JS Front-end: Exam Preparation 2

Link to contest: https://judge.softuni.org/Contests/4828/JS-Front-End-Retake-Exam-22-August-2024

01. Science Experimentation

You are a scientist conducting experiments. Follow protocols to mix chemicals, observe reactions, and document results to make groundbreaking discoveries. Each experiment involves a series of steps and specific chemical requirements.

On the first line of the standard input, you will receive an integer **n** – the number of chemicals available in your lab.

On the next **n** lines, the chemical details will follow with their names and quantities separated by a hashtag in the following format:

`"{chemical name} # {quantity}"`

Chemical quantity represents the current quantity in the lab, ranging from 0 to 500.

After you have your chemicals listed, you will receive different commands, each on a new line, separated by "**#**", until the "End" command is given. There are three actions you can perform:

`"Mix # {chemical name 1} # {chemical name 2} # {amount}"`

- Mix a specified amount of two chemicals together to observe a reaction.
- If the amount of both chemicals is sufficient to be mixed, **reduce** the amount of both chemicals and **print**:
  `"{chemical name 1} and {chemical name 2} have been mixed. {amount} units of each were used."`
- If either chemical does not have enough quantity, **print**:
  `"Insufficient quantity of {chemical name 1}/{chemical name 2} to mix."`

`"Replenish # {chemical name} # {amount}"`

- Replenish the quantity of a specific chemical.
- If the chemical does not exist, **print**:

  `"The Chemical {chemical name} is not available in the lab."`

- If the replenishment brings the quantity **above** the maximum storage capacity of 500 units, set the quantity to 500 units and **print**:

  `"{chemical name} quantity increased by {addedAmount} units, reaching maximum capacity of 500 units!"`

- Otherwise, **print**:

  `"{chemical name} quantity increased by {amount} units!"`

`" Add Formula # {chemical name} # {formula}"`

- Add a chemical formula to an existing chemical name in the lab.
- If the chemical exists, set its formula and **print**:

  `"{chemical name} has been assigned the formula {formula}."`

- Otherwise, **print**:

  `"The Chemical {chemical name} is not available in the lab."`

Input

- On the first line of the standard input, you will receive an integer **n**
- On the following **n** lines, the **chemicals** themselves will follow with their **quantity,** separated by a hashtag in the following format
- You will be receiving different **commands**, each on a new line, separated by **" # "**until the **"End"** command is given

Output

- Every command should **print** its own template sentence. In the **End**, **print** all chemicals with their updated quantities. If a chemical has a formula, include it in the output:
  `"Chemical: {chemical name}, Quantity: {quantity}, Formula: {formula}"`
- If a chemical does not have a formula, **print**:
  `"Chemical: {chemical name}, Quantity: {quantity}"`

Constraints

- The **names** of the chemicals will **always** be **unique**.
- All given **commands** will be **valid**.

Examples

Input	Output

`[ '4',`
`  'Water # 200',`
`  'Salt # 100',`
`  'Acid # 50',`
`  'Base # 80',`
`  'Mix # Water # Salt # 50',`
`  'Replenish # Salt # 150',`
`  'Add Formula # Acid # H2SO4',`
`  'End']`
	`Water and Salt have been mixed. 50 units of each were used.`
	`Salt quantity increased by 150 units!`
	`Acid has been assigned the formula H2SO4.`
	`Chemical: Water, Quantity: 150`
	`Chemical: Salt, Quantity: 200`
	`Chemical: Acid, Quantity: 50, Formula: H2SO4`
	`Chemical: Base, Quantity: 80`

Input	**Output**

`[ '3',`
`'Sodium # 300',`
`'Chlorine # 100',`
`'Hydrogen # 200',`
`'Mix # Sodium # Chlorine # 200',`
`'Replenish # Sodium # 250',`
`'Add Formula # Sulfuric Acid # H2SO4',`
`'Add Formula # Sodium # Na',`
`'Mix # Hydrogen # Chlorine # 50',`
`'End']`
	`Insufficient quantity of Sodium/Chlorine to mix.`
	`Sodium quantity increased by 200 units, reaching maximum capacity of 500 units!`
	`The Chemical Sulfuric Acid is not available in the lab.`
	`Sodium has been assigned the formula Na.`
	`Hydrogen and Chlorine have been mixed. 50 units of each were used.`
	`Chemical: Sodium, Quantity: 500, Formula: Na`
	`Chemical: Chlorine, Quantity: 50`
	`Chemical: Hydrogen, Quantity: 150`

02. Laptop Wishlist

Environment Specifics

Please be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

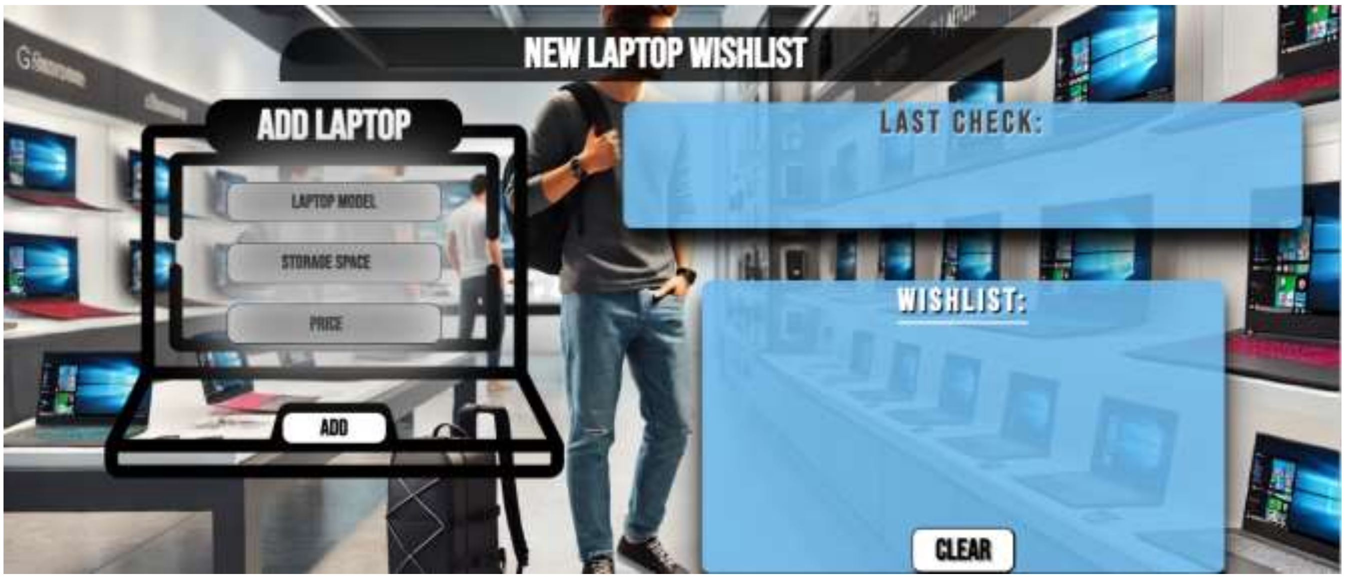
The following actions are **NOT** supported:

- `.forEach()` with **NodeList** (returned by `querySelector()` and `querySelectorAll()`)
- `.forEach()` with **HTMLCollection** (returned by `getElementsByClassName()` and `element.children`)
- using the **spread-operator** (`...`) to convert a **NodeList** into an array
- `append()` (use only `appendChild()`)
- `prepend()`
- `replaceWith()`
- `replaceAll()`
- `closest()`
- `replaceChildren()`

If you want to perform these operations, you may use `Array.from()` to first convert the collection into an array.

Use the provided skeleton to solve this problem.

Note: You **can't** and you have no permission to **change** directly the given HTML code (index.html file).



Your Task

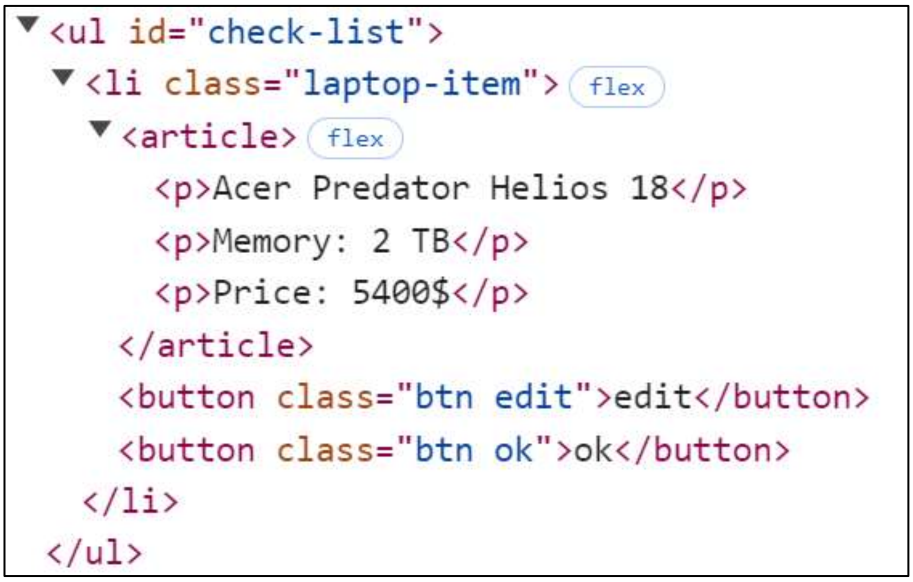
Write the missing JavaScript code to make the **Laptop Wishlist** work as expected:

- **Laptop Model, Storage Space,** and **Price** should be **non-empty strings**. If any of them are empty, the program should not do anything.

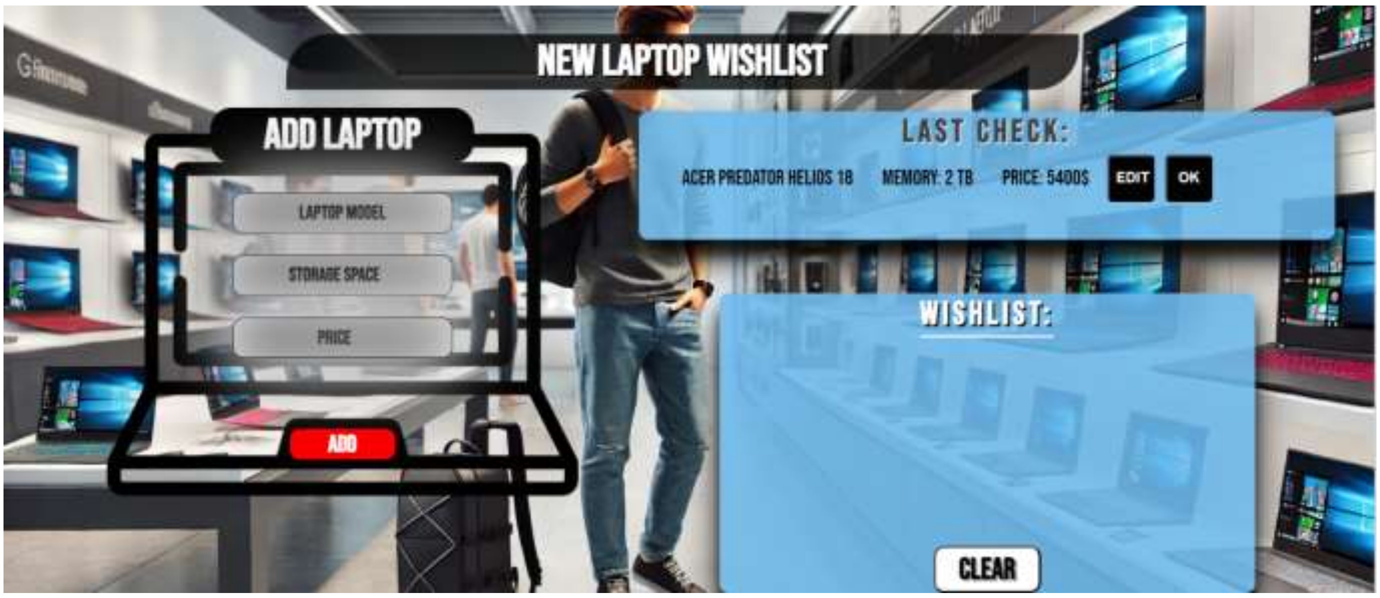
Getting the information from the form

When you click the **[Add]** button, the information from the input fields must be added to the **`<ul>`** with the **id `"check-list"`,** the **[Add]** button must be **disabled** and **the input fields should be cleared**.

