

Project 5

In this assignment you are asked to implement a student management system using HTML, CSS and vanilla JS. You may use any of the CSS Frameworks in your project, but you are **NOT** allowed to use any JS frameworks.

Student Management System: The data and the backend

You are given a database of student records maintained as JSON objects (**data.json**). Here are the fields (schema) of the student table:

```
{
  "id": 1,
  "fname": "Isim1",
  "lname": "Soyisim1",
  "num": "152120171101",
  "dept": "2",
  "pob": "Eskişehir",
  "dob": "1999-03-23"
}
```

As you can see, each student record has a unique identifier (id: 1) which is assigned by the database server when the student is created. The other fields are the firstname (fname), the lastname (lname), student number (num), department (dept), place of birth (pob) and date of birth (dob). Notice that the date of birth is in the “YYYY-MM-DD” format. Here is the list of departments by number.

```
const depts = {
  "1": "Bilgisayar Müh.",
  "2": "Elektrik-Elektronik Müh.",
  "3": "Endüstri Müh.",
  "4": "İnşaat Müh."
};
```

You are given a collection of 25 students in a file named **data.jsonon**. To serve these student records from the backend, you need to use “**json-server**” as we discussed in class, which emulates a REST api for the student collection. You can then access this REST api using the “**fetch**” api of the browser.

Student Management System: The front-end

In this project you will be implementing the front-end of your project in HTML, CSS and **vanilla JS**. You are **NOT** allowed to use any JS frameworks. When your application starts, it needs to contact the json-server to fetch all student records and then dynamically create the following web page:

ESOGU Öğrenci Yönetim Sistemi

+

http://127.0.0.1:5501/index.html

Merhaba, John Doe

Çıkış Yap



Eskişehir Osmangazi Üniversitesi
Öğrenci Yönetim Sistemi

Öğrenci Listesi



İsim Soyisim	Öğrenci Numarası	Bölüm	Yetkiler
İsim1 Soyisim1	152120171101	Elektrik-Elektronik Müh.	Sil Düzenle Detay
İsim2 Soyisim2	152120171102	Bilgisayar Müh.	Sil Düzenle Detay
İsim3 Soyisim3	152120171103	Endüstri Müh.	Sil Düzenle Detay
İsim4 Soyisim4	152120171104	İnşaat Müh.	Sil Düzenle Detay
İsim5 Soyisim5	152120171105	Bilgisayar Müh.	Sil Düzenle Detay
İsim6 Soyisim6	152120171106	Elektrik-Elektronik Müh.	Sil Düzenle Detay
İsim7 Soyisim7	152120171107	Endüstri Müh.	Sil Düzenle Detay
İsim8 Soyisim8	152120171108	İnşaat Müh.	Sil Düzenle Detay

25 öğrenciden 1-8 arası gösteriliyor

1 2 3 4

5 8 10

As you can see, the page starts with some heading at the top and then a title (Öğrenci Listesi) and a button to add a new students (the blue button on the right with a person icon inside). The page then continues with a table of students (initially 8 students/page) as you see in the table. Notice that the table is striped; that is, alternative rows of the table has different background colors. At the bottom of the table, i.e., in the footer, we first give information about which students are currently being displayed and then you have some page controls: At the far right we have 3 radio buttons to change the number of students to display in each page. When the page loads, you need to display 8 students/page. The user can change this number to 5 or 10 by selecting the appropriate radio button. The page should then re-render itself appropriately. In the middle of the footer we show the buttons representing the page numbers. Because we have 25 students and each page displays 8 students, we currently have 4 pages: The first 3 pages will display 8 students and the last page will display only 1 student as shown below:

ESOGU Öğrenci Yönetim Sistemi x +

http://127.0.0.1:5501/index.html

Merhaba, John Doe

Çıkış Yap

 **Eskişehir Osmangazi Üniversitesi**
Öğrenci Yönetim Sistemi

Öğrenci Listesi

İsim Soyisim	Öğrenci Numarası	Bölüm	Yetkiler
İsim25 Soyisim25	152120171125	İnşaat Müh.	Sil Düzenle Detay

25 öğrenciden 25-25 arası gösteriliyor

1 2 3 4 5 6 10

Notice that the button representing the current page has a background of blue and a color of white, whereas the other buttons have a gray background and a black color.

Responsiveness

The information displayed in a row will be responsive. We have the following break-points:

Large	Screen-size ≥ 992 px
Medium	$768 \leq$ Screen-size < 992
Small	Screen-size < 768


As the screen gets smaller, you will start showing less information at a row. Here is how the screen should look like for medium and small screens respectively:

