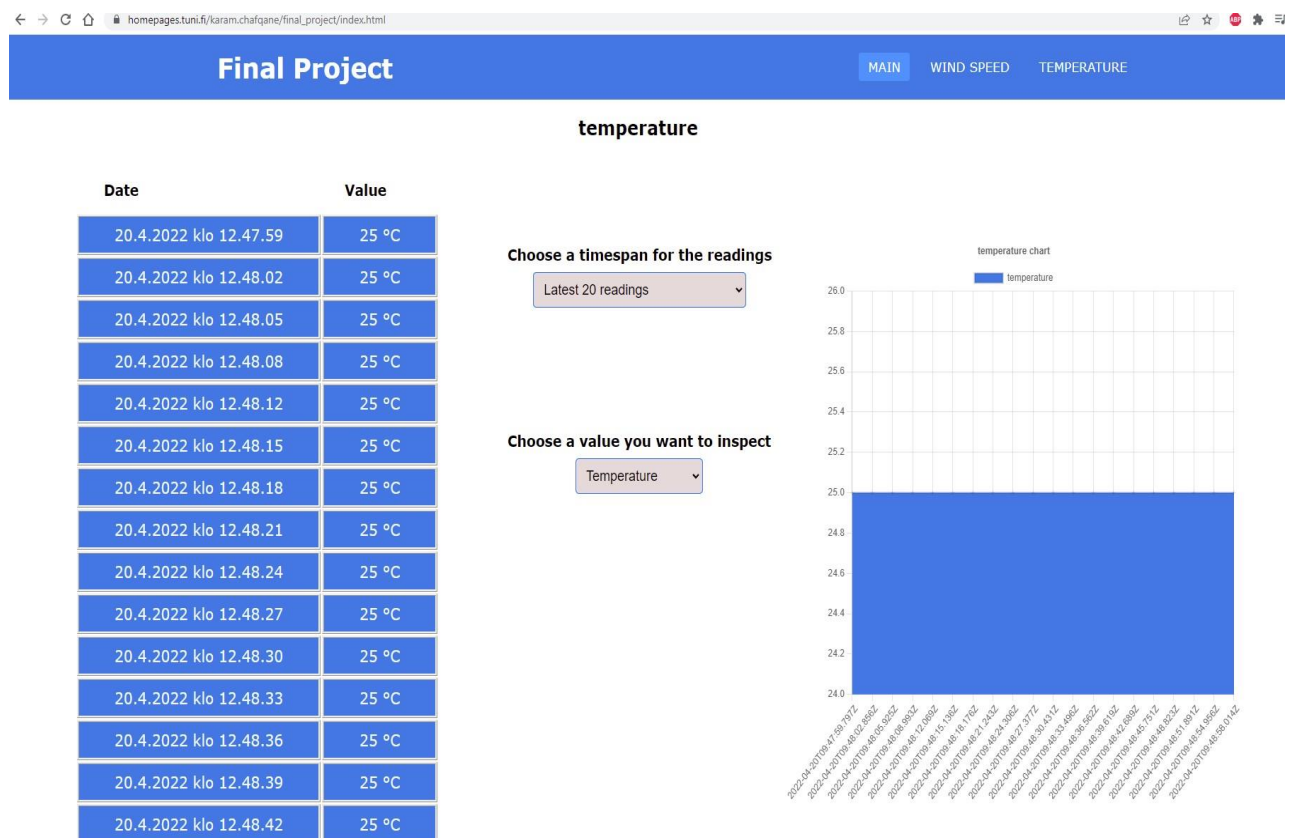


Final Project

Karam Chafqane

The final project works like a weather station for the school premises. The program consists of 3 views or pages. Main page, wind speed page and a temperature page. I aimed for a grade 4 in my project.

Main Page



Picture of the main page perfectly Shows all the attributes needed for the maximum grade. The main page consists of the fetched table data, chart that displays the data and a couple of selectors, to change the values and range of the data displayed. As a user, you can choose from variable different values temperature, wind speed, humidity etc. and timespan all the way from average values per hour form the past month to latest 20 readings.

CSS

```
final_project > style.css > td
1  ∨ *{
2      padding: 0;
3      margin: 0;
4      text-decoration: none;
5      list-style: none;
6      box-sizing: border-box;
7  }
8
9  ∨ body{
10     font-family: Verdana, Geneva, Tahoma, sans-serif;
11 }
12
13
14 ∨ nav{
15     background: #0082e6;
16     height: 80px;
17     width: 100%;
18 }
19
20 ∨ label.logo{
21     color: white;
22     font-size: 35px;
23     line-height: 80px;
24     padding: 0 300px;
25     font-weight: bold;
26 }
27
28 ∨ nav ul {
29     float: right;
30     margin-right: 300px;
31 }
32
33 ∨ nav ul li{
34     display: inline-block;
35     line-height: 80px;
```

CSS is one of my weakest points in web development. I have not yet fully figured out how you can take full advantage of CSS elements when constructing a page. Nevertheless, after some struggles, I did manage to make the page look somewhat nice.

I did try to implement bootstrap to the page, but I did it after I was pretty much done with the project, so when I tried to add it in the whole thing kind of collapsed and nothing really worked anymore as intended. I learned that you should start with the bootstrap and not add it in later.