The HTML structure should look like this:

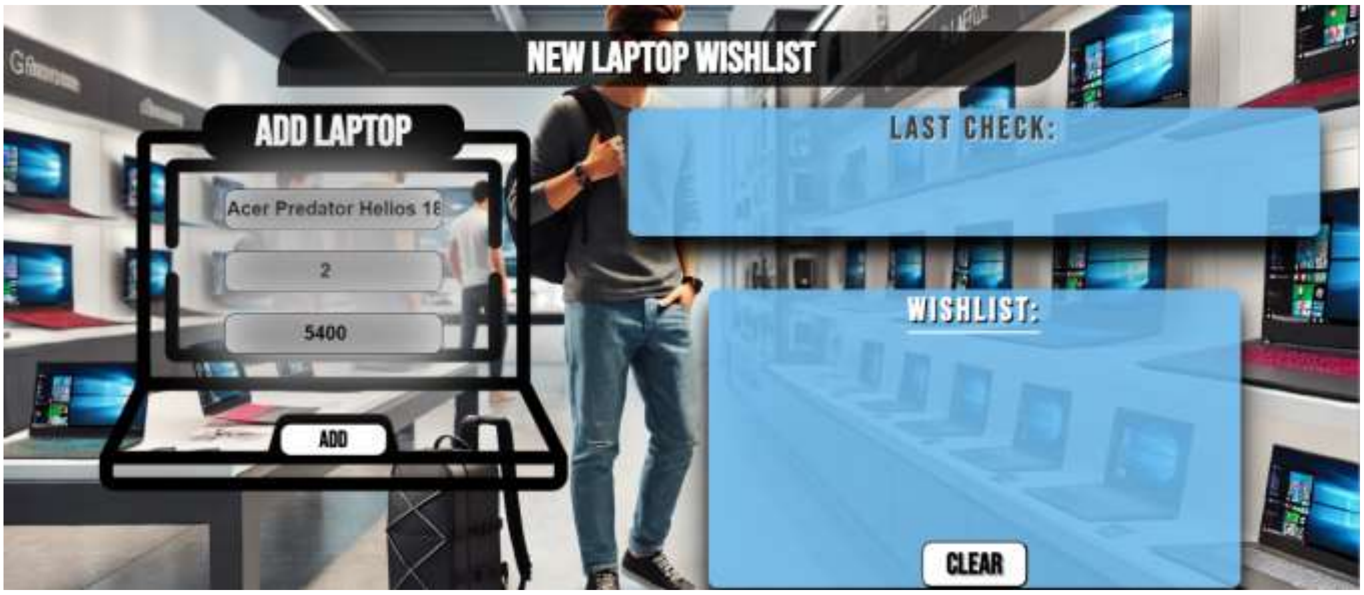


```
▼<ul id="check-list">
  ▼<li class="laptop-item"> flex
    ▼<article> flex
        <p>Acer Predator Helios 18</p>
        <p>Memory: 2 TB</p>
        <p>Price: 5400$</p>
     </article>
     <button class="btn edit">edit</button>
     <button class="btn ok">ok</button>
   </li>
 </ul>
```

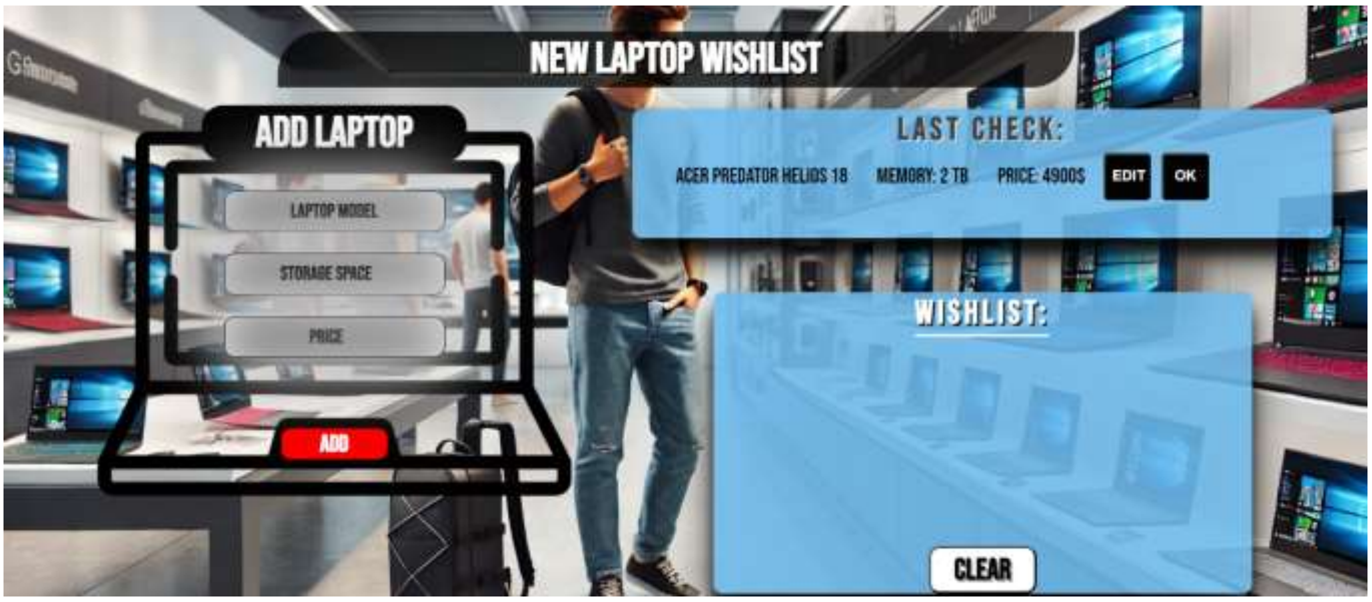


Edit information

When the **[Edit]** button is clicked, the information from the post must be sent to the input fields on the left side and the record should be deleted from the **`<ul>` `"check-list"`** and **[Add]** button must be **enabled** again.



