Community Healthcare Management System Interface

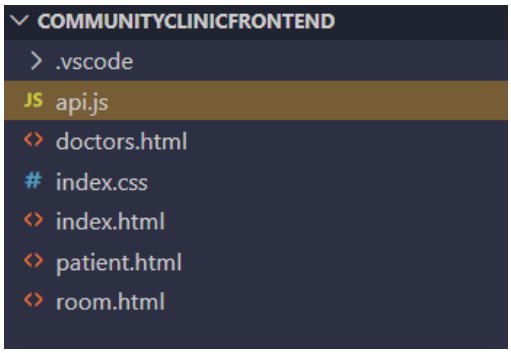
Irmak Kahya

Introduction:

The community healthcare management system interface aims to add, delete, and show clinics, doctors, patients, and rooms. The website has four HTML pages, one CSS file, and one JS file. This interface uses the Community Healthcare Management System API that I created in the first project. There is only one difference that I made to the API. I added CORS to each controller to ensure that the frontend and backend communicate safely.

```
@CrossOrigin(origins = "*")
```

The interface elements :



HTML pages:

1. index.html
2. doctors.html
3. patient.html
4. room.html

CSS file:

1. index.css

JS file:

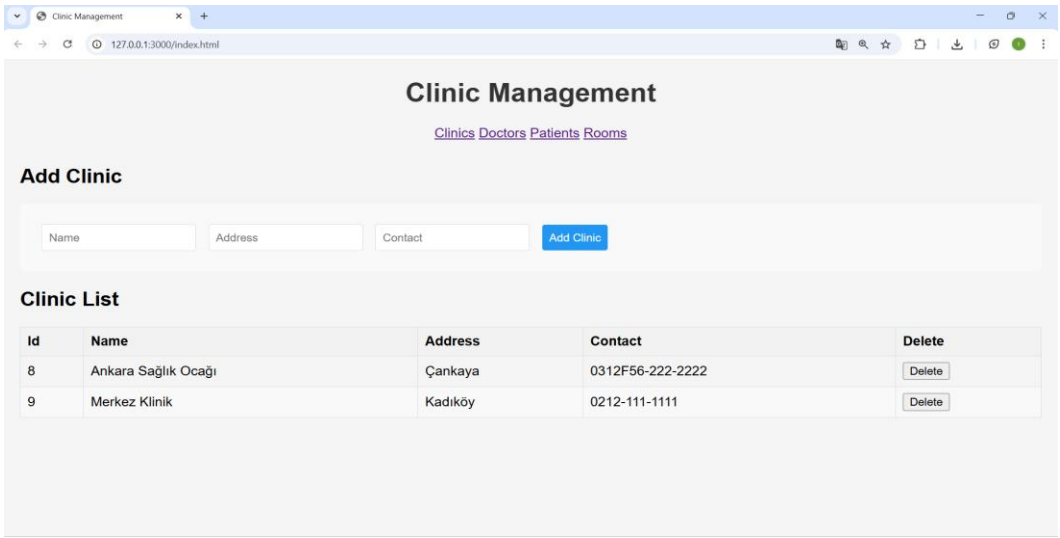
1. api.js

Content :

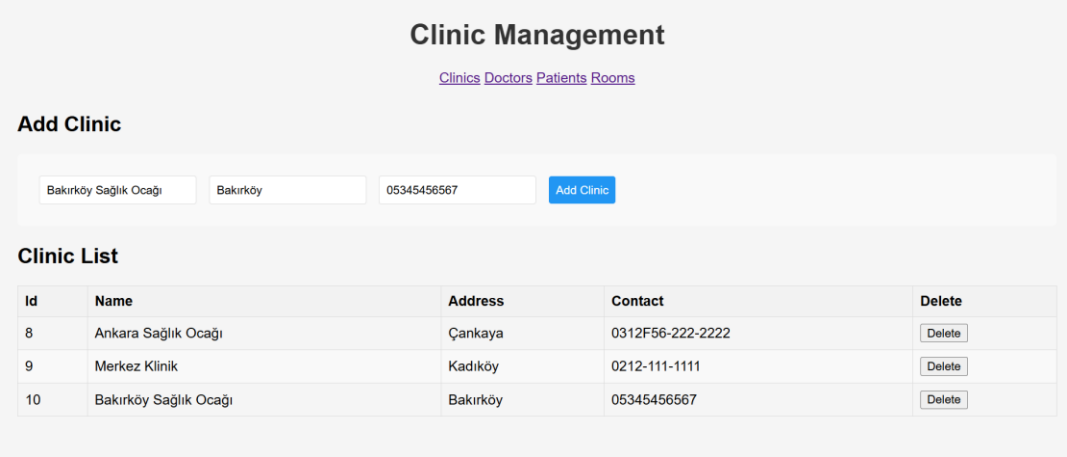
1. index.html:

The clinic management page (index.html) displays the clinic list on the table and allows us to add a new clinic to the system. On the table, each row has a delete button, which takes the id of the clinic to delete it from the system by sending a delete request to the right endpoint on the API. The page has a menu under the heading connecting html pages to each other.

