MIS203 - Web Programming Languages
Term Project
Report 3: Realization of Our Web Page

Team Members Involved in Part 3	
1.	Hazal Serençelik
Finalised on 17/12/2024	

1. Visual design of our web page:

1.1. Where did we locate the menu?

As in our planning, we have placed the navigation menu at the top-center of the page which follows the established UX pattern we see in Khas institutional pages such as Learn and MyKhas. The menu has core tabs mentioned.

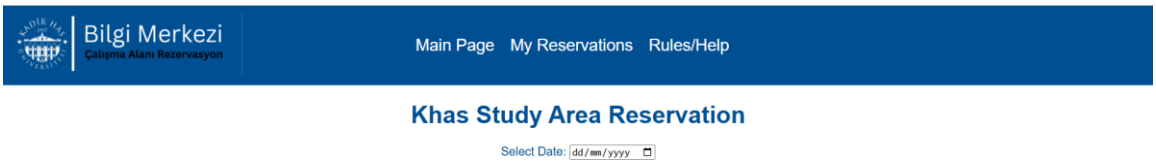

Figure 1: Logo, navbar, first load of the page

1.2. Which content did we add? How many pages are linked to our main page? What kind of interactions does it have?

Instead of new pages everytime, we preferred pop-ups on main page which sequences the user events on the page. Other pages are “My Reservations” and “Rules” that only display texts.

Main Page: To have more directive page, we started from date selection. After the date is selected, seat layout appears. We couldn't start with the seat layout because the layout changes depending on the selected date. If a seat is fully booked, it should be displayed in red. This status can be correct for 16/12/2024, but it is not shown correctly for 18/12/2024 maybe.

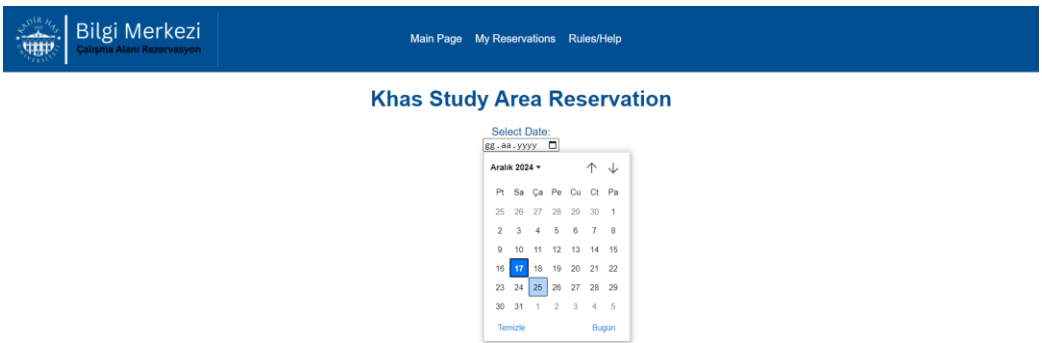

Figure 2: Main page; calender is a browser call, a built-in function

Slot Selection Pop-up: When a user clicks on a seat, a popup displays the seat's time slots in a table with 0.5-hour intervals. A special class assignment was needed since the opening hours differ on weekends. We did not initialize all slots with the "available" class since the 8:30–10:00 slots on weekends should not be selectable.