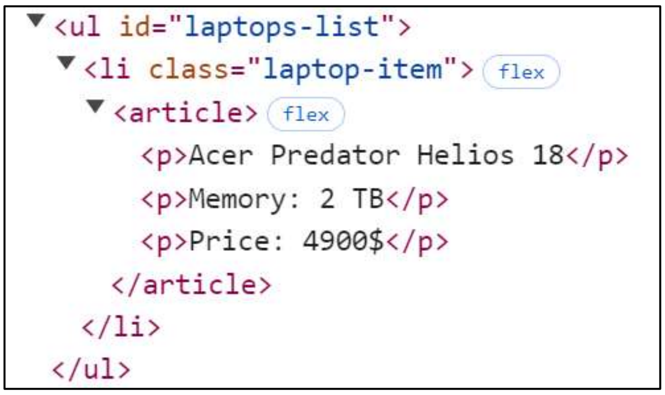
After editing the information, add a new item to the **`<ul>`** with the updated information.



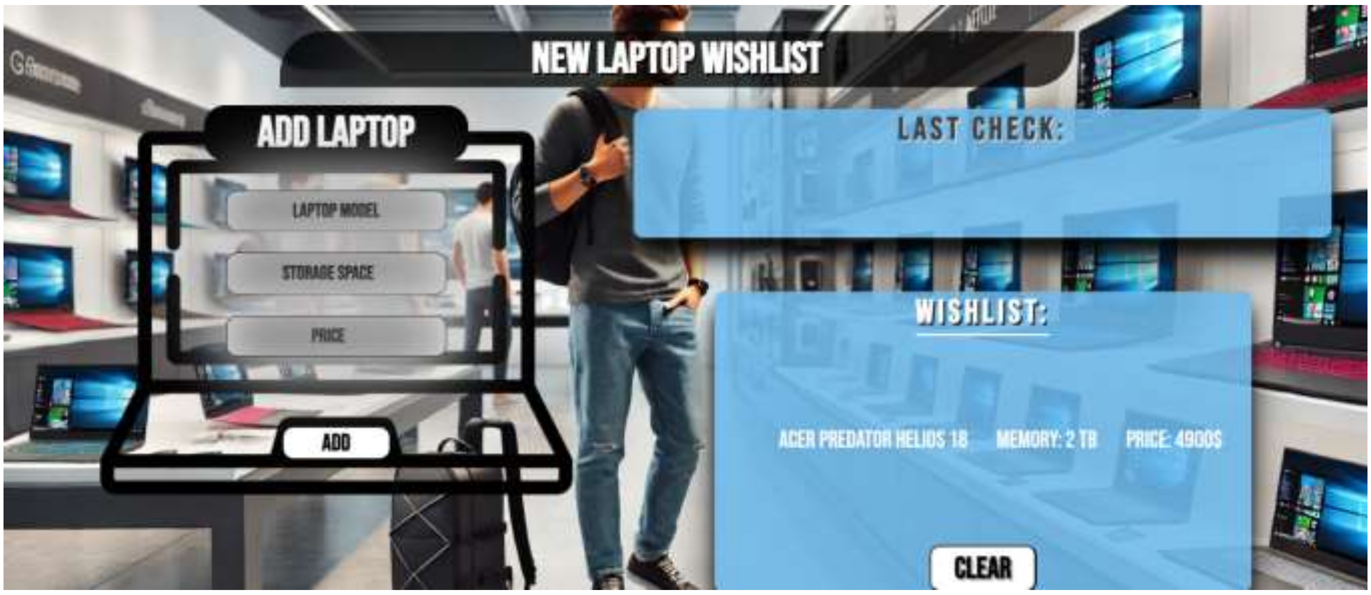
Add to Wishlist

When you click the **[Ok]** button, the record must be **deleted** from the **`<ul>`** with **id `"check-list"`** and appended to the **`<ul>`** with **id `"laptops-list"`**.

The **buttons [Edit]** and **[Ok]** should be removed from the **`<li>`** element and **the `[Add]`** button must be **enabled** again.



```
▼<ul id="laptops-list">
  ▼<li class="laptop-item"> flex
    ▼<article> flex
        <p>Acer Predator Helios 18</p>
        <p>Memory: 2 TB</p>
        <p>Price: 4900$</p>
      </article>
    </li>
  </ul>
```



Clear Wishlist

When the **[Clear]** button is clicked, you must **reload** the application**.**

Submission

Submit only your **solve()** function.

03. Car Maintenance Booking

Working with Remote Data

For the solution of some of the following tasks, you will need to use an up-to-date version of the **local REST service** provided in the lesson's resources archive. You can read the documentation here.

Environment Specifics

Please be aware that every JS environment may **behave differently** when executing code. Certain things that work in the browser are not supported in **Node.js**, which is the environment used by **Judge**.

The following actions are **NOT** supported:

- `.forEach()` with **NodeList** (returned by `querySelector()` and `querySelectorAll()`)
- `.forEach()` with **HTMLCollection** (returned by `getElementsByClassName()` and `element.children`)
- using the **spread-operator** (`...`) to convert a **NodeList** into an array
- `append()` (use only `appendChild()`)
- `prepend()`
- `replaceWith()`
- `replaceAll()`
- `closest()`
- `replaceChildren()`

If you want to perform these operations, you may use `Array.from()` to first convert the collection into an array.

Requirements

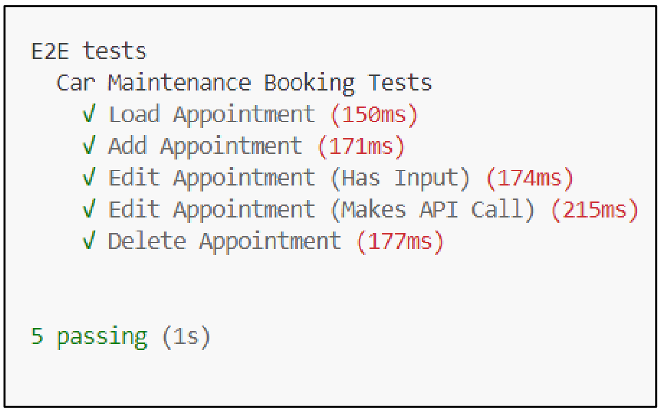
Write a JS program that can load, create, remove and edit a list of step Appointments. You will be given an HTML template to which you must bind the needed functionality.