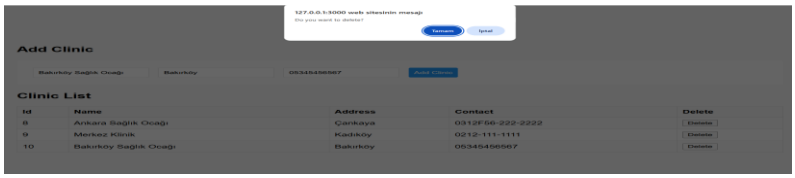
The page:



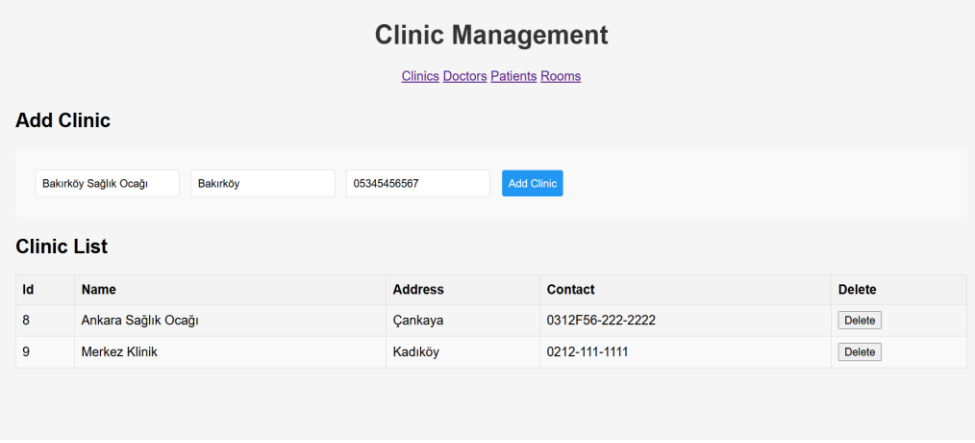
The new clinic is added :



Message to ensure whether a manager wants to delete the clinic:



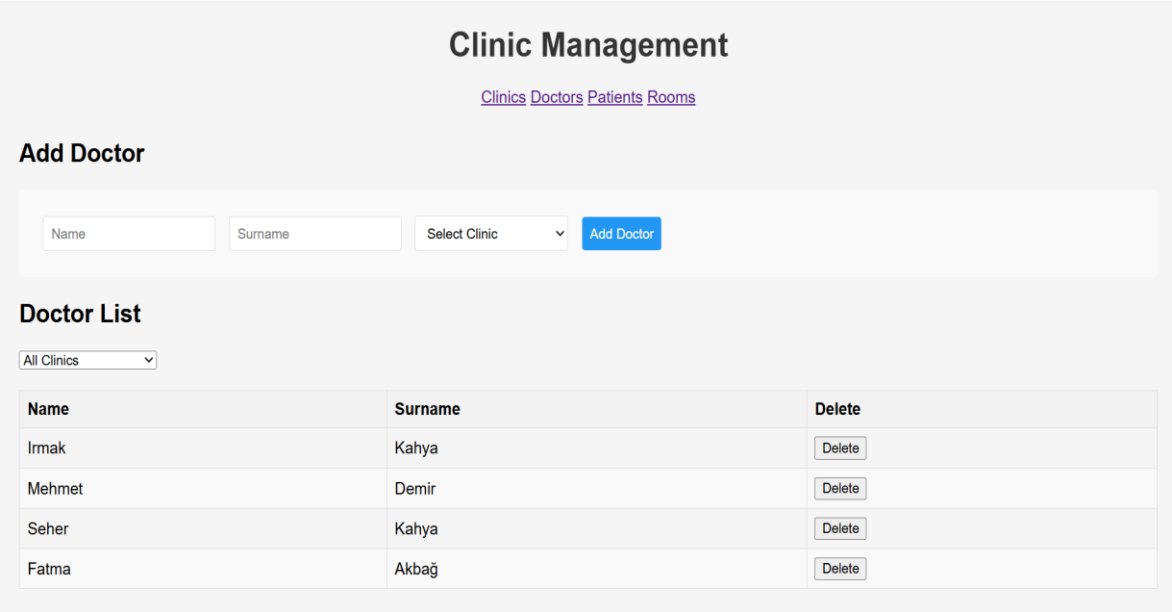
The table is updated after deleting the clinic :



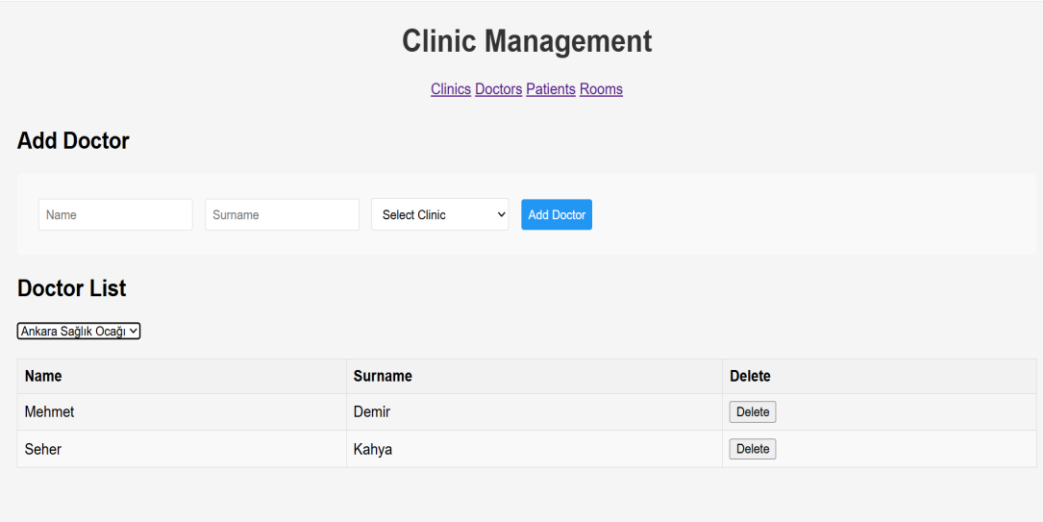
2.doctors.html :

