# Exam Preparation – 9 April 2025

**Link to contest**: <https://judge.softuni.org/Contests/5155/JS-Front-End-Retake-Exam-19-December-2024>

## Problem 1 – Space Exploration Crew Management

*You are the captain of a space exploration mission. Your team consists of astronauts, each with their assigned spacecraft section and a set of skills they can perform.*

On the **first line** of the input, you will **receive** an integer **n** – the **number of astronauts** in your crew. On the next **n** lines, the **details of each astronaut** will follow in the format:

"{astronaut name} {spacecraft section} {skill 1,skill 2,...}"

The spacecraft section represents the area where the astronaut works (e.g., command module, engineering bay, research lab), and the list of skills includes the tasks they can perform in that section.

After forming your crew, you will receive different commands, separated by " / " (see the example below for clarity on the delimiter), until the command "End" is given. There are three types of actions astronauts can perform:

**"Perform / {astronaut name} / {spacecraft section} / {skill}"**

* If the astronaut is in the correct spacecraft section and can perform the skill, print:

**"{astronaut name} has successfully performed the skill: {skill}!"**

* Otherwise, print:

**"{astronaut name} cannot perform the skill: {skill}."**

**"Transfer / {astronaut name} / {new spacecraft section}"**

* The astronaut changes their assigned spacecraft section. Print:

**"{astronaut name} has been transferred to: {new spacecraft section}"**

**"Learn Skill / {astronaut name} / {new skill}"**

* If the astronaut already knows the skill, print:

**"{astronaut name} already knows the skill: {new skill}."**

* Otherwise, add the skill to their list and print:

**"{astronaut name} has learned a new skill: {new skill}."**

### Input

* On the **first line**, you will receive an integer **n**.
* On the next **n** lines, you will receive **details** about the **astronauts**.
* After that, you will receive **commands** until the "**End**" command.

### Output

* **Print** the results **of each command**.
* At the end, print a list of all astronauts. Show their updated spacecraft sections and skill lists. Arrange the **skills in alphabetical order**:

**"Astronaut: {astronaut name}, Section: {spacecraft section}, Skills: {skill 1, skill 2, ...}"**

### Constraints

* The names of the astronauts will be unique.
* All given commands will be valid.

### Examples

|  |  |
| --- | --- |
| **Input** | **Output** |
| [  "2",  "Alice command\_module piloting,communications",  "Bob engineering\_bay repair,maintenance",  "Perform / Alice / command\_module / piloting",  "Perform / Bob / command\_module / repair",  "Learn Skill / Alice / navigation",  "Perform / Alice / command\_module / navigation",  "Transfer / Bob / command\_module",  "Perform / Bob / command\_module / maintenance",  "End"  ] | Alice has successfully performed the skill: piloting!  Bob cannot perform the skill: repair.  Alice has learned a new skill: navigation.  Alice has successfully performed the skill: navigation!  Bob has been transferred to: command\_module  Bob has successfully performed the skill: maintenance!  Astronaut: Alice, Section: command\_module, Skills: communications, navigation, piloting  Astronaut: Bob, Section: command\_module, Skills: maintenance, repair |
| **Input** | **Output** |
| [  "3",  "Tom engineering\_bay construction,maintenance",  "Sara research\_lab analysis,sampling",  "Chris command\_module piloting,communications",  "Perform / Tom / engineering\_bay / construction",  "Learn Skill / Sara / robotics",  "Perform / Sara / research\_lab / robotics",  "Transfer / Chris / research\_lab",  "Perform / Chris / research\_lab / piloting",  "Learn Skill / Tom / diagnostics",  "Perform / Tom / engineering\_bay / diagnostics",  "End"  ] | Tom has successfully performed the skill: construction!  Sara has learned a new skill: robotics.  Sara has successfully performed the skill: robotics!  Chris has been transferred to: research\_lab  Chris has successfully performed the skill: piloting!  Tom has learned a new skill: diagnostics.  Tom has successfully performed the skill: diagnostics!  Astronaut: Tom, Section: engineering\_bay, Skills: construction, diagnostics, maintenance  Astronaut: Sara, Section: research\_lab, Skills: analysis, robotics, sampling  Astronaut: Chris, Section: research\_lab, Skills: communications, piloting |