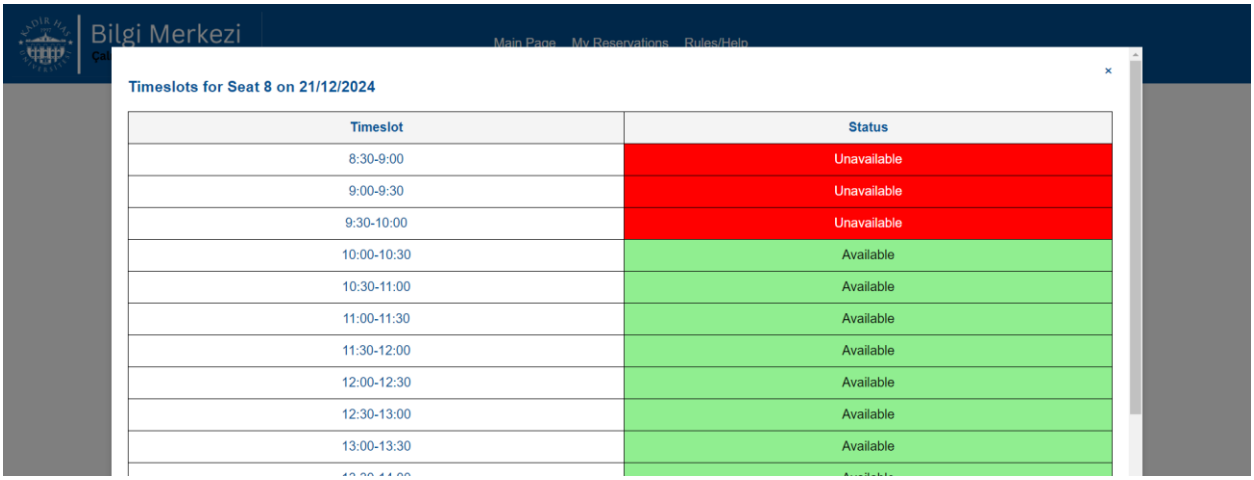

Figure 3: Slot block for weekends

Figure 4: Color-text transitions from "Available" to "Selected" when slots are clicked before confirmation

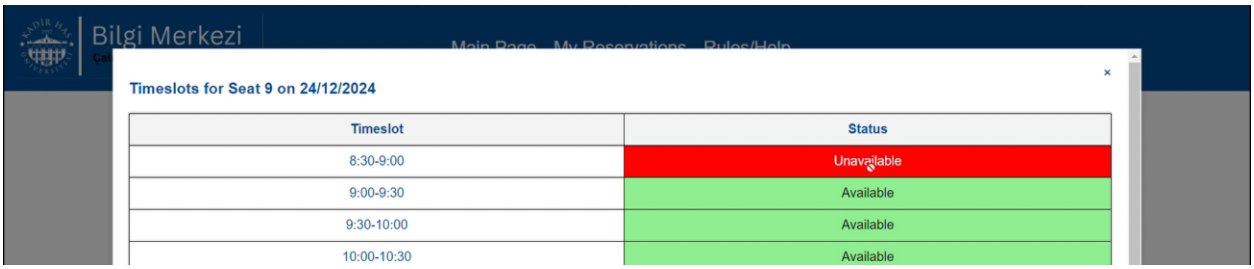

Figure 5: Color-text transitions and availability on the timeslot table after visiting a seat-hour had been selected and confirned before

Confirmation Pop-up:

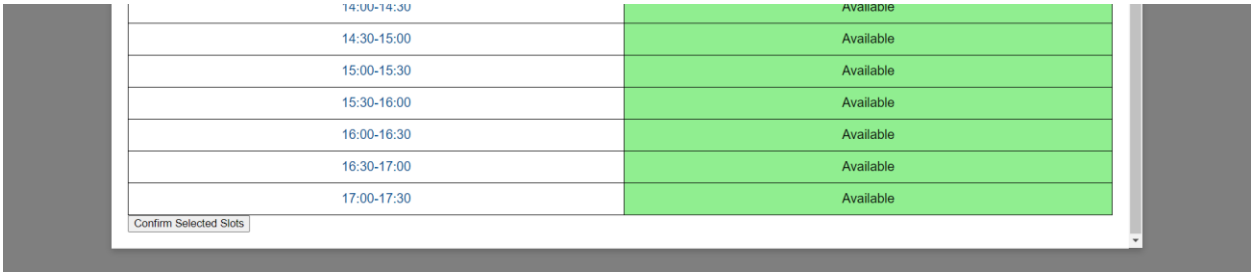

Figure 6: The button from timeslot table triggers the confirmation pop-up, added "Cancel" later