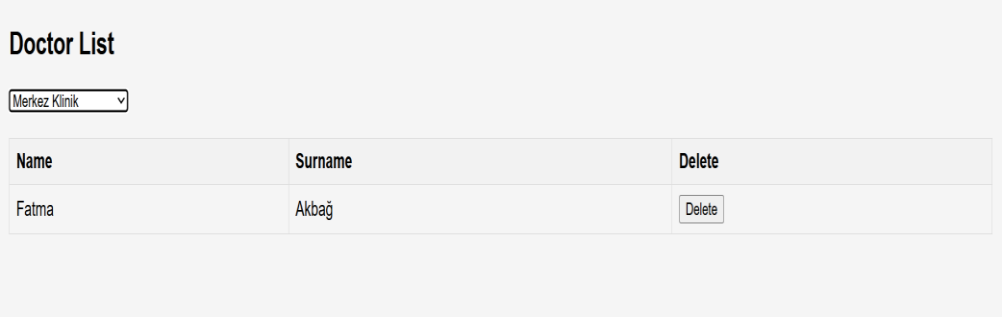
In this page (doctors.html), the interface enables us to add a doctor by choosing the clinic that the doctor will be assigned to. The other functionality of the interface is to display the doctors with the delete button on the table according to their clinic.

The page :

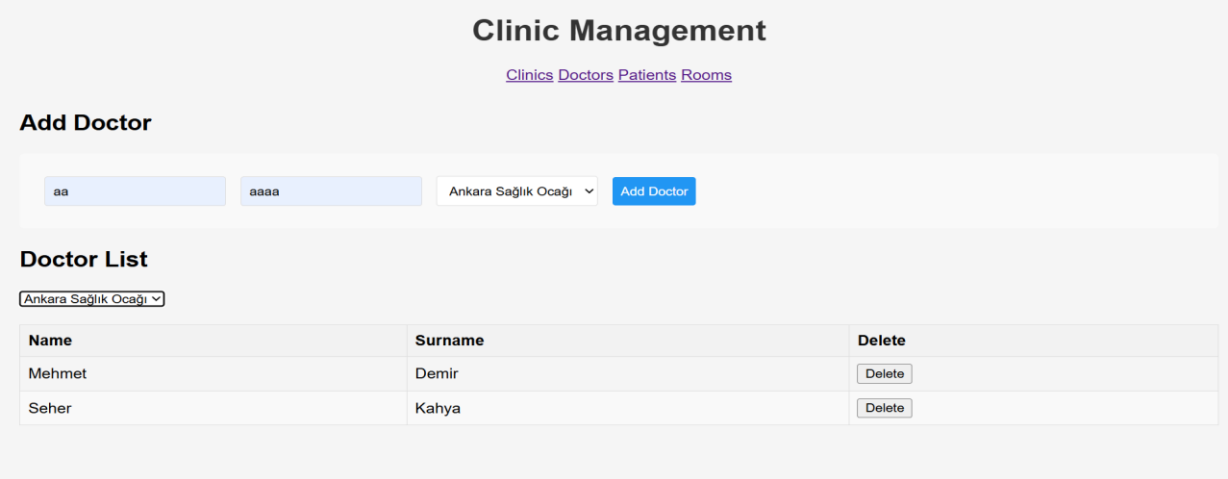


In this part, the combo box was selected for a specific community clinic to display only the doctors belonging to it.