## Problem 2.Event Tracker

**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**.

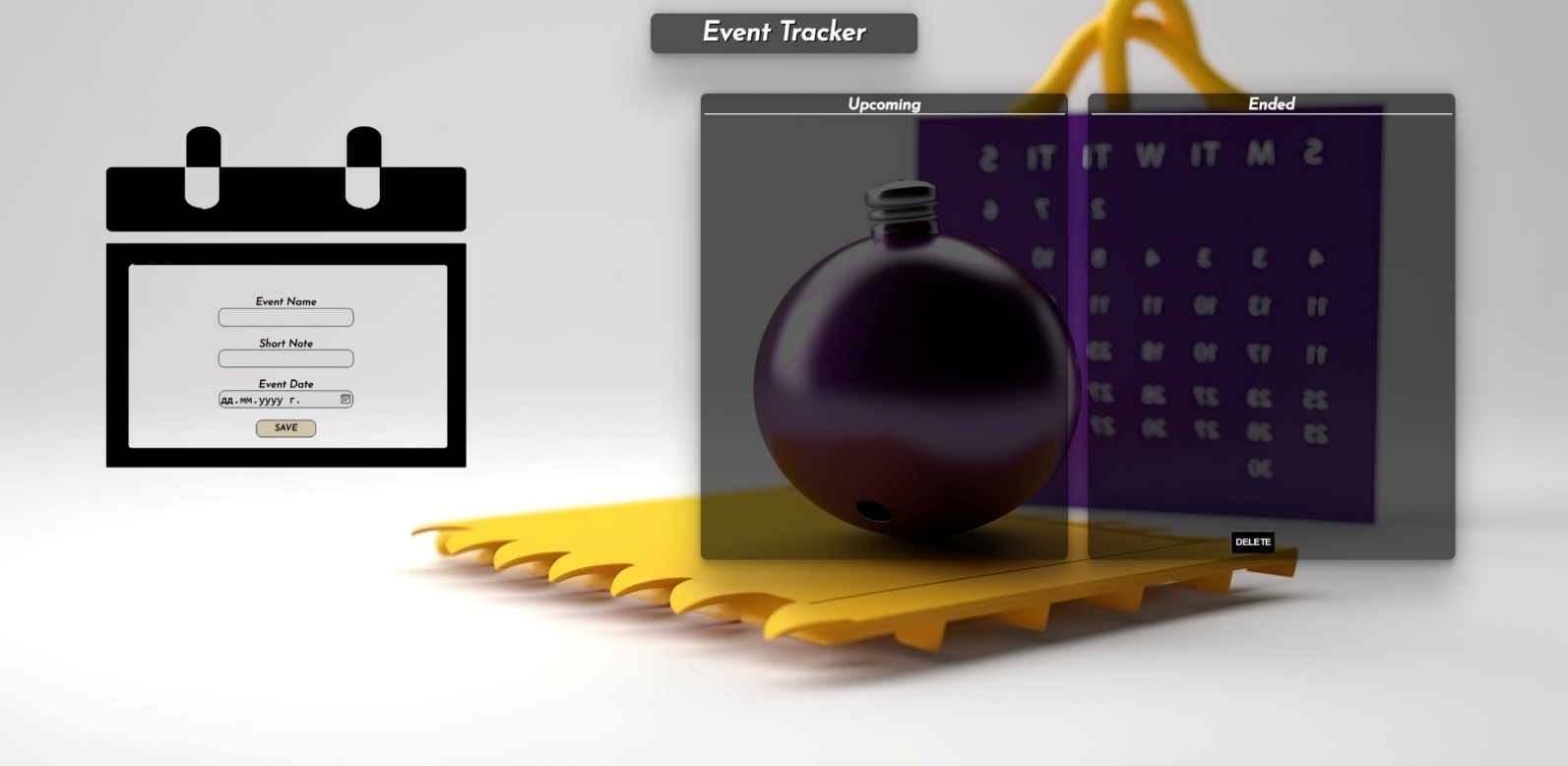
The following actions are **NOT** supported:

* **.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 **Event Tracker** work as expected:

* + **Event Name, Note,** and **Date** should be **non**-**empty** **strings**. If any of them are empty, the program should not do anything.