First, you need to install all dependencies using the **`npm install`** command

Then, you can start the front-end application with the **`npm start`** command

You also must start the ***server.js*** file in the ***server*** folder using the **`node server.js`** command in another console **(BOTH THE CLIENT AND THE SERVER MUST RUN AT THE SAME TIME)**.

At any point, you can open up another console and run **`npm test`** to test the **current state** of your application. It's preferable for **all of your tests to pass locally** before you submit to the Judge platform, like this:



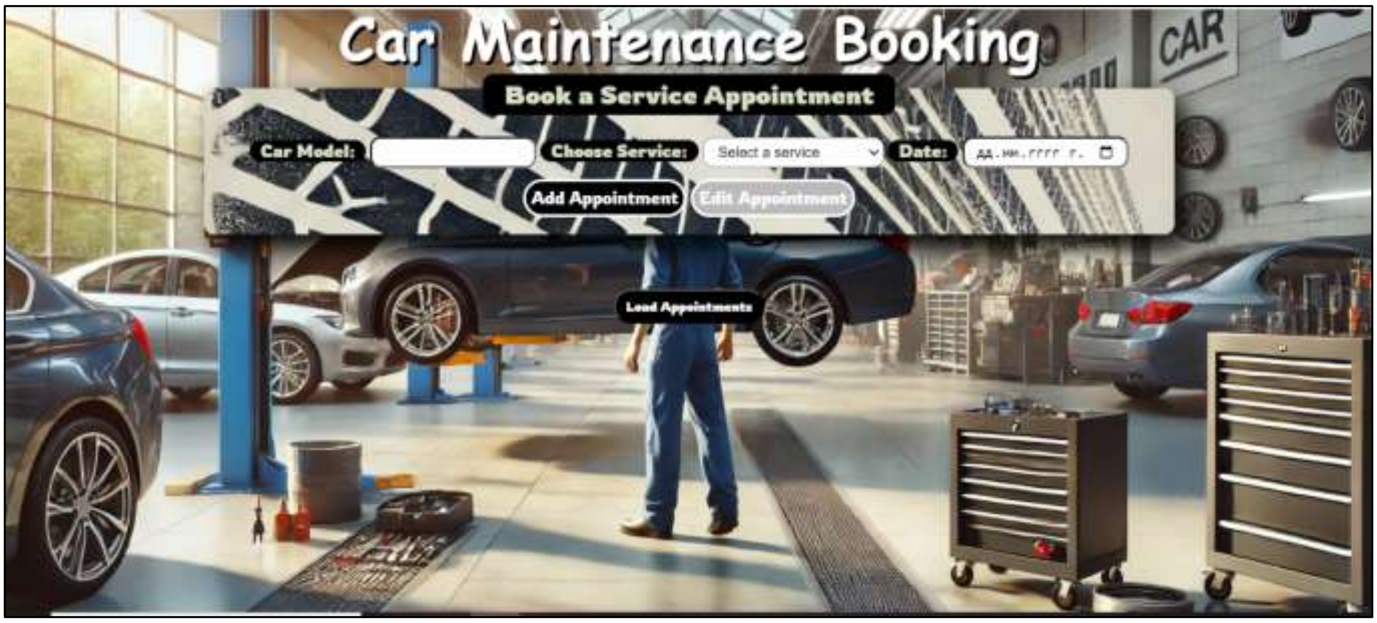
```
E2E tests
  Car Maintenance Booking Tests
    √ Load Appointment (150ms)
    √ Add Appointment (171ms)
    √ Edit Appointment (Has Input) (174ms)
    √ Edit Appointment (Makes API Call) (215ms)
    √ Delete Appointment (177ms)


5 passing (1s)
```

Endpoints

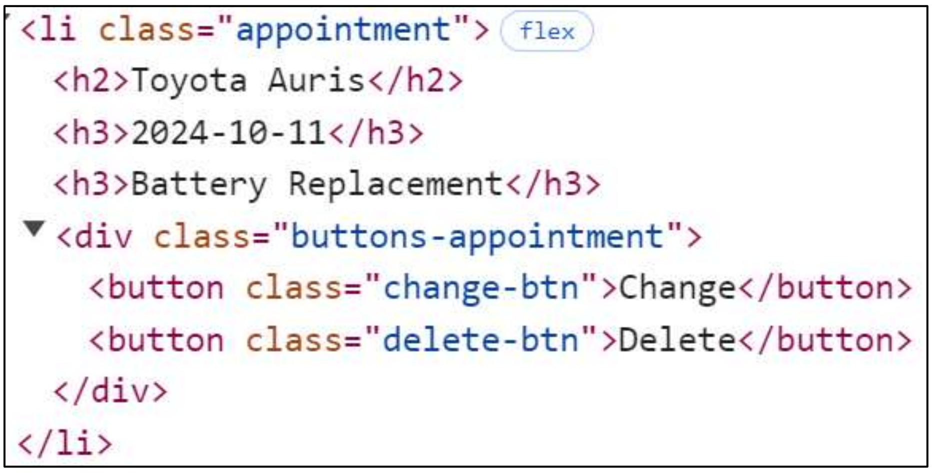
- http://localhost:3030/jsonstore/appointments/
- http://localhost:3030/jsonstore/appointments/:id