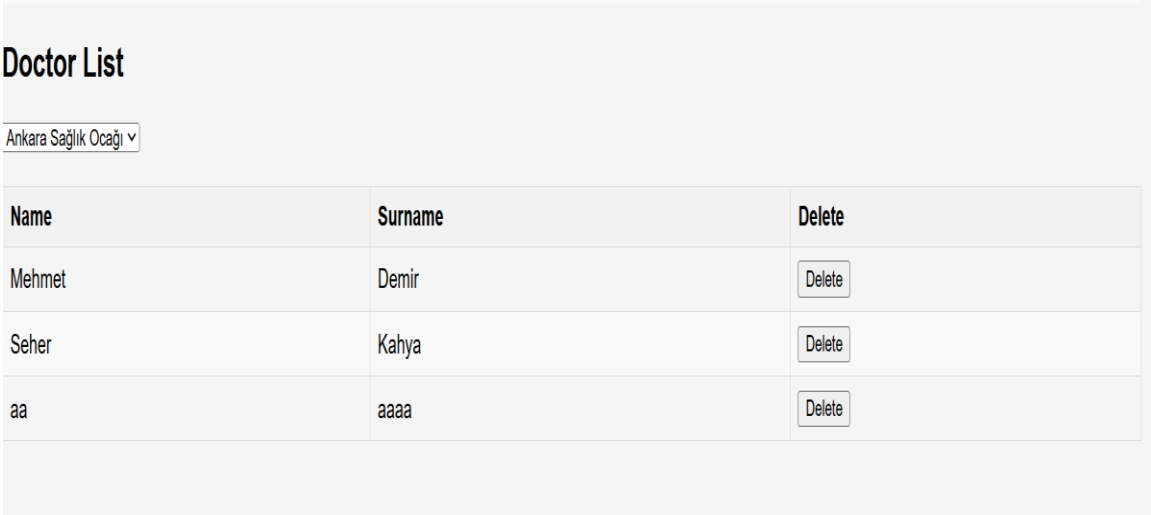




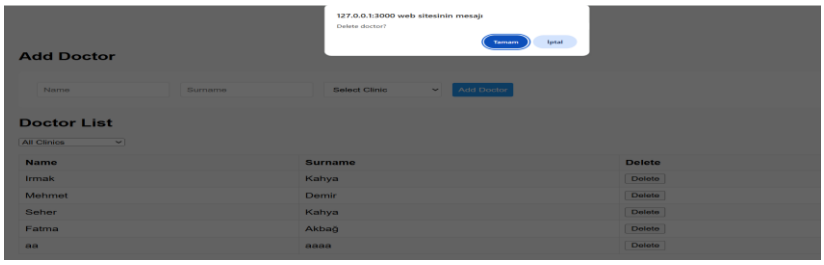
In this part, a new doctor is added :

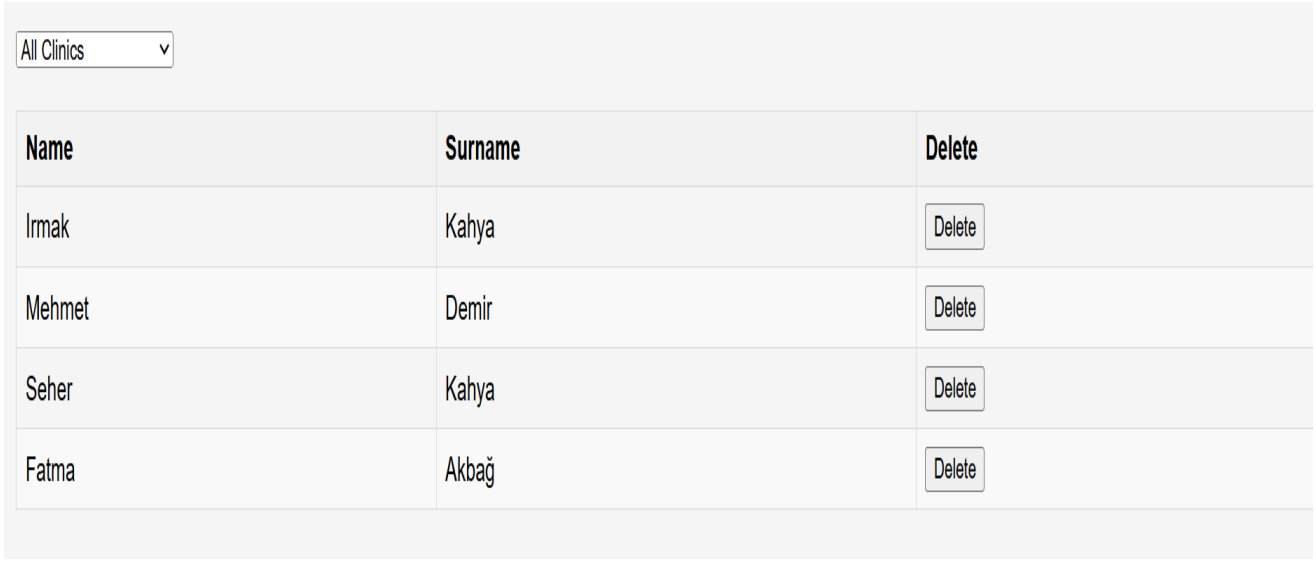


The new test data is added to the system in its clinic, and the table is updated as shown below :



In this part, the doctor will be deleted from our database by clicking the delete button provided, and the table will be updated:

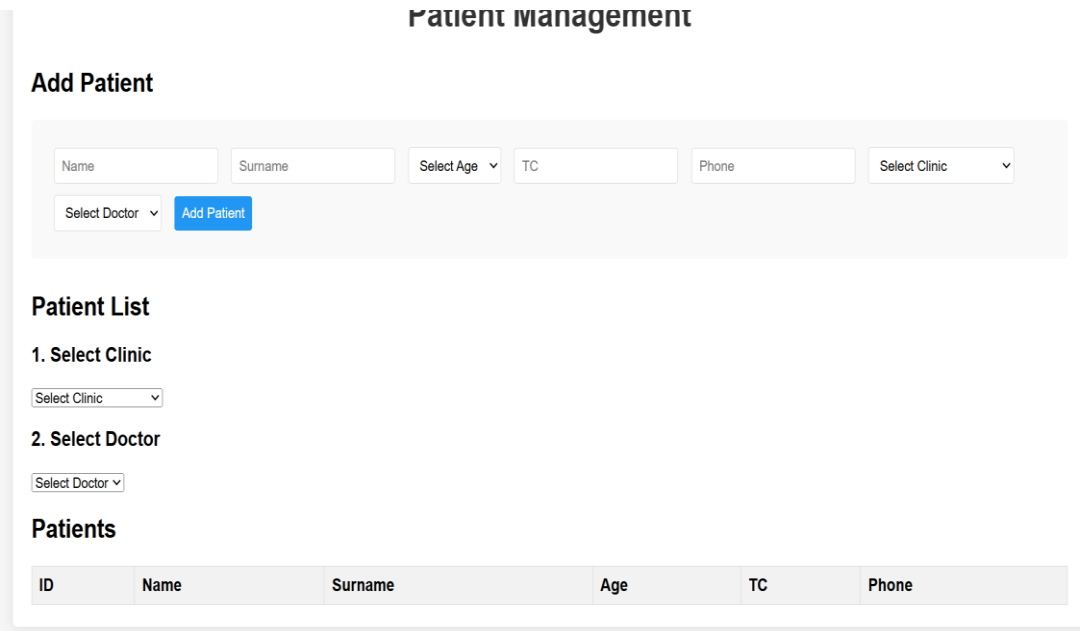


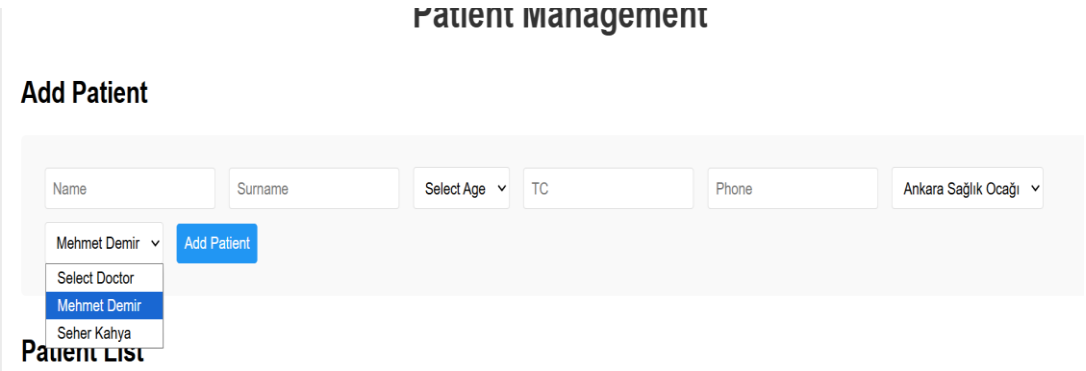


3.patient.html:

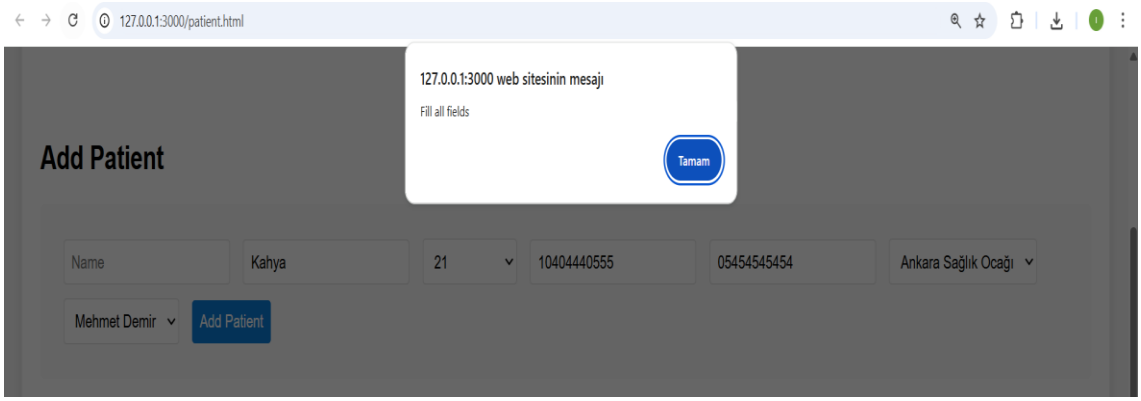
On this page, the patients are added by first choosing the clinic and then assigned to the doctor who is chosen from the clinic, and all the text fields are required to submit a patient. By the same principal, the system displays the patients.

The page :