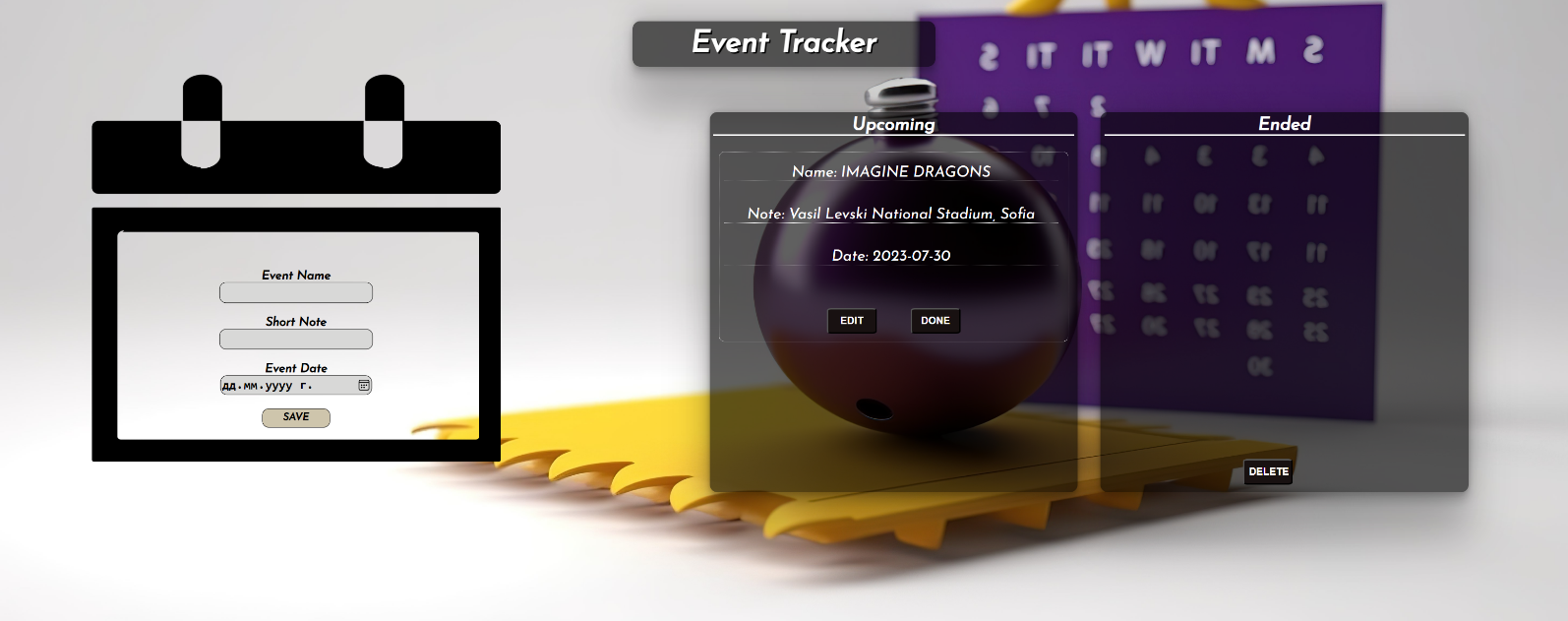
### Get the information from the form

When you click the **[Save]** button, the information from the input fields must be added to the <ul> with the id "upcoming-list",**[Save]** button **should allow multiple event adding** and **the input fields should be cleared**.

The HTML structure should look like this:

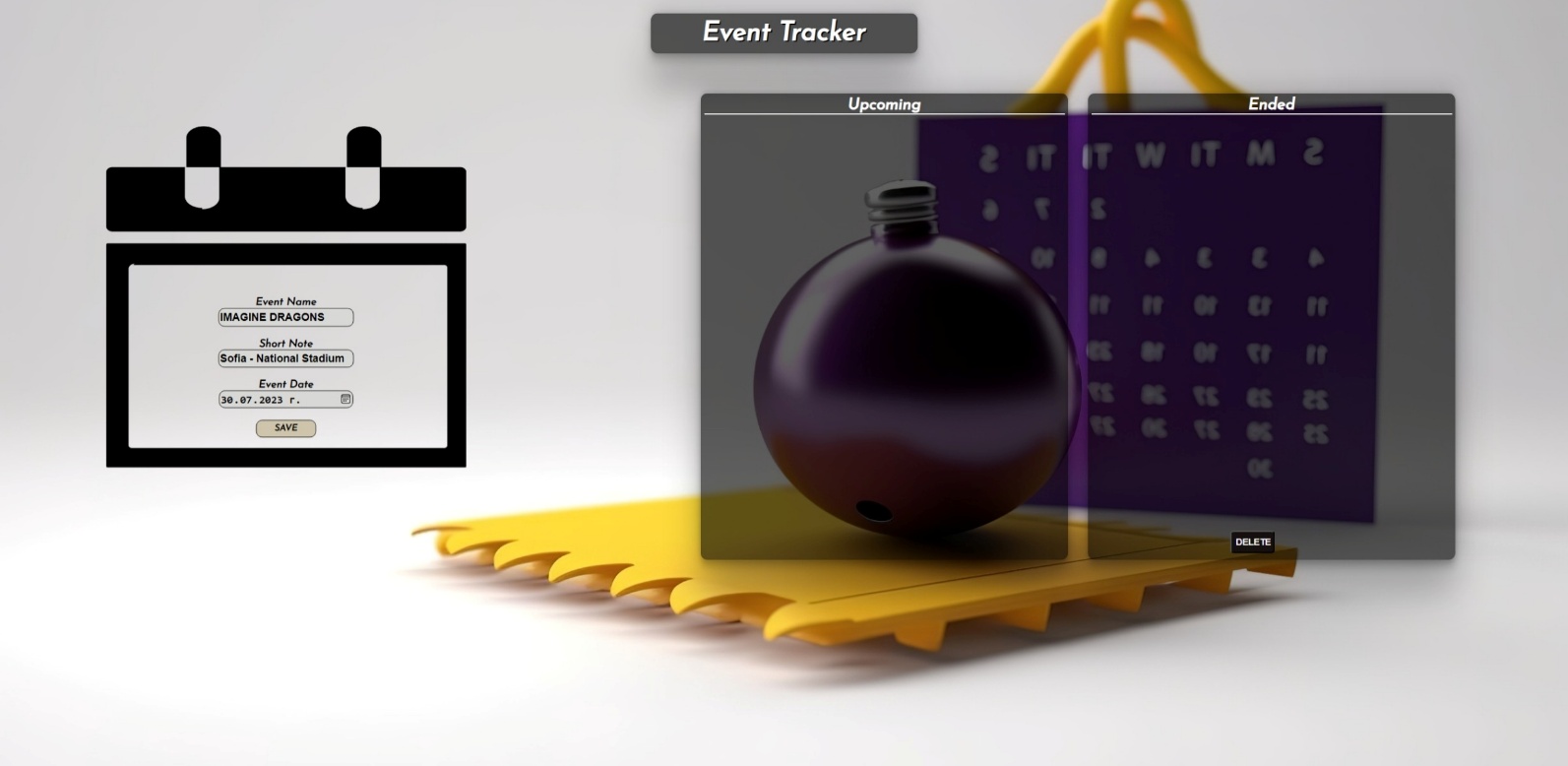
Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично

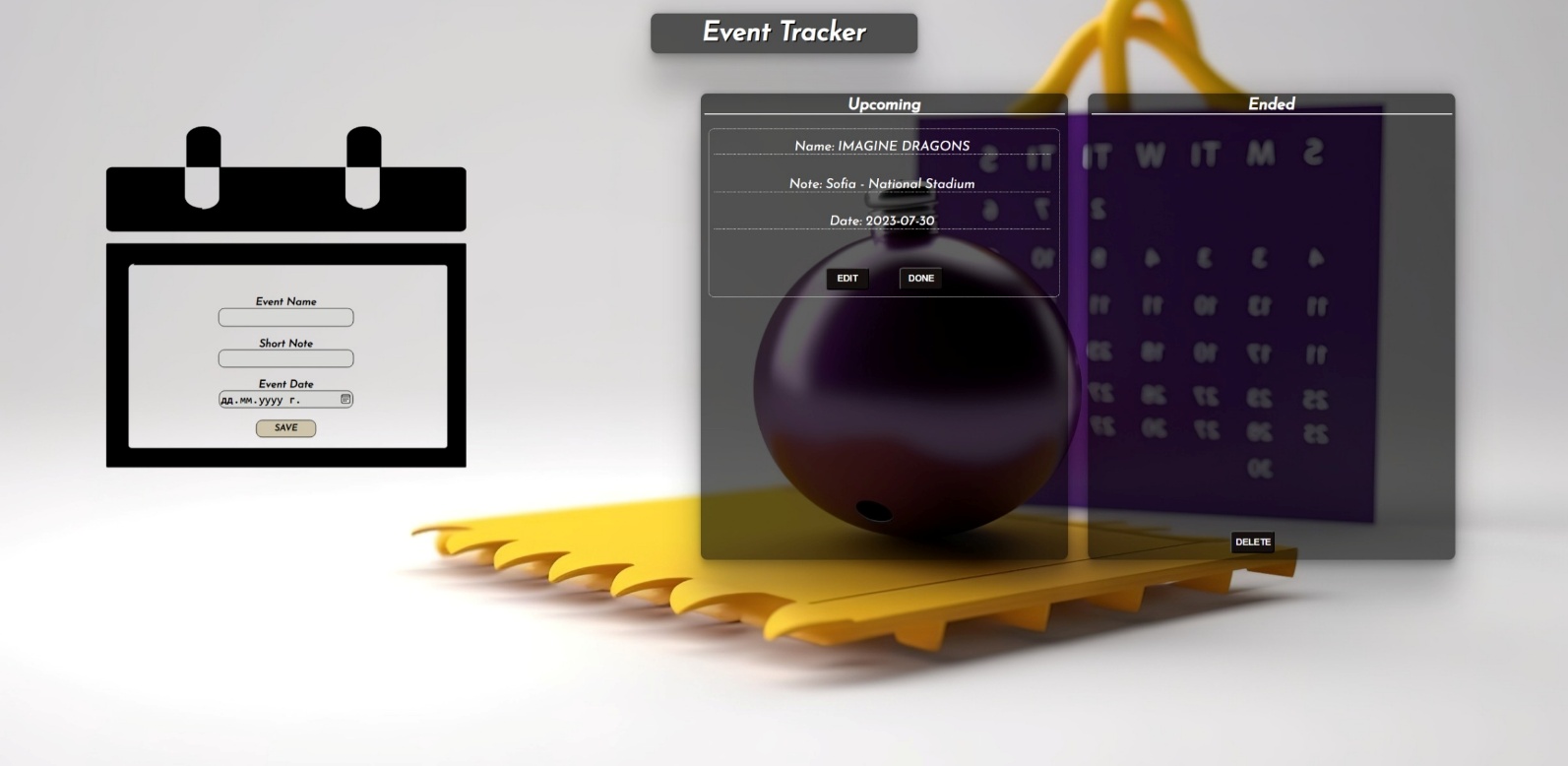


### Edit the Information

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



After editing the information, add a new item to the <ul> "upcoming-list" with the updated information.