Load Appointments

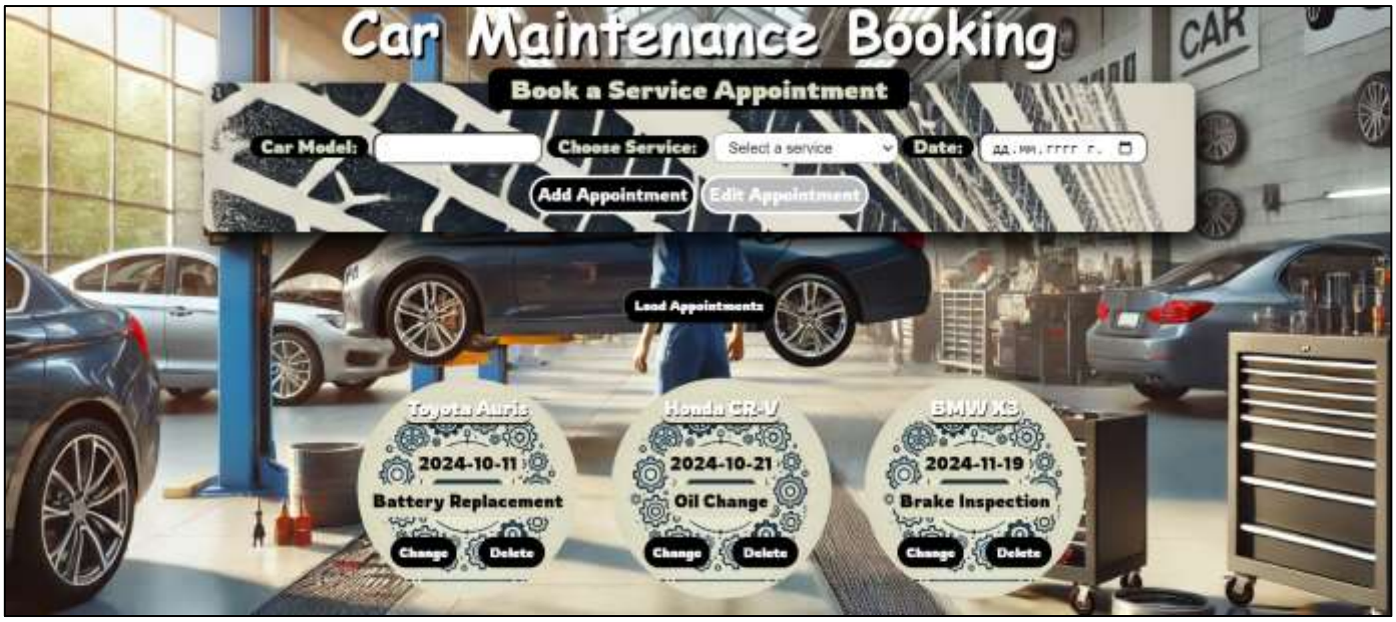


Clicking the **[Load Appointments]** button should send a **GET** request to the server to fetch **all appointments** from your local database. You must add each task to the **`<ul>`** with **`id="appointments-list"`**. The **[Edit Appointment]** button should be deactivated.

Each appointment has the following **HTML structure**:

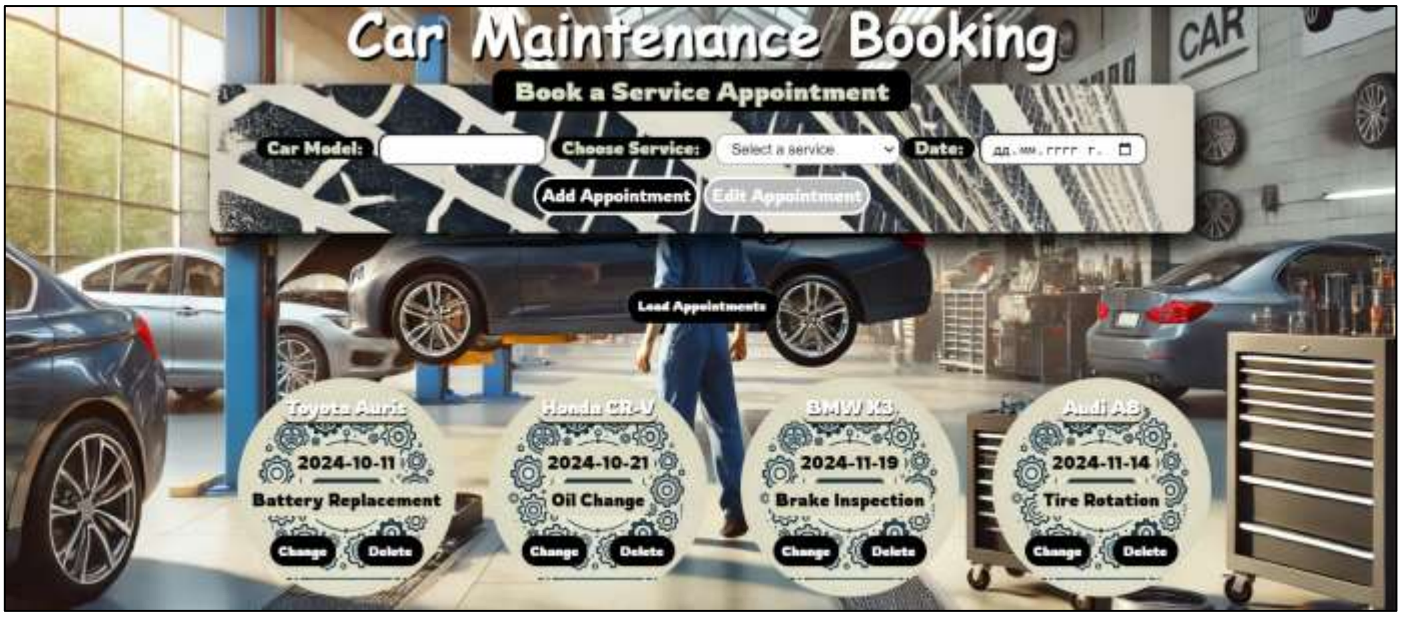


```
<li class="appointment"> flex
  <h2>Toyota Auris</h2>
  <h3>2024-10-11</h3>
  <h3>Battery Replacement</h3>
▼ <div class="buttons-appointment">
    <button class="change-btn">Change</button>
    <button class="delete-btn">Delete</button>
  </div>
</li>
```