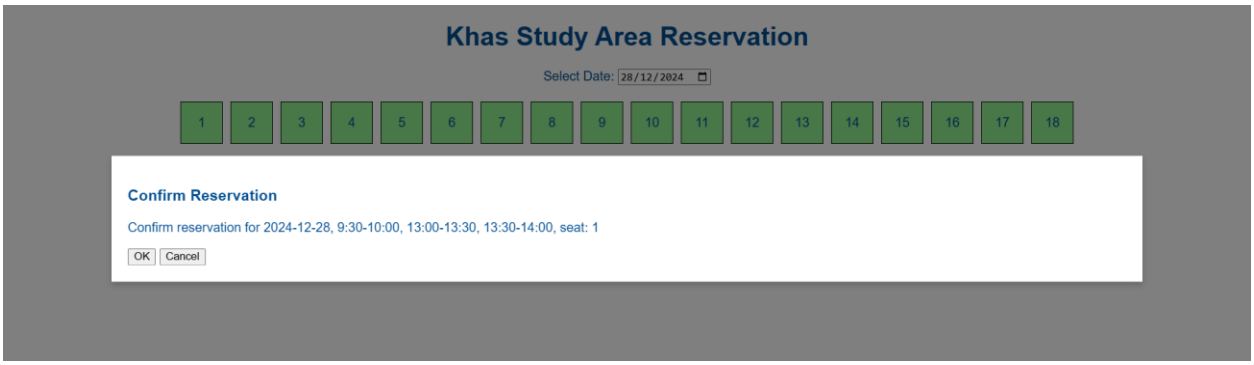

Figure 7: Confirmation pop-up. "Ok" triggers to save date, time and seat to transfer them to "My Reservations".

We used to think about a timetable similar to the Figure 7 below. However, we decided to change it because of the total number of availability from row*column, we have lots of combinations to manage/save. Additionally, waiting until all slots were taken to change the seat color was not satisfactory. So, we decided to make status (seat's) change on a daily basis which is more practical/meaningful.