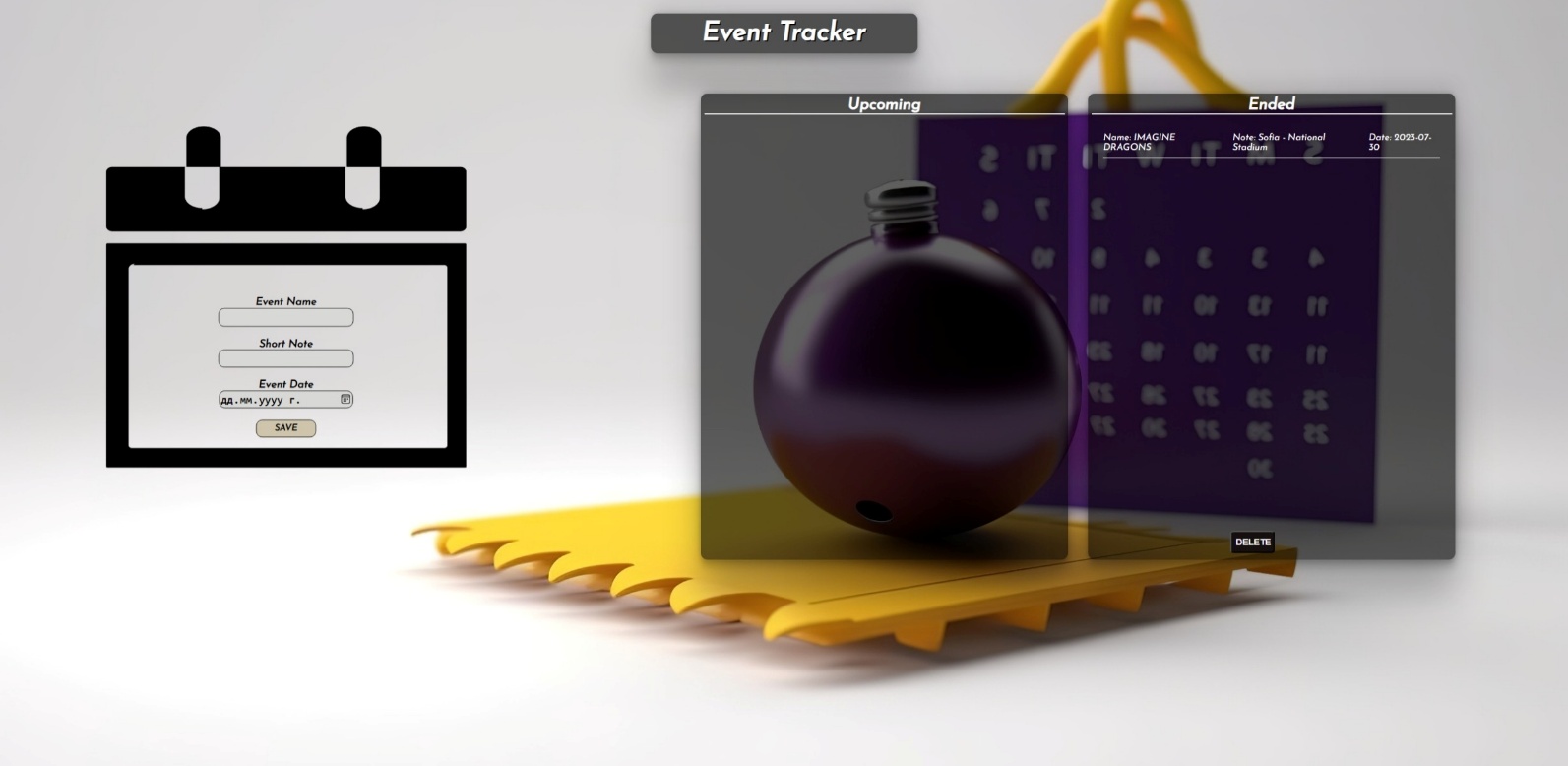
### End the event

When you click the **[Done]** button, the record must be **deleted** from the <ul>withid"upcoming-list" and appended to the <ul> with id "events-list".