ESOGU Öğrenci Yönetim Sistemi x +

http://127.0.0.1:5501/index.html

Merhaba, John Doe

Çıkış Yap

 **Eskişehir Osmangazi Üniversitesi**
Öğrenci Yönetim Sistemi

Öğrenci Listesi

İsim Soyisim	Öğrenci Numarası	Yetkiler
İsim1 Soyisim1	152120171101	Sil Düzenle Detay
İsim2 Soyisim2	152120171102	Sil Düzenle Detay
İsim3 Soyisim3	152120171103	Sil Düzenle Detay
İsim4 Soyisim4	152120171104	Sil Düzenle Detay
İsim5 Soyisim5	152120171105	Sil Düzenle Detay
İsim6 Soyisim6	152120171106	Sil Düzenle Detay
İsim7 Soyisim7	152120171107	Sil Düzenle Detay
İsim8 Soyisim8	152120171108	Sil Düzenle Detay

25 öğrenciden 1-8 arası gösteriliyor


1 2 3 4 5 6 10

ESOGU Öğrenci Yönetim Sistemi x +

http://127.0.0.1:5501/index.html

Merhaba, John Doe

Çıkış Yap

 **Eskişehir Osmangazi Üniversitesi**
Öğrenci Yönetim Sistemi

Öğrenci Listesi

İsim Soyisim	Yetkiler
İsim1 Soyisim1	Sil Düzenle Detay
İsim2 Soyisim2	Sil Düzenle Detay
İsim3 Soyisim3	Sil Düzenle Detay
İsim4 Soyisim4	Sil Düzenle Detay
İsim5 Soyisim5	Sil Düzenle Detay
İsim6 Soyisim6	Sil Düzenle Detay
İsim7 Soyisim7	Sil Düzenle Detay
İsim8 Soyisim8	Sil Düzenle Detay

25 öğrenciden 1-8 arası gösteriliyor

1 2 3 4 5 6 10

As you can see, in medium screens we hide the department. In small screens, we hide both the department and the student number, and only show the student name and the controls to manipulate the student.

Adding a new Student

To add a new student, we press the blue button that has the person icon inside:



When we press this button, a modal opens up that shows a form where you must enter the required fields as shown below:

The screenshot shows a web application interface for Eskişehir Osmangazi Üniversitesi Öğrenci Yönetim Sistemi. A modal titled "Eklenicek Öğrenci Bilgileri" is open, displaying a form with the following fields:

- İsim: Cem
- Soyisim: Kurt
- Öğrenci Numarası: 152120171101
- Bölüm: Bölüm seçiniz (dropdown menu)
- Doğum Yeri: Eskişehir
- Doğum Tarihi: 11/04/2022

At the bottom of the modal are "Close" and "Add" buttons. The background shows a table of students with columns for İsim Soyisim, Öğrenci Numarası, Bölüm, and Yetkiler. The table is currently showing 8 students, and the modal is positioned over it.

We used Bootstrap modals, forms, and form-validation to implement the project. If you decide NOT to add a new student and cancel the operation, simply press the Close button or the ESC key in your keyboard. You can also press the Close icon 'x' on the top right corner. The modal will then disappear and no changes will be made. If however, you enter all the required data correctly and press the Add button, then the new student is added. At this point you must make a POST request to json-server, add the student, get its "id", add the new student to your local state and then re-render the page to reflect the changes.

Assuming that we were showing the last page when we add the new student, here is how the page must look like before and after we add the new student:

The left screenshot shows the application state before adding a new student. The modal is open, and the table shows 8 students. The right screenshot shows the application state after adding a new student. The modal is closed, and the table now shows 9 students, with the new student (Cem Kurt) added at the end of the list. The pagination shows 26 students in total, and the current page is 4.

As you can see, the new student is added to the end of the dataset and therefore gets displayed in the last page, which is where we are right now.

The modal must also be responsive: It will display as shown above for medium and large screens. But in small screens, each field must occupy 100% width of the modal. This means that the modal fields will all align vertically one after the other as shown below:

You must also implement form validation as we discussed in class. Specifically, the firstname, lastname and the place of birth must contain only letters (Turkish letters included) and each must have at least 3 chars. The student number must have 12 digits. The department must be selected from the departments menu. The date of birth must be entered using a datepicker. We used Bootstrap form validation to implement our project. The figure above on the right shows some error messages when the user enters invalid input. Your project must also have similar error messages when invalid input is entered.