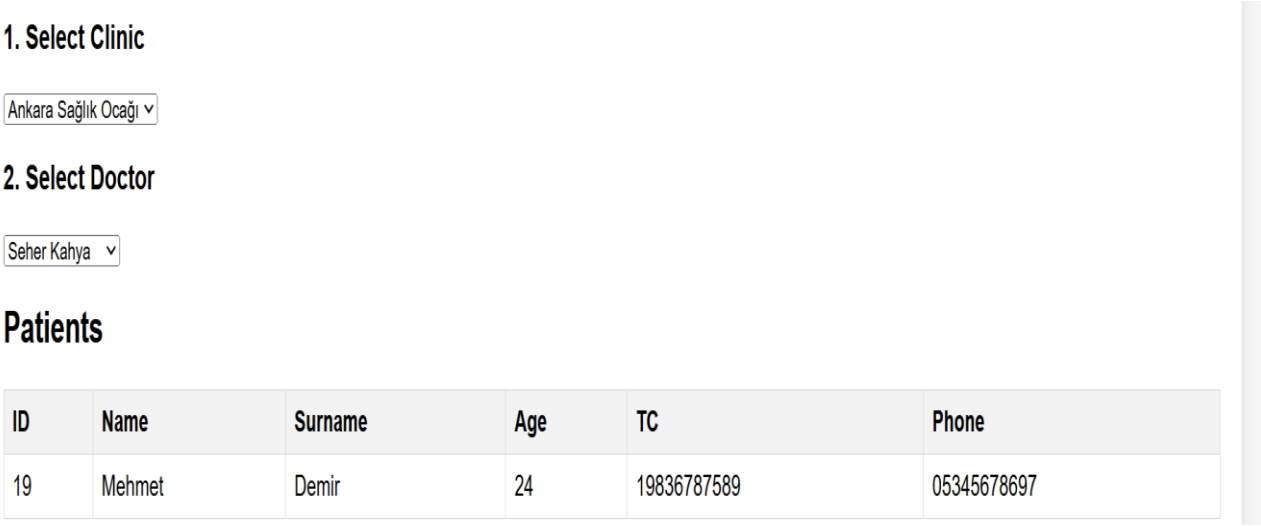

Patient Management
Add Patient
Name
Surname
Select Age
TC
Phone
Select Clinic
Select Doctor
Add Patient
Patient List
1. Select Clinic
Select Clinic
2. Select Doctor
Select Doctor
Patients
ID
Name
Surname
Age
TC
Phone


Patient Management
Add Patient
Name
Surname
Select Age
TC
Phone
Ankara Sağlık Ocağı
Mehmet Demir
Add Patient
Select Doctor
Mehmet Demir
Seher Kahya
Patient List

An error to warn the manager to fill all the fields :

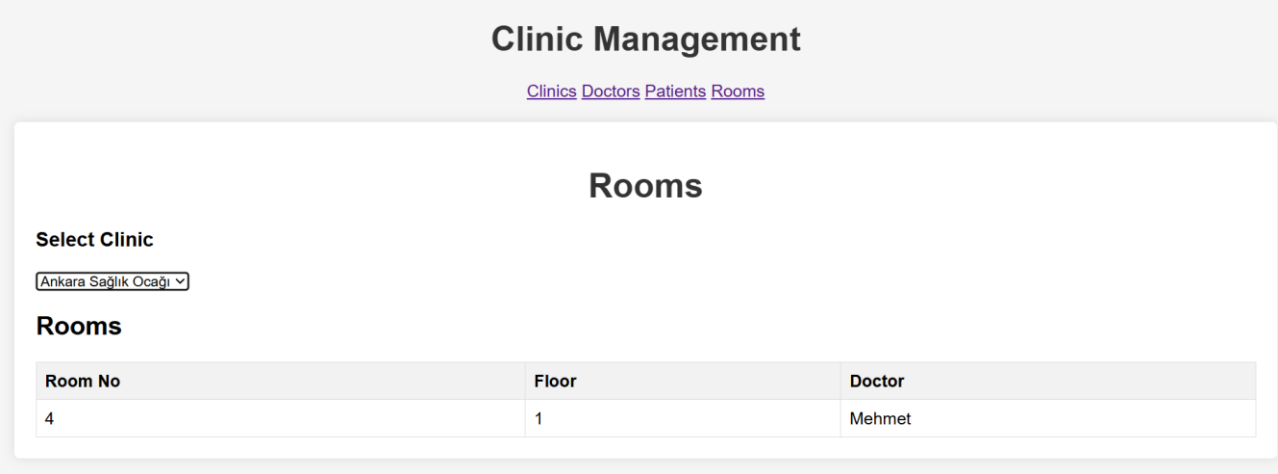


In here, the system displays the patients by their clinic and the doctor. First, we choose the clinic from the first combobox, then the second combobox displays the doctors on that clinic. When we choose the doctor, the table displays the patients with their id, name, surname, age, TC, and phone number.



4.room.html:

On the room.html, the interface displays the rooms and their doctors on the table according to the clinic that is selected.



The codes of each html ,css, and javascript :

1. index.html:

```
<!DOCTYPE html>
<html lang="en">
<html><head>
      <title>Community Health Care Management</title>
    <link rel="stylesheet" href="index.css">
</head> <body>
        <h1> Clinic Management</h1>
        <div class="menu">
            <a href="index.html">Clinics</a>
            <a href="doctors.html">Doctors</a>
            <a href="patient.html">Patients</a>
            <a href="room.html">Rooms</a></div>
        <div id="clinics" class="section">
            <h2>Add Clinic</h2>
            <div class="form">
                <input type="text" id="clinicName" placeholder="Name">
                <input type="text" id="clinicAddress" placeholder="Address">
                <input type="text" id="clinicContact" placeholder="Contact">
                <button onclick="addClinic()">Add Clinic</button> </div>
            <h2>Clinic List</h2>
              <table>
                <thead>
                    <tr>
                        <th>Id</th>
                        <th>Name</th>
                        <th>Address</th>
```

```
                        <th>Contact</th>
                        <th>Delete</th>
                    </tr>
                </thead>
                <tbody id="clinicList">
                </tbody>
            </table>
        </div>
    <script src="api.js"></script>
</body>
</html>
```

2.doctors.html:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Community Health Care Management</title>
    <link rel="stylesheet" href="index.css">
</head>
<body>
    <h1> Doctor Management</h1>
    <div class="menu">
        <a href="index.html">Clinics</a>
        <a href="doctors.html">Doctors</a>
        <a href="patient.html">Patients</a>
        <a href="room.html">Rooms</a>
    </div>
    <div id="doctors" class="section">
        <h2>Add Doctor</h2>
        <div class="form">
            <input type="text" id="docName" placeholder="Name">
            <input type="text" id="docSurname" placeholder="Surname">
            <select class="clinicSelect">
                <option value="">Select Clinic</option>
            </select>
            <button onclick="addDoctorToClinic()">Add Doctor</button>
        </div>
        <h2>Doctor List</h2>
        <div class="filter">
            <select class="clinicSelect">
                <option value="">All Clinics</option>
```