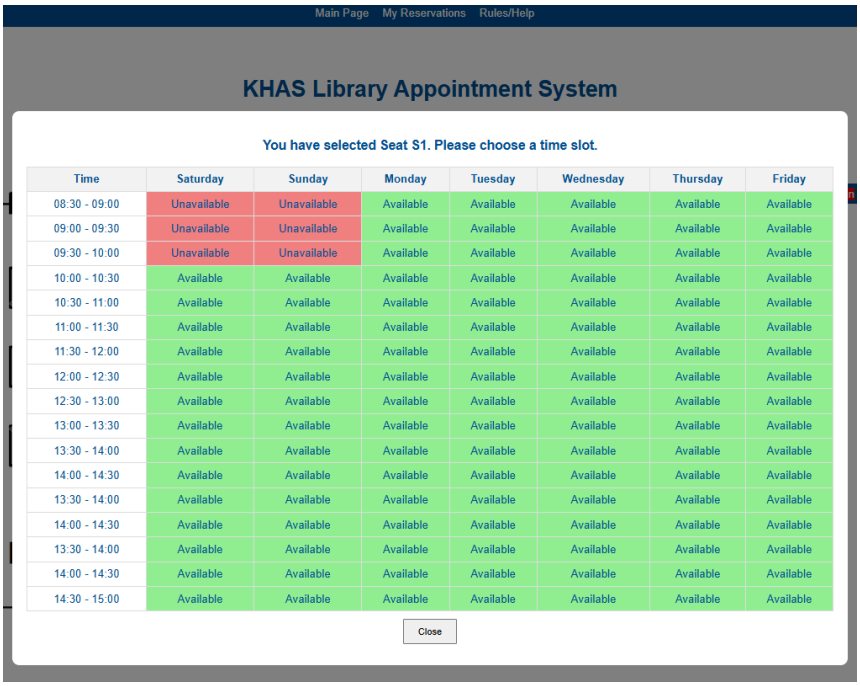

Figure 8: First did this, then changed into daily basis

1.3. Case of a full seat:

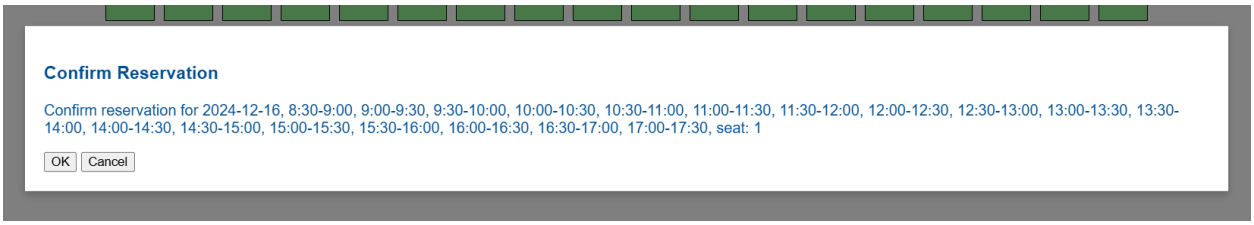

Figure 9: All slots are selected

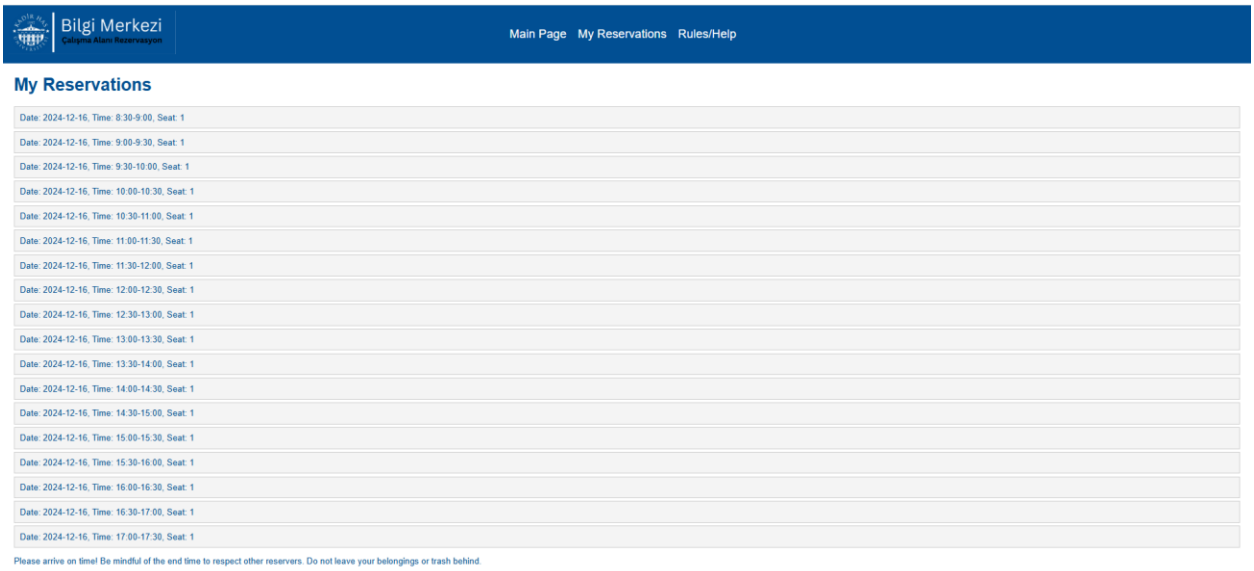

Figure 10: Status transference to "My reservations" page

Figure 11: All slots of this seat have been selected and confirmed. Consequently, the seat's status has been updated and its color has been changed accordingly

We wanted to color the seats and changed our approach three times:

1. We sourced an image of a seat layout that we drew it on Canva. With the help of https://www.image-map.net/ , we found the coordinates of each seat and coded as, for example, clicking on the coord. area of seat 1 would display its timeslots. However, this approach failed because we couldn't dynamically change the image for every scenario. It required creating 2^18 different images to change them for all possible seat coloring patterns.

```
<div class="seat-selection">
    <img src="Layout.png" usemap="#image-map" alt="Library Layout">
    <div class="seat-status">Legend: <span class="available">Available</span> | <span class="taken">Taken</span> | <span class="blocked">Blocked</span></div>

    <!-- Define the Image Map -->
    <map name="image-map">
        <area shape="rect" coords="153,190,86,126" alt="S1" title="S1" class="available" onclick="selectSeat('S1')">
        <area shape="rect" coords="279,124,346,188" alt="S2" title="S2" class="available" onclick="selectSeat('S2')">
        <area shape="rect" coords="85,238,154,305" alt="S3" title="S3" class="available" onclick="selectSeat('S3')">
        <area shape="rect" coords="281,238,345,303" alt="S4" title="S4" class="available" onclick="selectSeat('S4')">
        <area shape="rect" coords="84,347,153,416" alt="S5" title="S5" class="available" onclick="selectSeat('S5')">
        <area shape="rect" coords="279,349,346,416" alt="S6" title="S6" class="available" onclick="selectSeat('S6')">
        <area shape="rect" coords="406,125,473,191" alt="S7" title="S7" class="available" onclick="selectSeat('S7')">
        <area shape="rect" coords="600,126,665,190" alt="S8" title="S8" class="available" onclick="selectSeat('S8')">
        <area shape="rect" coords="406,238,471,301" alt="S9" title="S9" class="available" onclick="selectSeat('S9')">
        <area shape="rect" coords="601,239,664,302" alt="S10" title="S10" class="available" onclick="selectSeat('S10')">
        <area shape="rect" coords="406,351,470,415" alt="S11" title="S11" class="available" onclick="selectSeat('S11')">
        <area shape="rect" coords="601,352,664,415" alt="S12" title="S12" class="available" onclick="selectSeat('S12')">
        <area shape="rect" coords="730,124,795,191" alt="S13" title="S13" class="available" onclick="selectSeat('S13')">
        <area shape="rect" coords="925,127,989,190" alt="S14" title="S14" class="available" onclick="selectSeat('S14')">
        <area shape="rect" coords="727,238,794,302" alt="S15" title="S15" class="available" onclick="selectSeat('S15')">
        <area shape="rect" coords="925,239,990,303" alt="S16" title="S16" class="available" onclick="selectSeat('S16')">
        <area shape="rect" coords="731,351,797,418" alt="S17" title="S17" class="available" onclick="selectSeat('S17')">
        <area shape="rect" coords="923,351,989,415" alt="S18" title="S18" class="available" onclick="selectSeat('S18')">
    </map>
</div>
```