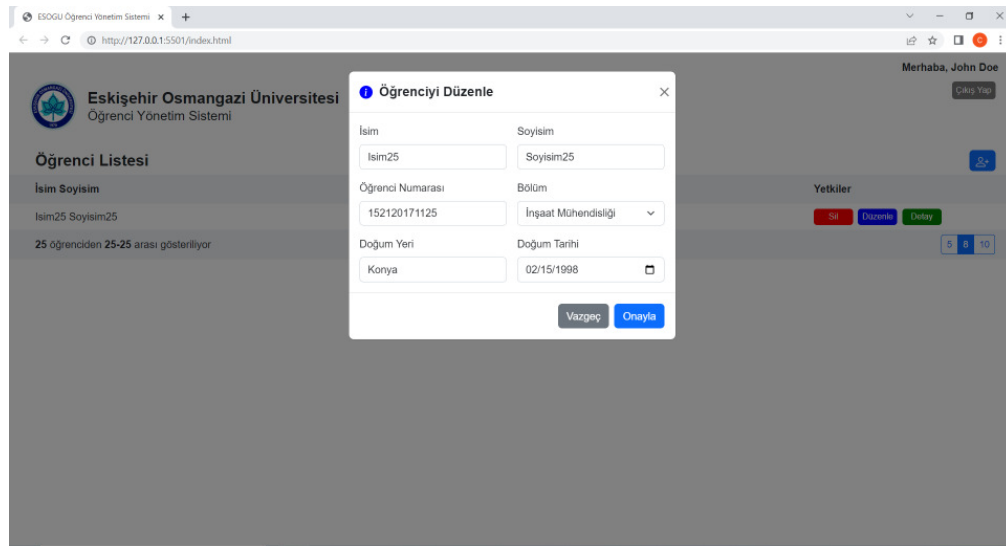
Seeing Student Details

As you have seen, we only display the student's name, number and dept. in each row. But there are other information we may want to see for a student. To do this, there are 3 controls associated with each student: Sil, Düzenle, Detay. To show all the details of a student, simply press "Detay" button for that student. You will display the details in a modal as shown below:

Notice that the details canNOT be edited. They are just displayed. Although not shown in the figure, this modal must also be responsive similar to the modal above, where we enter new information.

Editing Student Info

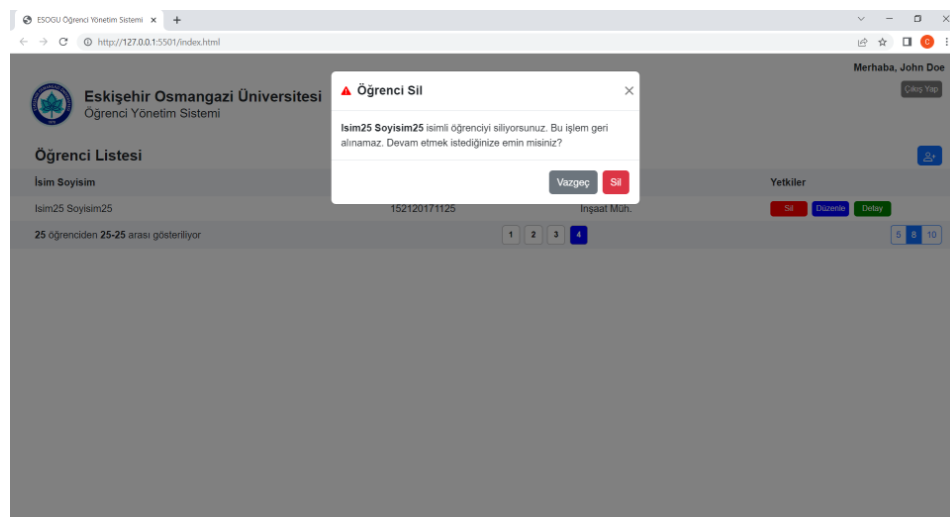
To edit the information pertaining to a student, press the "Düzenle" button for the student. A modal similar to the one shown below will appear, where you can edit the student's information. Here is how the page will look like:



In this modal, we should be able to change the information. This modal must also be responsive and also have form validation similar to the modal to add a new student. If the user wants to commit the changes to the database by pressing the “Onayla” button, you must send a PUT request to the json-server, get the response, update your local state, and then re-render the page with the new information.

Deleting a Student

To delete a student, press the “Sil” button of the student. A new dialog similar to the one shown below will appear:



If the user chooses to delete the student by pressing the “Sil” button, you must send a DELETE request to the json-server, get the response, delete the student from your local state, and then re-render the page. One thing to pay attention to is the following: Assume that you are in the last page, and that the last page contains only one student as is the case in the figure above. When the user deletes this student, there will be no more students to be shown in this page because we now have one less page. In this case, your app must NOT fail. It should continue showing the last available page full of students.

Implementation

This is NOT an easy project. It requires you to incorporate everything you learned so far in this class including HTML, CSS, CSS libraries, and JS. To simplify things, we used Bootstrap components: Buttons, Modals, responsiveness etc. all come from Bootstrap components. Furthermore, some icons shown in the modal come from Bootstrap icons:

<https://icons.getbootstrap.com/>. To use a Bootstrap icon, you can simply copy its “svg” and embed it in your HTML. This is how we implemented our project.

One important note: Make sure that you store “data.json” in a separate directory than where you store your project files. Then run the json-server for data.json. If you store data.json in the same directory where you store your project files, the following problem will occur: Because we are using the “live-server” VS code extension to start and serve our project, when a change is made to the json-server through POST, PUT or DELETE requests, the json-server changes data.json. Because a file in the current project directory has now changed, the live server reloads your app! This is definitely not something you want. So, make sure that data.json is stored in a separate directory.

Another note: data.json that we are giving you contains only 25 students. Do not make any assumptions as to how many students data.json will contain when your project starts. Obviously, when your project starts, it must contact the json-server and get all students by sending a GET request. When we test your projects, the initial number of students may be less or more. Your project must work with any data size.

Submission

Zip all project files and submit them through UZEM. Do not forget to include the group member names.