": "624f28d1aede06531ff954a5",
    "tag": "5",
    "name": "Vigge",
    "gender": "male",
    "date": "2022-04-07T18:09:21.557Z",
    "__v": 0
}
```

Figure 6.

iiii) Now we want to make it possible to delete a specific user. Here we use the URL "http://localhost:3000/api/users/:id". To delete a user, there was a button, which fetched the following http method and took the tag (id) as parameter,



tag	name	gender			
13	Thea	Female	Edit	Delete	Show info
10	Hugo	male	Edit	Delete	Show info
3	Leia	Female	Edit	Delete	Show info
15	Luke	Male	Edit	Delete	Show info
12	Yoda	Male	Edit	Delete	Show info
5	vigge	male	Edit	Delete	Show info

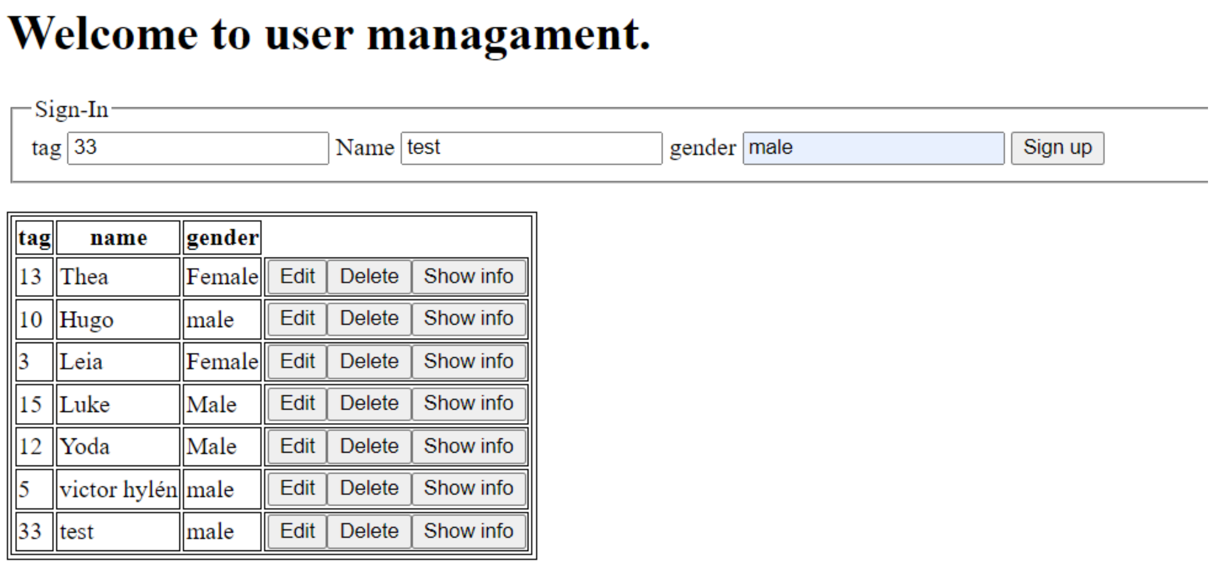
Figure 7.

Task 4.

1. Now we wanted to make it possible to show all users on one table. To do that we had to store all json object in variable, then push all our keys in to list to be able to create our table. But to create a table in HTML, we must have element "table", "tr" and "th". Then we visit all our keys that are stored in our list. Then we create a "th" for each value and "th" for every header. When we then have a table. We can create a cell for each entry of object and just leave the cell empty if not value is given. See figure 7.

2. Each row will have edit, delete, show info buttons. These button all fetch different http method and give different result. We have already presented about what each action will have in result.

3. We also wanted make it possible for the user to their own user account which submit button will fetch a http method, POST. Example,



Welcome to user managament.

Sign-In

tag 33 Name test gender male Sign up

tag	name	gender	
13	Thea	Female	Edit Delete Show info
10	Hugo	male	Edit Delete Show info
3	Leia	Female	Edit Delete Show info
15	Luke	Male	Edit Delete Show info
12	Yoda	Male	Edit Delete Show info
5	victor hylén	male	Edit Delete Show info
33	test	male	Edit Delete Show info

Figure 8.