Figure 12: <area> to set initial classes then functions to select a seat

2. We tried to show seats with div elements styled as simple boxes labelled S1, S2, ..., S18 (Figure 11). This approach worked functionally but was visually unappealing and did not align with our goal of creating visually engaging interface. Thought that a new at first sign wouldn't understand that it was a seat/study room reservation page.

3. We uploaded two mirrored seat png images and treated them as div elements. However, we couldn’t manage to change the color of the entire image dynamically. Instead, we decided to change the color of the seat number displayed over the images, which was positioned using css.

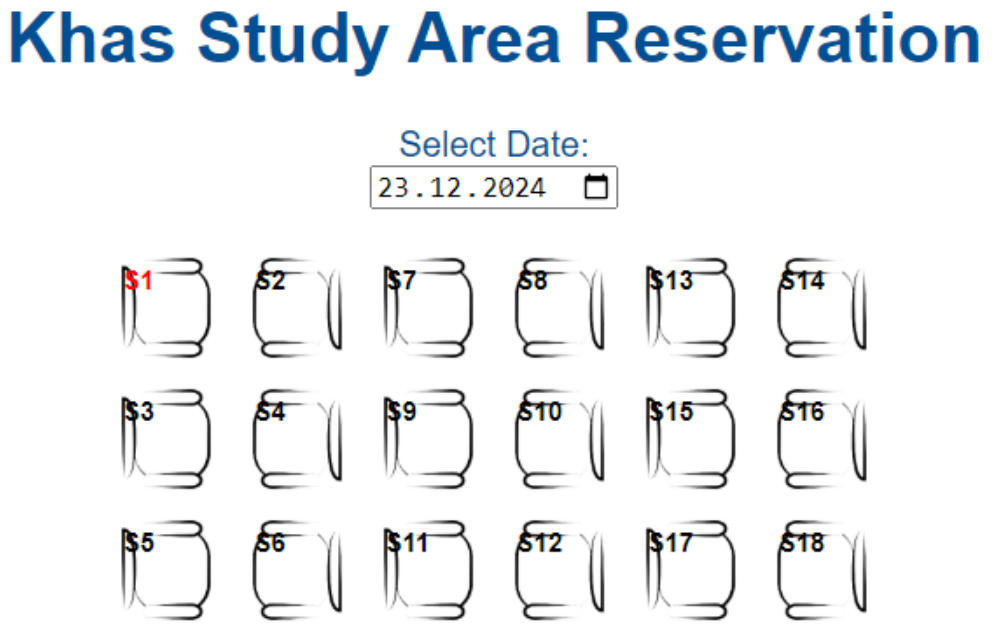

Figure 13: Visually enhanced version of Figure 10.