```
            </select>
        </div>
        <table>
            <thead>
                <tr>
                    <th>Name </th>
                    <th>Surname</th>
                    <th>Delete</th>
                </tr>
            </thead>
            <tbody id="doctorTable">
            </tbody>
        </table>
    </div>
    <script src="api.js"></script>
</body></html>
```

3.patient.html:

```
<!DOCTYPE html>
<html>
<head>
    <title>Community Health Care Management</title>
    <link rel="stylesheet" href="index.css">
</head>
<body>
    <h1> Patient Management</h1>
    <div class="menu">
        <a href="index.html">Clinics</a>
        <a href="doctors.html">Doctors</a>
        <a href="patient.html">Patients</a>
        <a href="room.html">Rooms</a>
    </div>
    <div class="container">
        <h1>Patient Management</h1>
        <div class="section">
            <h2>Add Patient</h2>
            <div class="form">
                <input type="text" id="patientName" placeholder="Name">
                <input type="text" id="patientSurname" placeholder="Surname">
                <select id="patientAge">
                    <option value="">Select Age</option>
                </select>
                <input type="text" id="patientTC" placeholder="TC">
               <input type="text" id="patientPhone" placeholder="Phone">
                <select class="clinicSelect">
                    <option value="">Select Clinic</option>
                </select>
```

```
                <select class="doctorSelect">
                    <option value="">Select Doctor</option>
                </select>
                <button onclick="addPatientToClinicAndDoctor()">Add
Patient</button>
            </div>
        </div>
        <div class="section">
            <h2>Patient List</h2>
            <div class="filter-box">
                <h3>1. Select Clinic</h3>
                <select class="clinicSelect">
                    <option value="">Select Clinic</option>
                </select>
            </div>
            <div class="filter-box" id="doctorFilter">
                <h3>2. Select Doctor</h3>
                <select class="doctorSelect" onchange="">
                    <option value="">Select Doctor</option>
                </select>
            </div>
            <div id="patientSection">
                <h2>Patients</h2>
                <div id="selectedDoctorInfo"></div>
                <table>
                    <thead>
                        <tr>
                            <th>ID</th>
                            <th>Name</th>
                            <th>Surname</th>
                            <th>Age</th>
                            <th>TC</th>
                            <th>Phone</th>
                        </tr>
                    </thead>
                    <tbody id="patientTable">
                    </tbody>
                </table>
            </div>
        </div>
        <script src="api.js"></script>
</body> </html>
```

4.room.html :

```
<!DOCTYPE html>
```

```
<html>
<head>
    <title>Community Health Care Management</title>
    <link rel="stylesheet" href="index.css">
</head> <body> <h1> Room Management</h1>
     <div class="menu">
            <a href="index.html">Clinics</a>
            <a href="doctors.html">Doctors</a>
            <a href="patient.html">Patients</a>
            <a href="room.html">Rooms</a></div>
    <div class="container">
        <h1>Rooms</h1>
        <div>
            <h3>Select Clinic</h3>
            <select class="clinicSelect">
                <option value="">Select Clinic</option>
            </select>
        </div>
        <div id="roomsSection">
            <h2>Rooms</h2>
            <table>
                <thead>
                    <tr>
                        <th>Room No</th>
                        <th>Floor</th>
                        <th>Doctor</th>
                    </tr>
                </thead>
                <tbody id="roomTable">
                </tbody>
            </table>
        </div>
    </div>
    <script src="api.js"></script>
</body> </html>
```

5.index.css:

```
body {
    font-family: Arial, sans-serif;
    margin: 20px;
    background: #f5f5f5;
}
.container {
    max-width: 1200px;
    margin: 0 auto;
```

```
    background: white;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 10px rgba(0,0,0,0.1);
}
h1 {
    color: #333;
    text-align: center;
}

.menu {
    text-align: center;
    margin: 20px 0;
}

.section {
    margin-top: 30px;
}

.form {
    margin-left: 20px 0;
    padding: 20px;
    background: #f9f9f9;
    border-radius: 5px;
}

.form input, .form select, .form button {
    margin: 5px;
    padding: 8px;
    border: 1px solid #ddd;
    border-radius: 3px;
}

.form button {
    background: #2196F3;
    color: white;
    border: none;
    cursor: pointer;
}

.form button:hover {
    background: #0b7dda;
}

table {
    width: 100%;
```

```
    border-collapse: collapse;
    margin-top: 20px;
}

table th, table td {
    border: 1px solid #ddd;
    padding: 8px;
    text-align: left;
}

.delete-btn {
    background: #f44336;
    color: white;
    border: none;
    padding: 5px 10px;
    border-radius: 3px;
    cursor: pointer;
}