The **buttons [Edit]** and **[Done]** should be removed from the <li> element.

Картина, която съдържа текст, екранна снимка, Шрифт

Описанието е генерирано автоматично



### Delete Ended Events

When the **[Delete]** button is clicked, you must **delete all ended events.**

### Submission

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

## Problem 3 – StockFlow

**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](https://github.com/softuni-practice-server/softuni-practice-server).

**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 stock orders. 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:

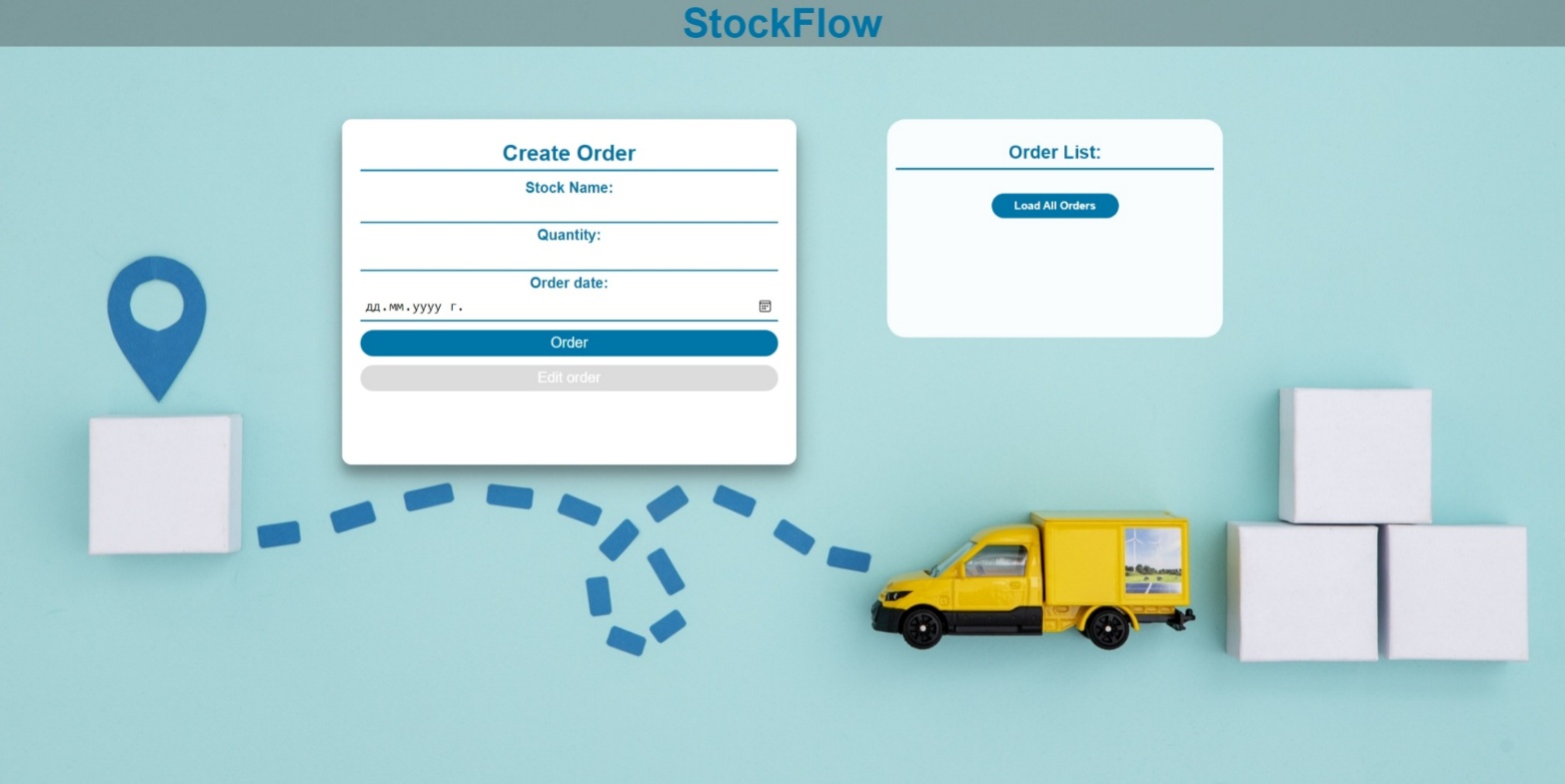
Картина, която съдържа текст, Шрифт, екранна снимка

