# Problem 3 – Weather Tracker

**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

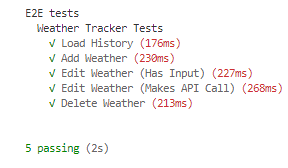
Write a JS program that can load, create, remove and edit a list of scheduled vacations. You will be given an HTML template to which you must bind the needed functionality.

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)**.

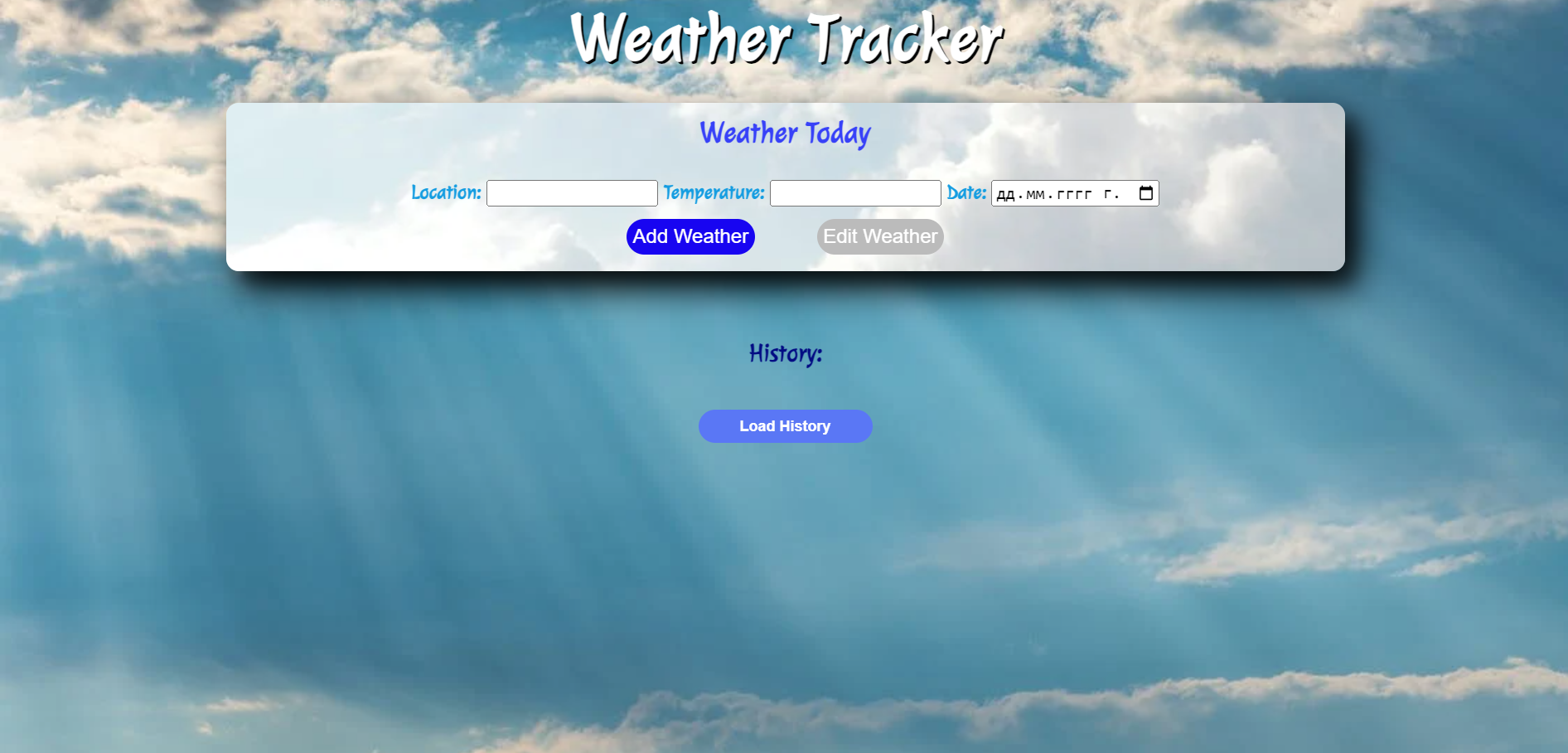
At any point, you can open up another console and run npm test to test the **current state** of your application. It’s preferable for **all of your tests to pass locally** before you submit to the Judge platform, like this:



## Endpoints

* [**http://localhost:3030/jsonstore/tasks/**](http://localhost:3030/jsonstore/tasks/)
* [**http://localhost:3030/jsonstore/tasks/**](http://localhost:3030/jsonstore/tasks/)**:id**

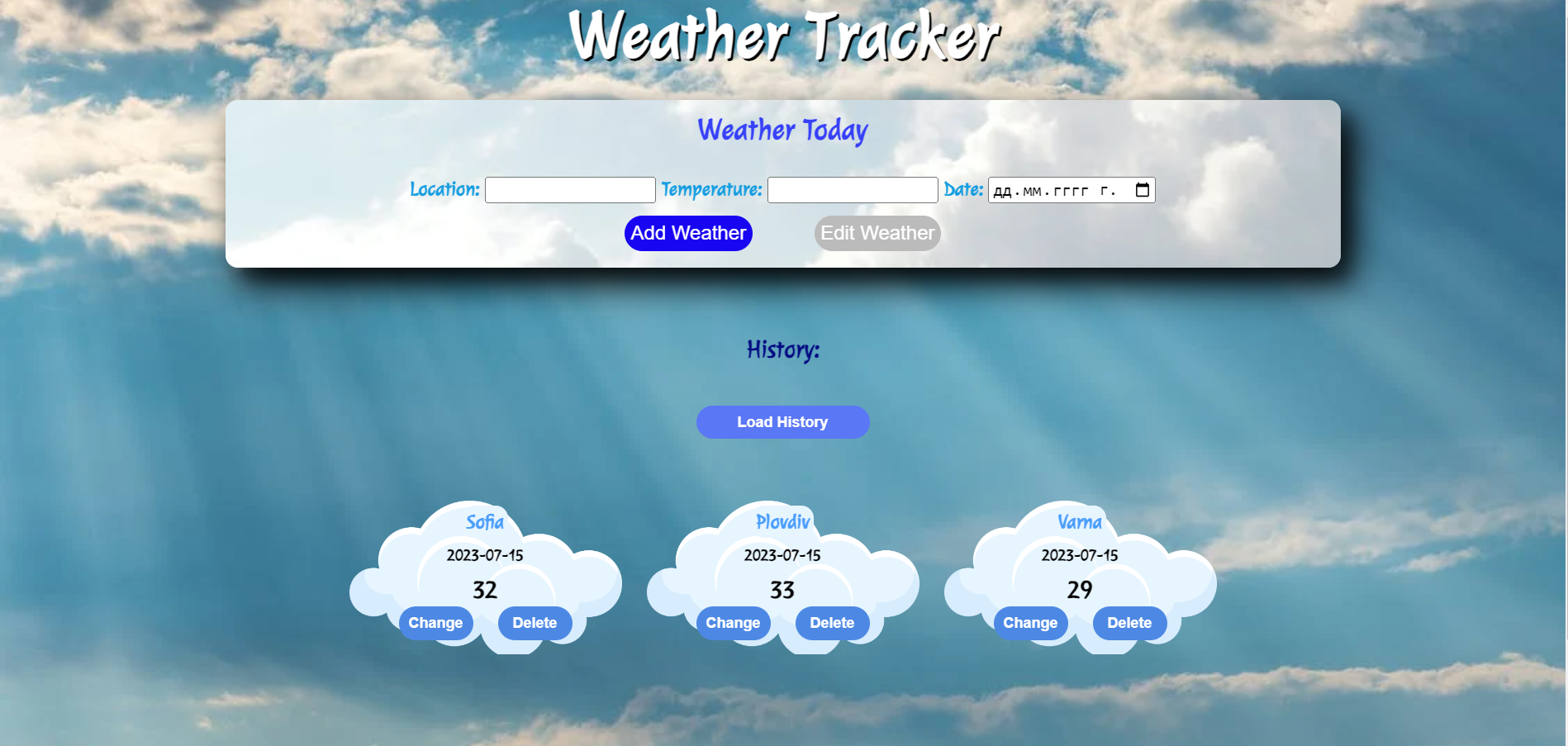
## Load History



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

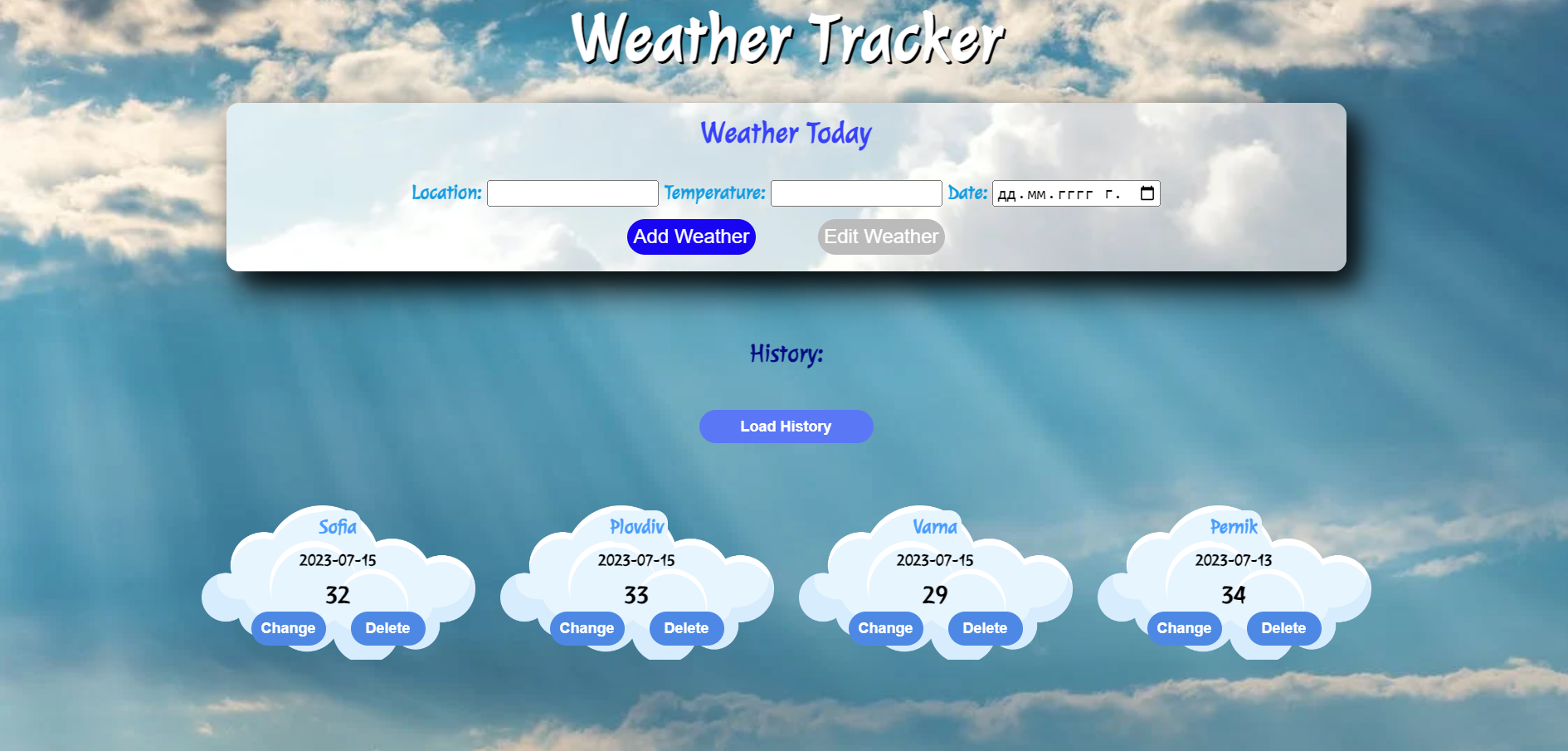
Each record has the following **HTML** **structure**:





## Add а Weather

Clicking the **[Add Weather]** button should send a **POST** request to the server, creating a new scheduled vacation with the **location**, **temperature, and date** from the input values. After a successful creation, you should send another **GET** request to fetch all the scheduled vacations, including the **newly added one** into the **History** column. You should also **clear all the input fields** after the creation!



## Edit a Weather

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** **Weather]** button **in the** **form** should be activated and the **[Add Weather]** one should be deactivated.

After clicking the [**Edit Weather]** button in the form, you should send a **PUT** request to the server to **modify the location, temperature and the 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** **Weather]** button should be deactivated and the **[Add Weather]** one should be activated.



## Delete

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 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