.delete-btn:hover {
    background: #d32f2f;
}
```

6.The functions for each HTML on api.js :

General Functions

- loadClinics() :Loads all clinics to combobox.

Clinics Page (index.html)

- loadClinicTable() : Loads clinics to the table.
- deleteClinic(id) :Deletes a clinic by id.
- addClinic() :Adds a new clinic to the system.

Doctors Page (doctors.html)

- loadAllDoctors() :Loads all doctors.

- loadDoctorsByClinic(clinicId) :Loads doctors by clinic id .
- renderDoctors(doctors) :Displays doctors in table.
- addDoctorToClinic() :Adds doctor to clinic.
- deleteDoctor(doctorId) :Deletes a doctor by id.

Patients Page (patient.html)

- loadDoctorsForClinicForm(clinicId) :Loads doctors for form dropdown.
- loadDoctorsForClinic(clinicId) :Loads doctors for filter dropdown for table.
- loadPatientsForDoctor(doctorId) :Loads patients for selected doctor.
- addPatientToClinicAndDoctor() : Adds patient and assigns to clinic and doctor.

Rooms Page (room.html)

- loadRoomsForClinic(clinicId) : Loads rooms for selected clinic by id.

Global Window Functions

- window.deleteClinic :Global clinic delete function.
- window.addClinic :Global clinic add function
- window.addDoctorToClinic :Global doctor add function
- window.deleteDoctor :Global doctor delete function
- window.addPatientToClinicAndDoctor : Global patient add function

6. Some part of code from api.js:

In here, the URL of my API is defined.

```
const API_URL = 'http://localhost:8080';
```

The system has asynchronous functions to delete, add, and display entities. In the below, I showed some of them.

First, the JavaScript code for index.html. I wrote the whole JS in one file. The code will be executed according to the current page. It is made as shown below:

```
if ( window.location.pathname.includes('index.html') ||
window.location.pathname === '/'){
document.addEventListener('DOMContentLoaded', () => { loadClinicTable();
loadClinics();});
```

In here, the system takes the whole clinic from the API. If there is no clinic on the database, it adds a row displaying the "no clinics" message. If it is found, then it adds a row and data to each clinic. There is a delete button on the row that deletes the clinic by taking the id of it.

```
async function loadClinicTable() {
        try {
            const res = await fetch(API_URL + '/clinics');
            const clinics = await res.json();

            const table = document.getElementById('clinicList');
            if (!table) return;

            table.innerHTML = clinics.length === 0
                ? '<tr><td colspan="5">No clinics</td></tr>'
                : clinics.map(c => `
                    <tr>
                        <td>${c.id}</td>
                        <td>${c.name}</td>
                        <td>${c.address}</td>
                        <td>${c.phoneNumber}</td>
                        <td>
                            <button
onclick="deleteClinic(${c.id})">Delete</button>
                        </td>
                    </tr>
                `).join('');
        } catch (e) {
            console.error('Failed to load clinics', e);
        }
    }
```

I used this structure for the other entities on different HTML pages as well to display on the table. On the code, I used a ternary operation to write if and else shortly.

```
function renderDoctors(doctors) {
    const tbody = document.getElementById('doctorTable');
    if (!tbody) return;
```

```
    tbody.innerHTML = doctors.length === 0
        ? `<tr><td colspan="3">No doctors</td></tr>`
        : doctors.map(d => `
            <tr>
                <td>${d.name}</td>
                <td>${d.surname}</td>
                <td>
                    <button onclick="deleteDoctor(${d.id})">Delete</button>
                </td>
            </tr>
        `).join('');
}
```

The function to delete the clinic:

Before deleting the clinic, the system asks the manager whether they want to delete the clinic or not to protect from accidental loss of any entity. We declare the method of our fetch; here we use the delete method as shown. In the other codes, I used a similar structure.

```
    window.deleteClinic = async function (id) {
        if (!confirm('Do you want to delete?')) return;
        await fetch(`${API_URL}/clinics/${id}`, { method: 'DELETE' });
        loadClinicTable();
        loadClinics();
    };
```

The example function for save a clinic:

In here, the system at least wants a clinic name to save a new clinic. First, the system gets a clinic name, then trims it to get a clear name without space. If there is no name, it alerts the manager. Then it creates an object with values; the address and phone number can be empty, and then they will be assigned as shown below. The object is sent by the POST method after being turned into JSON format with the JSON.stringify() method. Again, the same structure is used for other saving functions for different entities.

```
    window.addClinic = async function () {
        const name = document.getElementById('clinicName').value.trim();
        if (!name) {
            alert('Clinic name cannot be empty');
            return;
        }
```

```
        const clinic = {
            name,
            address: document.getElementById('clinicAddress')?.value || '',
            phoneNumber: document.getElementById('clinicContact')?.value || ''
        };
        await fetch(API_URL + '/clinics/save', {
            method: 'POST',
            headers: { 'Content-Type': 'application/json' },
            body: JSON.stringify(clinic)
        });
        loadClinicTable();
        loadClinics();
    };}
```

The codes for loading clinics:

```
async function loadClinics(onChangeCallback = null) {
    const selects = document.getElementsByClassName('clinicSelect');
    if (selects.length === 0) return;
    try {
        const res = await fetch(API_URL + '/clinics');
        const clinics = await res.json();

        Array.from(selects).forEach(select => {
            select.innerHTML = '<option value="">Select Clinic</option>';

            clinics.forEach(c => {
                select.innerHTML += `
                    <option value="${c.id}">${c.name}</option>
                `;
            });

            if (onChangeCallback) {
                select.onchange = () => onChangeCallback(select.value);
            }
        });

        console.log('Clinics loaded to dropdown');
    } catch (err) {
        console.error('Failed to load clinics:', err);
    }
}
```

In conclusion, this Community Healthcare Management Interface was created by html, CSS, and JavaScript. The data is fetched from a restful API that I wrote earlier which contains four entities and various functions on the service layer. I created a clear and easy to use interface for my API that enables managers to manage clinics, doctors, patients, and rooms.