Описанието е генерирано автоматично

### Endpoints

* <http://localhost:3030/jsonstore/orders/>
* <http://localhost:3030/jsonstore/orders/:id>

### Load All Orders

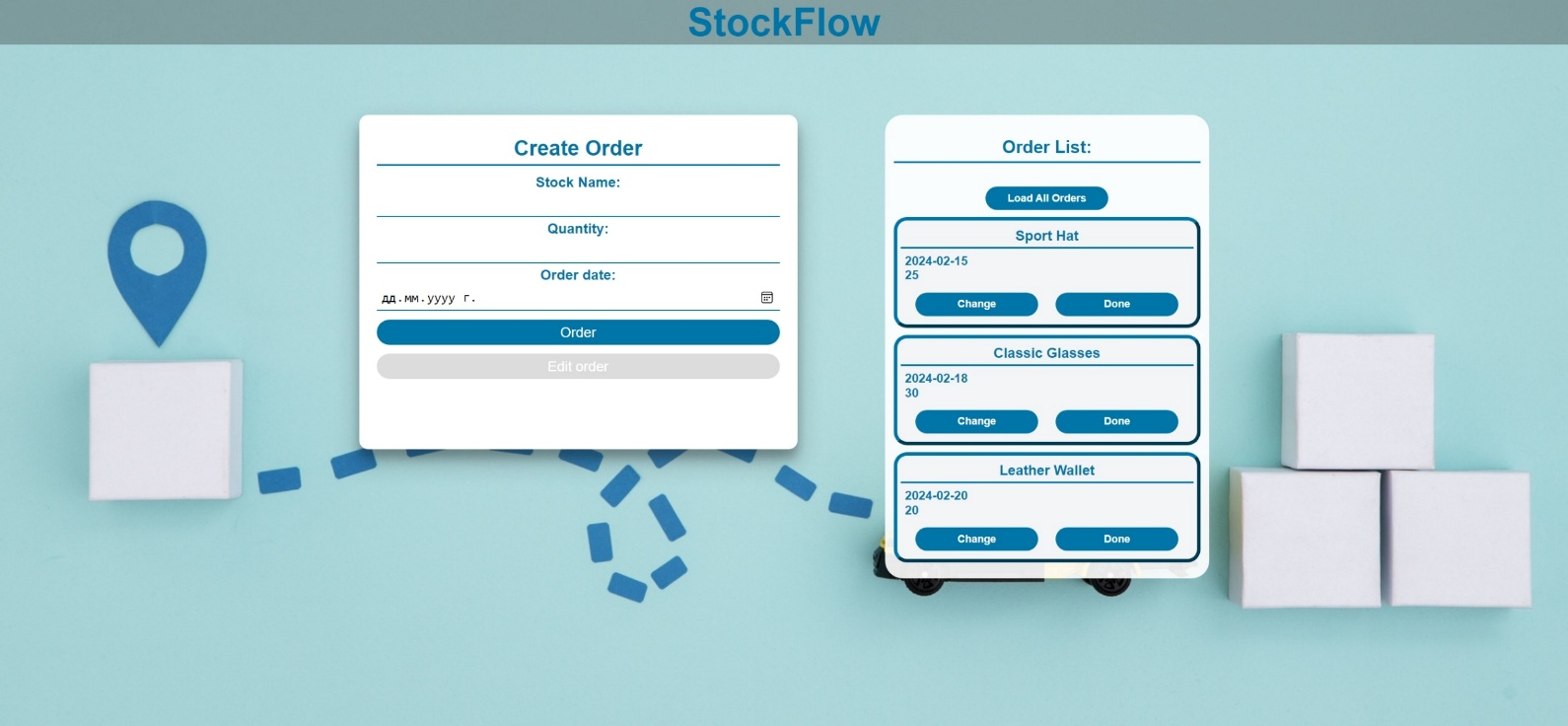


Clicking the **[Load All Orders]** button should send a **GET** request to the server to fetch **all stock orders** from your local database. You must add each task to the<div>with **id="list". [Edit order]** button should be deactivated.