Add a Appointment

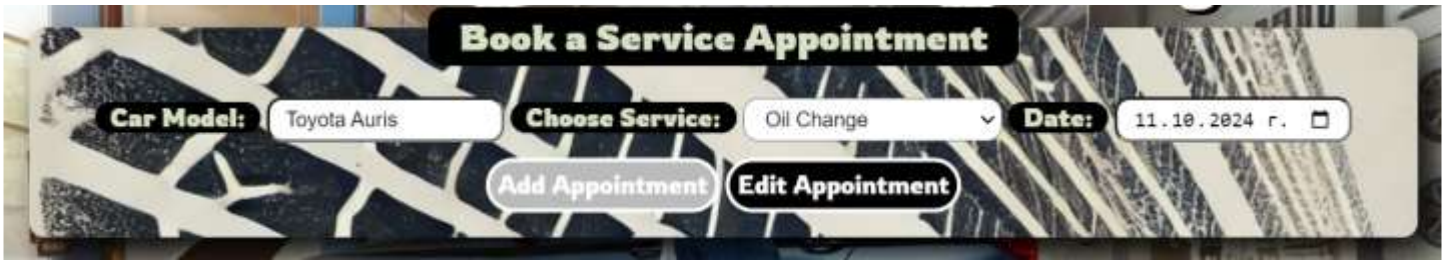
Clicking the **[Add Appointment]** button should send a **POST** request to the server, creating a new Appointment record with the **model**, **service and date** from the input values. After a successful creation, you should send another **GET** request to fetch all the Appointments including the **newly added one**. You should also **clear all the input fields** after the creation!



Edit an Appointment

Clicking the **[Change]** button should populate the info into the input fields above. The **[Edit Appointment]** button **in the form** should be activated and the **[Add Appointment]** one should be deactivated.

After clicking the [**Edit Appointment]** button in the form, you should send a **PUT** request to the server to **modify** the **model**, **service and date** of the changed item. After the successful request, you should **fetch the items again** and see that the changes have been made. After that, the **[Edit Appointment]** button should be deactivated and the **[Add Appointment]** one should be activated.