JavaScript

```
const fetchCollectors = async (value, valueTime) => {
  try {
    console.log("Value:", value)
    console.log("time:", valueTime)
    const response = await fetch ('https://webapi19sa-1.course.tamk.cloud/v1/weather/' + value + '/' + valueTime);
    const jsonData = await response.json();
    // console.log(jsonData)
    populateTable(jsonData, value)
    drawChart(jsonData, value)
  } catch (error) {
    console.error(error);
  }
}

const selectValue = document.getElementById("valueSelect");
selectValue.addEventListener('change', (event) => {

  fetchCollectors(selectValue.value, selectTime.value)

  console.log("time Inside select value: " ,selectTime.value)

  resetTable();

}))

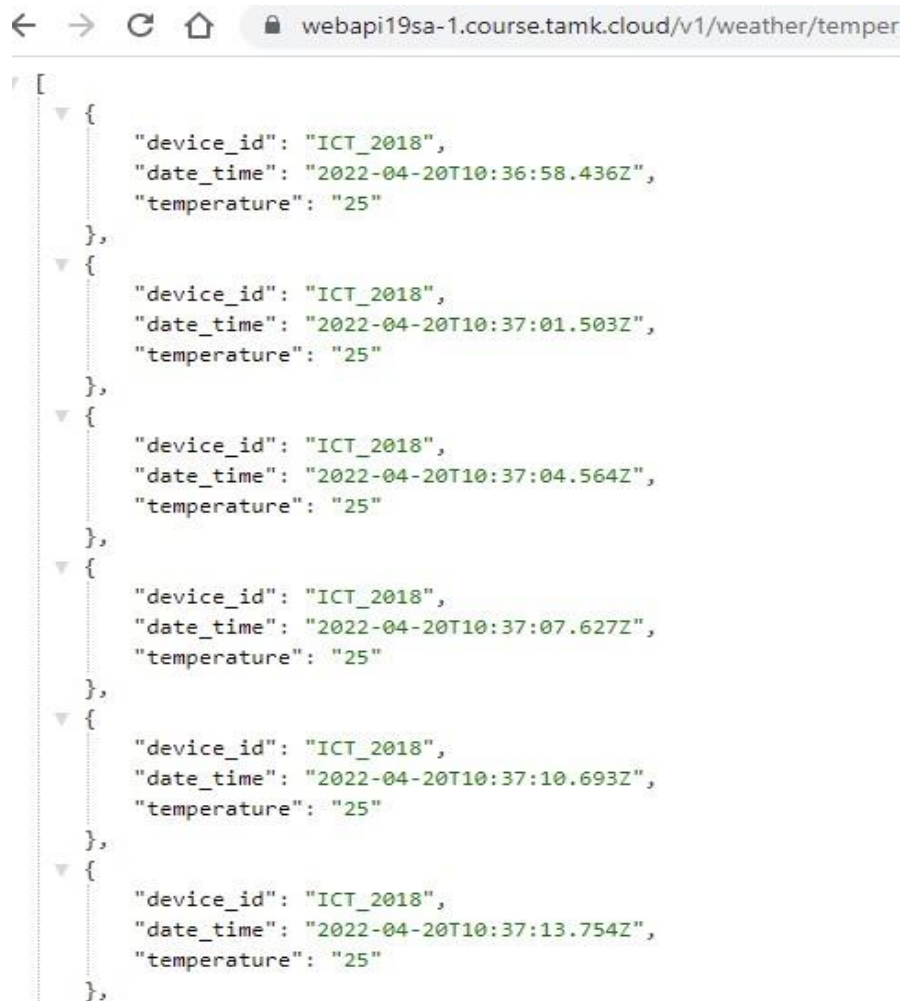
const selectTime = document.getElementById("timespanSelect");
selectTime.addEventListener('change', (event) => {

  fetchCollectors(selectValue.value, selectTime.value)
```

Picture above shows only a small part of the main page JavaScript, but it shows one of the key parts of the code.

We see how the user selected values affects which API endpoint is called, when fetching the data. After the data is fetched, it is sent to 2 separate functions which are the table and the chart. Chart and the table are then constructed by using the data that is being fetched from the endpoint with the user selected values.

JSON data



The screenshot shows a web browser window with the address bar displaying the URL `webapi19sa-1.course.tamk.cloud/v1/weather/temper`. The main content area displays a JSON array of six objects, each representing a temperature reading. The objects are expanded to show their internal structure. Each object contains three fields: `device_id` (always "ICT_2018"), `date_time` (ISO 8601 timestamps), and `temperature` (always "25").

```
[
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:36:58.436Z",
    "temperature": "25"
  },
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:37:01.503Z",
    "temperature": "25"
  },
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:37:04.564Z",
    "temperature": "25"
  },
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:37:07.627Z",
    "temperature": "25"
  },
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:37:10.693Z",
    "temperature": "25"
  },
  {
    "device_id": "ICT_2018",
    "date_time": "2022-04-20T10:37:13.754Z",
    "temperature": "25"
  }
]
```

<https://webapi19sa-1.course.tamk.cloud/v1/weather/temperature/>

When you open the page, this is what is being fetched from the API. We get the latest 20 readings from the temperature value. This of course changes with the users selected value and range.