2. Structure of our web page

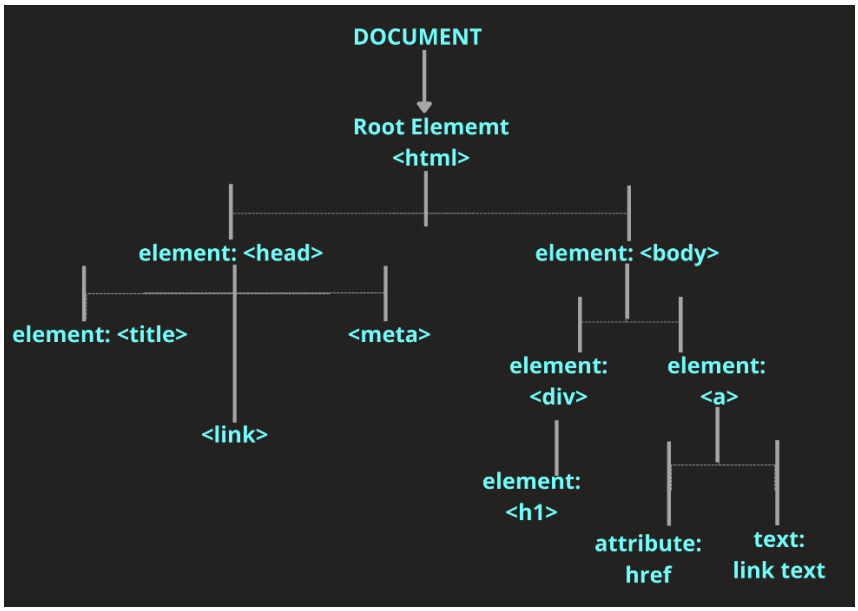

Figure 14: DOM from https://www.freecodecamp.org/news/what-is-the-dom-explained-in-plain-english/

We considered the *user's journey* and the *DOM structure* on our page while sequencing functions and searching inspiration online. Our primary source of inspiration was movie ticket/seat selection page and for js, we heavily relied on W3Schools and to add/modify code according to our needs.

The biggest challenge was treating all seats individually. In the initial stages, our code updated a status and it affected all other seats. It was difficult to combine each seat's status (class) with at least three values: that seat, that date and that hour(s). Sometimes, we attempted to def functions and failed, but then realized there are *built-in functions* already exists to handle those tasks. However, another challenge occurred since these built-in functions required specific value/array formats to work correctly and modifying our earlier assignments to match those formats was challenging. Constantly changing and error everytime.

The *class assignments* for our case were critical because they manage the visual changes of the seat and timeslot statuses, which are the core of the page. A class helps dynamically update and display the current state of the seat&timeslots.

Classes used:

- seat: base class for all seat divs
- taken: all timeslots (of that seat) are reserved. It helps changing seat color to red
- available: first, marks a timeslot as free and clickable for selection
- unavailable: indicates a timeslot that cannot be selected/clicked, when class is this, cell is in red with a disabled cursor
- selected: middle of transition. It helps coloring timeslots that the user has chosen for reservation to yellow

Class transformations:

- available → selected: clicking on a timeslot moves between available (default) and selected (highlighted state).
- seat → taken: when all timeslots of a seat are confirmed, the seat class receives the taken class, calls visual change of its reserved status
- unavailable: static class for timeslots that are pre-reserved or blocked, which does not change (for different opening hour on Saturdays and Sundays)

Our functions/calls that sequence and update the events:

- showSeats(): generates the seat layout and attaches event listeners for seat selection.
- showPopup(seatIndex): displays the popup with available timeslots for the selected seat
- showConfirmation(): displays a confirmation popup
- confirmReservation(): confirms the reservation, updates seat data and
- cancelReservation(): closes the confirmation popup and returns to the seat layout
- closePopup(): closes the timeslot selection popup and refreshes the seat layout
- showReservationsPage(): displays the list of confirmed reservations on a separate page
- showMainPage(): returns to the main page with the seat layout from the reservations page

Built in functions:

- document.getElementById(): retrieves an element by its ID to manipulate or update its content or behavior (in seatLayout, popup, confirmationText)
- Array.from(): Converts a structure like an iterable or array-like object into an actual array (used in typing hours for timeslots and picking selected many elements)
- map(): creates a new array by applying a function to every element in an array (selected timeslots)
- forEach(): iterates over an array and performs a specified action for each element (used in isssue of not spreading one seat's status on others, (seatLayout and reservations))
- alert(): displays a popup message (from browser, not as a popup created with js-css, etc.) to notify the user with message
- createElement(): dynamically creates new DOM elements (creating div, tr, td, li elements).
- innerHTML: Sets or gets the HTML content of an element (clearing or adding elements like seatLayout or timetable)
- classList.add() classList.remove() classList.toggle(): adds, removes, or changes CSS classes for an element (class transformations mentioned before)
- appendChild(): attaches a created element (many seats, many rows to table)

3. Changes needed before going online

- Proper storage to maintain seat status in many sessions
- Blocking past dates on the calendar
- Additional visual enhancements