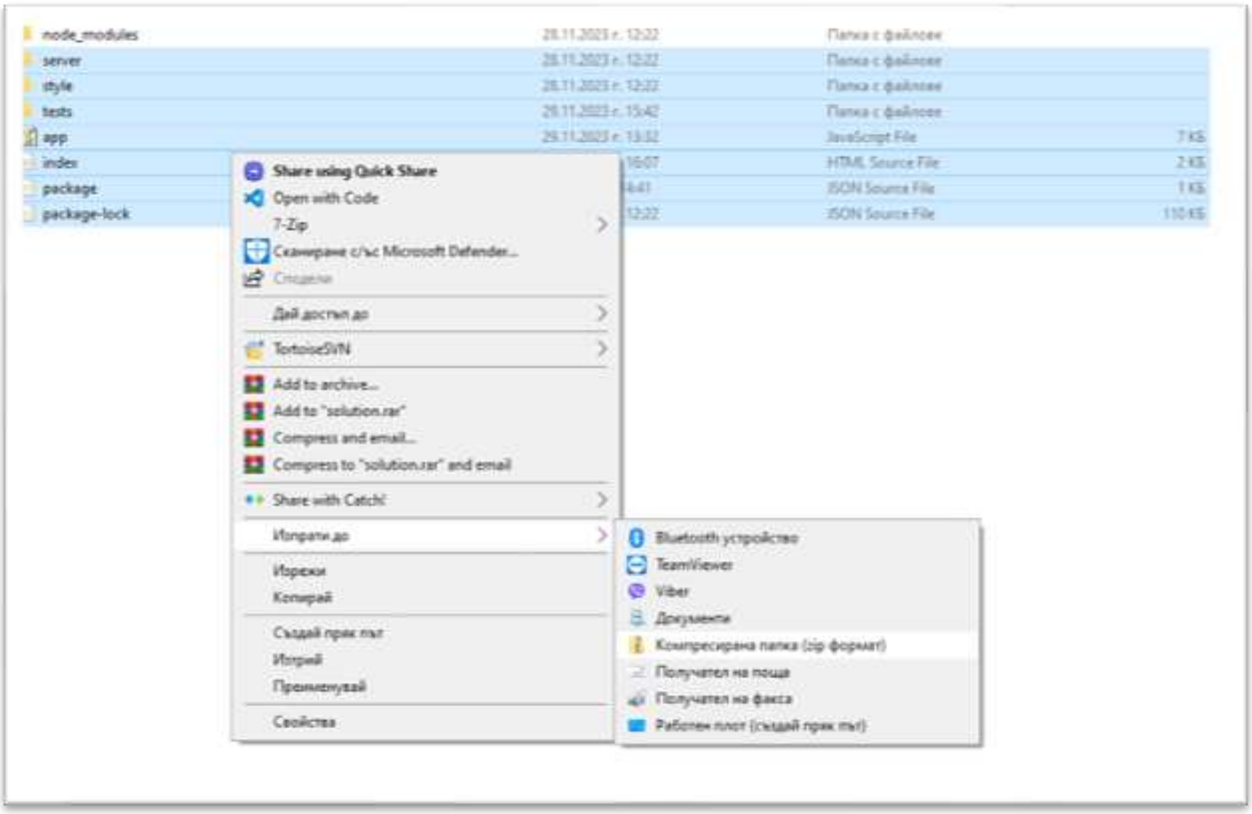


Delete Appointment

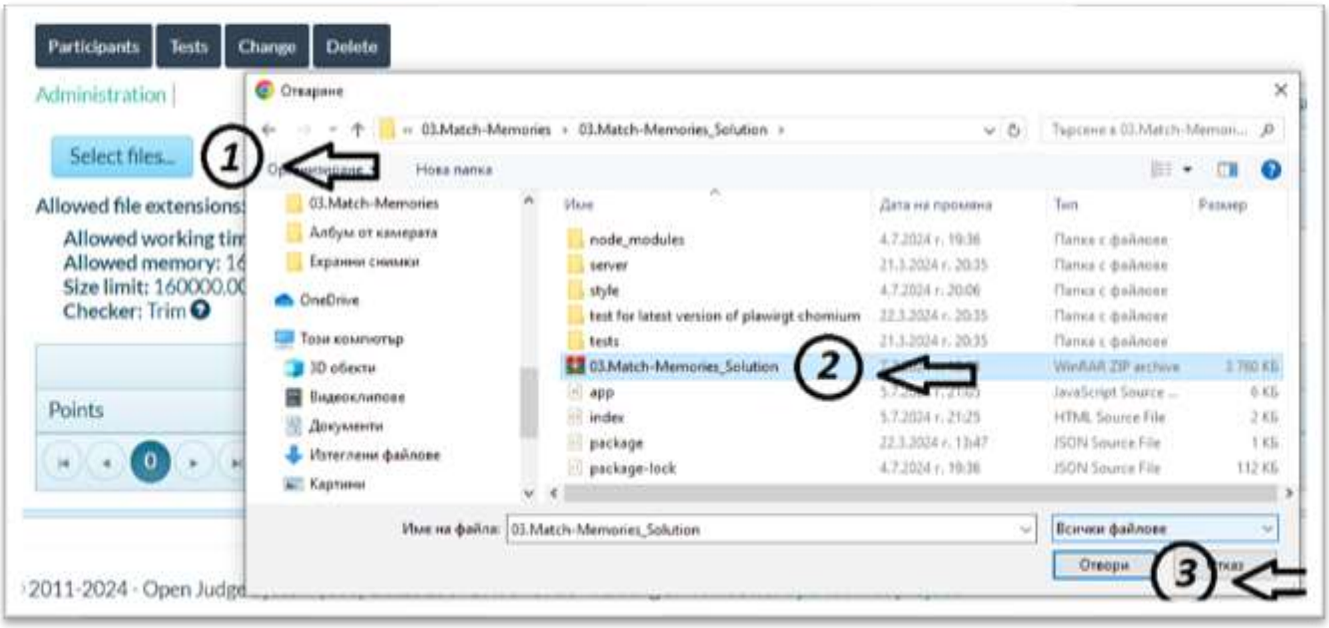
Clicking the **[Delete]** button should send a **DELETE** request to the server and remove the item from your local database. After you've removed it successfully, **fetch** the Appointments **again**.

Submitting Your Solution

Select the content of your working folder (the given resources). Exclude the *node_modules*. Archive the rest into a **ZIP** file and upload the archive to Judge.



node_modules	28.11.2023 г. 12:22	Папка с файлове	
server	28.11.2023 г. 12:22	Папка с файлове	
style	28.11.2023 г. 12:22	Папка с файлове	
tests	28.11.2023 г. 15:42	Папка с файлове	
app	29.11.2023 г. 13:32	JavaScript File	7 KB
index	16:07	HTML Source File	2 KB
package	4:41	JSON Source File	1 KB
package-lock	12:22	JSON Source File	110 KB



Allowed file extensions

Allowed working tim

Allowed memory: 16

Size limit: 160000.00

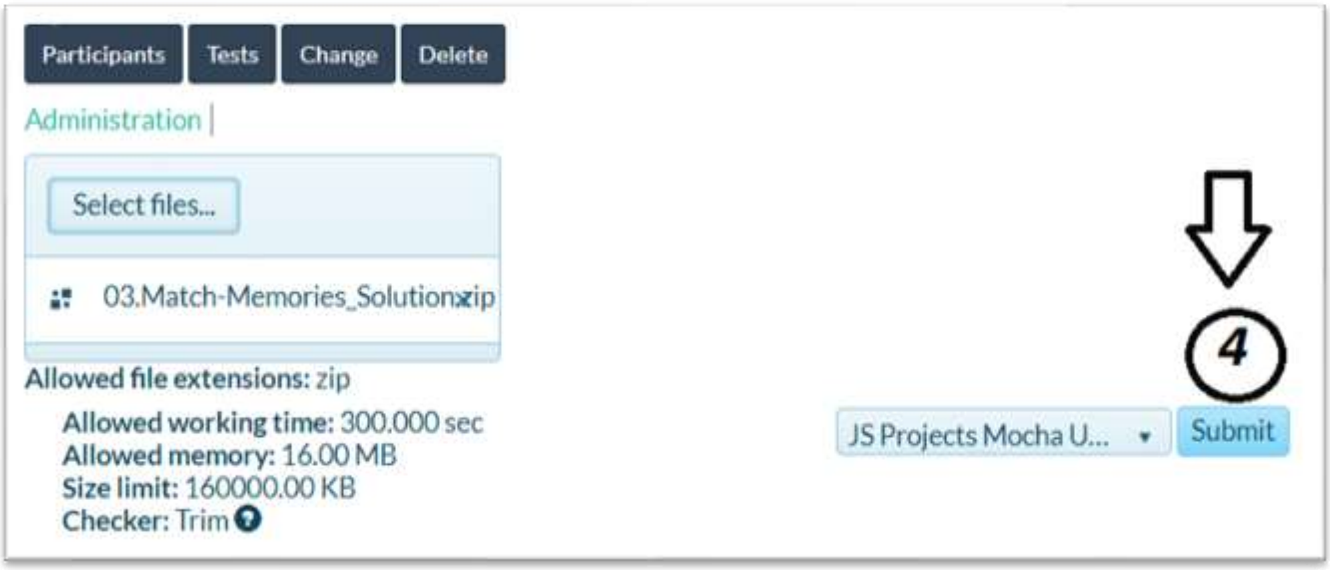
Checker: Trim

Име	Дата на промяна	Тип	Размер
node_modules	4.7.2024 г. 19:36	Папка с файлове	
server	21.3.2024 г. 20:35	Папка с файлове	
style	4.7.2024 г. 20:06	Папка с файлове	
test for latest version of plawirgt chomium	22.3.2024 г. 20:35	Папка с файлове	
tests	21.3.2024 г. 20:35	Папка с файлове	
03.Match-Memories_Solution		WinRAR ZIP archive	3 780 KB
app	5.7.2024 г. 21:03	JavaScript Source ...	6 KB
index	5.7.2024 г. 21:25	HTML Source File	2 KB
package	22.3.2024 г. 13:47	JSON Source File	1 KB
package-lock	4.7.2024 г. 19:36	JSON Source File	112 KB

Име на файла: 03.Match-Memories_Solution

Всички файлове

2011-2024 - Open Judge



Allowed file extensions: zip

Allowed working time: 300.000 sec
Allowed memory: 16.00 MB
Size limit: 160000.00 KB
Checker: Trim