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

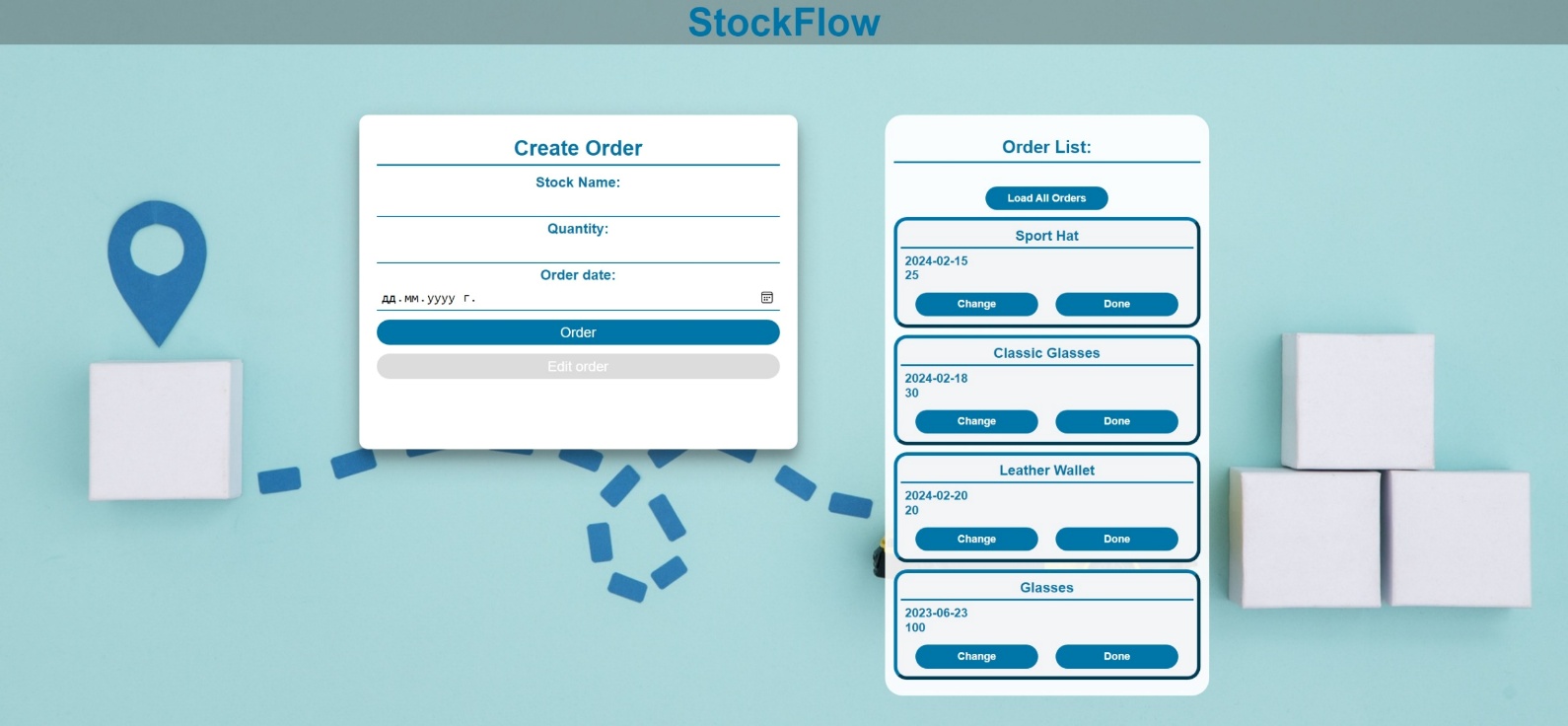
Картина, която съдържа текст, Шрифт, екранна снимка

Описанието е генерирано автоматично



### Create an Order

Clicking the **[Order]** button should send a **POST** request to the server, creating a new stock order with the **stock** **name**, **quantity, and order date** from the input values. After a successful creation, you should send another **GET** request to fetch all the stock orders, including the **newly created** into the **Order List** column. You should also **clear all the input fields** after the creation!



### Edit an Order

Clicking the **[Change]** button on a record should remove the record from the DOM structure and the information about the task should be populated into the input fields above. The **[Edit** **order]** button **in the** **form** should be activated and the **[Order]** one should be deactivated.

After clicking the [**Edit order]** button in the form, you should send a **PUT** request to the server to **modify the stock name, quantity and the date** of the changed order. After the successful request, you should **fetch the orders again** and see that the changes have been made. After that, the **[Edit** **order]** button should be deactivated and the **[Order]** one should be activated.

Картина, която съдържа текст, екранна снимка, Шрифт, номер

Описанието е генерирано автоматично

### Mark as Done

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

### Submitting Your Solution

Select the content of your working folder (the given resources). Exclude the *node\_modules* & *tests* folders. Archive the rest into a **ZIP** file and upload the archive to Judge.

Graphical user interface, application

Description automatically generated

Картина, която съдържа текст

Описанието е генерирано автоматично

Картина, която съдържа текст

Описанието е генерирано автоматично